# MODELING INTERCITY MODE CHOICE AND AIRPORT CHOICE IN THE U.S.

Senanu Ashiabor

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Civil Engineering

February 2, 2007
Blacksburg, Virginia

**Keywords**: Mode Choice, Airport Choice, Mixed Logit, Nested Logit, Intercity Travel Demand.

# Modeling Intercity Mode Choice and Airport Choice in the U.S.

Senanu Ashiabor

## Abstract

The aim of this study was to develop a framework to model travel choice behavior in order to estimate intercity travel demand at nation-level in the United States. Nested and mixed logit models were developed to study national-level intercity transportation in the United States. A separate General Aviation airport choice model to estimates General Aviation person-trips and number of aircraft operations though more than 3000 airports was also developed. The combination of the General Aviation model and the logit models gives the capability to estimate a full spectrum of intercity travel demand in the United States.

The logit models were calibrated using a nationwide revealed preference survey (1995 American Travel Survey). Separate models were developed for business and non-business trip purposes. An airport choice model is integrated into the mode choice model to estimate both the market share between any origin-destination pair and other modes of transportation, and the market share split between airports associated with the origin-destination pairs. The explanatory variables used in the utility functions of the models are travel time, travel cost, and traveler's household income. The logit models are used to estimate the market share of automobile and commercial air transportation between 3091 counties and 443 commercial service airports in the United States. The model was also used to estimate market share for on-demand air taxi services. Given an input county-to-county trip demand table, the models were used to estimate county-to-county travel demand by automobile and commercial airline between all counties and commercial service airports in the United States. The model has been integrated into a computer software framework called the Transportation Systems Analysis Model (TSAM) that estimates nationwide intercity travel demand in the United States.

## ACKNOWLEDGEMENTS

I want to thank my advisors Dr. Trani and Dr. Baik for their consistent support and guidance throughout my work on this dissertation. I have benefited immensely from the depths of their technical knowledge and I have also grown to know them as friends. I also want to express my gratitude the members of my committee who have helped in various ways. Dr. Rakha provided direction with very insightful comments during my proposal presentation, Dr. McGuirk was instrumental piquing my interest in econometric modeling and Dr. Teodorovic provided me with a lot of guidance and advice throughout the processes.

I am thankful for the comments and encouragement of all the team members of the Virginia Tech Air Transportation Systems Laboratory. I also want to thank Stuart Cooke, Jeff Viken (NASA Langley Research Center), Sam Dollyhigh and Jeremy Smith (Swales Aerospace) who at various times offered constructive criticism and suggestions of the mode choice model that is implemented in TSAM.

Several of my friends and colleagues have provided a lot of support and advice during time I have worked on this dissertation. My sincere thanks goes out you all, and though I cannot mention everyone by name I especially want to thank Dr. Nii Kwashie Allotey, Dr. Della Baker, Ted Asare, Richard Johnson and Emma Mensah. My ultimate thanks and appreciation is my God and Savior Jesus Christ who has been my guide and comfort over the past six years.

## DEDICATION

***To Daddy and MamaRee.***

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# 1 INTRODUCTION

## 1.1 BACKGROUND AND MOTIVATION FOR RESEARCH

In 2000, the National Aeronautics and Space Administration (NASA) proposed to Congress the development of a Small Aircraft Transportation System (SATS) to harness the potential of the nation's vast network of underutilized airports. The SATS concept is similar to the point-to-point, on-demand service offered by current operators of air-taxi and fractional ownership services. SATS is meant to reduce travel time, provide safe, affordable and improved access to rural and remote communities in the US.

Congress mandated NASA to prove four operational capabilities of the system. Namely, 1) Higher volume operations of SATS vehicles in non-radar airspace, 2) Lower landing minimums at minimally equipped airports, 3) Increased single-pilot crew safety and 4) En-route procedures and systems for integrated direct fleet operation. NASA assigned the Virginia Tech Air Transportation Systems Laboratory the task of developing an Integrated Systems Analysis Model to estimate potential demand for the SATS technology.

Using the classical four-step transportation planning procedure shown in Figure 1 Virginia Tech developed the Transportation Systems Analysis Model (TSAM) to estimate demand for small aircraft used in on-demand services. The four-step planning model is a sequential demand forecasting model made up of trip generation, trip distribution, mode choice and trip assignment.

**Trip generation** is used to predict the number of trips produced from each zone of activity and attracted to each zone of activity by trip purpose. Counties are used as zones of activity and there are 3091 counties in the model. As shown in Figure 2 the output of the procedure is and origin-destination matrix with two vectors for attractions and productions respectively.

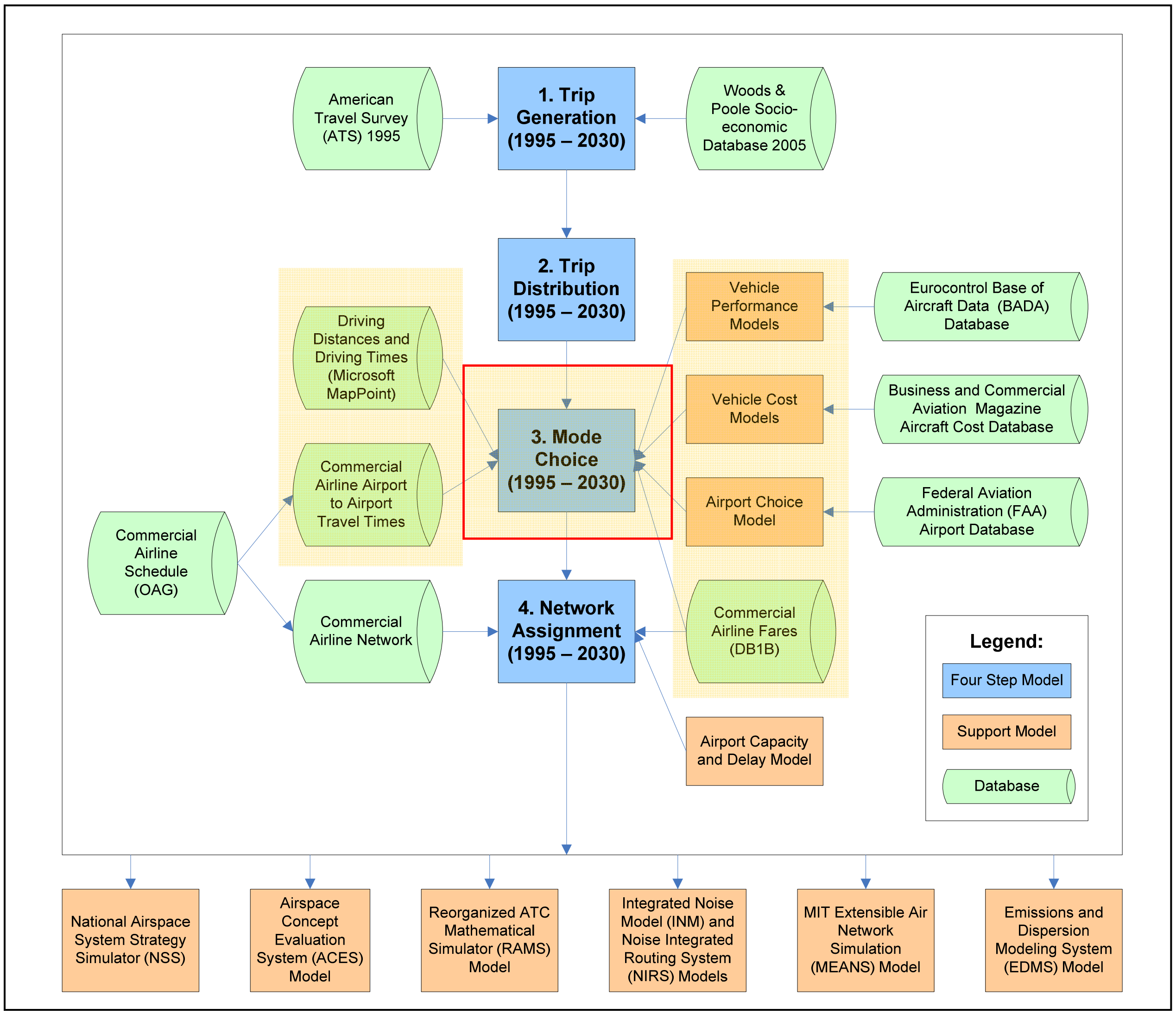
American Travel Survey (ATS) 1995
1. Trip Generation (1995 – 2030)
Woods & Poole Socio-economic Database 2005
2. Trip Distribution (1995 – 2030)
Driving Distances and Driving Times (Microsoft MapPoint)
Commercial Airline Airport to Airport Travel Times
3. Mode Choice (1995 – 2030)
Vehicle Performance Models
Eurocontrol Base of Aircraft Data (BADA) Database
Vehicle Cost Models
Business and Commercial Aviation Magazine Aircraft Cost Database
Airport Choice Model
Federal Aviation Administration (FAA) Airport Database
Commercial Airline Schedule (OAG)
Commercial Airline Network
4. Network Assignment (1995 – 2030)
Commercial Airline Fares (DB1B)
Airport Capacity and Delay Model
Legend:
Four Step Model
Support Model
Database
National Airspace System Strategy Simulator (NSS)
Airspace Concept Evaluation System (ACES) Model
Reorganized ATC Mathematical Simulator (RAMS) Model
Integrated Noise Model (INM) and Noise Integrated Routing System (NIRS) Models
MIT Extensible Air Network Simulation (MEANS) Model
Emissions and Dispersion Modeling System (EDMS) Model

**Figure 1: TSAM Framework Highlighting 4-Step Planning Procedure.**

**Trip distribution** is used to predict the origin-destination flows. This links the trip ends from the trip generation to form trip interchanges between zones. Thus the output of trip generation is the input into the trip distribution model. The result is a trip interchange table of person-trips per year between counties as shown in Figure 2. This model is segregated by business and non-business trip purposes and has five income groups. Thus, the output of the trip distribution process are ten 3091x3091 trip tables. Five tables for the business travelers and five for non-business travelers. Each of the five tables represents an income group.

**Mode choice** predicts the percentage of person-trips selecting each mode and is the focus of this dissertation. The output of the trip distribution serves as the input to the mode choice model. Hence, the mode choice model splits each of the ten tables by mode. If the modes under consideration are commercial airline and automobile then the output of the mode choice model will be twenty 3031x3091 trip tables. The mode choice model is the third step presented in Figure 2.

**Trip assignment** converts the origin-destination flows for each mode on specific routes through the respective networks. In this dissertation the airport choice model which is embedded in the mode choice model is used to estimate person-trips between each airport pair. This is the fourth step presented in Figure 2. The final step would be to convert the airport-to-airport person-trips into aircraft operations between the airports. The complete travel demand model is fully documented in Trani et al. (2002, 2003).

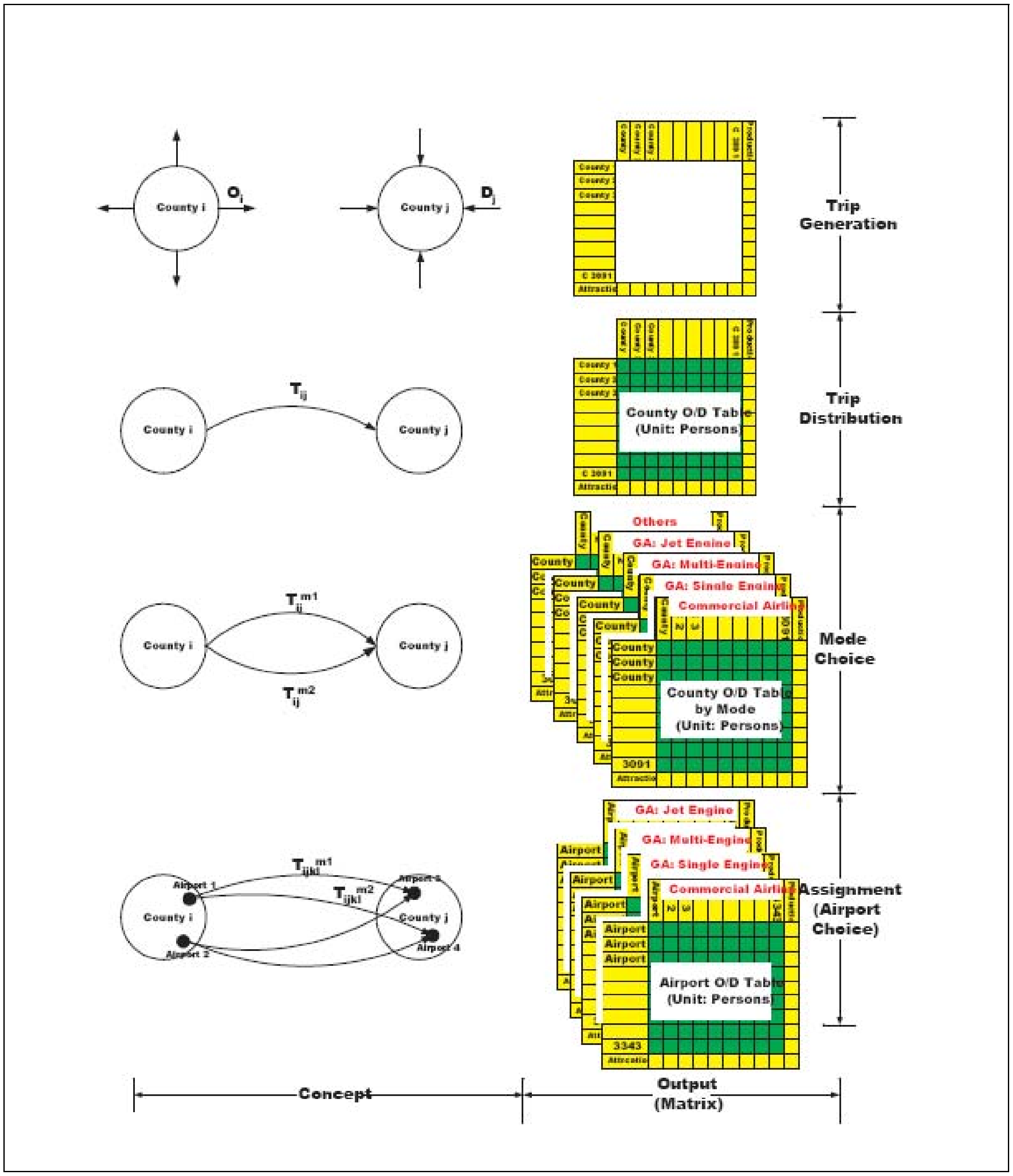

**Figure 2 Multi-step Illustration of Trip Demand Analysis.**

## 1.2 PROBLEM DEFINITION

The focus of this dissertation is the development of behavioral models to capture the decision making behavior of intercity travelers in the U.S. The aim is to determine how a traveler selects their mode of transport from a set of possible transportation alternatives when making a trip between any two locations. The models seek to generate probability estimates of selecting each mode of transport for each traveler. The probability estimates can then be used in conjunction with trip volumes to estimate the total number of travelers using each mode of transportation.

The model is developed within the random utility maximization framework. This framework presents individuals as utility maximizing entities. Faced with a choice set of $K$ alternatives, each individual $n$ attaches a utility $U_{nk}$ to the $k^{th}$ alternative that is a composite of the individuals attributes and attributes of the alternative. The modeler has information about certain attributes of the individual $a_n$ and attributes of the each alternative $x_{nk} \forall k$ that are postulated to have factored into the individual's decision making process. The utility of the individual can then be framed as $U_{nk} = V_{nk} + \varepsilon_{nk}$ where $\varepsilon_{nk}$ encapsulates the additional information in $U_{nk}$ that is known to the individual but not the modeler. $V_{nk}$ is usually referred to as the representative utility. The modeler treats the term $\varepsilon_{nk} \forall k$ as random hence the name random utility. The probability that an individual selects a specific alternative $i$ can be written as $P_{ni} = \Pr ob(U_{ni} \rangle U_{nk} \forall k \neq i)$ which decomposes to $P_{ni} = \ Prob(V_{ni} + \varepsilon_{ni} \rangle V_{nj} + \varepsilon_{nj} \ldots \forall j \neq i)$. The modeler seeks to specify an appropriate distribution for $V_{nk}$ that captures the individual's decision making behavior. Information about the individual, their choice and attributes of the alternative modes of transportation are usually contained in survey data that has been collected.

The model developed in this dissertation was initially calibrated using the 1995 American Travel Survey (ATS) database. The ATS is a nationwide survey of travelers in the U.S. conducted by the Census Bureau. The ATS has several limitations that hindered the calibration process, most notable is an existing privacy regulation enacted by the United States Congress that limits the level of geographical detail that can be released to the public. To improve model credibility, four revealed and stated preference personal travel surveys were developed and administered to collect more detailed information to augment the ATS. The survey was also used to ascertain traveler’s response to proposed aerospace technology similar to the Very Light Jets being developed.

The model developed is implemented in TSAM to estimate the market share for automobile, commercial airline and SATS for any pair of counties in the continental U.S.. The embedded airport choice model allows estimation of market share between commercial service airport routes between any origin-destination county pairs. Rail was incorporated as an additional mode in the model. Though the final model is implemented in the TSAM framework, the calibrated model can be implemented for any trip with the necessary socio-economic variables and trip characteristic data.

The focus of the dissertation is be the use of the ATS and the Personal Travel Surveys to develop a flexible modeling tool that can be used for credible intercity travel demand estimation and forecasting. In particular, various forms of logit models are developed and enhancements made to the airport choice model to improve the credibility of the model estimates.

## 1.3 RELEVANCE OF RESEARCH

The Program Development Office (JPDO) is currently using the model to plan the Next Generation Air Transportation System. NASA Langley is using the model to study demand for various aircraft technologies like supersonic business aircraft, tilt-rotors and Short Take-off and Landing (STOL) aircraft. In addition, Boeing Commercial Aircraft and the Embraer Aircraft Manufacturer have expressed interest in using the model to study demand for aviation technologies. The above shows the model is very relevant and the output is critical to policy makers.

Though the TSAM model was initially developed for NASA policy makers to estimate demand for SATS, the output in terms of automobile and commercial airline demand is critical to the decision making process of several federal agencies such as the Department of Transportation, Federal Highway Authority, State Metropolitan Authorities, Airport Authorities, Airlines and Transportation Planners.

## 1.4 OUTLINE OF DISSERTATION

The rest of the document is outlined as follows. The Literature Review chapter is a detailed review of how logit models have been applied in intercity travel demand modeling. The review was approached along two dimensions: past attempts of using logit models for travel demand models at the national level, and secondly a step by step review of the historical development of logit models in terms of the model structure to date.

Next a paper on the development and calibration of nested and mixed logit models for intercity mode choice in the United States is presented. The model presented in the paper was calibrated using the 1995 American Travel Survey. The model in the paper estimates market share for Automobile and Commercial Airline modes of transportation. The model was further used to estimate potential demand for Very Light Jets currently being introduced as a mode of transportation in the United States. The paper represents the state of the development of the model as at July 2006. The paper has been accepted for publication in the Transportation Research Record.

A one page abstract for a paper that will have nested and logit models developed using four personal travel surveys conducted by the Virginia Tech Air Transportation Systems Laboratory is included in the document. This paper has been accepted for presentation at the 29th International Air Transportation Conference of the American Society of Civil Engineers.

The next chapter presents a model to estimate demand for General Aviation Operations using a gravity based approach. The model estimates the annual number of General Aviation operations from more than three thousand public use airports in the United States. The gravity modeling approach was used for the General Aviation models because there is not enough data in the American Travel Survey to develop a robust model for General Aviation mode choice that could be used to estimate demand for that mode of transportation. The combination of the General Aviation demand estimates from this model and the model in chapter three gives us the capability to present a more complete picture of total intercity travel demand in the United States. This paper has been published in the 2006 Transportation Research Record (No. 1951). The appendices contain Matlab code for the nested logit, mixed logit and the General Aviation airport choice model.

# 2 Literature Review

## 2.1 Historical Development of Intercity Mode Choice Models

Active development of intercity travel demand models for transportation applications can be traced back to the North-East Corridor Transportation Project in the early 60's. The NE Corridor Project models were developed to evaluate current and future transportation requirements for the Boston-New York-Washington (DC) corridor. The models used demographic and economic characteristics of city pairs and level of service variables of travel modes to estimate travel demand. These early set of models include those by Kraft-SARC, Quandt and Baumol, and Mayberry as shown in Equations 1 to 3. The Quandt-Baumol's model is an enhancement to the Kraft-SARC to estimate demand for non-existent modes by comparing them with the fastest mode. While Mayberry's model is a further modification of Quandt's model to better represent inter-modal competition.

$$V_{kij} = \alpha_0 \left(P_i P_j\right)^{\alpha_1} \left(Y_i Y_j\right)^{\alpha_2} \left(E_i E_j\right)^{\alpha_3} A_J^{\alpha_4} \prod_m T_{mij}^{\beta_{1m}} C_{mij}^{\beta_{2m}} F_{mij}^{\beta_{3m}} \qquad (1)$$

$$V_{kij} = \alpha_0 P_i^{\alpha_1} P_j^{\alpha_2} Y_i^{\alpha_3} Y_j^{\alpha_4} \left(T_{ij}^b\right)^{\beta_1} \left(T_{kij}^r\right)^{\beta_2} \left(C_{ij}^b\right)^{\gamma_1} \left(C_{kij}^r\right)^{\gamma_2} \left(F_{ij}^b\right)^{\delta_1} \left(F_{kij}^r\right)^{\delta_2} \qquad (2)$$

$$V_{kij} = \alpha_0 \left(P_i P_j\right)^{\alpha_1} \left(Y_i Y_j\right)^{\alpha_2} \left[\sum T_{mij}^{\beta_1} c_{mij}^{\beta_2} F_{mij}\right]^{\gamma_1} \frac{T_{kij}^{\beta_1} c_{kij}^{\beta_2} F_{kij}}{\sum_m T_{mij}^{\beta_1} c_{mij}^{\beta_2} F_{mij}} \qquad (3)$$

where $V, P, Y, E, and, A$ represent travel volume, population, per capita income, city employment and measure of attractiveness between city pairs,
$b\, and\, r$ represent the best model and relative mode,
$k$ is the mode index,
$m$ is the index over all modes and
$i\, and\, j$ are the origin and destination city indices.

Separate models are estimated for each mode. Koppelman et al. (1984) in their review of travel demand models identified certain limitations of these early set of models, such as:

1. Biased model parameter estimates due to the use of aggregate data in estimation
2. Non-transferability and stability of model parameter values due to models being calibrated for specific city pairs (this creates issues if model needs to be used in different geographical region or context)
3. Difficulty in delineating city and region boundaries in corridor level models

4. The use of terminal to terminal level of service variables that do not adequately represent actual travel time and costs.

Aggregate bias is the major weakness of these early aggregate demand models. The aggregation bias severely limits transferability and hence usefulness of the model. Also, several of the early models structures are identified as being correlative rather than causal. Data availability and validation of model outputs is another key limitation.

Despite the above mentioned limitations of the aggregate demand models of the sixties, they played a crucial role in helping to identify key level of service variables (such as travel time, travel cost, service frequency) and relevant socioeconomic characteristics (income) that influence intercity travel demand decisions. In addition it became clear from these early studies that models need to be segregated by trip purpose: at minimum, for business and non-business trips. Income was also identified as a significant market segmentation feature. **A key strength of the mode choice model presented is the development of a framework for the continental U.S. in order to avoid issues related to geographically transferability of the model.**

The early 70's saw the introduction of travel demand forecasting techniques with mode choice models that had logit-type formulations. Examples of these include the binary share model between automobile and airline by Ellis et al. (1971) and Walmsley's (1979) model to predict mode share between airline and rail. Multinomial models were developed by Bennet et al. (1974) and Peat Marwick and Co. (1973) that estimate mode share for auto, air, bus and rail. Despite initial difficulties in extracting credible parameters all four authors concluded that disaggregate mode choice modeling techniques were a more theoretically sound and credible approach than aggregate techniques.

During this period disaggregate individual choice models were actively applied in urban travel demand modeling. Notable among these are the works of Domencich and McFadden (1975), Richards and Ben-Akiva (1975) and Daganzo (1979). The key conclusion of Koppelman's review is that there is a need to move towards more disaggregate individual choice models which is the same conclusion arrived at in an earlier review by Rice et al (1981) and Miller (1992). All of the above clearly indicate that disaggregate individual choice models are the model of choice for both by practitioners and researchers.

## 2.2 REVIEW OF DISAGGREGATE NATIONWIDE TRAVEL DEMAND MODELS

Four major attempts to develop national level intercity mode choice models in the U.S. using disaggregate techniques are reviewed in this section. The model developers in chronological order are Stopher and Prashker (1976), Grayson (1982), Morrison and Winston (1985) and Koppelman (1990). All four models were developed with various versions of the National Personal Travel Surveys conducted by the Bureau of Transportation and Statistics in 1969, 1977, 1983, 1990 and 1995. Stopher and Prashker used the 1972 National Travel Survey (NTS) database, while others used the 1997 NTS. The 1995 American Travel Survey was conducted to obtain information about long distance travel in the U.S. The mode choice model presented in this dissertation is the first known attempt to develop a national level intercity model using the 1995 American Travel Survey.

### 2.2.1 Stopher's Mode Choice Model

Stopher and Prashker (1976) document the earliest attempt to develop a nationwide intercity mode choice model for the U.S. using disaggregate modeling techniques. They developed a *multinomial logit model* for automobile, air, bus and rail modes using 2,085 observations from the 1972 NTS database. Separate models were calibrated for business and non-business trips. Due to the small sample size, the model was not segmented by income group. The analysis is only partially disaggregate since a typical record was created for each corridor and replicated in the analysis based on the number of records for each mode in the sample for that corridor.

The variables used in the model are Relative distance, Relative time and Relative cost and Relative access-egress time and frequency (departure frequency). The NTS database contains information on travelers' choices. However, estimates of travel times and costs were obtained from published fare and schedule guides and road maps. Relative values were defined as the value for a specific mode divided by the average for all modes. As would be expected travel time and costs vary across modes. To capture instantaneous Departure Frequency of automobile it was defined as 150 per day, this value was later questioned by the authors as some Commercial Air trips had frequencies as high as 85 per day.

Several model forms were tested and the final model had alternative specific constants for Air, Rail and Bus. Equation 4 shows the final model form and Table 1 has the coefficient estimates for business and non-business models. All the model coefficients had the expected apriori signs.

The coefficients in the Business model are significant. However, in the non-business model, the hypothesis that the coefficients of relative distance and relative access-time were not different from zero could not be rejected. The rail alternative specific constant was also found to be insignificant in the non-business model.

$$
\begin{aligned}
U_{ij}^{k} &= \alpha_0 + \alpha_1 \, Rel.Distance_{ij}^{k} + \alpha_2 \, Rel.Time_{ij}^{k} + \alpha_3 \, Rel.Cost_{ij}^{k} \ldots \\
&\quad + \alpha_4 \, Rel.AccessEgressTime_{ij}^{k} + \alpha_5 Frequency_{ij}^{k}
\end{aligned} \tag{4}
$$

**Table 1. Parameter Estimates from Stopher and Prashker Model**.

| **Trip Purpose** | Alternative Specific Constants | Relative Distance | Relative Time | Relative Cost | Relative Access-Egress Time | Relative Frequency |
|---|---|---|---|---|---|---|
| **Business** | Bus: -1.646<br>Rail: -0.393<br>Air: 3.128 | -10.64 | -0.632 | -3.957 | -0.517 | 0.00987 |
| **non-business** | Bus: -1.410<br>Rail: -0.365<br>Air: 2.476 | -0.521 | -1.609 | -4.252 | -0.196 | 0.0120 |

Several questions could be raised about the model coefficients and estimates. First, it is intriguing that holding all other attributes constant and using only the alternative specific constants air appears to be the dominant mode. The published sensitivity analysis from the model shows that base model estimates are very different from the NTS database observations. For example, from Los Angeles to San Francisco the NTS reports 28.6% of the trips are made by automobile while the model estimates 13.9%. It is understandable that the model would be weak in replicating observations for bus and rail, however, the poor estimates for automobile raises questions about the credibility of the model. Elasticity estimates from the model are also counterintuitive.

As will be shown in the section on ‘Review of Logit Models’, the multinomial logit models have very restrictive substitution patterns that renders it inappropriate for forecasting and conducting sensitivity analysis. The substitution pattern is such that if any single mode is improved it draws equally from all the other modes in order to keep the ratio of probabilities between other modes constant.

The authors attribute the model weaknesses to several tenuous assumptions and the poor quality of data used in developing the model. Koppelman et al. (1984) note the high level of geographic aggregation, poor information on the choice set, and lack of service variables as additional limitations. Koppelman et al. (1984) further argue that the poor model performance is due to poor model specification and the exclusion of demographic and economic variables rather than data quality. However it is more a combination of both factors than one or the other. The data issues are not trivial since in developing a similar model using the NTS database Koppelman (1990) faced estimation constraints he attributed to the limitation of the database.

### 2.2.2 Grayson's Mode Choice Model

The next major attempt to develop national intercity mode choice model was by Alan Grayson (1982). A multinomial logit model for automobile, air, rail and bus was developed using the 1977 NTS database. The model was calibrated for trips starting and ending in Metropolitan Statistical Areas. The variables in the model are cost, travel time, departure frequency, and access time. Grayson followed Stopher in supplementing the database with estimates of travel times and cost from published industry and fare guides, however he replaced access distance with access time as it was available in the 1977 NTS database.

The travel time is scaled by multiplying by household income, and frequency enters the model as a reciprocal and is also scaled by income. The form of Grayson's model is show in Equation 5. Three sets of models calibrated are shown in Table 2.

$$\begin{aligned} U_{ij}^{k} &= \alpha^{k} + \alpha^{1} TravelCost_{ij}^{k} + \alpha^{2} Income^{l} xTravelTime_{ij}^{k} \ldots \\ &\quad + \alpha^{3} \frac{Income^{l}}{2xFrequency_{ij}^{k}} + \alpha^{4} Income^{l} xAccessDis\tan ce_{ij}^{k} \end{aligned} \tag{5}$$

**Table 2. Models Developed by Grayson.**

| **Model** | **Sample Structure** | **Alternative Specific Coefficients** | **Travel Cost** | **Income x Travel Time** | **Wait Time = Income/ (2xFrequency)** | **Access Time** | $\rho^2$ |
|---|---|---|---|---|---|---|---|
| ***Model I*** | **Model A.** 1685 observations along 46 most heavily sampled routes in NTS | Air: -2.700<br>Bus: -2.552<br>Rail: -3.027 | -0.0161 | -0.0240 | -0.0055 | -0.0007 | 0.303 |
| | **Model B.** 1062 observations on routes with greatest number of passenger miles | Air: -0.582<br>Bus: -2.728<br>Rail: -2.185 | -0.0036 | -0.0050 | -0.0050 | -0.0005 | 0.321 |
| ***Model II*** | **550 Business trips** from Model I (A) | Air: -1.313<br>Bus: -3.159<br>Rail: -2.461 | -0.0328 | -0.0200 | -0.0244 | -0.0006 | 0.299 |
| | **717 Social trips** from Model I (A) | Air: -3.166<br>Bus: -3.016<br>Rail: -2.828 | -0.0111 | -0.0238 | -0.0090 | -0.0016 | 0.306 |
| | **191 Entertainment trips** from Model I (A) | Air: -3.435<br>Bus: -1.797<br>Rail: -3.288 | -0.0111 | -0.0079 | -0.0030 | -0.0015 | 0.301 |
| *Model* | **233 Light Drivers** from Model I (A) | Air: -1.995<br>Bus: -1.137<br>Rail: -1.253 | -0.0129 | -0.0167 | -0.0246 | -0.0003 | 0.141 |

Two sub-models A and B are developed in Model I. Model A samples the the most heavily sampled routes in the NTS, while Model B has routes with the greatest number of passenger miles. In a sense the routes on Model A will be relatively short compared to those of Model B. The coefficients in all the models have the expected apriori signs. Wait time variable is significant in Model A but not Model B at the 5% level. This is attributed to trips not originating at home and wide variations in access time. Grayson notes that in Model B longer trips dominate, and the transportation literature has shown that for longer trips travelers are willing to travel a bit further to access an airport that has more frequencies and lower fares hence they may not be sensitive to access time. As would be expected for a multinomial logit, sensitivity and elasticity analysis from the model yields unintuitive estimates. This is a defining characteristic of multinomial logit models as mentioned earlier.

Model II stratifies the base model by trip purpose. Though Grayson attempts to justify the high cost coefficient for business, it is still questionable. Apart from the alternative specific constants there is not much difference between coefficient estimates for the stratified models.

The model is a considerable improvement on Stopher's. Their application for policy analysis is limited because of the way the model was structured. The criteria of selecting city pairs along heavily trafficked corridors or city pairs with high passenger-miles is very subjective. When analyzing another city pair in the U.S. it is not clear which criteria to use to assign it to any of the sub-groups in the model, or if it belongs to any of those sub-groups at all. Secondly, the calibration of a business-only trip model for group A limits its application.

### 2.2.3 Morrison and Winston's Mode Choice Model

Morrison and Winston (1985) were the first to move from the Multinomial to a Nested logit Model using the 1997 NTS database. They used the log-sum variable to hierarchically nest three models. The models are decision to rent a car, destination choice and mode choice. The model was calibrated for automobile, bus, rail and air. Traveler choice information from the NTS database was supplemented with trip information from the Official Airline, Railroad, and Bus Guides. It is interesting to note that for the 1997 NTS, long distance-travel is defined as any trip greater than 200 miles. Separate models were developed for business and vacation travel trips. The Business model was calibrated using 2,325 trip records and the vacation with 1893 records.

The analysis as in previous studies was restricted to trips starting and ending in MSAs which greatly limits the spatial transferability and application of the model.

The variables used in the business model are cost, time, party size for automobile travelers and average time between departures. The non-business model is stratified by income for travel time and has an additional log-sum variable to link it to the rental car model. Statistical analysis showed the log-sum variable to be insignificant for non-business travel leading the authors to conclude that the expected maximum utility derived from the rental-car decision at the destination does not have an important influence on the travelers choice of mode. The rental car and destination choice models were dropped from the analysis of business travelers due to difficulty in obtaining credible parameter estimates.

Model parameter estimates have the expected apriori signs. An interesting feature of Morrison's business model was that it obtained credible coefficients without interacting any financial constraint (such as income) with any of the other variables. Elasticity and value of time between departure estimates for business travelers in absolute and percentage terms indicate that business travelers have a higher value of time than non-business travelers.

### 2.2.4 Koppelman's Mode Choice Model

Koppelman (1990) extends Morrison's hierarchical approach to nest a *trip frequency, trip destination, mode choice* and *fare class* choice models. The 1997 NTS database was used for this model and level of service variables were obtained from published official guides. The variables used in the mode choice model were travel time, travel cost, departure frequency, distance between city pair and household income. Due to co-linearity issues the ratio between travel time and cost was constrained using judgmentally selected values of time for high and low income travelers.

The models are linked hierarchically using log-sum values. The log-sum value for the fare class model (estimates probability of choosing a discount, coach or first class fares) enters the mode choice model as a variable. Variables in the fare class model are cost, number of daily departures, household income and trip purpose. Even though integrating the fare class model into the mode choice model provided flexibility, it introduces aggregation bias since fares vary significantly between any two city pairs. The likelihood ratio of 0.33 for the fare class model is quite low; however given the small sample size of 235 fares it is acceptable. Credibility issues arise with regard to the small sample size and limited spatial extent of city pairs used if the model has to be transferred.

The estimated model parameters had the expected apriori signs. The fare class composite utility variable was only significant in the non-business model. Though the author attributes this to sample size, this is not surprising since it is well documented in the transportation literature that business travelers are more sensitive to time than cost hence fares do not enter significantly into their decision making process. The overall statistical fit of the model measured by likelihood ratio index is good for both models with the business model having a better fit to the data than the non-business. The fare class model parameter estimates are shown in Table 3 while those for the mode choice model are shown in Table 4.

**Table 3. Parameter Estimates from Koppelman's Fare Class Model (1990).**

| Variable Names | | Parameter Estimate (t-statistic) |
|---|---|---|
| Alternative Specific Constants | Discount Class<br>Fare Class | -0.311 (0.6)<br>-0.889 (1.2) |
| Fare Cost ($) | | -0.010 (2.7) |
| Daily Departures | | 0.055 (4.1) |
| Income | Discount Class<br>Fare Class | -0.263 (1.3)<br>0.350 (2.1) |
| Trip Purpose | Discount Class<br>Fare Class | -1.605 (3.7)<br>-0.160 (0.3) |
| **Statistical Information** | | |
| Likelihood Ratio Index | $\rho^2$ | 0.333 |
| Number of cases | | 235 |

**Table 4. Parameter Estimates from Koppelman's Mode Choice Model (1990).**

| **Variable Names** | | **Business Trip Parameter Estimate (t-statistic)** | **Non-business Trip Parameter Estimate (t-statistic)** |
|---|---|---|---|
| Alternative Specific Constants | Car<br>Bus<br>Rail | -0.883 (1.5)<br>-1.703 (2.2)<br>-2.227 (2.8) | 1.687 (4.0)<br>0.386 (0.6)<br>0.136 (0.2) |
| Cost ($) | | -0.00460 (3.0) | -0.00256 (3.8) |
| Travel Time (Minutes) | High Income<br>Low Income | -0.00460 (3.0)<br>-0.00153 (3.0) | -0.00193 (3.8)<br>-0.00064 (3.8) |
| Bus/Rail Frequency | | | 0.03990 (1.9) |
| Composite Air Class Utility | | 0.324 (1.5) | 0.4560 (4.0) |
| Income | Car<br>Bus/Rail | -0.0865 (0.4)<br>0.3540 (1.3) | 0.0460 (0.3)<br>-0.4910 (2.4) |
| Distance <250 miles | Car<br>Bus/Rail | 2.2360 (4.3)<br>1.9940 (2.9) | 1.7030 (3.8)<br>0.8570 (1.5) |
| Distance >500 miles | Car<br>Bus/Rail | | 1.7960 (3.5)<br>-0.8160 (1.3) |
| **Statistical Information** | | | |
| Likelihood Ratio Index $\rho^2$ | | 0.623 | 0.465 |
| Number of cases | | 251 | 356 |

### 2.2.5 Summary of Review of Disaggregate National-level Model Choice Models

The four models reviewed used some version of the Census Bureau National Travel Surveys to develop disaggregate national level intercity mode choice and travel demand models. The outcome of the modeling attempts show that disaggregate modeling procedures can be effectively applied to estimate mode choice models with certain limitations.

A key limitation of the National Travel Surveys is the absence of level of service variables such as travel time and cost. This means modelers have to supplement the survey database with external data sources such as the Official Airline and Railroad Guides. A high level of aggregation is introduced in the process because in linking the external database to the NTS sample average travel times and cost have to be estimated for each record, thereby introducing aggregation bias. For example, ticket prices vary widely both geographically and based on the time of purchase and departure times. In order to estimate a representative fare between any city pair one has to average over all these dimensions. The high level of aggregation bias introduced undermines one of the key reasons for the shift from aggregate to disaggregate modeling techniques.

The second issue is that the public version of the NTS does not contain detailed information on the exact location of travelers. Hence modelers have to make tenuous assumptions about where the trips start and end. The lack of geographical information makes it difficult to credibly estimate access and egress costs and times. This is a key reason for all the reviewed models restricting the area of analysis to only MSAs. The restriction helps provide more credible estimates but, renders the model almost useless for application to non-MSA areas. The absence of geographical information also makes it difficult to identify both the airport used by the traveler and the alternative airports that were considered. Due to this issue none of the models incorporates airport choice and they only model competition between the ground modes and air. The absence of airport choice is a crucial deficiency since demand through airports is a key component in analyzing the National Airspace System.

There are very few nationwide intercity travel demand models in the U.S. The only other model with a similar scope developed recently is Lewe et al (2003). It is an agent-based approach to model traveler’s response to SATS. Travelers in Lewe’s model are agents that seek to complete trips comfortably and safely with less travel time however; the mode choice portion has an embedded multinomial logit model.  Thus Lewe's model breaks down to a multinomial logit model with travelers maximizing their utility based on travel time and travel cost.  Giving the documented weakness and limitations of the multinomial logit model such a model will be severely constrained for forecasting purposes.

Koppelman and Hirsh (1986) highlight the data requirements that would be needed for researchers and practitioners to develop accurate and useful intercity travel demand models. However to date there does not seem to be any attempt by either the Census Bureau, the Bureau of Transportation statistics or any of the key Federal Agencies to collect such data.

A key contribution of the mode choice model presented in this dissertation is to extend the work of intercity travel demand modeling in three dimensions.  The spatial extent of the model is extended to include non-MSAs so the model can be applied nationally.  Secondly, a heuristic is introduced to implement airport choice modeling so that the model can estimate market share and by extension demand through airports, making the applied model more useful to policy makers.  Thirdly, the data used in calibrating and applying the model is aggregated from the county level giving the model better precision.  This model is unique because it is the first national level, intercity, multi-mode choice model to model both mode choice and airport choice at the county-to-county level in the U.S.

## 2.3 REVIEW OF LOGIT MODELS

This section examines the underlying structure of various logit models described in the transportation literature. Three major classes of models covered are the Multinomial logit, the Generalized Extreme Value (GEV) and Mixed logit Models. The underlying structure and derivations of the models are presented, and their strengths and weaknesses for different behavioral regimes examined. Issues related to estimation of the models are discussed.

### 2.3.1 Multinomial Logit Model

McFadden (2001) in his Nobel Laureate Lecture traces the theoretical development of the logit models back to the psychophysical works of Thurstone (1927), Luce's (1959) famous axiom of Independence of Irrelevant Alternatives, and Marschak's (1960) work on maximization of utilities. McFadden (1973) derived the conditional logit model (later known as the multinomial logit) using the utility maximizing theory axioms proposed by Luce (1959) and Arrow (1951). The model was saddled with the well documented 'independence of irrelevant alternatives' property usually referred to as IIA due to Luce's axiom which states that "*If one is comparing the two alternatives according to some algebraic criterion, say preference, this comparison should be unaffected by the addition of new alternatives or the subtraction of old ones" – Luce 1959.* The logit model was already in use in other fields including economics and biostatics prior to McFadden's derivation, but his key contribution was linking the statistical model to the axioms of individual choice behavior. This link and his application of his model to transportation problems were critical in establishing the logit model as the model of choice for choice related problems in transportation.

The multinomial logit model developed by McFadden (1973) had several limitations due to the restrictive assumptions used in deriving the model. Basically, econometric models can be classified in terms of three criteria: 1) the underlying distribution of the random variable characterizing the population; 2) the level of correlation or independence and 3) whether or not the random sample is being drawn from an identical distribution. The last two properties are tied to the first and second moments of the distribution. It is noted here that though in practice most empirical models are usually classified by their first and second moments there are higher order moments (skewness, kurtosis, etc) that are usually ignored not because they do not factor into the characterization of the models but, more due to that fact that techniques have not been fully developed that would make them easily integrated into the estimation process Spanos (1986, 2000).

In terms of the distribution the multinomial logit model assumes an underlying Gumbel distribution and a random sample that is Independent and Identically Distributed. The last two restrictions are usually referred to as IID (I for independence and ID for identical distribution) in the econometric literature; hence this model is Gumbel IID. This implies the alternatives being considered are independent of each other and have the same variance. This is actually the most restrictive form of any econometric model.

This can be illustrated by considering a model to characterize $n$ alternatives. Assume the joint distribution of the random variables ( $X_1, X_2, X_3, \ldots\ldots X_n$ ) to be normal. Then the function $f(x_1, x_2, x_3 \ldots\ldots, x_n; \phi)$ takes the form shown in Equation 6.

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{pmatrix} \sim N\left(\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_n \end{bmatrix}\begin{bmatrix} \sigma_{11}\,\sigma_{12}\;\sigma_{13}\ldots\sigma_{1n} \\ \sigma_{21}\,\sigma_{22}\;\sigma_{23}\ldots\sigma_{2n} \\ \sigma_{31}\,\sigma_{32}\;\sigma_{33}\ldots\sigma_{3n} \\ \vdots \\ \sigma_{n1}\,\sigma_{n2}\;\sigma_{n3}\ldots\sigma_{nn} \end{bmatrix}\right) \tag{6}$$

Clearly there are $n$ $\alpha' s$ and considering symmetry $\frac{1}{2}(n(n+1))$ $\sigma' s$ to be estimated. Clearly this model is analytically intractable due to the large number of unknown parameters that increase with number of alternatives. If the assumption of independence is introduced the off-diagonal elements all reduce to zero as shown in Equation 7. This is a considerable improvement but the modeler still has to deal with an increasing number of parameters as $n$ grows.

$$\sigma_{ij} = \begin{cases} \sigma_{ii}, \; for i = j \\ 0, \; for i \neq j \end{cases}$$

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{pmatrix} \sim N\left(\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_n \end{bmatrix}\begin{bmatrix} \sigma_{11}\,0\,0\ldots0 \\ 0\,\sigma_{22}\;0\ldots0 \\ 0\,0\,\sigma_{33}\ldots0 \\ \vdots \\ 0\,0\,0\ldots\sigma_{nn} \end{bmatrix}\right) \tag{7}$$

The further imposition of the assumption of identical distributions basically makes the model tractable as shown in Equation 8.

$$\sigma_{ij} = \sigma \ \forall \ i, j$$
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{pmatrix} \sim N\left(\begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ \vdots \\ \alpha \end{bmatrix}\begin{bmatrix} \sigma\, 0\, 0 \ldots 0 \\ 0\, \sigma\, 0 \ldots 0 \\ 0\, 0\, \sigma \ldots 0 \\ \vdots \\ 0\, 0\, 0 \ldots \sigma \end{bmatrix}\right) \tag{8}$$

The initial model is now reduced to $X_k \sim N(\alpha, \sigma^2)$. The imposition of the IID assumption leads to an easily tractable model. In the case of the logit model the Gumbel IID model leads to a probability of the form shown in Equation 9. However, the advantage of having analytically tractable form is offset by the famous IIA property that can produce unrealistic substitution patterns rendering the model estimates unreliable in many practical applications.

The form of the probability for the multinomial logit model is derived in the section on GEV models. The probability of any alternative $i$ has the form in Equation 9.

$$P(i) = \frac{e^{V_i}}{\sum_{j=1}^{J} e^{V_j}} \tag{9}$$

Then it is clear that for any two alternatives $k$ and $l$ the ratio of their probabilities $\frac{P(k)}{P(l)} = \frac{e^{V_k}}{e^{V_l}}$ is independent of any other alternatives in the model. This property follows Luce's axiom and is what is referred to as the IIA property. Theoretically, it can be argued that the way a decision maker perceives the utility of any two alternative modes should not be affected by the presence of other modes. The constant nature of this ratio however yields unintuitive estimates in practice. Consider the now famous red bus, blue bus problem (Ben-Akiva and Lerman, 1985). Say you have a logit model that initially has two alternatives, automobile and a blue bus and, the model calibrated model estimates a probability of 2/5 for automobile and 3/5 for the bus alternative. Then the ratio between the automobile and blue bus probabilities is 2/3. If the transit authority is to paint half of the blue buses red, the question is what would the new market shares be? The real result in practice will be for the net market share of the bus to remain constant at 3/5 with 0.3 each going to the red and blue buses. This will then make the ratio of automobile and blue bus probabilities (0.4/0.3) change to 1.33. Since the structure of the multinomial logit model keeps the ratio constant the model will predict very different estimates. Basically, it will yield automobile share of 0.25 and red and blue bus shares 3/8. Thereby the ratio of automobile to red bus is (2/8)/(3/8)=2/3. Clearly, the model estimates are unrealistic.

The problem is further compounded when one examines the cross-elasticities of logit probabilities. Consider the impact of the change in an attribute of an alternative $j$ on the probability $P_{ni}$ of all other alternatives in the model. Following Train (2003), the change in $P_{ni}$ with respect to a change in the attribute of $j$ is shown in Equation 10.

$$E_{iZ_{nj}} = -\beta_z Z_{nj} P_{nj} \tag{10}$$

Where $Z_{nj}$ is the attribute of alternative $j$ faced by individual $n$ where $\beta_z$ is its coefficient. Since the cross-elasticity is the same for all $i$, the implication is that an improvement in any one alternative reduces the probabilities of all the other alternatives by the same amount (that is $E_{iZ_{nj}}$ is fixed for all $i's$). This means if a model has automobile, bus, rail and air alternatives, and a policy is implemented that reduces say rail cost, then the multinomial logit model will draw the same percentage from bus, automobile and air alternatives. Obviously such a result is unrealistic. It is not surprising that model elasticity estimates from Grayson (1982) and Stopher and Prashker (1976) did not yield intuitive estimates. The property to an extent also affects nested logit models. The problem is not with the percentage increase in the improved mode but rather with the equal drop across all modes. The restrictive cross-substitution pattern renders the multinomial logit unsuitable for policy studies that seek to investigate the impact of improving alternatives or introducing new alternatives.

McFadden (1976) used the multinomial logit model to study criteria employed by the California State Highway department in selecting urban freeway routes and later on Domencich and McFadden (1975) developed a behavioral shopping model of '*choice of mode for trips, choice of destination for trips made at an observed time by preferred mode, and choice of whether or not to make a trip, given preferred time, mode and destination*'. One of the major studies that placed the multinomial logit model at the forefront of mode choice and travel demand modeling was the use of the model to study the introduction of a new rail system in the San Francisco Bay Area called the Bay Area Rapid Transit (BART). McFadden obtained a National Science Foundation grant to develop a logit model to predict BART ridership. During the study, data was collected before the construction of BART and used to estimate logit models. After BART was built, respondents were re-sampled and their behavior compared with predicted estimates from the logit models. The logit models were found to perform better than other models used by BART consultants in predicting the actual behavior of commuters. The results of the study are fully documented in the final report of McFadden et al (1978).

The beauty of the multinomial logit model lies in its analytically tractable closed form. Despite these early successes, in using the model for empirical studies it came under heavy criticism due to the IIA property and the restrictive substitution patterns than can led to unreasonable estimates during empirical studies, especially when used for forecasting. In order to develop more flexible empirical models either the independence or identical distribution assumptions need to be relaxed while maintaining the closed form model. The first attempt was the nested logit model that relaxed the independence assumption by grouping alternatives that are similar into nests (McFadden (1978), Daly and Zachary, (1978)). Since then a plethora of models were developed in this direction, however, McFadden derived a model called the General Extreme Value (GEV) model that is an overarching framework over all these models including the logit model.

Next, we present the GEV model framework and show how the multinomial and the Nested logit can be derived from the framework. After that we turn to a fairly new class of logit models called Mixed logit, that relax both the independence and identical distribution assumptions. These models cannot be solved analytically and require simulation techniques.

### 2.3.2 Generalized Extreme Value Models

Following Train (2003), McFadden (1978), and Ben-Akiva and Lerman (1985) we derive the form for Generalized Extreme Value (GEV) models from which we can derive both the multinomial logit and Nested logit (the most widely used forms) models, showing how the IIA problem arises in the logit and how it is mitigated by the Nested logit model.

Define $Y_j \equiv e^{V_j}$ and a function $G(Y_1, Y_2, \ldots, Y_J)$ for $(Y_1, Y_2, \ldots, Y_J \geq 0)$ with the following properties

1. $G$ is non-negative for all values of $y_j$ for all $j$.

2. $G$ is homogeneous of degree $\mu > 0$; that is G($\alpha Y_1, \alpha Y_2, \ldots, \alpha Y_J$) = $\alpha^{\mu} G(Y_1, Y_2, \ldots, Y_J)$.
3. $Lim_{Y_j \to \infty} G(Y_1, Y_2, \ldots, Y_J) = \infty,\ for\ j = 1,2,\ldots, J_n$).
4. The $i^{th}$ partial derivative of $G$ with respect to any combination of $i$ distinct $Y_j{}'s$, is non-negative if $i$ is odd, and non-positive if $i$ is even.

If all the above conditions hold and defining the partial derivative of $G$ as $G_i = \partial G / \partial Y_i$ then $P(i) = \frac{Y_i G_i}{G}$ is the choice probability for a discrete choice model that is consistent with utility maximization.

***Multinomial logit Model***

For the logit model define $G = \sum_{j=1}^{J} Y_j$. The summation is always positive, raising all $Y_j$'s by a factor $\mu$ raises $G$ by the same factor. If any $Y_j$ rises without bound $G$ does also. $G_i = 1$ and all higher derivatives are equal to zero satisfying all the four criteria above. The logit formula follows in Equation 11.

$$P(i) = \frac{Y_i G_i}{G} = \frac{Y_i}{\sum_{j=1}^{J} Y_j} = \frac{e^{V_i}}{\sum_{j=1}^{J} e^{V_j}} \tag{11}$$

Clearly, for any two alternatives $k$ and $l$ the ratio of their probabilities are independent of any other alternatives in the model (IIA axiom) as shown in Equation 12.

$$\frac{P(k)}{P(l)} = \frac{e^{V_k}}{e^{V_l}} \tag{12}$$

### 2.3.3 The Nested Logit Model

First partition $J$ alternatives are into $K$ nests $(B_1, B_2, \ldots B_K)$. Then specify $G = \sum_{l=1}^{K} \left( \sum_{j \in B_l} Y_j^{1/\lambda_l} \right)^{\lambda_l}$, with each $\lambda_k$ between zero and one. The first three properties can be shown to hold with ease, the fourth is shown in Equation 13.

$$G_i = \lambda_k \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l - 1} \frac{1}{\lambda_k} Y_i^{(1/\lambda_l) - 1} = Y_i^{(1/\lambda_l) - 1} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l - 1} \tag{13}$$

and for $i \in B_k$ since $Y_j \geq 0$ for all $j$, we have $G_i \leq 0$, as required. The second cross partial derivative is given in Equation 14.

$$G_{im} = \frac{\partial G_i}{\partial Y_m} = (\lambda_k - 1) Y_i^{(1/\lambda_l)-1} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l - 2} \frac{1}{\lambda_k} Y_m^{(1/\lambda_l)-1} = \ldots$$
$$\frac{\lambda_k - 1}{\lambda_k} (Y_i Y_m)^{(1/\lambda_l)-1} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l - 2} \qquad (14)$$

for $m \in B_k$ and $m \neq i$. With $\lambda_k \leq 1, G_{ij} \leq 0$, as required. For $j$ in a different nest then $i$, $G_{ij} = 0$, which also meets the criterion. Higher cross partials are calculated similarly; they exhibit the desired property if $0 \leq \lambda_k \leq 1$.

The choice probability for Nested logit then becomes (Equation 15),

$$P(i) = \frac{Y_i G_i}{G} = \frac{Y_i Y_i^{(1/\lambda_l)-1} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l}} = \frac{Y_i^{1/\lambda_k} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l}}$$

*substituting* $e^{V_i}$ (15)

$$= \frac{\left(e^{V_i}\right)^{\lambda_k} \left( \sum_{j \in B_k} \left(e^{V_j}\right)^{1/\lambda_k} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} \left(e^{V_j}\right)^{1/\lambda_l} \right)^{\lambda_l}} = \frac{e^{V_i/\lambda_k} \left( \sum_{j \in B_k} e^{V_j/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} e^{V_j/\lambda_l} \right)^{\lambda_l}}$$

Clearly for any two alternatives $i \in B_k$ and $m \in B_l$ in different nests

$$\frac{P(i)}{P(m)} = \frac{e^{V_i/\lambda_k} \left( \sum_{j \in B_k} e^{V_j/\lambda_k} \right)^{\lambda_k - 1}}{e^{V_m/\lambda_l} \left( \sum_{j \in B_l} e^{V_j/\lambda_l} \right)^{\lambda_l - 1}} \qquad (16)$$

IIA does not hold because the ratio of their probabilities are tied to all alternatives in their respective nests. However, since the ratio only applies to alternatives within nests there is a form of IIA that Train (2003) refers to as 'independence from irrelevant nests' (IIN). Note that if the two alternatives are in the same nest (i.e. $k = l$) then it can be derived from Equation 15 that

$\frac{P(i)}{P(m)} = \frac{e^{V_i/\lambda_k}}{e^{V_m/\lambda_l}}$ this ratio is independent of all other alternatives, so for the nested logit IIA only holds within nests, or in other words the independence assumption is only applied within nests. The Nested logit model is part of the GEV family and is the most frequently used because of its ability to overcome the IIA weakness while maintaining an analytically tractable and closed form. Several applications of the use of logit models to study traveler's mode choice behavior abound in the literature.

The initial form of the Nested logit model restricted each alternative to one nest, however, since then several models have been developed that allow flexible forms of dependence. These new cadre of flexible models include the cross-nested logit (Vovsha (1997), Bierlaire (2006, 1998), and Ben-Akiva and Bierlaire (1999)), which is a nested model that allows alternatives to belong to more than one nest. Small (1987) proposed an Ordered Generalized Extreme Value (OGEV) model that captures dependence by constructing the correlation between alternatives based on their proximity in ordering, which is a form of Markov dependence in econometric terms. The OGEV has been combined with the cross-nested by Small (1994) and Bhat (1998) to create to a model in which the alternatives in overlapping nests are correlated by their ordering. Other models that relax the dependence assumption are Chu's (1981, 1989) paired combinatorial logit and Wen and Koppelman's (2001) Generalized Nested logit. McFadden's GEV model basically specifies an Extreme Value joint distribution that allows for any form of correlation; hence all the above models can be derived from the GEV framework.

### 2.3.4 Heteroskedastic Extreme Value Model

More recently another group of models that focus on relaxing the Identical Distribution assumption have surfaced in the transportation literature. These models allow the variance of the population or alternatives to vary. The variance is usually referred to as 'skedastic' in econometrics hence these models are usually referred to as Heteroskedastic Extreme Value (HEV) models if they incorporate Extreme Value Distributions. In terms of relaxing the identical distribution assumption Steckel and Vanhonacker (1998) specified a model that allowed heterogeneity in the population, while Bhat (1995) on the other hand specified one that allowed heterogeneity in the variance of the alternatives. Recker (1995), specified an odd-ball model in which he postulates there is one alternative whose variance is significantly different from the others. An example is modal choice between automobile and two transit modes (say bus and train). The automobile is the odd-ball alternative. Hensher (1997) developed a stated choice Heteroscedastic Extreme Value (HEV) model to study competition between air and rail in the Sydney-Canberra corridor. Logically the next step was to develop a model that would allow for relaxing both the independence and identical distribution simultaneously. These are usually referred to as the Mixed logit Models.

### 2.3.5 Mixed Logit Model

Mixed logit models go a step further in giving the researcher the ability to relax both the independence and identically distribution assumptions. Two specifications of mixed logit model appear in the literature; the **random-coefficients** and the **error-components** specifications. These are actually not two unique models because the underlying statistical model remains the same. The two definitions refer to the behavioral mechanism (theoretical model) that the researcher uses to justify the interpretation of the model. The random-coefficients model is presented first and then it is then shown how the error-components is just a different way of looking of the same statistical model.

#### 2.3.5.1 Random-coefficients Mixed logit

In all the logit models considered so far the utility takes the form $U_{nj} = \alpha x_{nj} + \varepsilon_{nj}$ where $x_{nj}$ is a vector of attributes that relate to the individual $n$ and the alternatives $j$. The error term $\varepsilon_{nj}$ is IID Extreme Value. The coefficient $\alpha$ are considered fixed for each attribute $x_{nj}$. In the random-coefficients mixed logit utility is specified as shown in Equation 17 with the vector of coefficients $\alpha_n$ considered to vary over individuals '$n$' with a density $f(\alpha)$.

$$U_{nj} = \alpha_n \; x_{nj} + \varepsilon_{nj} \tag{17}$$

The decision maker knows the complete value of their utility in the form of the $\alpha_n$ *and* $\varepsilon_{nj}$ and selects the alternative with the highest utility, however the researcher only observes the choice and the $x_{nj}$'s but not $\alpha_n$ and $\varepsilon_{nj}$. The unconditional probability over all possible values of $\alpha_n$ takes the form shown in Equation 18.

$$P_{ni} = \int \left( \frac{e^{\alpha x_{ni}}}{\sum_j e^{\alpha x_{nj}}} \right) f(\alpha) d\alpha \tag{18}$$

The research specifies a distribution for the coefficients $\alpha_n$ and estimates the parameters of the distributions (say mean and variance). Forms of the distribution used in practice are Normal, Log-normal, Triangular and Uniform. The functional form in Equation 18 is a weighted average of the logit formula estimated at different values of $\alpha$ with weights given by the density $f(\alpha)$. The weighted average of several functions in the statistics literature is called a mixed function and the density that provides the weight the mixing distribution, hence the name Mixed logit.

### 2.3.5.2 Error Components Mixed Logit

The error-components form of the mixed logit decomposes the utility apriori into fixed and random components as shown in Equation 19.

$$U_{nj} = \delta' x_{nj} + \beta_n' z_{nj} + \varepsilon_{nj} \tag{19}$$

where $x_{nj}$ and $z_{nj}$ are vectors of observed variables relating to alternative $j$, and $\delta$ is a vector of fixed coefficients, $\beta$ is a vector of random terms with zero mean, and $\varepsilon_{nj}$ is iid extreme value. The variables in $z_{nj}$ are the ones referred to as **error-components** since they are correlated with the iid error $\varepsilon_{nj}$. Together they define the stochastic components of the utility $(\beta_n' z_{nj} + \varepsilon_{nj})$.

Now, consider the distribution of $\alpha_n$ from (18 with mean $\delta'$ and standard deviation $\beta_n'$ clearly the utility becomes $U_{nj} = \delta' x_{nj} + \beta_n' x_{nj} + \varepsilon_{nj}$ such that if $x_{nj}$ is replaced with $z_{nj}$ in the second term the two models are equivalent statistically. The mixed logit family of models has been shown by McFadden and Train (2000) as being capable of approximating the full family of logit models with the appropriate choice of mixing distributions. The flexibility of relaxing the restrictive model assumptions gained by freedom to specify various mixing distributions also means in practice the model can only be estimated using simulation techniques.

The earliest mixed models were by Byod and Mellman (1980), and Cardell and Dunbar (1980), since then several models have been estimated in the literature Train et. al. (1987), Ben-Akiva et. al. (1993), Bhat (1998), Revelt and Train (1998), and Brownstone and Train (1999). The 1996 San Francisco Bay Area Travel Study was used by Bhat and Castelar (2002) to develop a Mixed logit model of choice between driving alone, car-pooling, walking and using the Bay Area Rapid-Transit system. Recently, Hess and Polak (2005) use the same data to develop a Mixed logit model for airport choice of travelers in the San Francisco Bay Area. A defining characteristic of the above models is that they are all spatially localized and suffer from transferability issues mentioned earlier.

Table 5 summarizes the broad classes of logit models used in transportation planning in terms of the assumptions they attempt to relax.

**Table 5. Classification of forms of logit models.**

| **Model Type** | **Restrictive Model** | **GEV Models** | | **Relax Dependence & ID Assumptions** |
|---|---|---|---|---|
| | | **Relax ID assumption** | **Relax Dependence Assumption** | |
| **Distribution** | Type I Extreme Value or Gumbel | Type I Extreme Value or Gumbel | Type I Extreme Value or Gumbel | Type I Extreme Value + Specified Distribution |
| **Independence** | Independent | Independent | Correlation between alternatives | Correlation between alternatives |
| **Identical Distribution** | Identically Distributed | Variance of unobserved factors differ over alternatives | Identically Distributed | Variance of unobserved factors differ over alternatives |
| **Example** | *McFadden's Conditional logit* | *Heteroskedastic Extreme Value (HEV)* | *Nested logit* | *Mixed logit* |

One aim of the research work is to test the hypothesis that the mixed logit model can approximate the nested logit and produce more credible estimates in a large-scale modeling framework such as TSAM. An initial version of the mode choice model has been developed using the nested logit model. The variables used are travel time, travel cost and household income. Additional variables such as whether trips started or ended in MSAs are introduced in the new model.  In addition, data from stated preference personal travel surveys are combined with the ATS survey during calibration to improve the model fit.

Currently, policy makers and planners have only national/regional level statistics to plan policies for a system spanning several geographical areas with different characteristics.  In the case where localized studies are implemented to supplement regional level statistics the outputs are usually not transferable spatially.  Based on the above weakness we identified the need for a multi-mode travel demand model at the county to county level to improve decision making ability of policy makers and planners.  Three key strengths of the mixed logit model is the ability to predict how market share changes with policy (e.g. if a policy increases travel cost for a specific mode then demand for the mode should decrease), ability to overcome the IIA structure of the multinomial logit model, and the ease of integrating new modes of transportation (SATS).

## 2.4 References

Arrow, K. (1951). *Social choice and individual values*, Yale University Press, CT.

Ben-Akiva, M., and Bierlaire, M. (1999). Discrete choice methods and their applications in short term travel decisions, *in R. Hall. Ed, The Handbook of Transportation Science*, Kluwer, Dordrecht, The Netherlands, pp 5-33.

Ben-Akiva, M., and Lerman, S. (1985). *Discrete Choice Analysis – Theory and Application to Travel Demand*, MIT Press, Cambridge, MA.

Ben-Akiva, M., Bolduc, D., and Bradley, M. (1993). Estimation of travel model choice models with randomly distributed values of time, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1413*, pp 88-97.

Bhat, C. (1998). Accommodating variations in responsiveness to level-of-service measures in travel mode choice modeling, *Transportation Research Part A*, Vol.32, No. 7, pp 495-507.

Bhat, C. (1995). A heteroskedastic extreme value model of intercity model choice, *Transportation Research Part B*, Vol. 29, pp 417-483.

Bhat, C. and Castelar, S. (2002). A unified mixed logit framework for modeling revealed and stated preferences: Formulation and application to congestion pricing analysis in the San Francisco Bay area, *Transportation Research Part B*, Vol. 36, pp 577-669.

Bierlaire, M. (2006). A theoretical analysis of the cross-nested logit model, Annals of Operations Research, Vol. 144, No 1, pp 287-300.

Bierlaire, M. (1998). Discrete choice models, In *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management* (Edited by Labbe, M., Laporte, G., Tanczos, K., Toint, P.), Springer-Verlag, Heidlberg, Germany, pp 203-227.

Brownstone, D. and Train, K. (1999). Forecasting new product penetration with flexible substitution patterns, *Journal of Econometrics*, Vol. 89, pp 109-129.
Bureau of Transportation and Statistics (2000). Airline Origin and Destination Survey (DB1B), http://www.transtats.bts.gov/.

Bureau of Transportation and Statistics (1995). *American Travel Survey: An overview of the survey design and methodology*, Bureau of Transportation Statistics.

Byod, J.H. and Mellman, R.E. (1980). The effect of fuel economy standards on the U.S. automotive market: An hedonic demand analysis, *Transportation Research Part A,* Vol. 14A, pp 367-378.

Cardell, N.S. and Dunbar, F.C. (1980). Measuring the societal impacts of automobile downsizing, *Transportation Research Part A*, Vol. 14A, pp 423-434.

Chu, C. (1989). A paired combinatorial logit model for travel demand analysis, *Proceedings of the 5th World Conference on Transportation Research,* Vol. 4, pp 295-309.

Chu, C. (1981). Structural issues and sources of bias in residential location and travel choice models, Ph.D. Thesis Northwestern University.

Daly, A. and Zachary, S. (1978). Improved multiple choice models, In *Determinants of Travel Choice* (Edited by Hensher, D., Dalvi, M.), Saxon House, Sussex.

Grayson, A. (1982). Disaggregate model of mode choice in intercity travel, In *Transportation Research Record: Journal of the Transportation Research Board, No. 385,* pp 36-42.

Hensher, D. (1997). A practical approach to identifying the market for high speed rail: A case study in the Sydney-Canberra corridor, *Transportation Research Part A*, Vol. 31, No. 6, pp 431-446.

Hess, S. and Polak, J.W. (2005). Mixed logit modeling of airport choice in multi-airport regions, *Journal of Air Transport Management*, Vol. 11, pp 59-68.

Koppelman, F. S. (1990). Multidimensional model system for intercity travel choice behavior, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1241,* pp 1-8.

Koppelman, F. S. (1986). and Hirsh, M. Intercity passenger decision making: conceptual structure and data implications, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1085*, pp 70-75.

Koppelman, F. S., Kuah, G. and Hirsh, M. (1984). Review of Intercity Passenger Travel Demand Modeling: Mid 60's to the Mid 80's, *The Transportation Center,* Department of Civil Engineering, Northwestern University.

Lewe, J., Upton, E., Marvis, D., and Schrage, D. (2003). An Agent-Based Framework for Evaluating Future Transportation Architectures, AIAA 3rd Annual Aviation Technology, Integration and Operations (ATIO) Forum, Nov. 17-19.Luce, D. R. (1959). *Individual Choice Behavior*, New York: Wiley.

Morrison, S. and Winston, C. (1985). An Econometric Analysis of the Demand for Intercity Passenger Transportation, *Research in Transportation Economics*.

Marschak, J. (1960). Binary choice constraints on random utility indications, *in K. Arrow ed, Stanford Symposium on Mathematical Methods in the Social Sciences*, Stanford University Press, Stanford, CA, pp 312-329.

McFadden, D. (2001). Economic Choices, *American Economic Review*, Vol. 91, pp 351-387.

McFadden, D. (1978). Modeling the choice of spatial location, In *Spatial Interaction Theory and Planning Models* (Edited by Karlqvist, A., Lundqvist, L., Snickars, F., and Weibull, J.), North-Holland, Amsterdam, pp 75-96.

McFadden, D. (1973). Conditional logit Analysis of Qualitative Choice Behavior, in P. Zarembka, *Frontiers in Econometrics*, Academic Press: New York, pp 105-142.

Microsoft Corporation (2004). *MapPoint: Business Mapping and Data Visualization Software*, CD-ROM.

Official Airline Guide (2000). Official Airline Guide CD-ROM.

Revelt, D. and Train, K. (1998). Mixed logit with repeated choices, Review of Economics and Statistics, Vol. 80, pp 647-657.

Recker, W.W. (1995). Discrete choice with an oddball alternative, *Transportation Research Part B*, Vol. 29B, No. 3, pp 201-211.

Small, K. (1994). Approximate generalized extreme value models of discrete choice, *Journal of Econometrics*, Vol. 62, Issue 2, pp 351-382.

Small, K. (1987). A discrete choice model for ordered alternatives, *Econometrica*, Vol. 55, pp 105-130.

Steckel, J.H. and Vanhonacker, W.R. (1998). A heterogeneous conditional logit model of choice, *Journal of Business & Economic Statistics*, Vol. 6, No. 3, pp 381-389.

Stopher, P. and Prashker, J. (1976). Intercity Passenger Forecasting: The Use of Current Travel Forecasting Procedures, *Transportation Research Forum: Annual Meeting Proceedings*, pp 67-75.

SAS Institute, http://www.sas.com, version 9.1.3.

Spanos, A. (2000). *Probability theory and statistical inference: Econometric modeling with observational data*, Cambridge University Press.

Spanos, A. (1986). *Statistical foundations of econometric modeling*, Cambridge University Press.

Thurstone, L. (1927). A law of comparative judgment, *Psychological Review*, Vol. 34, pp 273-286.

Train, E. K., McFadden, D, and Ben-Akiva, M. (1987). The demand for local telephone service: A fully discrete model of residential calling patterns and service choice, *Rand Journal of Economics,* Vol. 18, pp 109-123.

Train, E. K. (2003). *Discrete Choice Methods with Simulation*, Cambridge University Press, Cambridge, UK.

Vovsha, P. (1997). Cross-nested logit model: an application to mode choice in the Tel-Aviv metropolitan area, *Transportation Research Board, 76th Annual Meeting*, Washington DC. Paper #970387.

Wen, C. and Koppelman, F.S. (2001). The generalized nested logit. *Transportation Research Part B*, Vol. 35, pp 627-641.

# 3 LOGIT MODELS TO FORECAST NATIONWIDE INTERCITY TRAVEL DEMAND IN THE U.S.

**Accepted for Publication in upcoming Transportation Research Record, 2007.**

Senanu Ashiabor
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 257-3830
Fax. (540) 231-7352
sashiabo@vt.edu

Hojong Baik
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 231-4418
Fax. (540) 231-7352
hbaik@vt.edu

Antonio Trani
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 231-2362
Fax. (540) 231-7352
vuela@vt.edu

## ABSTRACT

Nested and mixed logit models were developed to study national-level intercity transportation in the United States. The models are used to estimate the market share of automobile and commercial air transportation between 3091 counties and 443 commercial service airports in the United States. Models were calibrated using the 1995 American Travel Survey and separate models were developed for business and non-business trip purposes. The explanatory variables used in the utility functions of the models are travel time, travel cost, and traveler's household income. Given an input county-to-county trip demand table, the models were used to estimate county-to-county travel demand by automobile and commercial airline between all counties and commercial service airports in the United States. The model has been integrated into a computer software framework called the Transportation Systems Analysis Model (TSAM) that estimates nationwide intercity travel demand in the United States.

Key Words: Mode Choice, Nested Logit, Mixed Logit.

**Attributions:**

The mode choice model is a module in an intercity travel demand forecasting model called the Transportation Systems Analysis Model (TSAM) developed at the Virginia Tech Air Transportation Systems Laboratory.

TSAM was developed by Dr. Antonio Trani and Dr. Hojong Baik.

The mode choice model in TSAM presented in this paper was developed by Senanu Ashiabor. Senanu was responsible for developing the structure of the model, analyzing the survey data and development of the MATLAB program codes use to run the model.

## 3.1 INTRODUCTION

In 2000, the National Aeronautics and Space Administration (NASA) proposed to Congress the development of a Small Aircraft Transportation System (SATS) to harness the potential of the nation's vast network of underutilized airports. As part of the SATS program, NASA assigned the Virginia Tech Air Transportation Systems Laboratory the task to develop a transportation systems analysis model to estimate the demand for SATS vehicles. Virginia Tech used the classical four-step transportation planning procedure to develop a framework called the Transportation Systems Analysis Model (TSAM) to estimate demand for intercity trips when a novel mode of transportation such as SATS is introduced. The four-step planning model is a sequential demand forecasting model made up of trip generation, trip distribution, mode choice and trip assignment.

Trip generation estimates the number of trips produced and attracted to each zone of activity by trip purpose. Trip distribution estimates origin-destination flows thereby linking trip ends from the trip generation to form trip interchanges between zones. Mode choice estimates the percentage of travelers using each mode of transportation between each origin-destination pair. Trip assignment loads the origin-destination flows of each mode on specific routes through the respective transportation networks.

There are 3091 counties in TSAM that serve as the zones of travel activity in the continental U.S. The trip generation output in is made up of two 3091 vectors for attractions and productions. Trip distribution fills up the cells between the vectors creating a person-trip interchange table of demand between the two counties. Mode choice splits the demand between each county by mode of transportation. The mode choice model in TSAM and this paper estimates both the demand by mode between counties and the demand flows in the airport network associated with the counties. This is achieved by embedding an airport choice model in the mode choice model. Hence the model is both a mode choice and partial trip assignment model. The framework for the process is shown in Figure 3. The modes of transportation considered in the TSAM model are commercial airline, automobile, SATS and train. However, the focus in this paper is on the baseline model which has only automobile and commercial airline modes. The trip assignment in TSAM involves converting the airport-to-airport person-trips into aircraft operations, generating flights using a time of day profile, and loading the flights on the National Airspace

System to estimate the impact of aircraft operations in the system.  The complete travel demand model is fully documented in Trani et al.  *(1,2,3)*.

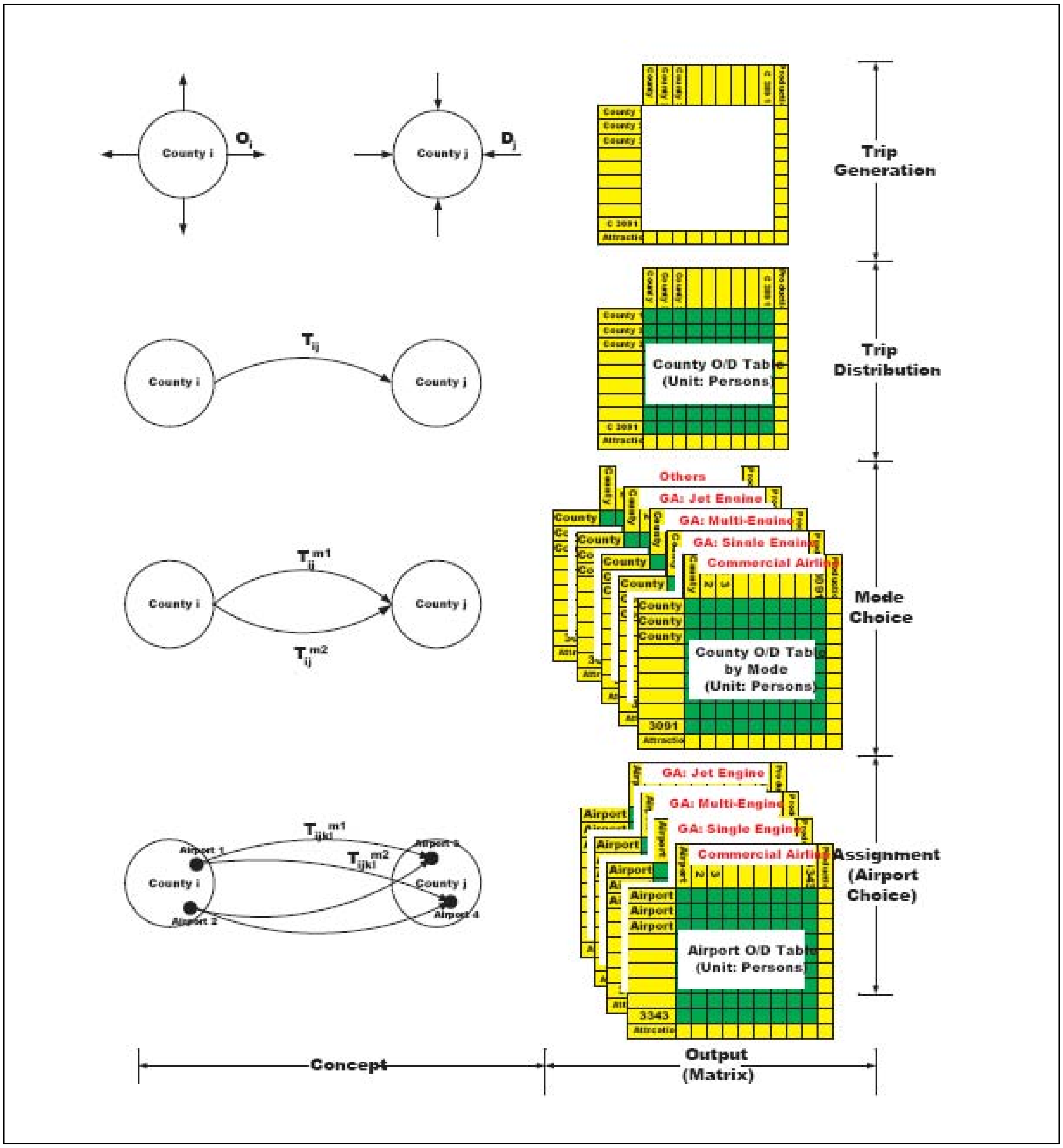

**Figure 3 Multi-step Illustration of Intercity Transportation Modeling Process.**

NASA is currently using TSAM to forecast future airport demands to assist the Joint Planning and Development (JPDO) plan the Next Generation Air Transportation System.  NASA is also using TSAM to study demand for supersonic aircraft, tilt-rotors and Short Take-off and Landing (STOL) aircraft. The above shows the model output is critical to policy makers and hence very relevant. This paper presents a family of logit models that have been developed since then to estimate intercity travel demand in the United States.

## 3.2 Literature Review

### 3.2.1 Review of Disaggregate Nationwide Travel Demand Models

Between 1976 and 1990 four major attempts at developing disaggregate national-level intercity mode choice models are documented in the transportation literature from. All the models used versions of National Travel Surveys (NTS) conducted by the Census Bureau and the Bureau of Transportation Statistics (BTS). The first is a Multinomial Logit (ML) model by Stopher and Prashker *(4)* in 1976 using the 1972 NTS. Alan Grayson *(5)* developed a ML model using the 1977 version of the NTS. Morrison and Winston *(6)* were the first to apply a Nested Logit (NL) formulation using the log-sum variable to hierarchically nest three models: 'Decision to rent a car', Destination choice and Mode Choice. Later on Koppelman *(7)* extended Morrison's approach to hierarchically nest a set of trip frequency, trip destination, mode choice and fare class choice models using log-sum values and the 1997 NTS database. Automobile, air, bus and rail are the main transportation modes in all the models above. Details of the four models and the variables in their utility function are summarized in Table 6.

Travelers' modal choice information was extracted from the NTS surveys. However these surveys did not contain information on level of service variables. Thus the authors developed synthetic travel time and cost data from published fare and schedule guides such as the Official Airline, Railroad, and Bus Guides. They all restricted their analyses to trips starting and ending in Metropolitan Statistical Areas (MSA). The likely reason for this is that, trips in the surveys are only identified by State and whether or not they are in a MSA. It is very difficult to estimate travel times and costs for any trip originating or ending in non-MSA areas given the size of most States.

The calibrated coefficients for the models above had the expected signs however, in the case of the two ML models the elasticity estimates were counterintuitive. The authors attributed model weaknesses to the poor quality of the NTS data, and tenuous assumptions made in derivation of the level of service variables. Koppelman et al., *(8)* also note that a high level of geographic aggregation, poor information on the choice set, and lack of service variables are additional limitations in the development of robust models.

The major constraint in developing credible models is related to the NTS databases more than the modeling techniques.  The two major issues are, the restriction of the minimum level of geographical detail to MSA's, and the absence of information related to airports (such as access and egress distances to airports and terminals).  Koppelman and Hirsh *(9)* expound on the data requirements for researchers and practitioners to develop accurate and useful intercity travel demand models.  However to date there does not seem to be any attempt by any of the key Federal Agencies (Census Bureau or BTS) to collect such data.

The mode choice models presented in this paper extend the work of national-level intercity travel demand modeling in three dimensions.  Procedures are developed to extend the spatial extent of the model to include non-MSA areas so the model can be applied nationally.  Secondly, an airport choice model is implemented with the mode choice so that the model can estimate market share of the airport network to make it more useful to policy makers.  Thirdly, level of service variables are aggregated from the county level giving the model a broader scope since county socio-economic variable forecasts exists at this level.  To the best of our knowledge, this appears to to be the first national level, intercity, multi-mode choice model incorporating both mode choice and airport choice at the county level in the U.S.  reported in the literature.

**Table 6: Major National Level Intercity Travel Demand Models for United States.**

| | **Model Type** | **Data and Scope** | **Modes of Transportation** | **Variables in Utility Function** | **Market Segmentation** |
|---|---|---|---|---|---|
| Stopher and Prashker (1976) | Multinomial Logit | Database: 1972 NTS<br>Scope: Trips that start and end in Metropolitan Statistical Areas (MSA)<br>**2,085** records from database | Automobile, Commercial Air, Bus, Rail. | Relative time, relative distance, relative cost, relative access-egress distance, departure frequency | Trip purpose (Business / Non-Business) |
| Alan Grayson (1982) | Multinomial Logit | Database: 1977 NTS<br>Scope: Trips that start and end in MSA's<br>Selected observations from database | Automobile, Commercial Air, Bus, Rail. | Travel time, travel cost, access time, and departure frequency | Trip purpose (Business / Non-Business) |
| Morrison and Winston (1985) | Nested Logit | Database: 1977 NTS<br>Scope: Trips that start and end in MSA's<br>**4,218** records from database | Automobile, Commercial Air, Bus, Rail. | Travel time, cost, party size, average time between departures | Trip purpose (Business / Non-Business) |
| Koppeleman (1990) | Nested Logit | Database: 1977 NTS<br>Scope: Trips that start and end in MSA's<br>Selected observations from database | Automobile, Commercial Air, Bus, Rail. | Travel time, cost, departure frequency, distance between city pairs, household income. | Trip purpose (Business / Non-Business) |
| Mode Choice model in TSAM | Nested Logit and Mixed Logit Models | Database: 1995 American Travel Survey<br>Scope: All trips irrespective of origin or destination type.<br>**402,295** records from database | Automobile, Commercial Air, Train, SATS. | Travel time, Travel Cost, Household Income, Region Type. | Trip purpose (Business / Non-Business)<br>Household Income |

### 3.2.2 Review of Logit Models

McFadden *(10)* developed the ML model based Luce's *(11)* axiom of independence of irrelevant alternatives (IIA). McFadden's model assumes an underlying Gumbel distribution and a random sample that is Independent and Identically Distributed (Gumbel IID) which implies that the alternatives considered in each choice set are independent of each other and have the same variance. The ML probability for selecting alternative $i$ for $J$ alternatives, where $j = 1,2,\ldots J$, has form shown in Equation 20.

$$P(i) = \frac{e^{V_i}}{\sum_{j=1}^{J} e^{V_j}} \tag{20}$$

Where $V$ is the systematic utility. It is clear from Equation 20 that for any two alternatives $k$ and $l$ the ratio of their probabilities $\frac{P(k)}{P(l)} = \frac{e^{V_k}}{e^{V_l}}$ is independent of any other alternatives in the model.

The constant nature of this ratio irrespective of the presence of other alternatives can lead to unrealistic substitution patterns associated with the IIA property. Ben-Akiva and Lerman, *(12)* use the now famous red bus, blue bus problem to illustrate how IIA can produce wrong estimates when modes with similar characteristics are present in the choice set. IIA also affects model cross-elasticity estimates. Consider the impact of the change in an attribute of an alternative $j$ on the probability $P_{ni}$ of all other alternatives in the model. The change in $P_{ni}$ with respect to a change in the attribute of $j$ is given as Equation 21, (see Train *(13)*)

$$E_{iZ_{nj}} = -\beta_z Z_{nj} P_{nj} \tag{21}$$

Where $Z_{nj}$ is the attribute of alternative $j$ faced by individual $n$ where $\beta_z$ is its coefficient.

Since the cross-elasticity is the same for all $i$, the implication is that an improvement in any one alternative reduces the probabilities of all the other alternatives by the same amount (that is $E_{iZ_{nj}}$ is fixed for all $i's$). This means in a model with say three alternatives, if a policy is implemented to improve one alternative, the ML model will draw the same percentage from the remaining modes. This yields unrealistic substitution patterns, and it is not surprising that elasticity and cross-elasticity estimates from Grayson *(5)* and Stopher's *(4)* ML models did not yield intuitive estimates. The ML model is analytically tractable due to its closed form, however the IIA property renders it unsuitable for policy studies that seek to investigate the impact of improving or introducing new alternatives if they have similar attributes to existing alternatives. In order to develop more flexible empirical models there was a shift towards relaxing the independence or identical distribution assumptions while maintaining the analytically closed form of the model.

The first attempt was the NL model that relaxed the independence assumption by grouping alternatives that are similar into nests (McFadden *(14)*, Daly and Zachary *(15)*), and later models that relax the independence assumption include cross-nested logits *(16,17),* ordered generalized extreme value models *(18,19),* Chu's *(20)* paired combinatorial logit and Wen and Koppelman's *(21)* generalized nested logit. McFadden specified a Generalized Extreme Value (GEV) joint distribution that allows for any form of correlation that is an overarching framework over all these models including the logit model.

A detailed discussion on GEV models is available in Train *(13),* and Ben-Akiva and Lerman *(12).* Define $Y_j \equiv e^{V_j}$ and a function $G\,(Y_1, Y_2, \ldots, Y_J)$ for $(Y_1, Y_2, \ldots, Y_J \geq 0)$ with the following properties

1. $G$ is non-negative for all values of $y_j$ for all $j$.
2. $G$ is homogeneous of degree $\mu > 0$; that is $\mathrm{G}(\alpha Y_1, \alpha Y_2, \ldots, \alpha Y_J) = \alpha^{\mu} G(Y_1, Y_2, \ldots, Y_J)$.
3. $\lim_{Y_j \to \infty} G(Y_1, Y_2, \ldots, Y_J) = \infty, \textit{ for } j = 1,2,\ldots, J_n$).
4. The $i^{th}$ partial derivative of $G$ with respect to any combination of $i$ distinct $Y_j$'$s$, is non-negative if $i$ is odd, and non-positive if $i$ is even.

If all the above conditions hold and defining the partial derivative of $G$ as $G_i = \partial G / \partial Y_i$ then $P(i) = \frac{Y_i G_i}{G}$ is the choice probability for a discrete choice model that is consistent with utility maximization. If we partition $J$ alternatives are into $K$ nests $(B_1, B_2, \ldots B_K)$. Then specify $G = \sum_{l=1}^{K} \left( \sum_{j \in B_l} Y_j^{1/\lambda_l} \right)^{\lambda_l}$, with each $\lambda_k$ between zero and one for the nested logit model it can be shown that the choice probability has the form in Equation 22,

$$P(i) = \frac{Y_i G_i}{G} \quad = \frac{Y_i Y_i^{(1/\lambda_l)-1} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l}} = \frac{Y_i^{1/\lambda_k} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} Y_j^{1/\lambda_l} \right)^{\lambda_l}}$$

$$\textit{substituting} \quad e^{V_i} \qquad (22)$$

$$= \frac{\left(e^{V_i}\right)^{\lambda_k} \left( \sum_{j \in B_k} \left(e^{V_j}\right)^{1/\lambda_k} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} \left(e^{V_j}\right)^{1/\lambda_l} \right)^{\lambda_l}} = \frac{e^{V_i/\lambda_k} \left( \sum_{j \in B_k} e^{V_j/\lambda_l} \right)^{\lambda_k - 1}}{\sum_{l=1}^{K} \left( \sum_{j \in B_k} e^{V_j/\lambda_l} \right)^{\lambda_l}}$$

Clearly for any two alternatives $i \in B_k$ and $m \in B_l$ in different nests

$$\frac{P(i)}{P(m)} = \frac{e^{V_i/\lambda_k} \left( \sum_{j \in B_k} e^{V_j/\lambda_k} \right)^{\lambda_k - 1}}{e^{V_m/\lambda_l} \left( \sum_{j \in B_l} e^{V_j/\lambda_l} \right)^{\lambda_l - 1}} \qquad (23)$$

and IIA does not hold because the ratio of their probabilities are tied to all alternatives in their respective nests. However, since the ratio only applies to alternatives within nests there is still a form of IIA referred to as ‘independence from irrelevant nests’ (IIN). Note that if the two alternatives are in the same nest (i.e. $k = l$) then $\frac{P(i)}{P(m)} = \frac{e^{V_i/\lambda_k}}{e^{V_m/\lambda_l}}$ this ratio is independent of all other alternatives, so for the NL IIA holds within nests. The NL model is part of the GEV family and is the most frequently used because of its ability to overcome the IIA weakness while maintaining an analytically tractable and closed form.

More recently the Heteroskedastic Extreme Value (HEV) was developed to relax the Identical Distribution assumption *(22,23,24)*. Logically the next step was to develop a model that relaxes both independence and identical distribution simultaneously. These models belong to the class of mixed logits.

There are two versions of mixed logit models in the literature; the random-coefficients and the error-components specifications. Statistically the models are equivalent but the specifications differ by the behavioral mechanism the researcher uses to justify the interpretation. The random-coefficients model is presented first and then we show that the error-components is a different way of looking at the same statistical model.

### 3.2.3 Random-coefficients Mixed logit

In all the logit models considered so far the utility takes the form $U_{nj} = \alpha x_{nj} + \varepsilon_{nj}$ where $x_{nj}$ is a vector of attributes that relate to the individual $n$ and $j$ alternatives. The error term $\varepsilon_{nj}$ is IID Extreme Value. The coefficient $\alpha$ are fixed for each attribute $x_{nj}$. In the random-coefficients mixed logit the vector of coefficients $\alpha_n$ is not fixed but rather varies over individuals '$n$' with a density $f(\alpha)$ in Equation 24.

$$U_{nj} = \alpha_n \, x_{nj} + \varepsilon_{nj} \tag{24}$$

The decision maker knows the complete value of their utility in the form of the values of $\alpha_n \textit{ and } \varepsilon_{nj}$ and selects the alternative with the highest utility, however the researcher only observes the choice and the $x_{nj}$'s but not coefficients $\alpha_n$ and error term $\varepsilon_{nj}$. The unconditional probability over all possible values of $\alpha_n$ takes the form shown in Equation 25.

$$P_{ni} = \int \left( \frac{e^{\alpha x_{ni}}}{\sum_j e^{\alpha x_{nj}}} \right) f(\alpha) d\alpha \tag{25}$$

The researcher specifies a distribution for the coefficients $\alpha_n$ and estimates the parameters of the distributions (say mean and variance). The utility function takes the form of a weighted average of the logit formula estimated at different values of $\alpha$ with weights given by the density $f(\alpha)$ as shown in Equation 25. Common distributions used in practice are the Normal, Log-normal, Triangular and Uniform.

### **3.2.4 Error Components Mixed logit**

The error-components form of the mixed logit decomposes the utility into fixed and random components as shown in Equation 26.

$$U_{nj} = \delta' \, x_{nj} + \beta_n' \, z_{nj} + \varepsilon_{nj} \tag{26}$$

where $x_{nj}$ and $z_{nj}$ are vectors of observed variables relating to alternative $j$, and $\delta$ is a vector of fixed coefficients, $\beta$ is a vector of random terms with zero mean, and $\varepsilon_{nj}$ is IID extreme value. The variables in $z_{nj}$ are the ones referred to as error-components since they are correlated with the IID error $\varepsilon_{nj}$. Together they define the stochastic components of the utility

$(\beta_n' z_{nj} + \varepsilon_{nj})$. Now, consider the distribution of $\alpha_n$ from (24 with mean $\delta'$ and standard deviation $\beta_n'$ clearly the utility becomes $U_{nj} = \delta' x_{nj} + \beta_n' x_{nj} + \varepsilon_{nj}$ such that if $x_{nj}$ is replaced with $z_{nj}$ in the second term the two models are equivalent statistically.

McFadden and Train *(25)* have shown that the mixed logit is capable of approximating the full family of logit models with the appropriate choice of mixing distributions. Early mixed logit applications were developed by Byod and Mellman *(26)*, and Cardell and Dunbar *(27)* and since then mixed logits have been actively use for modal choice modeling *(28,29,30)*. The flexibility gained by relaxing the restrictive assumptions is however offset by the need to use simulation techniques in estimation of the mixed logit model leading to increased computational time.

The 1995 American Travel Survey was used to develop a set of nested and mixed logit models. Data from a stated preference travel surveys conducted by Virginia Tech *(3)* are used in generating synthetic travel times. We experiment with various variables such as whether trips start or end in an MSA area, standard level of service variables such as travel time, cost and household income used in past national- level travel demand models. Strengths of the models include the ability to predict how market share changes with policy, ability to overcome the IIA structure, and the ease of integrating new modes of transportation in the model.

Currently, policy makers and planners have only national/regional level statistics to plan policies for a system spanning several geographical areas with different characteristics. In the case where localized studies are implemented to supplement regional level statistics the outputs are usually not transferable spatially. We therefore developed a nationwide multi-mode travel demand model at the county to county level to improve decision making ability of policy makers and planners.

## 3.3 METHODOLOGY

The main output of any logit model is an estimate of the probability in Equation 27.

$$P_i = \frac{e^{V_i}}{\sum_i e^{V_i}} \tag{27}$$

where $P_i$ is the probability of using mode of transportation $i$, and $V_i$ the utility value associated with mode $i$ with the form in Equation 28.

$$U_i = \alpha_j X_{ij} \tag{28}$$

where $X_{ij}$ is the $j$ variable(s) in the model and $\alpha_j$ are the model coefficients. Calibration of the model involves estimating coefficients $\alpha_j$ that give a best fit to the observed data.

### 3.3.1 American Travel Survey

In this analysis the 1995 American Travel Survey (ATS) constitutes the source of traveler information supplemented with a random survey of 2,000 records designed and conducted by the authors. The ATS is a survey of long-distance trips with route distance greater then 100 miles (one-way) conducted by Census Bureau for the Bureau of Transportation Statistics *(31)*. The database has 556,026 person trips records and 348 variables/fields for each record. Like the NTS, ATS has information on choices travelers made but has little information on the level of service variables. To calibrate the proposed models synthetic level of service variables were generated from external data sources as explained in the next section. ATS data is released at two levels. One is the actual database of 556,026 records and the other is published summary statistics projected from the sample. The ATS market share curves shown in Figure 4 indicates that 1) travelers tend to switch to faster modes of transportation for long trips and that 2) high income travelers tend to switch to the faster model earlier than low income travelers. This is the basis for stratifying the travel cost variable in the utility function by income categories.

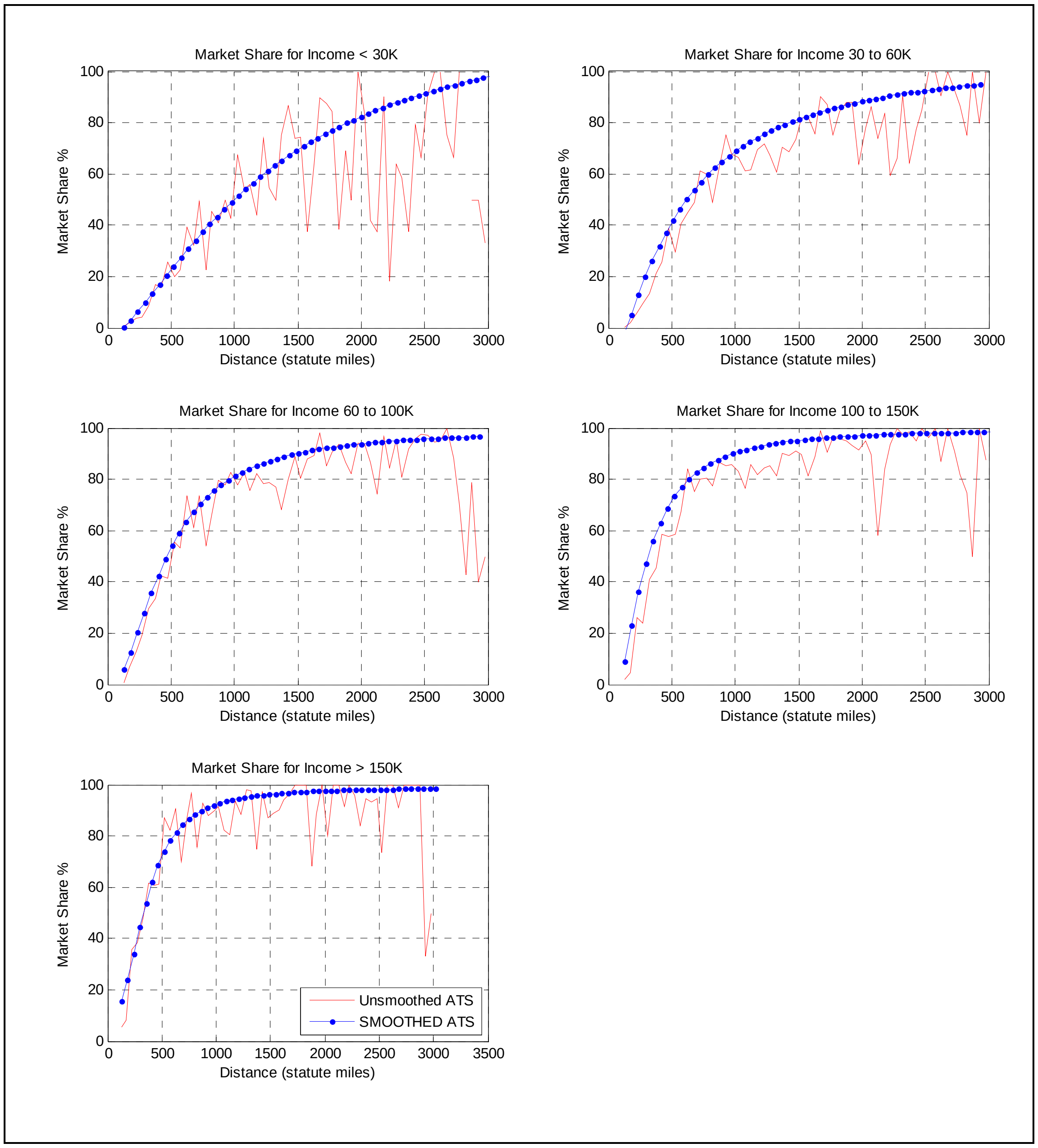

**Figure 4. Business Commercial Aviation Market Share Plots from ATS Sample Data.**

### 3.3.2 Development of Form of Logit model

The airport choice model was incorporated into the mode choice model because, this would allows the simultaneous estimation of the market share for commercial aviation between the counties and market share between airline routes available for that county pair. With this approach the applied model yields a county-to-county commercial airline demand table and an airport-to-airport demand table. The later is more useful to policy makers.

The form of the model is as follows: given any county pair, associate a set of airports with the county. Next create a set of feasible commercial airline routes for the county pair. Each route is characterized by the door-to-door level of service variables (i.e. travel times and costs). Each commercial airline route enters the NL model as an alternative as shown in Figure 5. Separate models were calibrated for business and non-business travelers. The impact of income on the behavior of traveler is incorporated in the model by splitting travelers into five income categories and incorporating the categories into the structure of the cost variable in the utility function.

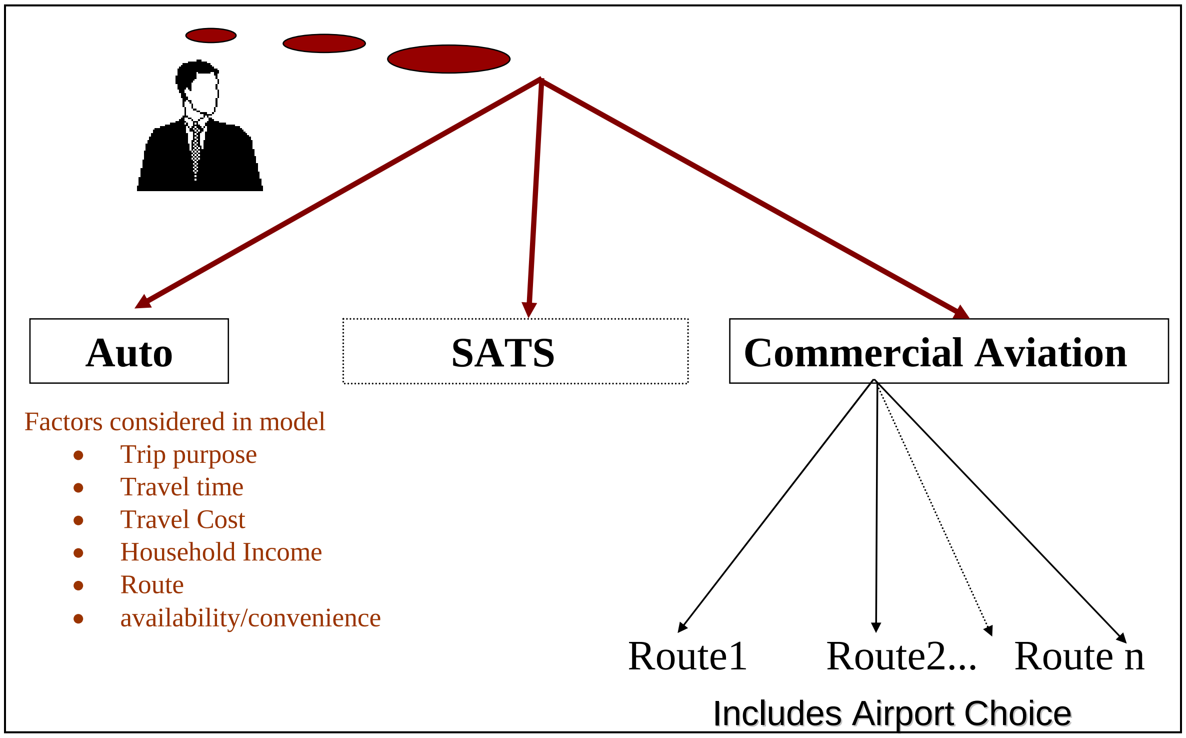

**Figure 5: Concept of Nested Logit Model.**

### 3.3.3 Form of Nested and Mixed Logit Utility Functions

#### 3.3.3.1 Nested logit utility function

After experimenting with various forms the utility structure in Figure 5 was selected for the logit model formulation. The variables used in the model are travel time, travel cost, household income and the location of the trip origin and destination (MSA or non-MSA). After testing different combinations of the utility function we selected the form shown in Equation 29,

$$U_{ij}^{klm} = \alpha_0 \ Travel\ Time_{ij}^{k} + \alpha_1 \ Travel\ Cost_{ij}^{k1} + \alpha_2 \ Travel\ Cost_{ij}^{k2} + \alpha_3 \ Travel\ Cost_{ij}^{k3} \ldots$$
$$+ \alpha_4 \ Travel\ Cost_{ij}^{k4} + \alpha_5 \ Travel\ Cost_{ij}^{k5} + \alpha_6 IncomeDummy_{ij}^{m} \tag{29}$$

where $U_{ij}^{klm}$ is the utility value of a trip maker of income group $l$ traveling from origin county $i$ to destination county $j$ using mode of transportation $k$. The subscript $m$ is a categorical income dummy for income greater than one hundred thousand and less than one hundred thousand. $\alpha_0$ is the travel time coefficient, and $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \ and\ \alpha_5$. are income specific coefficients related to travel cost for each income group. The coefficient for the categorical income dummy variable is $\alpha_6$. The dummy was derived based on empirical examination of travelers' choice patterns observed in the ATS data.

We test an extension of the model with a dummy variable for short trips. These are trips for which both the synthetic travel time and cost for commercial air are higher than that for a corresponding automobile trip for the same route distance. We defined a short trip as the breakeven point at which the travel times for automobile and commercial air coincide. Due to the geographical location of airports this breakeven point is likely to vary by region (i.e. MSA/non-MSA) hence the dummy enters as a region specific dummy as shown in Equation 30,

$$U_{ij}^{klmn} = \alpha_0 \ Travel\ Time_{ij}^{k} + \alpha_1 \ Travel\ Cost_{ij}^{k1} + \alpha_2 \ Travel\ Cost_{ij}^{k2} + \alpha_3 \ Travel$$
$$+ \alpha_4 \ Travel\ Cost_{ij}^{k4} + \alpha_5 \ Travel\ Cost_{ij}^{k5} + \alpha_6 IncomeDummy_{ij}^{m} + shortTripDun \tag{30}$$

where $shortTripDummy_{ij}^{n}$ is the subscript $n$ is related to whether the trip starts or ends in an MSA. All other coefficients have the same meaning as in Equation 10. Plots of the synthetic travel times showed that the breakeven point is about 800 miles for non-MSA's and 350 miles otherwise.

### 3.3.3.2 Mixed logit utility function

The variables in the mixed logit utility function are the same as the NL formulations explained before. The difference is in the fact that the time coefficient is no longer fixed, and the mixed logit has no nests. Hence the airline routes and automobile are all at the same level. To illustrate, the mixed logit is now reformulated Equation 31 as:

$$U_{ij}^{klm} = (\alpha_0 + \beta) Travel\ Time_{ij}^{k} + \alpha_1 \ Travel\ Cost_{ij}^{k1} + \alpha_2 \ Travel\ Cost_{ij}^{k2} + \ldots$$
$$\alpha_3 \ Travel\ Cost_{ij}^{k3} + \alpha_4 \ Travel\ Cost_{ij}^{k4} + \alpha_5 \ Travel\ Cost_{ij}^{k5} + \alpha_6 IncomeDummy_{ij}^{m} \tag{31}$$

where $\alpha_0$ is the fixed coefficient for travel time and $\beta$ is the random component. The travel time parameter in the mixed logit application was modeled using a normal distribution. The NL and mixed logit models are calibrated using the PROC MDC function in the SAS statistical software *(32)*. SAS provides goodness of fit estimates in the form of various R-squared values and log-likelihood ratios, and p-values for each coefficient.

### 3.3.4 Synthetic Automobile Travel Times and Costs

Automobile drive times between all the 3091 counties in the U.S. were estimated using Microsoft MapPoint software *(33)*. This generates a 3091x3091 table of drive times where each row represents all the trips from one county to all the other counties in the United States. The Virginia Tech travel surveys indicate that travelers tend to stopover for an overnight stay after 8 and 10 hours for business and non-business trips, respectively. This information was used to adjust the drive time to obtain a total travel time between counties. This level of detail is adequate for applying the calibrated model in TSAM. However, since the lowest level of geographical detail in the ATS is the MSA area, the drive times (and all other variables) need to be aggregated up to that level.

The drive times are aggregated along three dimensions. By origin state, distance, and trip origin-destination type (MSA or non-MSA). The aggregated data is also weighted by number of trips for each county. Say for Virginia, extract drive times for all trips from any county in the State that is between 100 and 150 miles route distance one-way. Select those county pairs for which both the origin and destination counties are MSA's and generate the average travel time, weighting it by total number of trips from the counties. Repeat the procedure for MSA to non-MSA, non-MSA to MSA, and then non-MSA to non-MSA. If the procedure is repeated for increasing distance brackets up to 3,000 miles by State, the resulting input table has dimension of 50 States x 4 MSA regions x 58 distance brackets. For any trip in the ATS the appropriate aggregate travel time can be selected from this table.

The procedure for automobile travel cost is similar to that of drive times. Route drive distances obtained in MapPoint are multiplied by an average driving cost per mile to obtain the automobile trip cost. Automobile travel cost was judgmentally set at be 37 cents-per-mile based on reviews of rates published by the American Automobile Association. The overnight stay cost is the product of number of overnight days and daily lodging cost. The business lodging cost by income group from the highest to the lowest income levels were $70, 80, 90, 100, and 120, respectively. For non-business trips we used $50, 60, 70, 80, and 90, respectively. All cost values are adjusted by party size numbers extracted from the ATS and that vary by income group. Hence the travel cost tables have an additional dimension for income, (i.e. 50 states x 4 MSA regions x 58 distance brackets x 5 income groups). The business party size extracted from the ATS by income level was 2.44, 2.43, 2.01, 1.84, and 1.87. That for non-business was 2.98, 3.19, 3.24, 3.18, and 3.28. Ideally one would expect the values to increase monotonically however this was not the case for non-business values.

### 3.3.5 Synthetic Commercial Airline Travel Time and Costs

Airport-to-airport flight times between 443 commercial service airports were synthesized from the Official Airline *(34)*. The travel time between an airport pair is based on the number of possible routes between them in the OAG and weighted by the volume of traffic on each route. Schedule delay *(35)* a measure of the additional travel time penalty air travelers are forced to experience because flights are not scheduled at the time they want to depart is added on to the flight time. The full procedure to estimate the flight times is documented by Trani et al., 2003 *(3)*. The door-to-door travel time for commercial airline is made up of:

1. Access time (time spent traveling to the airport)
2. Origin Airport wait time (time from arrival at the airport till flight departs)
3. Air travel time
4. Schedule delay
5. Destination Airport wait time (time from disembarking till exiting the terminal)
6. Egress time (time from exiting the terminal till arrival at the destination).

The access and egress times for commercial aviation are computed in the same manner as for automobile.

Commercial airline travel costs are also synthesized from the Department of Transportation ten percent sample ticket survey referred to as DB1B *(36)*. Airport-to-airport flight cost table for the 443 commercial service airports is created from the ticket survey. The airports were classified into the four hub groupings currently used by the FAA and sixteen cost curves were created based on these groupings. When more than 5 observations are available in DB1B for an airport pair the average of those fares is inserted in the table. For those airports with few or no samples in the database the generic cost curves used to fill in the cells. The procedure is fully explained in Trani et al., 2003 *(3)*. The travel costs are made up of the access cost, air fare and the egress cost. The access and egress costs are computed the same way as for automobile.

### 3.3.6 Airport Choice Model Assumptions

The airport choice behavior was based on analysis of the ATS data. The access distance information in the ATS (Figure 6) shows that access distance to airports varies by MSA region type. From Figure 6 it is clear that the access distance is mainly related to trip origin type. The plots show that for trips originating from MSA areas the maximum access distance is 100 miles compared to about 250 miles for trips starting in non-MSA areas. Based on these observations the following rule was established for access distance. For any trips starting in an MSA area only airports within 100 miles radius of the population weighted county centroids are considered in the choice set irrespective of trip purpose. For trips starting in non-MSA areas the radius is 200 miles.

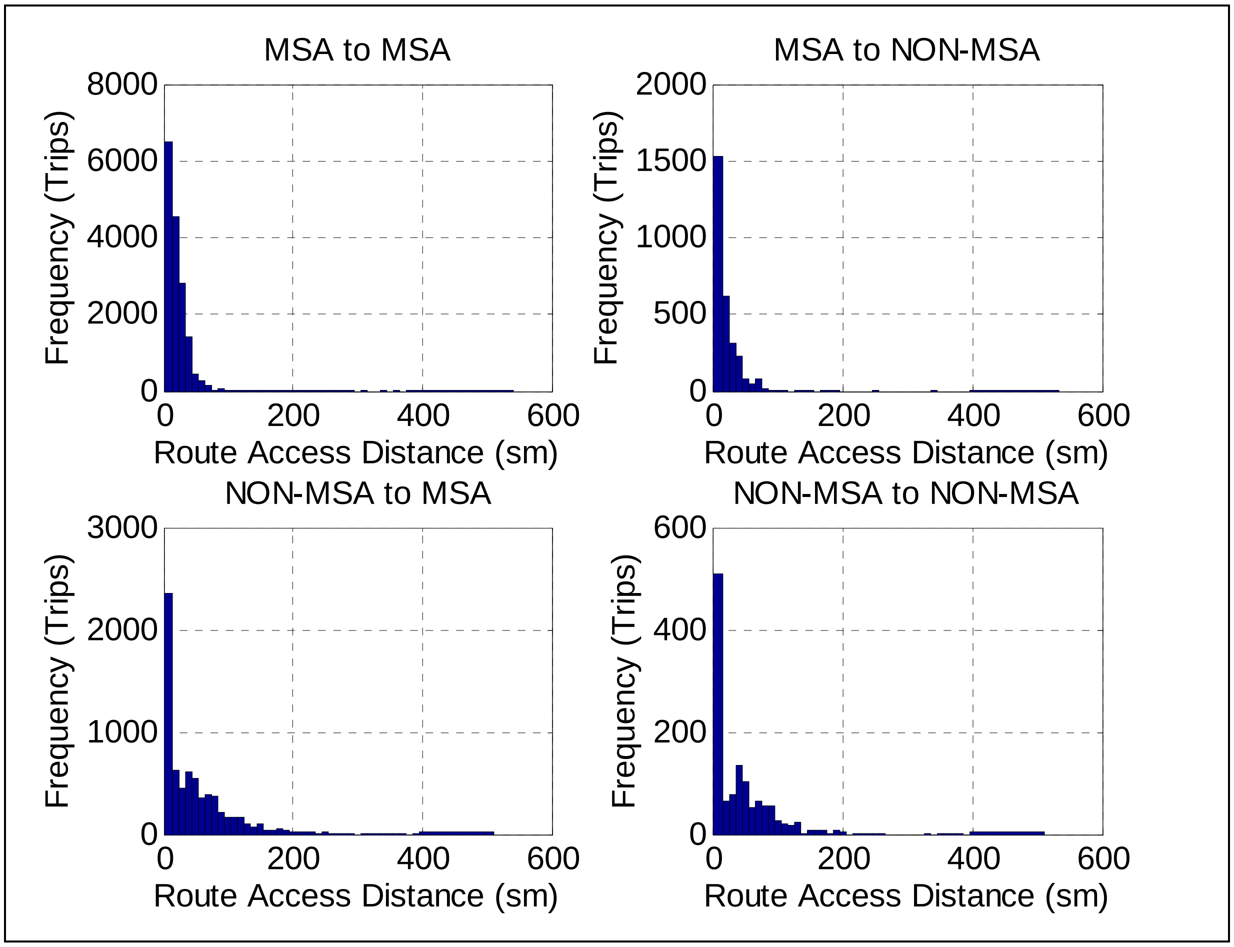

**Figure 6: Histogram of Access Distance for Business Trips in ATS Sample Data.**

The rules described above will generate several airports for each county. For practical purposes it is necessary to reduce the choice set to a manageable number of airports. It was decided to limit the number or airports associated with each county to three leading to a maximum of nine routes between each county pair. Three airports are selected using the following criteria: the closest airport to the population weighted county centroid, the airport with the lowest average fare from the remaining airports, and the airport with the highest average number of enplanements from the remaining airports. For time and convenience reasons some travelers will always consider the closest airport irrespective of cost. The airport choice literature shows that travelers prefer airports with low fares, high departure frequencies and high number of connections to other airports. We believe selecting the airports with the lowest fares and highest number of enplanements will adequately create a choice set with all the major attributes important to travelers. With these rules candidate airports sets can be pre-processed and assigned to each county before running the TSAM model.

Once a county pair is selected in the model the candidate airports for that county are automatically read and the level of service variable related to them can be use to create door-to-door travel times for all the possible routes between those counties.

#### 3.3.6.1 Elimination of Inappropriate Routes

When applying the calibrated coefficients to estimate demand certain limitations were identified. First, as mentioned earlier the dummy $shortTripDummy_{ij}^{n}$ was introduced for cases where the commercial travel time and cost are both considerably higher than that of automobile. However, due to the probabilistic nature of the logit models some market share is assigned to commercial air albeit small. A filter was implemented in the code that applies the calibrated model to delete such routes as alternatives from the choice set if commercial airline travel time takes more than 1.5 the automobile travel time in such cases.

The second issue has to do with the fact that in some non-hubs airports received a disproportionately high amount of demand due to their presence in the choice set of several counties. A second rule was applied where if both a large hub and non-hub were part of the choice set for a selected county and the non-hub was not the closet airport it was deleted from the choice set. This is based on and apriori assumption that if a large hub was present in the choice set very few people would consider the non-hub for travel unless it was really close to their origin or destination.  The rule may be further extended to small hubs in future versions of the model.

#### 3.3.6.2 Airport Choice Data for Calibration

As mentioned earlier the highest resolution of the ATS is the MSA level, and there is no airport related information in the ATS database.  Therefore for purposes of calibration all the travel times and costs for commercial air travel have to be aggregated like those of the automobile to state, MSA region, distance and income categories.  The presence of airports in the commercial air mode case adds another level of complexity.  For any county pair there can be one to nine routes.  In aggregating the data it was decided to limit the number of routes to three based on analysis of airport choice information in the surveys conducted by Virginia Tech.  The surveys showed that more than ninety percent of the time travelers only use three of the routes.  These are the routes between 1) closest airport at origin and closest airport at destination, 2) closest airport at origin and cheapest airport at destination, 3) cheapest airport at origin and closest airport at destination.  The data for calibration was therefore aggregated for only those three routes.  Hence the dimension for the travel time data for commercial air is 50 states x 4 MSA regions x 58 distance brackets x 3 routes.  The dimension for travel cost is 50 states x 4 MSA regions x 58 distance brackets x 5 income groups x 3 routes.

## 3.4 ANALYSIS OF RESULTS

The model coefficient estimates are presented in Table 7.  All coefficient estimates are negative, indicating that as travel times and costs increase the utility of any of the modes decreases.  All coefficients of variables in the NL model are significant except for the non-business MSA region dummy.  The r-squared estimates obtained for all the models are greater than 80% indicating an acceptable fit.  Examination of the travel cost coefficients over the range of income levels show they decrease with increasing income showing high income travelers are less sensitive to travel cost.

Comparing the mixed logit and the NL models the mixed logits always have a higher r-square value, and their log likelihood estimates indicate a better fit than the logit model.  Figure 7 is a plot comparing the commercial airline market share of the ATS against estimates using the NL model coefficients.  The plots are for the five income groups.  The oscillations observed in the ATS curves beyond fifteen hundred miles are due to the small sample size.  The plots and the model statistics both indicate the NL model presented is able to credibly predict market share for intercity travel demand. The application of the model to estimate nationwide demand is presented in Trani et al. *(3)*.

## Table 7: Model Coefficient Estimates.

| Variable Name | NESTED LOGIT | | | | | | | | MIXED LOGIT | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Business | | | | Non-Business | | | | Business | | | | Non-Business | | | |
| | Coefficient | Standard Error | t-value | p-value | Coefficient | Standard Error | t-value | p-value | Coefficient | Standard Error | t-value | p-value | Coefficient | Standard Error | t-value | p-value |
| **Without Region Dummy** | | | | | | | | | | | | | | | | |
| *Fixed Coefficients* | | | | | | | | | | | | | | | | |
| Travel Time | -0.0197 | 0.0011 | -17.33 | <.0001 | -0.0311 | 0.0006 | -50.33 | <.0001 | -0.0454 | 0.001429 | -31.78 | <.0001 | -0.0529 | 0.000742 | -71.33 | <.0001 |
| Travel Cost | | | | | | | | | | | | | | | | |
| Household Income (less than $30K) | -0.0102 | 0.0003 | -36.61 | <.0001 | -0.0080 | 0.0001 | -81.26 | <.0001 | -0.008463 | 0.000173 | -48.92 | <.0001 | -0.008203 | 0.0000784 | -104.58 | <.0001 |
| Household Income ($30 to $60K) | -0.0088 | 0.0002 | -49.93 | <.0001 | -0.0078 | 0.0001 | -98.3 | <.0001 | -0.007374 | 0.0000957 | -77.06 | <.0001 | -0.008151 | 0.0000488 | -166.94 | <.0001 |
| Household Income ($60 to $100K) | -0.0064 | 0.0001 | -48.14 | <.0001 | -0.0070 | 0.0001 | -97.33 | <.0001 | -0.005535 | 0.0000878 | -63.04 | <.0001 | -0.0073 | 0.0000506 | -144.17 | <.0001 |
| Household Income ($100 to $150K) | -0.0048 | 0.0001 | -38.82 | <.0001 | -0.0062 | 0.0001 | -84.03 | <.0001 | -0.004199 | 0.0000917 | -45.78 | <.0001 | -0.006438 | 0.0000654 | -98.44 | <.0001 |
| Household Income (greater than 150K) | -0.0032 | 0.0002 | -20.63 | <.0001 | -0.0041 | 0.0001 | -43.77 | <.0001 | -0.002765 | 0.000133 | -20.74 | <.0001 | -0.004281 | 0.000099 | -43.25 | <.0001 |
| Distance Dummy | -2.0486 | 0.0601 | -34.09 | <.0001 | -2.5981 | 0.0489 | -53.15 | <.0001 | -1.1171 | 0.0251 | -44.44 | <.0001 | -2.4101 | 0.0254 | -95.02 | <.0001 |
| Inclusive Value | 0.6226 | 0.0144 | 43.28 | <.0001 | 0.9536 | 0.0142 | 67.39 | <.0001 | - | - | - | - | - | - | - | - |
| *Random Coefficients* | | | | | | | | | | | | | | | | |
| Travel Time | - | - | - | - | - | - | - | - | -0.0655 | 0.001229 | -53.25 | <.0001 | 0.0588 | 0.001074 | 54.73 | <.0001 |
| *R-Square (Estrella)* | 0.8866 | | | | 0.9854 | | | | 0.892 | | | | 0.9859 | | | |
| *Log Likelihood* | -54,572 | | | | -92,929 | | | | -53624 | | | | -91656 | | | |
| **With Region Dummy** | | | | | | | | | | | | | | | | |
| *Fixed Coefficients* | | | | | | | | | | | | | | | | |
| Travel Time | -0.0189 | 0.0011 | -16.68 | <.0001 | -0.0302 | 0.0006 | -50.02 | <.0001 | -0.045 | 0.001426 | -31.57 | <.0001 | -0.0531 | 0.000744 | -71.38 | <.0001 |
| Travel Cost | | | | | | | | | | | | | | | | |
| Household Income (less than $30K) | -0.0094 | 0.0003 | -34.35 | <.0001 | -0.0079 | 0.0001 | -79.77 | <.0001 | -0.008239 | 0.000176 | -46.71 | <.0001 | -0.008326 | 0.000083 | -100.28 | <.0001 |
| Household Income ($30 to $60K) | -0.0083 | 0.0002 | -44.20 | <.0001 | -0.0078 | 0.0001 | -95.5 | <.0001 | -0.007108 | 0.0001 | -70.94 | <.0001 | -0.008264 | 0.0000553 | -149.36 | <.0001 |
| Household Income ($60 to $100K) | -0.0061 | 0.0001 | -44.14 | <.0001 | -0.0070 | 0.0001 | -96.92 | <.0001 | -0.005387 | 0.0000907 | -59.39 | <.0001 | -0.00737 | 0.0000535 | -137.79 | <.0001 |
| Household Income ($100 to $150K) | -0.0047 | 0.0001 | -36.79 | <.0001 | -0.0062 | 0.0001 | -84.46 | <.0001 | -0.004101 | 0.0000934 | -43.92 | <.0001 | -0.006495 | 0.0000666 | -97.55 | <.0001 |
| Household Income (greater than 150K) | -0.0031 | 0.0002 | -19.78 | <.0001 | -0.0041 | 0.0001 | -44.24 | <.0001 | -0.002659 | 0.000135 | -19.75 | <.0001 | -0.004341 | 0.0000998 | -43.48 | <.0001 |
| Region Dummy | -0.2081 | 0.0314 | -6.62 | <.0001 | 0.0165 | 0.0164 | 1 | 0.3163 | -1.1087 | 0.0253 | -43.84 | <.0001 | -2.4169 | 0.0254 | -95 | <.0001 |
| Distance Dummy | -1.9136 | 0.0591 | -32.38 | <.0001 | -2.5513 | 0.0478 | -53.41 | <.0001 | -0.1516 | 0.0222 | -6.84 | <.0001 | 0.0801 | 0.0181 | 4.43 | <.0001 |
| Inclusive Value | 0.6523 | 0.0162 | 40.27 | <.0001 | 0.9728 | 0.0144 | 67.68 | <.0001 | - | - | - | - | - | - | - | - |
| *Random Coefficients* | | | | | | | | | | | | | | | | |
| Travel Time | - | - | - | - | - | - | - | - | 0.0648 | 0.00126 | 51.45 | <.0001 | 0.059 | 0.001063 | 55.52 | <.0001 |
| *R-Square (Estrella)* | 0.8867 | | | | 0.9853 | | | | 0.892 | | | | 0.9859 | | | |
| *Log Likelihood* | -54559 | | | | -93065 | | | | -53619 | | | | -91639 | | | |

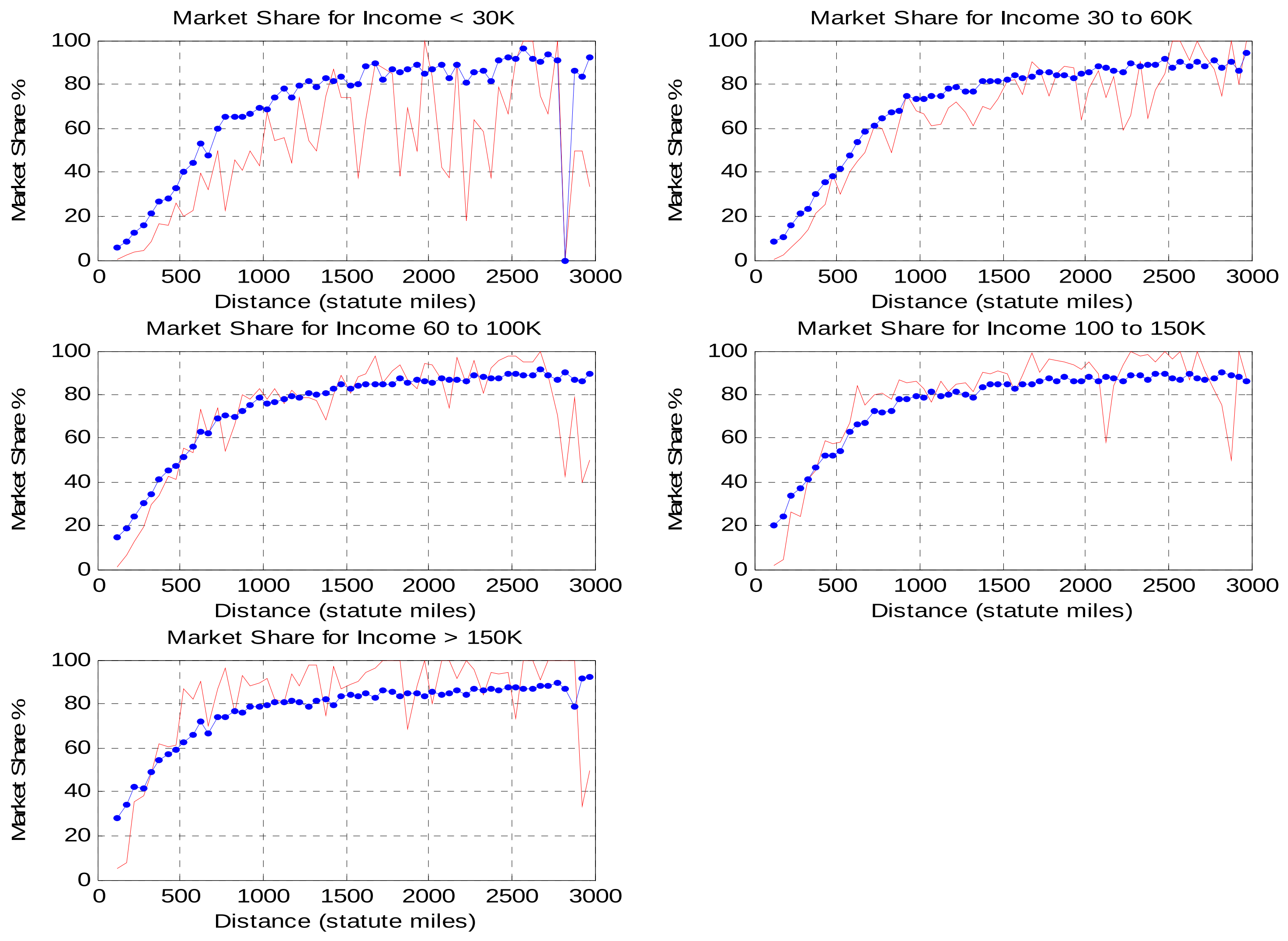

**Figure 7: Comparison of American Travel Survey and Mixed Logit Model Plots (Business Trips).**

## 3.5 CONCLUSIONS

A credible mode choice model and airport choice model has been developed to estimate market share for automobile and commercial airline modes between any pair of counties and airports in the United States. Given any county-to-county trip demand table the mode choice model can be used to estimate travel demand by automobile and commercial airline between all counties in the United States.

The model is unique in that is a first attempt to have a county-to-county nationwide choice model calibrated for the United States. The use of a NL model also means additional modes of transportation (such as rail and general aviation) can be integrated into the mode choice model with additional survey data. The model has been implemented in estimating demand for the automobile and commercial airline trips in the United States with satisfactory results. The current model with some simplifying assumptions has also been used to estimate demand for the Small Aircraft Transportation System (SATS) a new mode of air transportation being developed by NASA *(1)*.

In order to improve the model fit to the ATS for short trips in the range of hundred to five hundred miles. Virginia Tech has conducted four different personal travel surveys that are being used to supplement the ATS in order to improve the credibility of the model.
Travel demand estimates from the applied model could be useful to airlines, airport authorities and various Federal Agencies such as the Department of Transportation, the Federal Aviation Administration, and Federal Highway Administration.

## 3.6 RECOMMENDATIONS

The current process of data collection and collation of the ATS needs to be modified to make it more useful for research and decision support applications. Specifically, a mechanism needs to be developed to release information about origin and destination zip codes and airports used for travel without compromising privacy of survey respondents.

The zip code and station information is critical in estimating credible travel time and costs. The airport information is needed to improve and validate airport choice model assumptions, especially for MSA areas where it is likely more than three airports are actively used for commercial airline operations.

The release of this information will help in developing a more credible model in order to give decision makers a valuable planning tool than can be used to plan transportation infrastructure improvements in the United States.

## 3.7 REFERENCES

1. Trani, A. A, Baik, H., Swingle, H., Ashiabor. S. An Integrated Model for Studying Small Transportation System, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1850*, TRB, National Research Council, Washington, D.C., 2003, pp. 1-10.
2. Trani, A., Baik, H., Ashiabor, A., Swingle, H. and Wingrove, E. Transportation Systems Baseline Assessment Study, *Virginia SATSLab Report*, 2002.
3. Trani, A., Baik, H., Ashiabor, A., Swingle., A., Sheshadri, A., Murthy, K., and Hinze, N. Transportation Systems Analysis of Small Aircraft Transportation, *Final Report submitted to NASA Langley,* 2003.
4. Stopher, P., Prashker, J. Intercity Passenger Forecasting: The Use of Current Travel Forecasting Procedures, *Transportation Research Forum: Annual Meeting Proceedings*, 1976, pp 67-75.
5. Grayson, A. Disaggregate model of mode choice in intercity travel, In *Transportation Research Record: Journal of the Transportation Research Board, No. 385*, 1982, pp 36-42.
6. Morrison, S., Winston, C. An Econometric Analysis of the Demand for Intercity Passenger Transportation, *Research in Transportation Economics*, 1985.
7. Koppelman, F. S. Multidimensional model system for intercity travel choice behavior, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1241*, 1990, pp 1-8.
8. Koppelman, F. S., Kuah, G., and Hirsh, M. Review of Intercity Passenger Travel Demand Modeling: Mid 60's to the Mid 80's, *The Transportation Center,* Department of Civil Engineering, Northwestern University, 1984.
9. Koppelman, F. S., and Hirsh, M. Intercity passenger decision making: conceptual structure and data implications, In *Transportation Research Record: Journal of the Transportation Research Board, No. 1085*, 1986, pp 70-75.
10. McFadden, D. Conditional logit Analysis of Qualitative Choice Behavior, in P. Zarembka, *Frontiers in Econometrics*, Academic Press: New York, 1973, pp 105-142.
11. Luce, D. R. *Individual Choice Behavior*, New York: Wiley, 1959.
12. Ben-Akiva, M., and Lerman, S. *Discrete Choice Analysis – Theory and Application to Travel Demand*, MIT Press, Cambridge, MA, 1985.
13. Train, E. K. *Discrete Choice Methods with Simulation*, Cambridge University Press, Cambridge, UK, 2003.
14. McFadden, D. Modeling the choice of spatial location, In *Spatial Interaction Theory and Planning Models* (Edited by Karlqvist, A., Lundqvist, L., Snickars, F., and Weibull, J.), North-Holland, Amsterdam, 1978, pp 75-96.
15. Daly, A., and Zachary, S. Improved multiple choice models, In *Determinants of Travel Choice* (Edited by Hensher, D., Dalvi, M.), Saxon House, Sussex, 1978.
16. Vovsha, P. Cross-nested logit model: an application to mode choice in the Tel-Aviv metropolitan area, *Transportation Research Board, 76th Annual Meeting*, Washington DC. Paper #970387, 1997.

17. Bierlaire, M. Discrete choice models, In *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management* (Edited by Labbe, M., Laporte, G., Tanczos, K., Toint, P.), Springer-Verlag, Heidlberg, Germany, 1998, pp 203-227.

18. Small, K. Aproximate generalized extreme value models of discrete choice, *Journal of Econometrics*, Vol. 62, 1994, Issue 2, pp 351-382.

19. Bhat, C. Accommodating variations in responsiveness to level-of-service measures in travel mode choice modeling, *Transportation Research Part A*, 1998, Vol.32, No. 7, pp 495-507.

20. Chu, C. A paired combinatorial logit model for travel demand analysis, *Proceedings of the 5th World Conference on Transportation Research*, Vol. 4, 1989, pp 295-309.

21. Wen, C. and Koppelman, F.S. The generalized nested logit. *Transportation Research Part B*, Vol. 35, 2001, pp 627-641.

22. Steckel, J.H., and Vanhonacker, W.R. A heterogeneous conditional logit model of choice, *Journal of Business & Economic Statistics*, Vol. 6, No. 3, 1998, pp 381-389.

23. Bhat, C. A heteroskedastic extreme value model of intercity model choice, *Transportation Research Part B*, Vol. 29, 1995, pp 417-483.

24. Recker, W.W. Discrete choice with an oddball alternative, *Transportation Research Part B*, Vol. 29B, No. 3, 1995, pp 201-211.

25. McFadden, D., and Train, K. Mixed MNL models for discrete response, *Journal of Applied Econometrics*, Vol. 15, 2000, pp 447-470.

26. Byod, J.H., and Mellman, R.E. The effect of fuel economy standards on the U.S. automotive market: An hedonic demand analysis, *Transportation Research Part A*, Vol. 14A, 1980, pp 367-378.

27. Cardell, N.S. and Dunbar, F.C. Measuring the societal impacts of automobile downsizing, *Transportation Research Part A*, Vol. 14A, 1980, pp 423-434.

28. Brownstone, D. Train, K., Forecasting new product penetration with flexible substitution patterns, *Journal of Econometrics*, Vol. 89, 1999, pp 109-129.

29. Bhat, C., Castelar, S. A unified mixed logit framework for modeling revealed and stated preferences: Formulation and application to congestion pricing analysis in the San Francisco Bay area, *Transportation Research Part B*, Vol. 36, 2002, pp 577-669.

30. Hess, S., and Polak, J.W., Mixed logit modeling of airport choice in multi-airport regions, *Journal of Air Transport Management*, Vol. 11, 2005, pp 59-68.

31. Bureau of Transportation Statistics. *American Travel Survey: An overview of the survey design and methodology*, Bureau of Transportation Statistics, 1995.

32. SAS Institute, http://www.sas.com, version 9.1.3.

33. Microsoft Corporation. *MapPoint: Business Mapping and Data Visualization Software*, CD-ROM, 2004.

34. Official Airline Guide, CD-ROM, 2000.

35. Teodorovic, D. *Airline Operations Research*, Gordon and Breach Science Publishers, New York, 1998.

36. Bureau of Transportation Statistics. Airline Origin and Destination Survey (DB1B), http://www.transtats.bts.gov/, 2000.

ACKNOWLEDGEMENT
Thank to NASA for their support in developing this model.  Especially we want to thank Stuart Cooke (NASA), Jeff Viken (NASA) and Sam Dollyhigh (Swales Aerospace) for their constructive criticisms, comments and contributions to the model.

# 4 Development of a Family of Intercity Mode Choice Models for New Aviation Technologies in the U.S. Using Revealed and Stated Preference Surveys

Senanu Ashiabor
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 257-3830
Fax. (540) 231-7352
sashiabo@vt.edu

Antonio Trani
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 231-2362
Fax. (540) 231-7352
vuela@vt.edu

Hojong Baik
Via Department of Civil Engineering, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Tel. (540) 231-4418
Fax. (540) 231-7352
hbaik@vt.edu

## ABSTRACT

A family of mixed and nested logit random utility models was developed to study intercity mode choice behavior in the United States. The models were calibrated using a nationwide revealed preference survey (1995 American Travel Survey) and two stated preference surveys conducted by Virginia Tech at selected airports in the U.S. The focus of this paper is the ability of the models to estimate market share for the new category of Very Light Jet (VLJ) aircraft used in on-demand air taxi services. Analysis was performed to test the impact of using only the stated preference survey versus combining it with the revealed preference survey to predict VLJ air taxi demand. The models were also compared by model type, vis-à-vis nested versus mixed logit. The main explanatory variables in the utility functions are travel time and travel cost stratified by household income.

An airport choice model is integrated into the model choice model to estimate both the market share between any origin-destination pair and other modes of transportation, and the market share split between airports associated with the origin-destination pairs. The model estimates market share for automobile, commercial air and on-demand air taxi services.

The model has been integrated into a large-scale computer software travel demand framework called the Transportation Systems Analysis Model (TSAM) to estimate nationwide intercity travel demand flow between 3,091 counties in the US, 443 commercial service airports and more than 3,000 general aviation airports in the U.S. A pared down version of the model will be integrated into the National Strategy Simulator that the FAA uses for strategic level planning the aviation system.

Keywords: Mode Choice, Airport Choice, Very Light Jets, Stated Preference, Revealed Preference.

# 5 DEVELOPMENT OF AIRPORT CHOICE MODEL FOR GENERAL AVIATION OPERATIONS

**Hojong Baik, Senanu Ashiabor and Antonio Trani**

**Published in Transportation Research Record 1951**

## ABSTRACT

The General Aviation Airport Choice Model (GAACM) presented estimates General Aviation (GA) person-trips and number of aircraft operations through a set of airports given initial trip demand (GA person trips) from a set of counties. A pseudo-gravity model embedded in the model is used to distribute the inter-county person-trips to the set of airports. The inter-airport person-trips are then split into person-trips by aircraft type (single, multi and jet engine). To split the trips an attractiveness factor based on average occupancy, level of utilization, a distance distribution factor and number of operations of each aircraft type is developed. The person-trips by aircraft type are then converted to aircraft operations using occupancy factors for each aircraft type. The final model output is number of aircraft operations by each aircraft type in the form of three inter-airport trip tables. The GA operations estimated provide a means of assessing the impact of GA activities on the National Airspace System. The model output may be used to assess the viability of GA aircraft serving as a competitive mode of transportation for intercity travel.

# 6 SUMMARY AND CONCLUSIONS

## 6.1 SUMMARY OF UPDATES TO MODEL

Earlier calibrations of the model showed an overestimation of commercial air demand for short trips, especially overestimating demand for high income travelers (Figure 7 above). The overestimation of demand for short trips was temporarily corrected in applications of the model using a step function that scaled up the travel time for business trips, and travel cost for non-business trips up based on a trigger ratio. **Changes presented below have eliminated the need for a step function and significantly corrected the overestimation of commercial air demand for short trips.**

Currently the scale parameter for the business model is set to be different for both nests while that for non-business has the same scale parameter for both nests. A review of the literature revealed it is more appropriate to set the nested logit scale parameter to be the same for both nests. Using the same scale parameter across both nests makes it easier to introduce new alternatives in the absence of a survey. Finally there is the issue of biased results when an individual choice model is calibrated with aggregate data.

In the process of making changes to correct the above, other issues with the model were discovered and addressed. The major ones are a change in the way schedule delay and airport-to-airport commercial air travel time was computed. Also the airport-to-airport fare structure was significantly modified based on inconsistencies identified with the way airlines reported fares in DB1B.

The initial supposition was that the aggregate nature of the model was affecting the calibration process. The initial model was calibrated using nationwide average values for the variables (travel time and travel cost) in the utility function. The need to average is due to the limited level of geographical detail in the American Travel Survey. The smallest unit of geographical detail in the survey is the Census Bureau Metropolitan Statistical Area (MSA) while the smallest unit of detail in TSAM is a county.

For calibration purposes, smoothed national level market share profiles over distance and by income group were created using the ATS. Based on these the travel times and costs had to be averaged along the same dimensions.  The travel times and costs were estimated by running TSAM for all the 3091 counties and averaging the variables at the national level along selected dimensions.  For a given dimension the variables are also weighted by the number of trips between the counties.  The weighting is to ensure the variables were biased towards the origin destination pairs with more trips based on the trip distribution.

The dimensions used in TSAM 4.1 were distance, region, and income group.  The distance range was from 100 miles to 3000 miles at 50 mile intervals.  There are four regions in the model.

1. Trips beginning in an MSA and ending in an MSA
2. Trips beginning in an MSA and ending in a non-MSA
3. Trips beginning in a non-MSA and ending in an MSA
4. Trips beginning in a non-MSA and ending in a non-MSA

The five income groups in the model are

1. Traveler with household income less than $30,0000
2. Traveler with household income from $30,000 to $60,000
3. Traveler with household income from $60,000 to $100,000
4. Traveler with household income from $100,000 to $150,000
5. Traveler with household income greater than $150,000

Base on the above, the average automobile travel time tables have the dimension in Table 8.

**Table 8: Schematic of Automobile Travel Times**

| Range (miles) | Midpiont (miles) | Auto Travel Time |
|---|---|---|
| 100 - 150 | 125 | |
| 150 - 200 | 126 | |
| 200 - 250 | 127 | |
| 250 - 300 | 128 | |
| 300 - 350 | 129 | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| ….. | ….. | |
| 2750 - 2800 | 2775 | |
| 2800 - 2850 | 2825 | |
| 2850 - 2900 | 2875 | |
| 2900 -2950 | 2925 | |
| 2950 -3000 | 2975 | |

The average automobile travel cost has an additional dimension for household income because cost are adjusted based on travel party size, which varies by household income. The schematic is shown in Table 9 below.

**Table 9: Schematic of Automobile Travel Cost**

| Range (miles) | Midpiont (miles) | <$30K | $30K to 60K | $60K to 100K | $100K to 150K | > 150K |
|---|---|---|---|---|---|---|
| 100 - 150 | 125 | | | | | |
| 150 - 200 | 126 | | | | | |
| 200 - 250 | 127 | | | | | |
| 250 - 300 | 128 | | | | | |
| 300 - 350 | 129 | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| 2750 - 2800 | 2775 | | | | | |
| 2800 - 2850 | 2825 | | | | | |
| 2850 - 2900 | 2875 | | | | | |
| 2900 -2950 | 2925 | | | | | |
| 2950 -3000 | 2975 | | | | | |

The commercial air tables are similar to the respective tables for automobile but with an additional dimension for possible routes. The schematic for travel cost for three prototyped routes are shown in Table 10 below.

**Table 10: Schematic of Commercial Air Travel Cost**

**Closest airport to Closest Airport**

**Closest Airport to Cheapest Airport**

**Cheapest Airport to Closest Airport**

| **Range (miles)** | **Midpiont (miles)** | **<$30K** | **$30K to 60K** | **$60K to 100K** | **$100K to 150K** | **> 150K** |
|---|---|---|---|---|---|---|
| 100 - 150 | 125 | | | | | |
| 150 - 200 | 126 | | | | | |
| 200 - 250 | 127 | | | | | |
| 250 - 300 | 128 | | | | | |
| 300 - 350 | 129 | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| ….. | ….. | | | | | |
| 2750 - 2800 | 2775 | | | | | |
| 2800 - 2850 | 2825 | | | | | |
| 2850 - 2900 | 2875 | | | | | |
| 2900 -2950 | 2925 | | | | | |
| 2950 -3000 | 2975 | | | | | |

### 6.1.1 Changes to Travel Time and Travel Cost Computations during Calibration

Initially aggregate data was used to calibrate an individual choice model. It is well documented in the economics literature that individual choice models calibrated using aggregate data yield biased estimates. Based on this we tried to use more of the disaggregate information in the ATS. So instead of the smoothed national level market share curves the actual ATS records were used for calibration. This necessitated a more detailed travel time computation.

Say, for a trip record in the ATS between two MSA's we can identify all the counties in those MSA's. So computed the travel times and costs between all the counties for those MSA's and use that instead of the average travel time at the national level. The travel times and cost computed in this manner are likely to be closer to that faced by the decision maker in their choice process compare to the national average. This is especially true for commercial air where times and cost change significantly for different airports. After several changes to the utility function it was realized that due to the small sample size for long trips the ATS the market shares by income group interact significantly (see Figure 8). Thus the model has travelers with household income less than $30,000 annually switching to commercial air travel from automobile faster than travelers with household income greater $150,000 annually.

**Based on this finding the approach of using more disaggregate modeling approach has been abandoned and the <u>smoothed market share curves</u> are now being used.** However in creating the travel times the detailed approach is used before summing up to the national level. **One major change that has been introduced is a penalty for overnight stay for commercial air travel, that was not in any of the earlier versions of TSAM**. This significantly increased the commercial air travel time, especially for short trips and contributed in lowering the demand helping eliminate the need to use a step function.

Hence the dimensions for travel time and cost tables remain the same as in 6.1. The actual values of travel times and cost are however different from the model in Section 3 due to the changes mentioned earlier. These are:

1. Change in the way schedule delay is computed
2. Change for average commercial air travel time to weighted commercial air travel time (between airports)
3. Changed in commercial air fares due to reclassification of fares

4. Change in the airport choice rules (elimination of non-hubs competing with large hubs)

### 6.1.2 Additional Changes

***Automobile Cost-per-Mile:*** Based on reviewer comments of the paper submitted to the Transportation Research Board the cost-per-mile for automobile non-business trips was reduced from 37 cents to 10 cents. The value of ten cents was judgmentally selected based on the current fuel cost. This value can be changed at anytime and it does not affect the model significantly as long as the same value is maintained in both the calibration process and during application.

***Fares for Non-Hubs:*** It was also noticed that for specific origin-destination pairs when applying the model in TSAM some specific routes with non-hub airports have a high market share than competing routes that have large hubs. This is due to the fact that in such cases the non-hub values are from the Harris model and are only based on the Harris model restricted fares. The large hubs have records in DB1B for both restricted and unrestricted fares. The average fare for the large hubs therefore tend to be lower and in the process the non-hubs origin destination pair gets assigned a higher market share due to its low fare.

A two pronged approach is being used to address this. In the first set we have created Harris Model unrestricted fare curves from DB1B. As shown in Figure 9 the fares from this model are much higher than the restricted fares. When running TSAM the fares for airports that do not have unrestricted fares are adjusted upwards using the fare from the unrestricted model. It is believed this would help ameliorate the current problem. Dr. Baik and colleagues are currently designing a more sophisticated approach to address the issue, that might involving sampling data from the internet.

The form of the utility function has also been changed after testing various structures as explained in the next section.

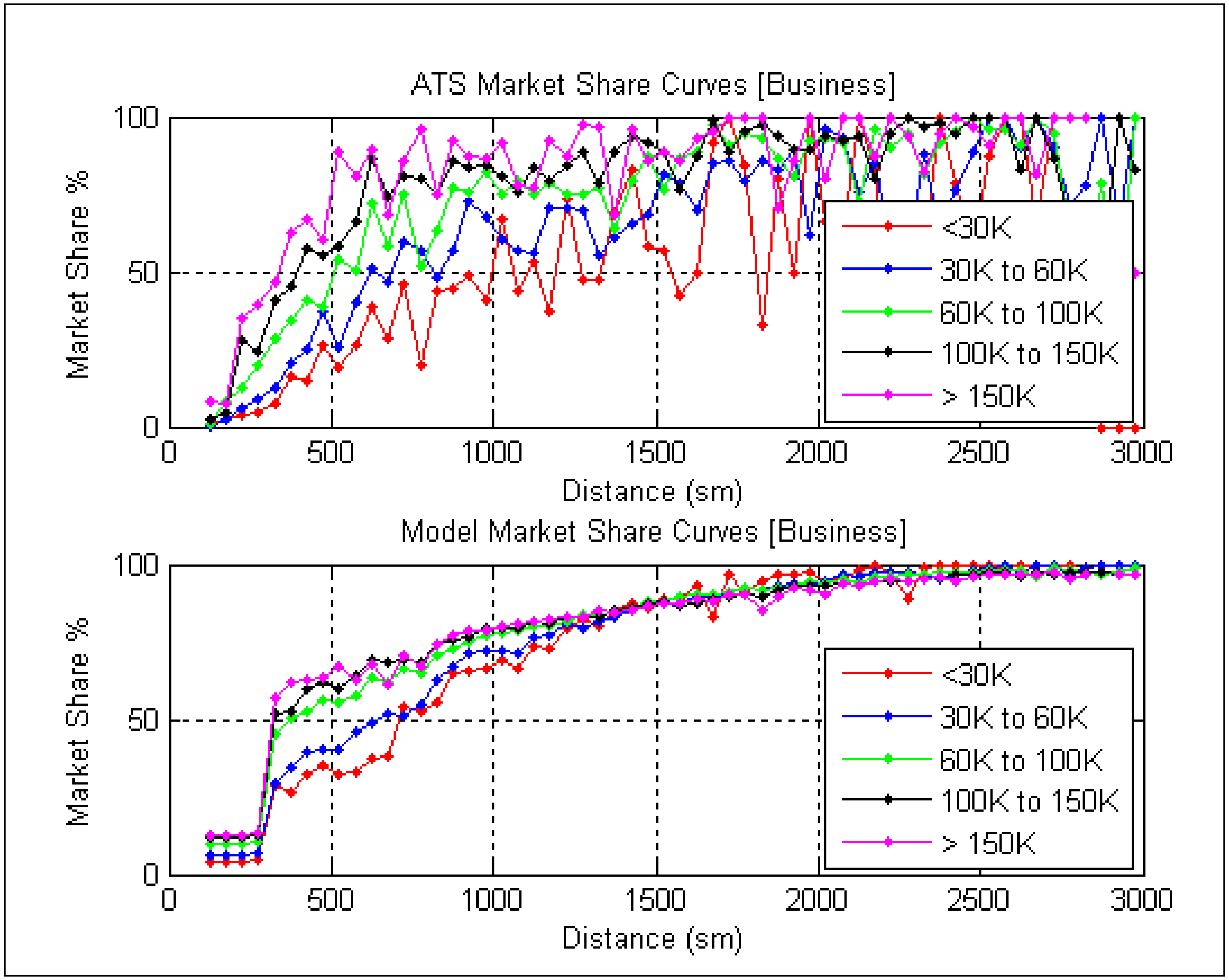

**Figure 8: Calibrating Using Unsmoothed ATS Market Shares for Business (by Income Group).**

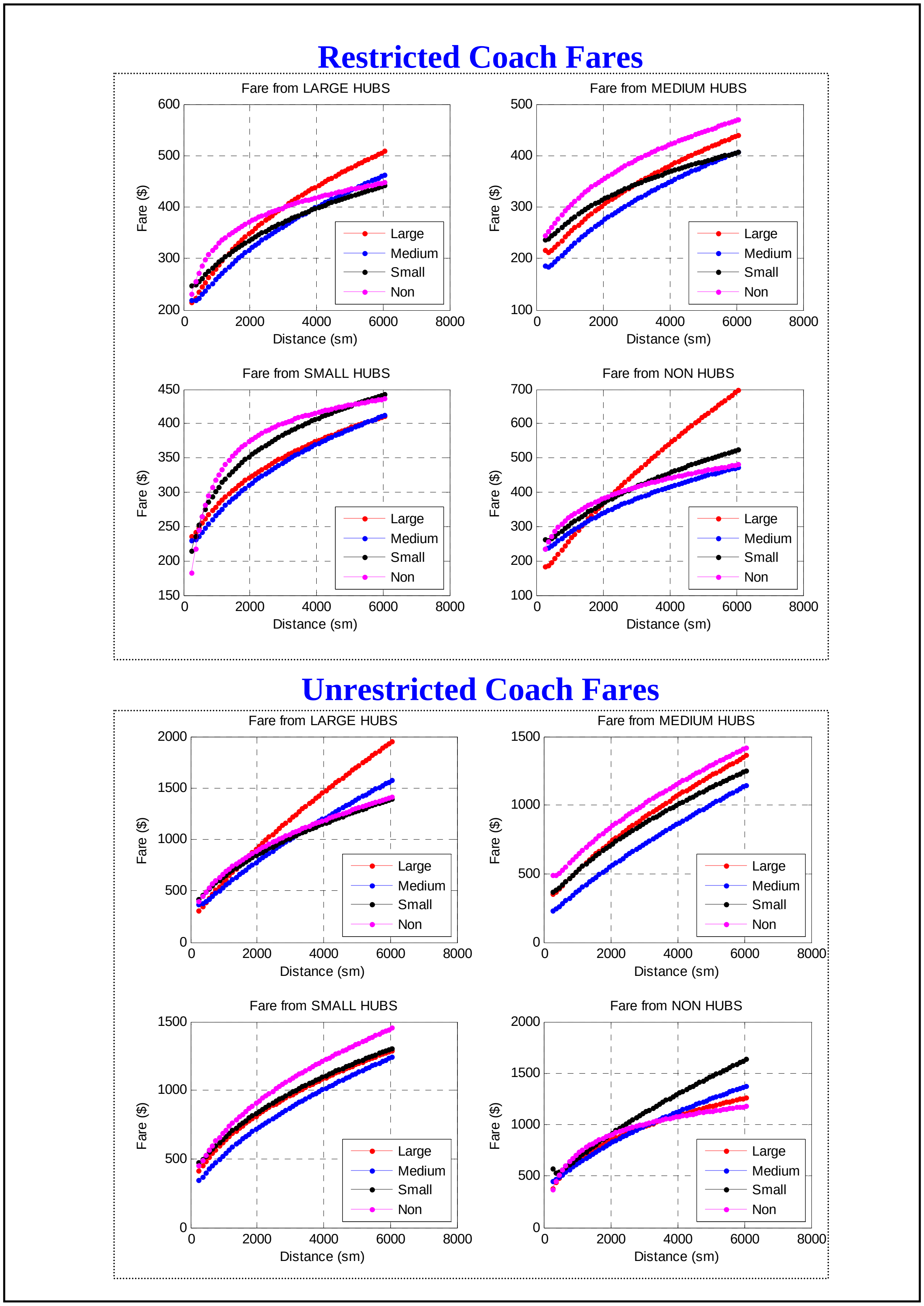

Figure 9: Comparison of Restricted and Unrestricted Fares from DB1B

### 6.1.3 Changes to the Utility Function

Using the smoothed curves we tested the form of the utility function of several dimensions. More than twenty different models were calibrated and finally the form that yielded the best fit to the ATS market share curves had travel time and cost for each income group. The form of the utility function in the model is as shown in Equation 32.

$$U_{ij}^{kl} = \alpha_1^l Travel\ Time_{ij}^{kl} + \alpha_2^l\ Travel\ \ Cost_{ij}^{kl} \tag{32}$$

Where:

$Travel\ Time_{ij}^{kl}$ is the travel time from origin $i$ to destination $j$ using mode of transportation $k$ for a traveler in income group $l$
$\alpha_1^l$ are the five travel time coefficients for travelers in income group $l$
$Travel\ \ Cost_{ij}^{kl}$ is the travel cost from origin $i$ to destination $j$ using mode of transportation $k$ for a traveler with household income $k$
$\alpha_1^l$ are the five travel cost coefficients for travelers income group $l$

Table 4 has the coefficients of the calibrated nested and mixed logit models using the utility equation above and the changes discussed. The model results are discussed in the Section 6.2.

**Table 11: Updated Calibrated Model Coefficients**

| Variable | NESTED LOGIT MODEL |  |  |  | MIXED LOGIT MODEL |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  | BUSINESS |  | NON-BUSINESS |  | BUSINESS |  | NON-BUSINESS |  |
|  | Parameter Estimate | P-values | Parameter Estimate | P-values | Parameter Estimate | P-values | Parameter Estimate | P-values |
| Travel Time (<30K) | -0.0269 | <.0001 | -0.1219 | <.0001 | 0.0007 | <.0001 | -0.2098 | <.0001 |
| Travel Time (30 to 60K) | -0.0776 | <.0001 | -0.1329 | <.0001 | 0.0005 | <.0001 | -0.2028 | <.0001 |
| Travel Time (60 to 100K) | -0.1084 | <.0001 | -0.1508 | <.0001 | 0.0007 | <.0001 | -0.1493 | <.0001 |
| Travel Time (100 to 150K) | -0.2087 | <.0001 | -0.1827 | <.0001 | 0.0013 | <.0001 | -0.2211 | <.0001 |
| Travel Time (>1500K) | -0.2608 | <.0001 | -0.2444 | <.0001 | 0.0019 | <.0001 | -0.2382 | <.0001 |
| Travel Cost (<30K) | -0.0309 | <.0001 | -0.0275 | <.0001 | 0.0001 | <.0001 | -0.0567 | <.0001 |
| *Scale Parameter Cost (<30K)* |  |  |  |  | 0.0015 | 0.948 | -0.0327 | <.0001 |
| Travel Cost (30 to 60K) | -0.0235 | <.0001 | -0.0274 | <.0001 | 0.0001 | <.0001 | -0.0501 | <.0001 |
| *Scale Parameter Cost (30 to 60K)* |  |  |  |  | 0.0009 | 0.9922 | 0.0281 | <.0001 |
| Travel Cost (60 to 100K) | -0.0183 | <.0001 | -0.0249 | <.0001 | 0.0001 | <.0001 | -0.0278 | <.0001 |
| *Scale Parameter Cost (60 to 100K)* |  |  |  |  | 0.0007 | 0.9591 | 0.0147 | <.0001 |
| Travel Cost (100 to 150K) | -0.0117 | <.0001 | -0.0223 | <.0001 | 0.0001 | <.0001 | -0.0297 | <.0001 |
| *Scale Parameter Cost (100 to 150K)* |  |  |  |  | 0.0009 | 0.9827 | -0.0180 | <.0001 |
| Travel Cost (>150K) | -0.0094 | <.0001 | -0.0191 | <.0001 | 0.0001 | <.0001 | -0.0197 | <.0001 |
| *Scale Parameter Cost (>150K)* |  |  |  |  | 0.0009 | 0.8472 | -0.0120 | <.0001 |
| *Nested Logit Scale Parameter* | 0.3401 | <.0001 | 0.4409 | <.0001 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| *R-square (Adjusted Estrella)* | *84.30* |  | *97.55* |  | *82.58* |  | *97.59* |  |
| *Log-Likelihood* | *-55,803* |  | *-28,812* |  | *-57,941* |  | *-28,640* |  |

## 6.2 COMMENTS AND CONCLUSIONS

The updated model has both a time and cost coefficient for each of the five income groups. Conceptually this is akin to having five separate models. The nested logit model has only one scale parameter for both nests. In the mixed logit models each cost coefficient has two parameters, the mean and scale parameter. Since the normal distribution was used the scale parameter is analogous to the standard deviation. All the mean coefficients in both the nested and mixed logit models are negative. Indicating given any two transportation alternatives if cost is held constant travelers prefer the faster mode, and if time is held constant travelers prefer the cheaper mode.

Comparing the two business models, the nested logit r-square of 84.3% relative to 82.5% for the mixed logit indicates the nested logit out performs the mixed logit. The log-likelihood value of -55,803 for the nested logit is also better than the -57,941 for the mixed logit. The closer the log-likelihood is to zero the better the model. A closer examination of the p-values shows that all the scale parameters of the business mixed logit model are >0.01. This indicates a poor fit for the mixed logit model. A formulation with scale parameters varied for time did not yield an improved fit either. These results do not necessarily mean that nested logit models are better than mixed logit models. What is happening is that the new formulation with variations in both travel time and cost by income is so efficient at capturing the behavioral information related to business travelers that estimating scale parameters for either time or cost does not yield any additional information.

As can be seen in the case of non-business models the mixed logit slightly out performs the nested logit. It's r-square is higher by 0.04 and the log likelihood of -28,640 is better than the value of -28,812 for the nested logit. Given than all the p-values for all the estimates in the non-business model are <0.0001. It is the model on which a comparison of the performance of nested and mixed logit should be made. Clearly, the mixed logit out performs the nested logit model. However for the data at hand using a nested logit does not offer much in terms of increased explanatory power and hence it is recommended that the nested logit model be used for modeling intercity travel demand using the 1995 ATS dataset, and smoothed market share curves.

# 7 APPENDIX A: MATLAB AND SAS CODE FOR CALIBRATING MIXED AND NESTED LOGIT MODELS

## 7.1 PRE-PROCESS TRAVEL COSTS FOR COMMERCIAL AIR

#### *7.1.1 Prepare tables for Harris Model Curve Fitting*

---------------------------------------------------------------------------------------------------------------

```
function prepareTables_for_CurveFitting()

%-----------------------------------------------------------
% This function read in DB1B data creates files by airport
% type
% The funtion also deletes all records for airport pairs with
% less than 5 records in DB1B
%-----------------------------------------------------------

clear all; clc; tic

Install_Dir = 'C:\Program Files\TSAM\4.2\data\mode_choice\input';
Load_Dir    = 'D:\Mode Choice\Calibration_February_07\DB1B_Y2000_$5000_Limit';
Year        = '2000';

load ([Load_Dir, '\TotalRecords_CoachClass_2000_2way'])
load ([Install_Dir, '\airportlist_oag.mat']);
load ([Install_Dir, '\CA_Hub_Type'])
load ([Install_Dir, '\distA2A_CA_443'])

% load ('C:\Program Files\TSAM 3.8.4\data\mode_choice\input\Airport_List_CA')
% numberOf_CA_Apts = size(Airport_List_CA, 2);
% for cnt_arpt = 1 : numberOf_CA_Apts
%     airportlist_oag(cnt_arpt,1:3) = Airport_List_CA(cnt_arpt).OAG_ID;
% end % for cnt_arpt = 1 : numberOf_CA_Apts
% save ('C:\Program Files\TSAM 3.8.4\data\mode_choice\input\airportlist_oag', 'airportlist_oag')


for cntFareType = 2 : 2

   if cntFareType == 1
       fareType = 'BusinessClass';
       fid_read = fopen([Load_Dir, '\DB1B_Ticket_Market_Coupon_Merged_', Year,
'_BusinessClass.txt'], 'r'); % Input File
       save_Dir = ([Load_Dir, '\SAS Data\BusinessClass']);
   else
       fareType = 'CoachClass';
       fid_read  = fopen([Load_Dir, '\DB1B_Ticket_Market_Coupon_Merged_', Year, '_CoachClass.txt'],
'r'); % Input File
```

```
    save_Dir = ([Load_Dir, '\SAS Data\CoachClass']);
end % if cntFareType == 1

% save used records in Coach and Business Class files
fid_L2L = fopen([save_Dir, '\DB1B_', fareType, 'FareData_L2L.txt'], 'w'); % Output File
fid_L2M = fopen([save_Dir, '\DB1B_', fareType, 'FareData_L2M.txt'], 'w'); % Output File
fid_L2S = fopen([save_Dir, '\DB1B_', fareType, 'FareData_L2S.txt'], 'w'); % Output File
fid_L2N = fopen([save_Dir, '\DB1B_', fareType, 'FareData_L2N.txt'], 'w'); % Output File

fid_M2L = fopen([save_Dir, '\DB1B_', fareType, 'FareData_M2L.txt'], 'w'); % Output File
fid_M2M = fopen([save_Dir, '\DB1B_', fareType, 'FareData_M2M.txt'], 'w'); % Output File
fid_M2S = fopen([save_Dir, '\DB1B_', fareType, 'FareData_M2S.txt'], 'w'); % Output File
fid_M2N = fopen([save_Dir, '\DB1B_', fareType, 'FareData_M2N.txt'], 'w'); % Output File

fid_S2L = fopen([save_Dir, '\DB1B_', fareType, 'FareData_S2L.txt'], 'w'); % Output File
fid_S2M = fopen([save_Dir, '\DB1B_', fareType, 'FareData_S2M.txt'], 'w'); % Output File
fid_S2S = fopen([save_Dir, '\DB1B_', fareType, 'FareData_S2S.txt'], 'w'); % Output File
fid_S2N = fopen([save_Dir, '\DB1B_', fareType, 'FareData_S2N.txt'], 'w'); % Output File

fid_N2L = fopen([save_Dir, '\DB1B_', fareType, 'FareData_N2L.txt'], 'w'); % Output File
fid_N2M = fopen([save_Dir, '\DB1B_', fareType, 'FareData_N2M.txt'], 'w'); % Output File
fid_N2S = fopen([save_Dir, '\DB1B_', fareType, 'FareData_N2S.txt'], 'w'); % Output File
fid_N2N = fopen([save_Dir, '\DB1B_', fareType, 'FareData_N2N.txt'], 'w'); % Output File

% skip the first line
if(fid_read >= 0)
    firstLine = fgets(fid_read);
end % if(fid_read >= 0)

counter = 0;
while ~feof(fid_read)

    if mod(counter, 100000) == 0
        disp(['Processed Records: ' num2str(counter), ' Elapsed Time: ', num2str(toc/60), ' Minutes']);
    end % if mod(counter, 50000) == 0

    line = fgets(fid_read);
    % Field1: ItinID
    % Field2: Origin
    % Field3: Dest
    % Field4: ItinYield
    % Field5: Passengers
    % Field6: ItinFare
    % Field7: MilesFlown
    % Field8: FirstOfFareClass
    % Field9: RPCarrier
    double_line = textscan(line, '%s%s%s%f%f%f%f%s%s', 'delimiter', ',');
    counter = counter + 1;

    originAirport   = double_line{2}; % Reads the origin
    destnAirport    = double_line{3}; % Reads the destination
```

```
        ItinYield       = double_line{4}; % Reads the itineary yield
        %         Passenger       = double_line{5}; % Reads for passenger
        %         ItinFare        = double_line{6}; % Reads the itineary fare
        milesFlown      = double_line{7}; % Reads the itineary fare

        table_OrgApt      = strmatch(originAirport, airportlist_oag);  % Determines the table_OrgApt airport
in the O-D table
        if isempty(table_OrgApt)
           continue;
        end % if (isempty(table_OrgApt)

        table_DesApt  = strmatch(destnAirport,  airportlist_oag); % Determines the table_DesApt airport in
the O-D table
        if isempty(table_DesApt)
           continue;
        end % if (table_DesApt)

        numberOfRecords = TotalRecords_CoachClass(table_OrgApt, table_DesApt);

        if numberOfRecords > 4

           orgHubTypeIndex = CA_Hub_Type(table_OrgApt);
           desHubTypeIndex = CA_Hub_Type(table_DesApt);
           a2a_dist        = distA2A_CA(table_OrgApt,table_DesApt)*2;

           if orgHubTypeIndex == 1 && desHubTypeIndex == 1
              fprintf(fid_L2L, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 1 && desHubTypeIndex == 2
              fprintf(fid_L2M, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 1 && desHubTypeIndex == 3
              fprintf(fid_L2S, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 1 && desHubTypeIndex == 4
              fprintf(fid_L2N, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 2 && desHubTypeIndex == 1
              fprintf(fid_M2L, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 2 && desHubTypeIndex == 2
              fprintf(fid_M2M, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 2 && desHubTypeIndex == 3
              fprintf(fid_M2S, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 2 && desHubTypeIndex == 4
              fprintf(fid_M2N, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 3 && desHubTypeIndex == 1
              fprintf(fid_S2L, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 3 && desHubTypeIndex == 2
              fprintf(fid_S2M, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 3 && desHubTypeIndex == 3
              fprintf(fid_S2S, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 3 && desHubTypeIndex == 4
              fprintf(fid_S2N, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
           elseif orgHubTypeIndex == 4 && desHubTypeIndex == 1
              fprintf(fid_N2L, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
```

```
            elseif orgHubTypeIndex == 4 && desHubTypeIndex == 2
                fprintf(fid_N2M, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
            elseif orgHubTypeIndex == 4 && desHubTypeIndex == 3
                fprintf(fid_N2S, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
            elseif orgHubTypeIndex == 4 && desHubTypeIndex == 4
                fprintf(fid_N2N, '%d %5.6f %d\n', a2a_dist,ItinYield,milesFlown);
            else
                disp('Record Not Saved')
                counter
                pause
            end % if orgHubTypeIndex == 1 && desHubTypeIndex == 1
        end % if numberOfRecords > 4

    end % while ~feof(fid_read)
    fclose('all');
end % for cntFareType = 2 : 2

return
---------------------------------------------------------------------------------------------------------------
```

### *7.1.2 Fit DB1B Data to Harris Model*

--------------------------------------------------------------------------------------------------------------

```
function fitData_withHarrisModel()

primary_Dir = 'D:\Mode Choice\Calibration_February_07\DB1B_Y2000_$5000_Limit\SAS
Data\CoachClass\';
for cnt_orgAptType = 1 : 4
   cnt_orgAptType
   for cnt_desAptType = 1 : 4
      rowIndex = ((cnt_orgAptType - 1) * 4) + cnt_desAptType;
      if cnt_orgAptType == 1 && cnt_desAptType == 1
         load([primary_Dir, 'DB1B_CoachClassFareData_L2L.txt']);
         dist    = DB1B_CoachClassFareData_L2L(:,3);
         fpm     = DB1B_CoachClassFareData_L2L(:,2); clear DB1B_CoachClassFareData_L2L
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 1 && cnt_desAptType == 2
         load([primary_Dir, 'DB1B_CoachClassFareData_L2M.txt']);
         dist    = DB1B_CoachClassFareData_L2M(:,3);
         fpm     = DB1B_CoachClassFareData_L2M(:,2); clear DB1B_CoachClassFareData_L2M
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 1 && cnt_desAptType == 3
         load([primary_Dir, 'DB1B_CoachClassFareData_L2S.txt']);
         dist    = DB1B_CoachClassFareData_L2S(:,3);
         fpm     = DB1B_CoachClassFareData_L2S(:,2); clear DB1B_CoachClassFareData_L2S
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 1 && cnt_desAptType == 4
         load([primary_Dir, 'DB1B_CoachClassFareData_L2N.txt']);
         dist    = DB1B_CoachClassFareData_L2N(:,3);
         fpm     = DB1B_CoachClassFareData_L2N(:,2); clear DB1B_CoachClassFareData_L2N
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 2 && cnt_desAptType == 1
         load([primary_Dir, 'DB1B_CoachClassFareData_M2L.txt']);
         dist    = DB1B_CoachClassFareData_M2L(:,3);
         fpm     = DB1B_CoachClassFareData_M2L(:,2); clear DB1B_CoachClassFareData_M2L
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 2 && cnt_desAptType == 2
         load([primary_Dir, 'DB1B_CoachClassFareData_M2M.txt']);
         dist    = DB1B_CoachClassFareData_M2M(:,3);
         fpm     = DB1B_CoachClassFareData_M2M(:,2); clear DB1B_CoachClassFareData_M2M
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 2 && cnt_desAptType == 3
         load([primary_Dir, 'DB1B_CoachClassFareData_M2S.txt']);
         dist    = DB1B_CoachClassFareData_M2S(:,3);
         fpm     = DB1B_CoachClassFareData_M2S(:,2); clear DB1B_CoachClassFareData_M2S
         betaIni = [-1.5;0.26;0.4];
      elseif cnt_orgAptType == 2 && cnt_desAptType == 4
         load([primary_Dir, 'DB1B_CoachClassFareData_M2N.txt']);
         dist    = DB1B_CoachClassFareData_M2N(:,3);
         fpm     = DB1B_CoachClassFareData_M2N(:,2); clear DB1B_CoachClassFareData_M2N
```

```
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 3 && cnt_desAptType == 1
            load([primary_Dir, 'DB1B_CoachClassFareData_S2L.txt']);
            dist    = DB1B_CoachClassFareData_S2L(:,3);
            fpm     = DB1B_CoachClassFareData_S2L(:,2); clear DB1B_CoachClassFareData_S2L
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 3 && cnt_desAptType == 2
            load([primary_Dir, 'DB1B_CoachClassFareData_S2M.txt']);
            dist    = DB1B_CoachClassFareData_S2M(:,3);
            fpm     = DB1B_CoachClassFareData_S2M(:,2); clear DB1B_CoachClassFareData_S2M
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 3 && cnt_desAptType == 3
            load([primary_Dir, 'DB1B_CoachClassFareData_S2S.txt']);
            dist    = DB1B_CoachClassFareData_S2S(:,3);
            fpm     = DB1B_CoachClassFareData_S2S(:,2); clear DB1B_CoachClassFareData_S2S
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 3 && cnt_desAptType == 4
            load([primary_Dir, 'DB1B_CoachClassFareData_S2N.txt']);
            dist    = DB1B_CoachClassFareData_S2N(:,3);
            fpm     = DB1B_CoachClassFareData_S2N(:,2); clear DB1B_CoachClassFareData_S2N
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 4 && cnt_desAptType == 1
            load([primary_Dir, 'DB1B_CoachClassFareData_N2L.txt']);
            dist    = DB1B_CoachClassFareData_N2L(:,3);
            fpm     = DB1B_CoachClassFareData_N2L(:,2); clear DB1B_CoachClassFareData_N2L
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 4 && cnt_desAptType == 2
            load([primary_Dir, 'DB1B_CoachClassFareData_N2M.txt']);
            dist    = DB1B_CoachClassFareData_N2M(:,3);
            fpm     = DB1B_CoachClassFareData_N2M(:,2); clear DB1B_CoachClassFareData_N2M
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 4 && cnt_desAptType == 3
            load([primary_Dir, 'DB1B_CoachClassFareData_N2S.txt']);
            dist    = DB1B_CoachClassFareData_N2S(:,3);
            fpm     = DB1B_CoachClassFareData_N2S(:,2); clear DB1B_CoachClassFareData_N2S
            betaIni = [-1.5;0.26;0.4];
        elseif cnt_orgAptType == 4 && cnt_desAptType == 4
            load([primary_Dir, 'DB1B_CoachClassFareData_N2N.txt']);
            dist    = DB1B_CoachClassFareData_N2N(:,3);
            fpm     = DB1B_CoachClassFareData_N2N(:,2); clear DB1B_CoachClassFareData_N2N
            betaIni = [-1.5;0.26;0.4];
        end % if cnt_orgAptType == 1 && cnt_orgAptType == 1

        harris_coef(rowIndex,:) = nlinfit(dist,fpm,@harrisModel,betaIni);
        clear dist fpm

    end % for cnt_desAptType = 1 : 4
end % for cnt_orgAptType = 1 : 4

save ([primary_Dir, 'harris_coef_businessClass'], 'harris_coef')
```

return

### 7.1.3 *Create Plots of Harris Model Curves using Coefficients*

------------------------------------------------------------------------------------------------------------------

```
function plotHarrisModelCurves()

%-----------------------------------------
% Plot Harris Model Curve using coefficients
%-----------------------------------------

clear all; clc

distArray_2way = 2 * [125 : 50 : 3050];
fareValues     = zeros(length(distArray_2way),4,4);
yieldValues    = fareValues;

load_dir = 'D:\Mode Choice\Calibration_February_07\DB1B_Y2000_$5000_Limit\';
load ([load_dir, '\SAS Data\CoachClass\harris_coef_coachClass'])
saveDir    = [load_dir, '\CoachClass'];
ticketType = '_CoachClass';

for cnt_orgAptType = 1 : 4
   if     cnt_orgAptType == 1; orgName = '_largeHub';
   elseif cnt_orgAptType == 2; orgName = '_mediumHub';
   elseif cnt_orgAptType == 3; orgName = '_smallHub';
   elseif cnt_orgAptType == 4; orgName = '_nonHub';
   end % if     orgName = '_largeHub';

   for cnt_desAptType = 1 : 4
      if     cnt_desAptType == 1; desName = '_largeHub';
      elseif cnt_desAptType == 2; desName = '_mediumHub';
      elseif cnt_desAptType == 3; desName = '_smallHub';
      elseif cnt_desAptType == 4; desName = '_nonHub';
      end % if     desName = '_largeHub';
      rowIndex = (cnt_orgAptType - 1) * 4 + cnt_desAptType;

      plotharris_coef = harris_coef(rowIndex, :);
      yieldValues(:,cnt_orgAptType,cnt_desAptType) = 1 ./ (plotharris_coef(1) + plotharris_coef(2) .*
distArray_2way.^plotharris_coef(3));
      fareValues(:,cnt_orgAptType,cnt_desAptType)  = distArray_2way' .*
yieldValues(:,cnt_orgAptType,cnt_desAptType);

      %          if cnt_tktType == 2
      %              figure;
      %              load (strcat(saveDir, '\fareData', ticketType, orgName, desName, '_2way'))
      %              plot(distArray_2way, yieldValues(:,cnt_orgAptType,cnt_desAptType), '-r',
fareData(:,1), fareData(:,2), '.b');
      %              grid on; xlabel('Distance (sm)'); ylabel('Yield ($/mile');
      %          end % if cnt_tktType == 2

   end % for cnt_desAptType = 1 : 4
end % for cnt_orgAptType = 1 : 4

save ([saveDir, 'yieldValues'], 'yieldValues')
save ([saveDir, 'fareValues'], 'fareValues')

% Plot fares FROM each of the 4 airport types
figure;
subplot(2,2,1); plot(distArray_2way, fareValues(:,1,1), '.-r', distArray_2way, fareValues(:,1,2), '.-b', ...
    distArray_2way, fareValues(:,1,3), '.-k', distArray_2way, fareValues(:,1,4), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare from LARGE HUBS')
subplot(2,2,2); plot(distArray_2way, fareValues(:,2,1), '.-r', distArray_2way, fareValues(:,2,2), '.-b', ...
    distArray_2way, fareValues(:,2,3), '.-k', distArray_2way, fareValues(:,2,4), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare from MEDIUM HUBS')
subplot(2,2,3); plot(distArray_2way, fareValues(:,3,1), '.-r', distArray_2way, fareValues(:,3,2), '.-b', ...
    distArray_2way, fareValues(:,3,3), '.-k', distArray_2way, fareValues(:,3,4), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare from SMALL HUBS')
subplot(2,2,4); plot(distArray_2way, fareValues(:,4,1), '.-r', distArray_2way, fareValues(:,4,2), '.-b', ...
    distArray_2way, fareValues(:,4,3), '.-k', distArray_2way, fareValues(:,4,4), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare from NON HUBS')

saveas (gcf, [saveDir, 'FarePlot_FROM'], 'bmp')


% Plot fares TO each of the 4 airport types
figure;
subplot(2,2,1); plot(distArray_2way, fareValues(:,1,1), '.-r', distArray_2way, fareValues(:,2,1), '.-b', ...
    distArray_2way, fareValues(:,3,1), '.-k', distArray_2way, fareValues(:,4,1), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare to LARGE HUBS')
subplot(2,2,2); plot(distArray_2way, fareValues(:,1,2), '.-r', distArray_2way, fareValues(:,2,2), '.-b', ...
    distArray_2way, fareValues(:,3,2), '.-k', distArray_2way, fareValues(:,4,2), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare to MEDIUM HUBS')
subplot(2,2,3); plot(distArray_2way, fareValues(:,1,3), '.-r', distArray_2way, fareValues(:,2,3), '.-b', ...
    distArray_2way, fareValues(:,3,3), '.-k', distArray_2way, fareValues(:,4,3), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare to SMALL HUBS')
subplot(2,2,4); plot(distArray_2way, fareValues(:,1,4), '.-r', distArray_2way, fareValues(:,2,4), '.-b', ...
    distArray_2way, fareValues(:,3,4), '.-k', distArray_2way, fareValues(:,4,4), '.-m'); grid on
legend('Large', 'Medium','Small','Non','location','southeast'); xlabel('Distance (sm)'); ylabel('Fare ($)');
title('Fare to NON HUBS')

saveas (gcf, [saveDir, 'FarePlot_TO'], 'bmp')

return
```

-----------------------------------------------------------------------------------------------------------

### *7.1.4 Fill-up Coach Fare Table cells with Harris Model Curve Values*

```
------------------------------------------------------------------------------------------------------------------

function preAllocate_A2A_travelCosts()

%-------------------------------------------------------
% Pre-allocate Airport-to-Airport fare table
%
% 1) Load table pre-filled with fares from DB1B
% 2) Search through table and replace empty cells
%    in coach tables with Harris model curve estimate
%-------------------------------------------------------

clc; tic
Install_Dir = 'C:\Program Files\TSAM\4.2';
Load_Dir    = 'D:\Mode Choice\Calibration_February_07';

% load mat files
%-----------------
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalPax_BusinessClass_2000_2way'])     % Airport to
airport business class fare table (419x419)
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalRevenue_BusinessClass_2000_2way'])  % Airport
to airport business class fare table (419x419)
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalRecords_BusinessClass_2000_2way'])

load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalPax_CoachClass_2000_2way'])     % Airport to
airport business class fare table (419x419)
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalRevenue_CoachClass_2000_2way']) % Airport to
airport business class fare table (419x419)
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\TotalRecords_CoachClass_2000_2way'])

load (strcat(Install_Dir, '\data\mode_choice\input\distA2A_CA_443'))   % Airport to airport distance table
(443x443)
load (strcat(Install_Dir, '\data\mode_choice\input\airport_hub_type'))


highIndex = find(TotalRecords_BusinessClass > 0 & TotalRecords_BusinessClass < 5);
TotalRevenue_BusinessClass(highIndex) = 0;
A2A_CA_BusinessClass_Fare_2000 = TotalRevenue_BusinessClass ./ TotalPax_BusinessClass;

nanValues = isnan(A2A_CA_BusinessClass_Fare_2000);
A2A_CA_BusinessClass_Fare_2000(nanValues) = 0;

highIndex = find(TotalRecords_CoachClass > 0 & TotalRecords_CoachClass < 5);
TotalRevenue_CoachClass(highIndex) = 0;
A2A_CA_CoachClass_Fare_2000 = TotalRevenue_CoachClass ./ TotalPax_CoachClass;

nanValues = isnan(A2A_CA_CoachClass_Fare_2000);
A2A_CA_CoachClass_Fare_2000(nanValues) = 0;
```

```
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_CoachClass_Fare_2000'),
'A2A_CA_CoachClass_Fare_2000')
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_BusinessClass_Fare_2000'),
'A2A_CA_BusinessClass_Fare_2000')

%-------------------------------------------
% check to make sure diagonals sum to zero
%-------------------------------------------
sum_diag_BUSINESS = sum(diag(A2A_CA_BusinessClass_Fare_2000));
if sum_diag_BUSINESS > 0; disp('Error with BUSINESS diagonals'); end
sum_diag_COACH = sum(diag(A2A_CA_CoachClass_Fare_2000));
if sum_diag_COACH > 0; disp('Error with COACH diagonals'); end

%-------------------------------------------
% check if all elements > 0
%-------------------------------------------
zero_business = find(A2A_CA_BusinessClass_Fare_2000 < 0);
if zero_business > 0; disp('Error with business zero values'); end
zero_coach = find(A2A_CA_CoachClass_Fare_2000 < 0);
if zero_coach > 0; disp('Error with coach values'); end

%------------------------------------
% load Harris Model Coefficients
%------------------------------------
dist_1way = 100 : 50 : 3000;
load ([Load_Dir, '\DB1B_Y2000_$5000_Limit\SAS Data\CoachClass\harris_coef_coachClass'])
A2A_CA_CoachFare_Index_2000 = zeros(443,443,2);

% loop for origin airports
counter_1 = 0;
problemAirportPairs = [];
counter3 = 0;
counter4 = 0;
for orgCnt_airportID = 1 : size(A2A_CA_BusinessClass_Fare_2000, 2)
   if mod(orgCnt_airportID, 10) == 0;
      disp(['Origin Airport: ', num2str(orgCnt_airportID), ' time: ', num2str(toc) ]);
   end % if mod(orgCnt_airportID, 10) == 0;

   % loop for destination airports
   for desCnt_airportID = 1 : size(A2A_CA_BusinessClass_Fare_2000, 2)

      % skip diagonals (orgin == destination)
      if orgCnt_airportID == desCnt_airportID
         A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID, desCnt_airportID) = 0;
         A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) = 0;
      else
         % extract fares from table
         bussClassFare_2way  = A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID,
desCnt_airportID);
         coachClassFare_2way = A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID);
```

```
        % if there is no fare from DB1B
        if bussClassFare_2way <= 0 || coachClassFare_2way <= 0

          counter4 = counter4 + 1;

          %-------------------------------------------------------------
          % use route distance to estimate coach fare
          interAirportDist_2way = 2 * distA2A_CA(orgCnt_airportID, desCnt_airportID); %Coach Fare
from model

          if interAirportDist_2way < 180
            interAirportDist_2way = 180;
            counter3 = counter3 + 1;
          end % if interAirportDist_2way < 180

          orgArpt_hubType = airport_hub_type(orgCnt_airportID);
          desArpt_hubtype = airport_hub_type(desCnt_airportID);
          rowIndex = ((orgArpt_hubType - 1) * 4) + desArpt_hubtype;

          coeff_a = harris_coef(rowIndex, 1);
          coeff_b = harris_coef(rowIndex, 2);
          coeff_c = harris_coef(rowIndex, 3);

          %-------------------------------------------------------------
          % CASE 1: NO Bussiness Fare, NO Coach Fare
          if bussClassFare_2way <= 0 && coachClassFare_2way <= 0
            % Use Coach fare and set Business fare to zero
            A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
interAirportDist_2way / (coeff_a + coeff_b * interAirportDist_2way ^ coeff_c); % regression coach fare
            A2A_CA_CoachFare_Index_2000(orgCnt_airportID, desCnt_airportID,1) = 1;

            % CASE 2: NO Bussiness Fare, Coach Fare
          elseif bussClassFare_2way <= 0 && coachClassFare_2way > 0
            % Set Business fare to zero, and use Coach fare
            A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID, desCnt_airportID)  = 0;

            % CASE 3: Bussiness Fare, NO Coach Fare
          elseif bussClassFare_2way > 0 && coachClassFare_2way <= 0
            % Use Coach fare from regression
            A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
interAirportDist_2way / (coeff_a + coeff_b * interAirportDist_2way ^ coeff_c); % regression coach fare
            A2A_CA_CoachFare_Index_2000(orgCnt_airportID, desCnt_airportID,1) = 1;

            % Reset Coach fare if > Business fare
            if A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) >
bussClassFare_2way
              %A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
bussClassFare_2way * 0.7;
              A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID);
              A2A_CA_CoachFare_Index_2000(orgCnt_airportID, desCnt_airportID,1) = 1;
```

```
                end % if coachClassFare_2way > bussClassFare_2way
            end % if bussClassFare_2way == -999 && coachClassFare_2way == -999
            %------------------------------------------------------------

        elseif coachClassFare_2way > bussClassFare_2way * 1.1
            counter_1 = counter_1 + 1;
            if (coachClassFare_2way > 200 || coachClassFare_2way > bussClassFare_2way * 1.25)
                bussClassFare_2way = coachClassFare_2way;
                A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
bussClassFare_2way;
                A2A_CA_CoachFare_Index_2000(orgCnt_airportID, desCnt_airportID,2) = 2;
                problemAirportPairs(counter_1,:) = [orgCnt_airportID desCnt_airportID
coachClassFare_2way bussClassFare_2way 1];
            else
                bussClassFare_2way = coachClassFare_2way;
                A2A_CA_BusinessClass_Fare_2000(orgCnt_airportID, desCnt_airportID) =
bussClassFare_2way;
                A2A_CA_CoachFare_Index_2000(orgCnt_airportID, desCnt_airportID,2) = 3;
                problemAirportPairs(counter_1,:) = [orgCnt_airportID desCnt_airportID
coachClassFare_2way bussClassFare_2way 2];
            end % if (coachClassFare_2way > 200 || coachClassFare_2way > bussClassFare_2way * 1.25)
        end % if bussClassFare_2way == -999 || coachClassFare_2way == -999

    end % if orgCnt_airportID ~= desCnt_airportID

    if A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) < 0
        ddd = 9;
    end % if A2A_CA_CoachClass_Fare_2000(orgCnt_airportID, desCnt_airportID) < 0

  end % for desCnt_airportID = 1 : size(A2A_CA_TravelTime, 2)
end % for orgCnt_airportID = 1 : size(A2A_CA_TravelTime, 2)

%--------------------------------------------
% check to make sure diagonals sum to zero
%--------------------------------------------
sum_diag_BUSINESS = sum(diag(A2A_CA_BusinessClass_Fare_2000));
if sum_diag_BUSINESS > 0; disp('Error with BUSINESS diagonals'); end
sum_diag_COACH = sum(diag(A2A_CA_CoachClass_Fare_2000));
if sum_diag_COACH > 0; disp('Error with COACH diagonals'); end

%--------------------------------------------
% check if all elements > 0
%--------------------------------------------
zero_business = find(A2A_CA_BusinessClass_Fare_2000 < 0);
if zero_business > 0; disp('Error with business zero values'); end
zero_coach = find(A2A_CA_CoachClass_Fare_2000 < 0);
if zero_coach > 0; disp('Error with coach values'); end

%--------------------------------------------
% rename and save files
%--------------------------------------------
```

```
A2A_CA_BusinessClassFare_DB1B_2000     = A2A_CA_BusinessClass_Fare_2000;
A2A_CA_CoachClassFare_DB1B_Filled_2000 = A2A_CA_CoachClass_Fare_2000;
A2A_CA_CoachFare_Index_DB1B_2000       = A2A_CA_CoachFare_Index_2000;
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_BusinessClassFare_DB1B_2000'),
'A2A_CA_BusinessClassFare_DB1B_2000')
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_CoachClassFare_DB1B_Filled_2000'),
'A2A_CA_CoachClassFare_DB1B_Filled_2000')
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_CoachFare_Index_DB1B_2000'),
'A2A_CA_CoachFare_Index_DB1B_2000')
save (strcat(Install_Dir, '\data\mode_choice\input\problemAirportPairs'), 'problemAirportPairs')

return
```

## 7.2 Pre-Process Travel Times for Commercial Air

### *7.2.1 Load OAG Airport to Airport Table and Fill Table with Regression Curve* **(Regression curve created in** A2A_TravelTime_Regression.m **file )**

```
------------------------------------------------------------------------------------------------------------------

function preAllocate_A2A_travelTimes()

%-----------------------------------------
% Part 1: (A2A_TravelTime_Regression.m)
%---------
% Use Weighted Airport-to-Airport travel times
% from Official Airline Guide (OAG) to create travel
% regression of travel time over distance
%
% Part 2:
%---------
% Fill empty cells with regression values
%
% Part 3:
%---------
% Save data
%-----------------------------------------

clc; tic

Install_Dir = 'C:\Program Files\TSAM\4.2';

load (strcat(Install_Dir, '\data\mode_choice\input\Weighted_travel_time')) % Inter-airporrt travel times in
hours (685x685)
load (strcat(Install_Dir, '\data\mode_choice\input\schedule_delay_new'))
load (strcat(Install_Dir, '\data\mode_choice\input\airport_hub_type'))
load (strcat(Install_Dir, '\data\mode_choice\input\distA2A_CA_443'))       % Airport to airport distance
table (443x443)

A2A_CA_TravelTime      = travel_time;
A2A_CA_Schedule_Delay = schedule_delay_new;
airport_List_CA        = airport_hub_type;
% A2A_TravelTime_Regression(travel_time,       schedule_delay,        airport_hub_type);
%
A2A_TravelTime_Regression(A2A_CA_TravelTime, A2A_CA_Schedule_Delay, airport_List_CA,
distA2A_CA, Install_Dir);

% load (strcat(Install_Dir, '\data\mode_choice\input\A2A_scheduleDelay_Mode'))
load (strcat(Install_Dir, '\data\mode_choice\input\A2A_Coefficients_time'))
load (strcat(Install_Dir, '\data\mode_choice\input\hub_indeces'))

% loop for origin airports
numberRows = length(travel_time);
```

```
for orgAirport_ID = 1 : numberRows

   % display counter
   if mod(orgAirport_ID, 10) == 0; disp(['Origin Airport: ', num2str(orgAirport_ID), ' time: ',
num2str(toc) ]); end

   % loop for destination airports
   for desAirport_ID = 1 : size(travel_time, 2)

      % check if origin and destination airports are the same
      if orgAirport_ID ~= desAirport_ID

         % Apply regression model if airport-to-airport travel time is non-positive
         if travel_time(orgAirport_ID, desAirport_ID) <= 0

            % regression based model from travel time table
            %-----------------------------------------------
            A2A_DISTANCE    = distA2A_CA(orgAirport_ID, desAirport_ID);
            orgArpt_hubType = airport_hub_type(orgAirport_ID);
            desArpt_hubtype = airport_hub_type(desAirport_ID);
            intercept_arpt = A2A_Coefficients_time(desArpt_hubtype, 1, orgArpt_hubType);
            slope_arpt     = A2A_Coefficients_time(desArpt_hubtype, 2, orgArpt_hubType);

            if orgAirport_ID == 4 || desAirport_ID == 4
               travel_time(orgAirport_ID, desAirport_ID) = -9999;
            else
               travel_time(orgAirport_ID, desAirport_ID) = intercept_arpt + slope_arpt *
A2A_DISTANCE;
            end % if orgAirport_ID == 4 || desAirport_ID == 4

         end % if travel_time(orgAirport_ID, desAirport_ID) < 0

         if schedule_delay_new(orgAirport_ID, desAirport_ID) <= 0
            schedule_delay_new(orgAirport_ID, desAirport_ID) =
A2A_scheduleDelay_Mode(orgArpt_hubType, desArpt_hubtype);
         end % if schedule_delay(orgAirport_ID, desAirport_ID) <= 0

      end % if orgAirport_ID ~= desAirport_ID

   end % for desAirport_ID = 1 : size(travel_time, 2)

end % for orgAirport_ID = 1 : size(travel_time, 2)

% check to make sure diagonals sum to zero
sum_diag = sum(diag(travel_time));
if sum_diag > 0
   disp('Error with diagonals')
end % if sum_diag > 0

A2A_CA_TravelTime_no_SD_1way = travel_time;
```

```
A2A_CA_travelTime_no_SD_2way = travel_time + travel_time'; % roundtrip airport-to-airport travel
time (no schedule delay)
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_TravelTime_no_SD_1way'),
'A2A_CA_TravelTime_no_SD_1way')
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_travelTime_no_SD_2way'),
'A2A_CA_travelTime_no_SD_2way')

A2A_CA_Schedule_Delay_1way = schedule_delay_new;
A2A_CA_Schedule_Delay_2way = schedule_delay_new + schedule_delay_new'; % roundtrip airport-to-
airport schedule delay
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_Schedule_Delay_1way'),
'A2A_CA_Schedule_Delay_1way')
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_Schedule_Delay_2way'),
'A2A_CA_Schedule_Delay_2way')

A2A_CA_travelTime_with_SD_2way = A2A_CA_travelTime_no_SD_2way +
A2A_CA_Schedule_Delay_2way; % Airport-to-airport travel time (with schedule delay)
save (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_travelTime_with_SD_2way'),
'A2A_CA_travelTime_with_SD_2way')

return
```
--------------------------------------------------------------------------------------------------------------

### *7.2.2 Regression Curve of Travel Time (over distance)*

```
------------------------------------------------------------------------------------------------------------

function [A2A_distance_1way, A2A_scheduleDelay_1way] = A2A_TravelTime_Regression(travel_time,
schedule_delay, airport_hub_type, distA2A_CA, Install_Dir)

%-----------------------------------------------------
% (1)Group the 443 commercial service airports by airport type and
% create regression equations for each airport type pair for travel
% time
%
% (2) Compute 'mode' instead of 'mean' for schedule delay
%
%  Hub Identifiers
%       Large Hub  => 1
%       Medium Hub => 2
%       Small Hub  => 3
%       Non Hub    => 4
%
% Created by: Senanu Ashiabor
% Date:       March 8, 2006
% Modified:   March 9, 2006
%-----------------------------------------------------

% set-up distance range for plotting
xrange = 100:50:3000;

A2A_distance_1way(1,1).HubType       = 0;
A2A_pureTravelTime_1way(1,1).HubType = 0;
A2A_scheduleDelay_1way(1,1).HubType  = 0;

% loop for origin airport type
for org_type = 1 : 4

   % extract all rows in 'distance' and 'time' tables for selected airport type
   org_loop_Index = find(airport_hub_type == org_type);
   org_table_time      = travel_time(org_loop_Index, :);
   org_table_Scd_Delay = schedule_delay(org_loop_Index, :);
   org_table_distance  = distA2A_CA(org_loop_Index, :);

   % loop for destination airport type
   for des_type = 1 : 4

      % extract all columns in 'distance' and 'time' tables for selected airport type
      des_loop_Index = find(airport_hub_type == des_type);
      des_table_time      = org_table_time(:, des_loop_Index);
      des_table_Scd_Delay = org_table_Scd_Delay(:, des_loop_Index);
      des_table_distance  = org_table_distance(:, des_loop_Index);
```

```
%------------------------------------------------------------------------------
% reshape tables to vectors
%------------------------------------
[nRows nCols] = size(des_table_distance);
distance_x  = reshape(des_table_distance, 1, nRows*nCols)'; % Distance data
[nRows nCols] = size(des_table_Scd_Delay);
sDelay_y  = reshape(des_table_Scd_Delay, 1, nRows*nCols)'; % Schedule Delay data
[nRows nCols] = size(des_table_time);
time_y  = reshape(des_table_time, 1, nRows*nCols)'; % Time data

%------------------------------------------------------------------------------
% delete Distance < 0
zero_distance = find(distance_x <= 0);
distance_x(zero_distance) = [];
sDelay_y(zero_distance)   = [];
time_y(zero_distance)     = [];

% delete Travel time < 0
negative_time = find(time_y <= 0);
distance_x(negative_time) = [];
sDelay_y(negative_time)   = [];
time_y(negative_time)     = [];

% save data
A2A_distance_1way(org_type, des_type).HubType       = distance_x;
A2A_pureTravelTime_1way(org_type, des_type).HubType = time_y;
A2A_scheduleDelay_1way(org_type, des_type).HubType  = sDelay_y;

%------------------------------------------------------------------------------
% initialize regression arrays
xValues  = [ones(size(distance_x)) distance_x];
yValues_time   = time_y;
yValues_sDelay = sDelay_y;

if min(yValues_sDelay) < 0
   disp('negative Schedule Delay!!!')
   pause
end % if min(yValues_sDelay) < 0

%------------------------------------------------------------------------------
% preform regression and save data
coefficients_time   = xValues \ yValues_time;
coefficients_sDelay = xValues \ yValues_sDelay;

% create forecasting line and save in matrix
yrange_time   = coefficients_time(1) + coefficients_time(2) * xrange;
yrange_sDelay = coefficients_sDelay(1) + coefficients_sDelay(2) * xrange;

all_data_time(:, des_type, org_type)   = yrange_time';
all_data_sDelay(:, des_type, org_type) = yrange_sDelay';
```

```
        A2A_Coefficients_time(des_type, :, org_type)   = coefficients_time';
        A2A_Coefficients_sDelay(des_type, :, org_type) = coefficients_sDelay';
        %------------------------------------------------------------------------------

    end % for des_type = 1 : 4
end % for org_type = 1 : 4

for orgLoop = 1 : 4
%     plot1 = figure;
%     plot2 = figure;

    for desLoop = 1 :4
        dist_data   = A2A_distance_1way(orgLoop, desLoop).HubType;
        time_data   = A2A_pureTravelTime_1way(orgLoop, desLoop).HubType;
        sDelay_data = A2A_scheduleDelay_1way(orgLoop, desLoop).HubType;

        if desLoop == 1
            lineType = '.r';
        elseif desLoop == 2
            lineType = '.b';
        elseif desLoop == 3
            lineType = '.g';
        elseif desLoop == 4
            lineType = '.k';
        end % if desLoop == 1

%         figure(plot1);
%         subplot(2,2,desLoop); plot(dist_data, sDelay_data, lineType); hold on; grid on
        mean_SD    = mean(sDelay_data);
        stdDev_SD  = std(sDelay_data);
        count_SD   = length(sDelay_data);

        outliers_SD        = abs(sDelay_data - mean_SD(ones(count_SD,1),:)) > 3 *
stdDev_SD(ones(count_SD,1),:);
        number_of_outliers = sum(outliers_SD);
        sDelay_data(any(outliers_SD,2),:) = [];
        dist_data(any(outliers_SD,2),:)   = [];
%         figure(plot2);
%         subplot(2,2,desLoop); plot(dist_data, sDelay_data, lineType); hold on; grid on

        %         A2A_scheduleDelay_Mode(orgLoop, desLoop) = mode(sDelay_data);
    end % for desLoop = 1 :4
end % for orgLoop = 1 : 4
%------------------------------------------------------------------------------------------

for i = 1 : 2
    if i == 1
        all_data = all_data_time;
        ylabel_string = '1-way Airport-to-Airport Travel Time (Hours)';
    else
        all_data = all_data_sDelay;
        ylabel_string = '1-way Airport-to-Airport Schedule Delay (Hours)';
    end


    % plot from each hub type to all other airport types
    figure;
    subplot(2,2,1); plot(xrange, all_data(:,1,1), '-r', xrange, all_data(:,2,1), '-.r', xrange, all_data(:,3,1), '--r',
xrange, all_data(:,4,1), 'or'); hold on; grid on
    title('LARGE HUB to ALL HUBS'); xlabel('Airport-to-Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,2); plot(xrange, all_data(:,1,2), '-b', xrange, all_data(:,2,2), '-.b', xrange, all_data(:,3,2), '--b',
xrange, all_data(:,4,2), 'ob'); hold on; grid on
    title('MEDIUM HUB to ALL HUBS'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,3); plot(xrange, all_data(:,1,3), '-k', xrange, all_data(:,2,3), '-.k', xrange, all_data(:,3,3), '--k',
xrange, all_data(:,4,3), 'ok'); hold on; grid on
    title('SMALL HUB to ALL HUBS'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,4); plot(xrange, all_data(:,1,4), '-m', xrange, all_data(:,2,4), '-.m', xrange, all_data(:,3,4), '--
m', xrange, all_data(:,4,4), 'om'); hold on; grid on
    title('NON HUB to ALL HUBS'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')

    for cntOrigin = 1 : 4
        plotName = figure;
        for cntDestn = 1 : 4
            if cntDestn == 1
                lineType = '.r';
            elseif cntDestn == 2
                lineType = '.b';
            elseif cntDestn == 3
                lineType = '.g';
            elseif cntDestn == 4
                lineType = '.k';
            end % if cntDestn == 1
            tripDistance  = A2A_distance_1way(cntOrigin, cntDestn).HubType;
            scheduleDelay = A2A_scheduleDelay_1way(cntOrigin, cntDestn).HubType;
            figure(plotName); subplot(2,2,cntDestn);
            plot(tripDistance, scheduleDelay, lineType); grid on
        end % for cntDestn = 1 : 4
    end % for cntOrigin = 1 : 4


    % plot from all other airport types to each hub type
    %-------------------------------------------------------
    figure;
    subplot(2,2,1); plot(xrange, all_data(:,1,1), '-r', xrange, all_data(:,1,2), '-.r', xrange, all_data(:,1,3), '--r',
xrange, all_data(:,1,4), 'or'); hold on; grid on
    title('ALL HUBS to LARGE HUB'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,2); plot(xrange, all_data(:,2,1), '-b', xrange, all_data(:,2,2), '-.b', xrange, all_data(:,2,3), '--b',
xrange, all_data(:,2,4), 'ob'); hold on; grid on
    title('ALL HUBS to MEDIUM HUB'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,3); plot(xrange, all_data(:,3,1), '-k', xrange, all_data(:,3,2), '-.k', xrange, all_data(:,3,3), '--k',
xrange, all_data(:,3,4), 'ok'); hold on; grid on
    title('ALL HUBS to SMALL HUB'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
    subplot(2,2,4); plot(xrange, all_data(:,4,1), '-m', xrange, all_data(:,4,2), '-.m', xrange, all_data(:,4,3), '--
m', xrange, all_data(:,4,4), 'om'); hold on; grid on
    title('ALL HUBS to NON HUB'); xlabel('Airport to Airport Distance (sm)'); ylabel(ylabel_string);
    legend('Large Hub', 'Medium Hub', 'Small Hub', 'Non Hub')
end % for i = 1 : 2

% save in TSAM Directory
save ([Install_Dir, '\data\mode_choice\input\A2A_Coefficients_time'], 'A2A_Coefficients_time')
save ([Install_Dir, '\data\mode_choice\input\airport_hub_type'], 'airport_hub_type')

largeHub_Index  = find(airport_hub_type == 1);
mediumHub_Index = find(airport_hub_type == 2);
smallHub_Index  = find(airport_hub_type == 3);
nonHub_Index    = find(airport_hub_type == 4);
save ([Install_Dir, '\data\mode_choice\input\hub_indeces'], 'largeHub_Index', 'mediumHub_Index',
'smallHub_Index', 'nonHub_Index')

return
```
--------------------------------------------------------------------------------------------------------------------

### *7.2.3 Plots of Travel Time (for Validation)*

```
------------------------------------------------------------------------------------------------------------

function A2A_FareTable_Plots()

%-----------------------------------------------------
% (1) Load airport-to-airport fare tables and create
% plots to validate table
%
%  Hub Identifiers
%       Large Hub  => 1
%       Medium Hub => 2
%       Small Hub  => 3
%       Non Hub    => 4
%
% Created by: Senanu Ashiabor
% Date:       March 9, 2006
% Modified:
%-----------------------------------------------------

clear all; clc
Install_Dir = 'C:\Program Files\TSAM 3.8';

% load files
%--------------
load (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_TravelCost_Business_2000'))    % Airport to
airport business class fare table (419x419)
load (strcat(Install_Dir, '\data\mode_choice\input\A2A_CA_TravelCost_NonBusiness_2000')) % Airport
to airport coach class fare table (419x419)
load (strcat(Install_Dir, '\data\mode_choice\input\distA2A_CA_443'))     % Airport to airport distance
table (443x443)
load (strcat('D:\Mode Choice\Mixed Logit Model\Create Input Travel Time and Cost
Tables\hub_indeces'))

zerosBus = find(A2A_CA_TravelCost_Business_2000 == -999);
A2A_CA_TravelCost_Business_2000(zerosBus) = 0;
zerosNonBus = find(A2A_CA_TravelCost_NonBusiness_2000 == -999);
A2A_CA_TravelCost_NonBusiness_2000(zerosNonBus) = 0;

fare_1way(1,1).HubType = 0;
dist_1way(1,1).HubType = 0;
distanceTable = distA2A_CA;

% loop for trip purpose 1-> business, 2->nonbusiness
for tripType = 1 : 2
tripType
   % load fare tables
   if tripType == 1
      fareTable = A2A_CA_TravelCost_Business_2000;
```

```
    businessFigure1 = figure;
    businessFigure2 = figure;
else
    fareTable = A2A_CA_TravelCost_NonBusiness_2000;
    coachFigure1 = figure;
    coachFigure2 = figure;
end % if tripType == 1

% loop for origin airport type
for orgArpType = 1 : 4
    orgArpType
    if orgArpType == 1      orgArray = largeHub_Index;
    elseif orgArpType == 2  orgArray = mediumHub_Index;
    elseif orgArpType == 3  orgArray = smallHub_Index;
    elseif orgArpType == 4  orgArray = nonHub_Index;
    end % if orgArpType == 1

    % loop for destination airport type
    for desArpType = 1 : 4
        if desArpType == 1      desArray = largeHub_Index;
            lineType = '.r';
        elseif desArpType == 2  desArray = mediumHub_Index;
            lineType = '.b';
        elseif desArpType == 3  desArray = smallHub_Index;
            lineType = '.g';
        elseif desArpType == 4  desArray = nonHub_Index;
            lineType = '.k';
        end % if desArpType == 1

        % reshape fare and distance arrays to vectors
        extractFare_2way    = fareTable(orgArray, desArray);
        [nRows nCols]       = size(extractFare_2way);
        reshape_extractFare_2way  = reshape(extractFare_2way, 1, nRows*nCols)';

        extractDistance_1way = distanceTable(orgArray, desArray);
        [nRows nCols]        = size(extractDistance_1way);
        reshape_extractDistance_1way = reshape(extractDistance_1way, 1, nRows*nCols)';

        % delete fares <= 0
        zeroFare = find(reshape_extractFare_2way <= 0);
        reshape_extractFare_2way(zeroFare) = [];
        reshape_extractDistance_1way(zeroFare) = [];

        % delete non-intercity trips
        Dist_100 = find(reshape_extractDistance_1way < 100);
        reshape_extractFare_2way(Dist_100)     = [];
        reshape_extractDistance_1way(Dist_100) = [];

        dist_2way(orgArpType, desArpType).HubType  = 2 * reshape_extractDistance_1way;
        fare_2way(orgArpType, desArpType).HubType  = reshape_extractFare_2way;
```

```
            fare_per_mile = reshape_extractFare_2way ./ (2 * reshape_extractDistance_1way);
            if tripType == 1
                figure(businessFigure1);    subplot(2,2,orgArpType)
                plot(reshape_extractDistance_1way, reshape_extractFare_2way, lineType); hold on; grid on
                figure(businessFigure2);    subplot(2,2,orgArpType)
                plot(reshape_extractDistance_1way, fare_per_mile, lineType); hold on; grid on
            else
                figure(coachFigure1);   subplot(2,2,orgArpType)
                plot(reshape_extractDistance_1way, reshape_extractFare_2way, lineType); hold on; grid on
                figure(coachFigure2);   subplot(2,2,orgArpType)
                plot(reshape_extractDistance_1way, fare_per_mile, lineType); hold on; grid on
            end % if tripType == 1
        end % for desArpType = 1 : 4
    end % for orgArpType = 1 : 4

    if tripType == 1
        save('businessfare_2way_struct', 'fare_2way');
    else
        save('coachfare_2way_struct', 'fare_2way');
    end % if tripType == 1

end % for cntOrgApt_ID = 1 : size(A2A_CA_TravelTime, 2)
```
------------------------------------------------------------------------------------------------------------------

## 7.3 Origin County Airport Choice for Commercial Air

### *7.3.1 Select Airports within 200 miles of Origin County*

```
-------------------------------------------------------------------------------------------------------------
function Create_MapPoint_Input_Airport_Set()
%-----------------------------------------------------------------------------
% This function will import the standard CA airport set.
% A text file containing the airport ID, longitude and latitude
% needs to be provided. The text file must be delimited by a tab
%
% the text file is processed with Microsoft MapPoint/VB
% the output from MapPoint is then called by
% PostProcess_MapPoint_C2A_AirportSet.m
%
% Current Code by: Nick Hinze
% Initial Code: Senanu
%-----------------------------------------------------------------------------

tic; clear all; clc

Install_Dir = 'C:\Program Files\TSAM\4.2\data\mode_choice';

alsakaHawaiiIndex = [68:78 527:530];
Part            = 1;

% Pre processing before MapPoint
if Part == 1

   %--------------
   % load files
   %--------------
   % List of 443 Airports
   load (strcat(Install_Dir, '\input\Airport_List_CA'))
   % List of 3091 Counties
   load (strcat(Install_Dir, '\input\County_List'))
   % Great Circle Distance between airports (443x443)
   load (strcat(Install_Dir, '\input\distA2A_CA_443'))
   % Great Circle Distance between counties and airports (3091x443)
   load (strcat(Install_Dir, '\input\C2A_GCDistance_popCent'));

   TotalCounty = length(County_List);
   candidateAirports_MapPoint_Input_GCD = [];

   % Task 3: Create Candidate Airport list based on Great Circle      Distance (GCD)
   %-----------------------------------------------------------------
   for countyCnt = 1 : TotalCounty
      if countyCnt ~= alsakaHawaiiIndex
```

```
          if mod(countyCnt, 100) == 0
             disp(['County Number: ', num2str(countyCnt), ' Time: ', num2str(toc)])
          end
          C2ADistance = 200; colIndex = 0;

          % find airports within 200 miles GCD from county centroid
          while length(colIndex) <= 3 && C2ADistance <= 500
             rowIndex = 0;
             colIndex = 0;
             [rowIndex colIndex] = find(C2A_GCDistance_popCent(countyCnt, :) <= C2ADistance);
             C2ADistance = C2ADistance + 1;
          end % while length(colIndex) <= 3 && C2ADistance <= 500

          % If no airport is found within 500miles, assign closest airport based on GCD
          if isempty(rowIndex) && isempty(colIndex)
             disp(['County: ', num2str(countyCnt), ' has no airport within 500 miles'])
          else
             for arptIndex = 1 : size(colIndex, 2)
                candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_ID(arptIndex)       =
Airport_List_CA(colIndex(arptIndex)).Airport_ID;
                candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_Latitude(arptIndex)  =
Airport_List_CA(colIndex(arptIndex)).Latitude;
                candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_Longitude(arptIndex) =
Airport_List_CA(colIndex(arptIndex)).Longitude;
             end % for arptIndex = 1 : size(rowIndex, 1)
          end % if isempty(rowIndex) == 0 && isempty(colIndex) == 0

       end % if countyCnt ~= alsakaHawaiiIndex
    end % for countyCnt = 1 : TotalCounty

    save ([Install_Dir, '\input\candidateAirports_MapPoint_Input_GCD'],
'candidateAirports_MapPoint_Input_GCD');


    % Create VB/MapPoint input text file to calculate the driving distance to the selected airports
    fid = fopen(strcat(Install_Dir, '\input\candidateAirports_MapPoint_Input_GCD.txt'), 'w');

    for countyCnt = 1:TotalCounty
       County_FIPS_Temp      = char(County_List(countyCnt).County_FIPS);
       County_Longitude_Temp = County_List(countyCnt).County_Longitude;
       County_Latitude_Temp  = County_List(countyCnt).County_Latitude;

       numberOfAirports = length(candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_ID);
       for countyArpts = 1 : numberOfAirports

          Airport_ID_Temp =
char(candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_ID(countyArpts));
          Airport_Longitude_Temp =
candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_Longitude(countyArpts);
          Airport_Latitude_Temp =
candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_Latitude(countyArpts);
```

```
            % If true, do not add a new line. This prevents the empty line at the end of the text file.
            if countyArpts == numberOfAirports && countyCnt == TotalCounty
                fprintf(fid, '%s,%4.10f,%3.10f,%s,%4.10f,%3.10f', County_FIPS_Temp,
County_Longitude_Temp, County_Latitude_Temp, Airport_ID_Temp, Airport_Longitude_Temp,
Airport_Latitude_Temp);
            else
                fprintf(fid, '%s,%4.10f,%3.10f,%s,%4.10f,%3.10f\r', County_FIPS_Temp,
County_Longitude_Temp, County_Latitude_Temp, Airport_ID_Temp, Airport_Longitude_Temp,
Airport_Latitude_Temp);
            end % if countyArpts ==
length(candidateAirports_MapPoint_Input_GCD(countyCnt).Airport_ID)
        end % for countyArpts = 1 : numberOfAirports
    end % for countyCnt = 1:TotalCounty

    fclose(fid);

else % Post Processing after MapPoint

    C2AData_Custom(Install_Dir, AirportSetName);

end % if Part == 1
```
-------------------------------------------------------------------------------------------------------------------

### *7.3.2 Assign three Candidate Airports to each County*

```
-------------------------------------------------------------------------------------------------------------

function PostProcess_MapPoint_C2A_AirportSet()

%------------------------------------------------------------------------
% This function assigns three airports to each county using the following
% rules
%
% Airport 1) Closest airport by drive time from county population centroid.
% Airport 2) Airport with the cheapest average fare
% Airport 3) Airport with the highest emplanements within 100 and 200miles
%
% Airports are selected within a radius of 125miles for MSA counties and
% 200 miles for non-MSA counties
%
% The function is in two parts
%-------------------------------
% Part A: Reads in candidate airports processed with MapPoint
% from Create_MapPoint_Input_Airport_Set.m: and selects
% airports based on route distance intsead of GCD
%
% Part B: Selects the candidate airports
%
%------------------------------------------------------------------------

tic; clear all; clc

Install_Dir = 'C:\Program Files\TSAM\4.2';

%------------------------------
% Part A
%------------------------------

% Read in text file with information about all the airports within selected distance from all
% counties in the US
[FIPS, Airport_ID, Distance_To_Airport, Time_To_Airport_1, Time_To_Airport_2, Ferry] = ...
   textread(strcat(Install_Dir, '\data\mode_choice\input\candidateAirports_MapPoint_Output.txt'),'%s %s
%f %f %f %s','delimiter',',');

% List of 3091 counties in the US (with FIPS code and additional
% information)
load (strcat(Install_Dir, '\data\mode_choice\input\County_List'));

% List of 443 commericial air airports
[FAA_ENPLANE FAA_text FAA_raw] = xlsread(strcat(Install_Dir,
'\data\mode_choice\input\FAA_enplanements.xls'));
FAA_ID_LIST = FAA_text(2:end,2);
```

```
Position = 1; % counter

% start loop of counties
for countyCnt = 1 : 3091

   arptCounter = 0;
   countyFIPS  = County_List(countyCnt).County_FIPS;

   if mod(countyCnt, 200) == 0
      disp(['County Number : ', num2str(countyCnt), ' Time: ', num2str(toc)])
   end % if mod(countyCnt, 200) == 0

   countyCandidateAirport_MapPointSummaryFile(countyCnt).FIPS = FIPS(Position);
   while Position <= length(FIPS) && strcmp(FIPS(Position), countyFIPS) == 1
      arptCounter = arptCounter + 1;
      countyCandidateAirport_MapPointSummaryFile(countyCnt).Airport_ID(arptCounter)      =
Airport_ID(Position);
      countyCandidateAirport_MapPointSummaryFile(countyCnt).Distance_To_Airport(arptCounter) =
Distance_To_Airport(Position);
      countyCandidateAirport_MapPointSummaryFile(countyCnt).Time_To_Airport_1(arptCounter) =
Time_To_Airport_1(Position);
      countyCandidateAirport_MapPointSummaryFile(countyCnt).Time_To_Airport_2(arptCounter) =
Time_To_Airport_2(Position);
      countyCandidateAirport_MapPointSummaryFile(countyCnt).Ferry_value(arptCounter)     =
Ferry(Position);

      currentAirportID = Airport_ID(Position);
      airportIndex     = strmatch(currentAirportID, FAA_ID_LIST);
      if isempty(airportIndex)
         disp('Airport Index Error')
         countyCnt
         arptCounter
         pause
      else
         countyCandidateAirport_MapPointSummaryFile(countyCnt).Airport_Index(arptCounter) =
airportIndex;
      end % if isempty(airportIndex)

      Position = Position + 1;
   end % countyCandidateAirport_MapPointSummaryFile(countyCnt).FIPS = FIPS(Position);
   countyCandidateAirport_MapPointSummaryFile(countyCnt).Number_Of_Candidate_Airports =
arptCounter;

end % for countyCnt = 1 : 3091
save (strcat(Install_Dir, '\data\mode_choice\input\countyCandidateAirport_MapPointSummaryFile.mat'),
'countyCandidateAirport_MapPointSummaryFile');
clear arptCounter Airport_ID Airport_Index countyCnt currentAirportID
countyCandidateAirport_MapPointSummaryFile
clear Distance_To_Airport Ferry FIPS Position Time_To_Airport_1 Time_To_Airport_2
```

```
%-------------------------------
% Part B
%-------------------------------
load ([Install_Dir, '\data\mode_choice\input\countyCandidateAirport_MapPointSummaryFile'])
load ([Install_Dir, '\data\mode_choice\input\C2A_GCDistance_popCent.mat']);
alsakaHawaiiIndex      = [68:78 527:530];
for countyCnt = 1 : 3091

    if countyCnt ~= alsakaHawaiiIndex

        if mod(countyCnt, 100) == 0
            disp(['County Number: ', num2str(countyCnt), ' Time: ', num2str(toc)])
        end % if mod(countyCnt, 100) == 0
        % Find out if all the airports have an access time
        negativeAccessTime_Index =
find(countyCandidateAirport_MapPointSummaryFile(countyCnt).Time_To_Airport_1 == -999);
        if ~isempty(negativeAccessTime_Index) % At least one airport without access time
            realAirportIndex =
countyCandidateAirport_MapPointSummaryFile(countyCnt).Airport_Index(negativeAccessTime_Index);

            C2A_dist = C2A_GCDistance_popCent(countyCnt, realAirportIndex);

countyCandidateAirport_MapPointSummaryFile(countyCnt).Distance_To_Airport(negativeAccessTime_
Index) = C2A_dist;

countyCandidateAirport_MapPointSummaryFile(countyCnt).Time_To_Airport_1(negativeAccessTime_I
ndex)   = C2A_dist/35;

countyCandidateAirport_MapPointSummaryFile(countyCnt).Time_To_Airport_2(negativeAccessTime_I
ndex)   = C2A_dist/35;
            disp(['No MapPoint access time from County: ', num2str(countyCnt), ' to Airport: ',
num2str(realAirportIndex)]);
        end % if ~isempty(negativeAccessTime_Index) % At least one airport without access time
    end % if countyCnt ~= alsakaHawaiiIndex
end % for countyCnt = 1 : 3091
save ([Install_Dir, '\data\mode_choice\input\countyCandidateAirport_MapPointSummaryFile.mat'],
'countyCandidateAirport_MapPointSummaryFile');


% Select candidate airport pairs
load ([Install_Dir, '\data\mode_choice\input\countyCandidateAirport_MapPointSummaryFile']);
load ([Install_Dir, '\data\mode_choice\input\CA_Hub_Type.mat']);
load ([Install_Dir, '\data\mode_choice\input\distC2A_FULL_With_OEP.mat']);
load ([Install_Dir, '\data\mode_choice\input\msaindicatordata.mat']);
load ([Install_Dir, '\data\mode_choice\input\A2A_CA_CoachClass_Fare_2000.mat']);

C2A_MapPointSummaryFile = countyCandidateAirport_MapPointSummaryFile;

[FAA_ENPLANE FAA_text FAA_raw] = xlsread([Install_Dir,
'\data\mode_choice\input\FAA_enplanements.xls']);
meanFares = zeros(size(A2A_CA_CoachClass_Fare_2000,1),6);
```

```
for i = 1 : size(A2A_CA_CoachClass_Fare_2000,2)
    meanFares(i,6) = CA_Hub_Type(i);
    row_values = nonzeros(A2A_CA_CoachClass_Fare_2000(i,:));
    if isempty(row_values)
    else
        meanFares(i,1) = mean(row_values);
        meanFares(i,4) = size(row_values,1);
    end % if isempty(row_values)

    col_values = nonzeros(A2A_CA_CoachClass_Fare_2000(:,i));
    if isempty(col_values)
    else
        meanFares(i,2) = mean(col_values);
        meanFares(i,5) = size(col_values,1);
    end % if isempty(row_values)
end % for i = 1 : size(A2A_CA_CoachClass_Fares_2000,2)

counter = 0;
single_and_zero_fare_index = zeros(400,1);
for i = 1 : size(meanFares,1)
    [maxValue maxIndex] = max(meanFares(i,1:2));
    meanFares(i,3) = maxValue;

    if meanFares(i,3) == 0 || meanFares(i,4) == 0 || meanFares(i,5) == 0
        counter = counter + 1;
        single_and_zero_fare_index(counter,1) = i;
    end % if meanFares(i,4) == 1 | meanFares(i,5) == 1
end % for i = 1 : size(meanFares,1)

zero_values_index = find(single_and_zero_fare_index == 0);
single_and_zero_fare_index(zero_values_index) = [];
tic;

alsakaHawaiiIndex      = [68:78 527:530];

% initialize tables
CA_candidateAirports   = [];
CA_candArpts_arptNumb  = zeros(3091, 3);
CA_candArpts_driveDist = zeros(3091, 3);
CA_candArpts_driveTime = zeros(3091, 3);
CA_candArpts_tripTime  = zeros(3091, 3);
CA_candArpts_hubType   = zeros(3091, 3);
tractCounties          = zeros(200,2);
counter                = 0;

for countyCnt = 1 : 3091
    if countyCnt == 8
        k = 9;
    end % if countyCnt == 359

    if msaIndicatorData(countyCnt,2) == 1
```

```
        countyType_cnt = 1;
        maxDistance = 100;
    else
        countyType_cnt = 2;
        maxDistance = 200;
    end % if msaIndicatorData(countyCnt,2) == 1

    % skip counties in alaska and hawaii
    if find(countyCnt == alsakaHawaiiIndex)
        CA_candidateAirports(countyCnt).arptName = [];
    else
        % display counter
        if mod(countyCnt, 50) == 0
            disp(['County Number: ', num2str(countyCnt), ' Time: ', num2str(toc)])
        end % if mod(countyCnt, 100) == 0

        %--------------------------------------
        % CASE 1: Select closest airport
        %--------------------------------------
        % check if there are airports within specified distance
        selectedIndex = find(C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport <=
maxDistance);

        airportIndex     = C2A_MapPointSummaryFile(countyCnt).Airport_Index(selectedIndex);
        average_fare_all = meanFares(airportIndex,1:2);
        non_fare_airport_index = find(sum(average_fare_all,2) == 0);
        if isempty(non_fare_airport_index)
        else
            selectedIndex(non_fare_airport_index) = [];
        end % if isempty(zeroFareCheck_2)

        % assign closest airport if there is no airport within selected radius
        if isempty(selectedIndex) % if there are no airports
            airportNumbers = C2A_MapPointSummaryFile(countyCnt).Airport_Index;
            check_hubType  = CA_Hub_Type(airportNumbers);

            % check if there is a large hub airport within airport set
            check_LH_index = find(check_hubType == 1);
            if ~isempty(check_LH_index)
                if length(check_LH_index) > 1
                    large_hub_airports = airportNumbers(check_LH_index);
                    for i = 1 : length(large_hub_airports)
                        enplanement_LH(i) = FAA_ENPLANE(large_hub_airports(i),2);
                    end
                    [maxValueEnp maxIndexEnp] = max(enplanement_LH);
                    check_LH_index = check_LH_index(maxIndexEnp);
                end % if length(check_LH_index) > 1
                selectedIndex = check_LH_index;
            else
                % if no Large Hub check if there is a medium hub airport
                check_MH_index  = find(check_hubType == 2);
                if ~isempty(check_MH_index)
                    if length(check_MH_index) > 1
                        medium_hub_airports = airportNumbers(check_MH_index);
                        for i = 1 : length(medium_hub_airports)
                            enplanement_MH(i) = FAA_ENPLANE(medium_hub_airports(i),2);
                        end
                        [maxValueEnp maxIndexEnp] = max(enplanement_MH);
                        check_MH_index = check_MH_index(maxIndexEnp);
                    end % if length(check_LH_index) > 1
                    selectedIndex = check_MH_index;
                else
                    [selectedIndexValue selectedIndex] =
min(C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1);
                end % if ~isempty(check_MH_index)
            end % if ~isempty(check_LH_index)
        end % if isempty(selectedIndex)

        % Assign information in file to variables
        airportIndex      = C2A_MapPointSummaryFile(countyCnt).Airport_Index(selectedIndex);
        candAirptID       = C2A_MapPointSummaryFile(countyCnt).Airport_ID(selectedIndex);
        distanceToAirport = C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport(selectedIndex);
        timeToAirport_1   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1(selectedIndex);
        timeToAirport_2   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_2(selectedIndex);
        airportHubType    = CA_Hub_Type(airportIndex);

        % CASE 1: Select closest airport and save
        [timeValue timeIndex] = min(timeToAirport_1);
        CA_candidateAirports(countyCnt).arptName(1) = candAirptID(timeIndex);
        CA_candArpts_arptNumb(countyCnt,1)  = airportIndex(timeIndex);
        CA_candArpts_driveDist(countyCnt,1) = distanceToAirport(timeIndex);
        CA_candArpts_driveTime(countyCnt,1) = timeValue;
        CA_candArpts_tripTime(countyCnt,1)  = timeToAirport_2(timeIndex);
        CA_candArpts_hubType(countyCnt,1)   = airportHubType(timeIndex);

        % delete saved airport from choice set
        candAirptID(timeIndex)       = [];
        airportIndex(timeIndex)      = [];
        distanceToAirport(timeIndex) = [];
        timeToAirport_1(timeIndex)   = [];
        timeToAirport_2(timeIndex)   = [];
        airportHubType(timeIndex)    = [];

        %-------------------------------------
        % CASE 2: Select cheapest airport
        %-------------------------------------
        % If array is empyty revert to old airport set
        % but remember to delete closest airport
        if countyType_cnt == 2
            if isempty(airportIndex)
                candAirptID       = C2A_MapPointSummaryFile(countyCnt).Airport_ID;
                airportIndex      = C2A_MapPointSummaryFile(countyCnt).Airport_Index;
                distanceToAirport = C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport;
                timeToAirport_1   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1;
                timeToAirport_2   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_2;
                airportHubType    = CA_Hub_Type(airportIndex);

                deleteIndex = find(C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1 ==
timeValue);
                candAirptID(deleteIndex)      = [];
                airportIndex(deleteIndex)     = [];
                distanceToAirport(deleteIndex) = [];
                timeToAirport_1(deleteIndex)   = [];
                timeToAirport_2(deleteIndex)   = [];
                airportHubType(timeIndex)      = [];
            end % if isempty(airportIndex)
        end % if countyType_cnt == 2

        if ~isempty(airportIndex)
            % delete all airports that have only 1 O-D pair link in DB1B
            airportMeanFares = meanFares(airportIndex,3);
            low_freq_airport = [];
            airport_numbers  = [];
            cheap_counter    = 0;
            for i = 1 : size(airportIndex,2)
                low_freq_airport_index = find(airportIndex(i) == single_and_zero_fare_index);
                if ~isempty(low_freq_airport_index)
                    cheap_counter = cheap_counter + 1;
                    low_freq_airport(cheap_counter,1) = i;
                    airport_numbers(cheap_counter,1)  = single_and_zero_fare_index(low_freq_airport_index);
                end % if ~isempty(low_freq_airport_index)
            end % for i = 1 : size(airportIndex,2)

            if ~isempty(low_freq_airport)
                candAirptID(low_freq_airport)      = [];
                airportIndex(low_freq_airport)     = [];
                distanceToAirport(low_freq_airport) = [];
                timeToAirport_1(low_freq_airport)   = [];
                timeToAirport_2(low_freq_airport)   = [];
                airportMeanFares(low_freq_airport)  = [];
            end % if ~isempty(low_freq_airport)

            if isempty(airportIndex)
                selectedIndex = find(C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport <=
maxDistance*1.5);
                candAirptID     = C2A_MapPointSummaryFile(countyCnt).Airport_ID(selectedIndex);
                airportIndex    = C2A_MapPointSummaryFile(countyCnt).Airport_Index(selectedIndex);
                airportHubType  = CA_Hub_Type(airportIndex)';

                find_NH_index    = [];
                find_LH_MH_index = find(airportHubType == 1 | airportHubType == 2);
                if ~isempty(find_LH_MH_index)
                    find_NH_index = find(airportHubType == 4);
                end % if isempty(find_LH_MH_index)
                selectedIndex(find_NH_index) = [];

                candAirptID      = C2A_MapPointSummaryFile(countyCnt).Airport_ID(selectedIndex);
                airportIndex     = C2A_MapPointSummaryFile(countyCnt).Airport_Index(selectedIndex);
                distanceToAirport =
            C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport(selectedIndex);
                timeToAirport_1   =
            C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1(selectedIndex);
                timeToAirport_2   =
            C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_2(selectedIndex);
                airportHubType   = CA_Hub_Type(airportIndex);

                deleteIndex_1 = find(CA_candArpts_arptNumb(countyCnt,1) == airportIndex);
                if ~isempty(deleteIndex_1)
                    candAirptID(deleteIndex_1)      = [];
                    airportIndex(deleteIndex_1)     = [];
                    distanceToAirport(deleteIndex_1) = [];
                    timeToAirport_1(deleteIndex_1)   = [];
                    timeToAirport_2(deleteIndex_1)   = [];
                    airportHubType(deleteIndex_1)    = [];
                end % if ~isempty(deleteIndex_1)

                cheap_counter_2 = 0;
                if ~isempty(airportIndex)
                    deleteIndex_2 = [];
                    for m = 1 : size(airportIndex,2)
                      low_freq_airport_index_2 = find(airportIndex(m) == single_and_zero_fare_index);
                      if ~isempty(low_freq_airport_index_2)
                        cheap_counter_2 = cheap_counter_2 + 1;
                        deleteIndex_2(cheap_counter_2, 1) = m;
                      end % if ~isempty(low_freq_airport_index_2)
                    end % for m = 1 : size(airportIndex,2)

                    candAirptID(deleteIndex_2)      = [];
                    airportIndex(deleteIndex_2)     = [];
                    distanceToAirport(deleteIndex_2) = [];
                    timeToAirport_1(deleteIndex_2)   = [];
                    timeToAirport_2(deleteIndex_2)   = [];
                    airportHubType(deleteIndex_2)    = [];
                end % if ~isempty(airportIndex)
            end % if isempty(airportIndex)

            airportMeanFares    = meanFares(airportIndex,3);
            airportHubIndicator = meanFares(airportIndex,6);
            LH_Indicator = find(airportHubIndicator == 1);
            NH_Indicator = find(airportHubIndicator == 4);
            if ~isempty(LH_Indicator) & ~isempty(NH_Indicator)
                airportMeanFares(NH_Indicator) = 99999;
            end % if ~isempty(LH_Indicator) & ~isempty(NH_Indicator)
```

```
            if ~isempty(airportIndex)
                [cheapValue cheapIndex] = min(airportMeanFares);
                CA_candidateAirports(countyCnt).arptName(2) = candAirptID(cheapIndex);
                CA_candArpts_arptNumb(countyCnt,2)  = airportIndex(cheapIndex);
                CA_candArpts_driveDist(countyCnt,2) = distanceToAirport(cheapIndex);
                CA_candArpts_driveTime(countyCnt,2) = timeToAirport_1(cheapIndex);
                CA_candArpts_tripTime(countyCnt,2)  = timeToAirport_2(cheapIndex);
                CA_candArpts_hubType(countyCnt,2)   = CA_Hub_Type(airportIndex(cheapIndex));

                cheap_single_fare = find(airportIndex(cheapIndex) == single_and_zero_fare_index);
                if ~isempty(cheap_single_fare)
                    counter = counter + 1;
                    tractCounties(counter,1) = countyCnt;
                    tractCounties(counter,2) = airportIndex(cheapIndex);
                    tractCounties(counter,3) = CA_Hub_Type(airportIndex(cheapIndex));
                end

                candAirptID(cheapIndex)       = [];
                airportIndex(cheapIndex)      = [];
                distanceToAirport(cheapIndex) = [];
                timeToAirport_1(cheapIndex)   = [];
                timeToAirport_2(cheapIndex)   = [];
            end % if ~isempty(airportIndex)
        end % if isempty(airportIndex)

        %-------------------------------------
        % CASE 3: Select airport with highest enplanements
        %-------------------------------------
        % first check if there are any airport in array else revert to
        % old airport set but remember to delete closest airport
        if countyType_cnt == 2
            if isempty(airportIndex)
                candAirptID       = C2A_MapPointSummaryFile(countyCnt).Airport_ID;
                airportIndex      = C2A_MapPointSummaryFile(countyCnt).Airport_Index;
                distanceToAirport = C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport;
                timeToAirport_1   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1;
                timeToAirport_2   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_2;

                deleteIndex_time  = find(C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1 ==
timeValue);
                deleteIndex_cheap = find(C2A_MapPointSummaryFile(countyCnt).Airport_Index ==
CA_candArpts_arptNumb(countyCnt,2));
                deleteIndex = [deleteIndex_time deleteIndex_cheap];
                candAirptID(deleteIndex)       = [];
                distanceToAirport(deleteIndex) = [];
                timeToAirport_1(deleteIndex)   = [];
                timeToAirport_2(deleteIndex)   = [];
                airportIndex(deleteIndex)      = [];
            end % if isempty(airportIndex)
        end % if countyType_cnt == 2
        if isempty(airportIndex)
        else
            airportEnplanements = FAA_ENPLANE(airportIndex,2);
            [enplaneValue enplaneIndex] = max(airportEnplanements);
            CA_candidateAirports(countyCnt).arptName(3) = candAirptID(enplaneIndex);
            CA_candArpts_arptNumb(countyCnt,3)  = airportIndex(enplaneIndex);
            CA_candArpts_driveDist(countyCnt,3) = distanceToAirport(enplaneIndex);
            CA_candArpts_driveTime(countyCnt,3) = timeToAirport_1(enplaneIndex);
            CA_candArpts_tripTime(countyCnt,3)  = timeToAirport_2(enplaneIndex);
            CA_candArpts_hubType(countyCnt,3)   = CA_Hub_Type(airportIndex(enplaneIndex));

            candAirptID(enplaneIndex)       = [];
            distanceToAirport(enplaneIndex) = [];
            timeToAirport_1(enplaneIndex)   = [];
            timeToAirport_2(enplaneIndex)   = [];
            airportIndex(enplaneIndex)      = [];
        end % if isempty(airportIndex)

        if msaIndicatorData(countyCnt,2) == 1
            selectedAirports_indexFinal   = CA_candArpts_arptNumb(countyCnt,:);
            selectedAirports_hubTypeFinal = CA_candArpts_hubType(countyCnt,:);
            check_4_LH_MH  = find((selectedAirports_hubTypeFinal == 1) |
    (selectedAirports_hubTypeFinal == 2));
            if isempty(check_4_LH_MH)
                deleteIndex       = [];
                candAirptID       = C2A_MapPointSummaryFile(countyCnt).Airport_ID;
                airportIndex      = C2A_MapPointSummaryFile(countyCnt).Airport_Index;
                distanceToAirport = C2A_MapPointSummaryFile(countyCnt).Distance_To_Airport;
                timeToAirport_1   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_1;
                timeToAirport_2   = C2A_MapPointSummaryFile(countyCnt).Time_To_Airport_2;
                selectHubType     = CA_Hub_Type(airportIndex)';

                k = 0;
                for i = 1 : size(selectedAirports_indexFinal,2)
                    if selectedAirports_indexFinal(i) == 0
                    else
                        k = k + 1;
                        deleteIndex(k) = find(selectedAirports_indexFinal(i) == airportIndex);
                    end % if selectedAirports_indexFinal(i) == 0
                end % for i = 1 : size(selectedAirports_indexFinal)

                nonHub_smallHub_index = find(selectHubType == 3 | selectHubType == 4);
                deleteIndex = unique([deleteIndex nonHub_smallHub_index]);
                candAirptID(deleteIndex)       = [];
                airportIndex(deleteIndex)      = [];
                distanceToAirport(deleteIndex) = [];
                timeToAirport_1(deleteIndex)   = [];
                timeToAirport_2(deleteIndex)   = [];
                selectHubType(deleteIndex)     = [];

                time_and_airportType = [timeToAirport_1; selectHubType; 1:length(timeToAirport_1)];
                time_and_airportType = sortrows(time_and_airportType',2)';

                largeHubs = find(time_and_airportType(2,:) == 1);
                [close_LH closeIndex_LH] = min(time_and_airportType(1,largeHubs));
                if isempty(closeIndex_LH)
                else
                    closeIndex_LH = find(time_and_airportType(1,:) == close_LH & time_and_airportType(2,:)
    == 1);
                    if length(closeIndex_LH) > 1
                        closeIndex_LH = closeIndex_LH(1);
                    end % if length(closeIndex_LH) > 1
                    closeIndex_LH = time_and_airportType(3, closeIndex_LH);
                end % if isempty(closeIndex_LH)

                mediumHubs = find(time_and_airportType(2,:) == 2);
                [close_MH closeIndex_MH] = min(time_and_airportType(1,mediumHubs));
                if isempty(closeIndex_MH)
                else
                    closeIndex_MH = find(time_and_airportType(1,:) == close_MH &
    time_and_airportType(2,:) == 2);
                    closeIndex_MH = time_and_airportType(3, closeIndex_MH);
                end % if isempty(closeIndex_LH)

                noData = 0;
                if isempty(close_LH) && isempty(close_MH)
                    noData = 1;
                elseif isempty(close_LH) && ~isempty(close_MH)
                    selected_index = closeIndex_MH;
                elseif ~isempty(close_LH) && isempty(close_MH)
                    selected_index = closeIndex_LH;
                elseif ~isempty(close_LH) && ~isempty(close_MH)
                    if close_LH <= close_MH
                        selected_index = closeIndex_LH;
                    else
                        LH_penalty = close_LH / close_MH;
                        if LH_penalty > 1.2
                           selected_index = closeIndex_MH;
                        else
                           selected_index = closeIndex_LH;
                        end % if LH_penalty > 1.2
                    end % if close_LH <= close_MH
                end % if isempty(close_LH) && isempty(close_MH)

                if isempty(candAirptID) | noData == 1
                else
                    setSize = size(CA_candidateAirports(countyCnt).arptName,2);
                    if setSize < 3
                        setIndex = setSize + 1;
                    else
                        airportEnplanements_2(1)    = FAA_ENPLANE(selectedAirports_indexFinal(2),2);
                        airportEnplanements_2(2)    = FAA_ENPLANE(selectedAirports_indexFinal(3),2);
```

```
                [valueSelect indexSelect] = min(airportEnplanements_2);
                setIndex = indexSelect + 1;
            end % if setSize < 3
            CA_candidateAirports(countyCnt).arptName(setIndex) = candAirptID(selected_index);
            CA_candArpts_arptNumb(countyCnt, setIndex)  = airportIndex(selected_index);
            CA_candArpts_driveDist(countyCnt, setIndex) = distanceToAirport(selected_index);
            CA_candArpts_driveTime(countyCnt, setIndex) = timeToAirport_1(selected_index);
            CA_candArpts_tripTime(countyCnt, setIndex)  = timeToAirport_2(selected_index);
            CA_candArpts_hubType(countyCnt, setIndex)   =
CA_Hub_Type(airportIndex(selected_index));
        end % if isempty(candAirptID)
        candAirptID       = [];
        distanceToAirport = [];
        timeToAirport_1   = [];
        timeToAirport_2   = [];
        airportIndex      = [];
      end % if isempty(check_4_LH_MH)
    end % if msaIndicatorData(countyCnt,3) == 1

  end % if find(countyCnt == alsakaHawaiiIndex)
end % for countyCnt = 1 : 3091

zeroCells = find(CA_candArpts_arptNumb == 0);
CA_candArpts_arptNumb(zeroCells)  = -9999;
CA_candArpts_driveDist(zeroCells) = -9999;
CA_candArpts_driveTime(zeroCells) = -9999;
CA_candArpts_tripTime(zeroCells)  = -9999;
CA_candArpts_hubType(zeroCells)   = -9999;

save ([Install_Dir, '\data\mode_choice\input\CA_candidateAirports'], 'CA_candidateAirports')
save ([Install_Dir, '\data\mode_choice\input\CA_candArpts_arptNumb'], 'CA_candArpts_arptNumb')
save ([Install_Dir, '\data\mode_choice\input\CA_candArpts_driveDist'], 'CA_candArpts_driveDist')
save ([Install_Dir, '\data\mode_choice\input\CA_candArpts_driveTime'], 'CA_candArpts_driveTime')
save ([Install_Dir, '\data\mode_choice\input\CA_candArpts_tripTime'], 'CA_candArpts_tripTime')
save ([Install_Dir, '\data\mode_choice\input\CA_candArpts_hubType'], 'CA_candArpts_hubType')

return
```

----------------------------------------------------------------------------------------------------------------

### 7.3.3 *Adjust Travel Time for Airports with Ferry in Trip Leg*

```
--------------------------------------------------------------------------------------------------------------

function Correct_For_Ferry_Distance()

%-------------------------------------------------------------------------
% THis m-file corrects the driving distance givin by MapPoint.
% MapPoint does not count a ferry trip in it's driving distance and driving
% time. However, the trip time given by MapPoint contains the ferry trip.
% Now the distance can be corrected using the difference between the trip
% time and driving time.
%
% Coded By: Nick Hinze
%-------------------------------------------------------------------------

clear all
clc

Estimated_Ferry_Average_Speed = 15; % statute mph

loadFileName  = 'C:\Program Files\TSAM\4.2\data\mode_choice\input';

%load files
load ([loadFileName, '/CA_candArpts_tripTime'])
load ([loadFileName, '/CA_candArpts_driveTime'])
load ([loadFileName, '/CA_candArpts_driveDist'])

%Difference between Trip Time and Driving Time
Ferry_Time_Difference = CA_candArpts_tripTime - CA_candArpts_driveTime;

%Calculate estimated ferry distance
Estiated_Ferry_Distance = Ferry_Time_Difference * Estimated_Ferry_Average_Speed;

%Add difference to driving distance
CA_candArpts_driveDist = CA_candArpts_driveDist + round(Estiated_Ferry_Distance);

% WARNING: Change trip time name to drive time name for mode choice model
CA_candArpts_driveTime = CA_candArpts_tripTime;

%Saving
save ([loadFileName, '\CA_candArpts_driveDist.mat'], 'CA_candArpts_driveDist')
save ([loadFileName, '\CA_candArpts_driveTime.mat'], 'CA_candArpts_driveTime')

return

--------------------------------------------------------------------------------------------------------------
```

### *7.3.4 Adjust airport-set for cases where large-hub and non-hub airports are in the same set*

```
----------------------------------------------------------------------------------------------------------------------
--

function update_candidate_airport_file()

%=============================================%
% Refill candidate airport table
% 1) if a large hub is present delete all
% non-hubs for that county
%=============================================%
clear all; clc

%=============================================%
% reload and save files to TSAM input dir        %
%=============================================%
saveFileName = 'C:\Program Files\TSAM\4.2\data\mode_choice\input';

load ([saveFileName, '\CA_candidateAirports'])
load ([saveFileName, '\CA_candArpts_arptNumb'])
load ([saveFileName, '\CA_candArpts_hubType'])
load ([saveFileName, '\CA_candArpts_driveDist'])
load ([saveFileName, '\CA_candArpts_driveTime'])

for cntCnty = 1 : 3091

   % check for hub type index
   largeHubIndex  = find(CA_candArpts_hubType(cntCnty,:) == 1); % Check if LH airport is present
   mediumHubIndex = find(CA_candArpts_hubType(cntCnty,:) == 2); % Check if MH airport is present
   smallHubIndex  = find(CA_candArpts_hubType(cntCnty,:) == 3); % Check if SH airport is present
   nonHubIndex    = find(CA_candArpts_hubType(cntCnty,:) == 4); % Check if a non-hub is present

   % if there is Large hub skip loop
   if isempty(largeHubIndex)
      % Check if non-hub is competing with a medium hub that is closet to county centroid
      if ~isempty(mediumHubIndex) & ~isempty(nonHubIndex)
         mediumclosest = find(mediumHubIndex == 1);
         if ~isempty(mediumclosest)
            numberOfNonHubs = size(nonHubIndex,2);
            for i = 1 : numberOfNonHubs
               hubPosition = nonHubIndex(i);
               if hubPosition == 1
               else
                  CA_candidateAirports(cntCnty).arptName(nonHubIndex(i)) = cellstr('');
                  CA_candArpts_arptNumb(cntCnty,nonHubIndex(i)) = -9999;
                  CA_candArpts_hubType(cntCnty,nonHubIndex(i)) = 4;
                  CA_candArpts_driveDist(cntCnty,nonHubIndex(i)) = -9999;
                  CA_candArpts_driveTime(cntCnty,nonHubIndex(i)) = -9999;
               end % if hubPosition == 1
            end % for i = 1 : numberOfNonHubs
            end % if ~isempty(mediumHubIndex) & ~isempty(nonHubIndex)
        else
            if ~isempty(smallHubIndex) & ~isempty(nonHubIndex)
                smallclosest = find(smallHubIndex == 1);
                if ~isempty(smallclosest)
                    numberOfNonHubs = size(nonHubIndex,2);
                    for i = 1 : numberOfNonHubs
                        hubPosition = nonHubIndex(i);
                        if hubPosition == 1
                        else
                            CA_candidateAirports_(cntCnty).arptName(nonHubIndex(i)) = cellstr('');
                            CA_candArpts_arptNumb(cntCnty,nonHubIndex(i)) = -9999;
                            CA_candArpts_hubType(cntCnty,nonHubIndex(i)) = 4;
                            CA_candArpts_driveDist(cntCnty,nonHubIndex(i)) = -9999;
                            CA_candArpts_driveTime(cntCnty,nonHubIndex(i)) = -9999;
                        end % if hubPosition == 1
                    end % for i = 1 : numberOfNonHubs
                end % if ~isempty(smallclosest)
            end % if ~isempty(smallHubIndex) & ~isempty(nonHubIndex)
        end % if ~isempty(mediumHubIndex) & ~isempty(nonHubIndex) & isempty(largeHubIndex)
    else
        % if there are non NH airports skip loop
        if isempty(nonHubIndex)
        else
            % if there is a non-hub competing with a large hub
            % delete it [L L N] [L N L] [*]
            numberOfNonHubs = size(nonHubIndex,2);
            for i = 1 : numberOfNonHubs
                hubPosition = nonHubIndex(i);
                if hubPosition == 1
                else
                    CA_candidateAirports(cntCnty).arptName(nonHubIndex(i)) = cellstr('');
                    CA_candArpts_arptNumb(cntCnty,nonHubIndex(i)) = -9999;
                    CA_candArpts_hubType(cntCnty,nonHubIndex(i)) = 4;
                    CA_candArpts_driveDist(cntCnty,nonHubIndex(i)) = -9999;
                    CA_candArpts_driveTime(cntCnty,nonHubIndex(i)) = -9999;
                end % if hubPosition == 1
            end % for i = 1 : numberOfNonHubs
        end % if isempty(nonHubIndex)
    end % if isempty(largeHubIndex)
end % for cntCnty = 1 : 3091

save ([saveFileName, '\CA_candidateAirports'], 'CA_candidateAirports')
save ([saveFileName, '\CA_candArpts_arptNumb'], 'CA_candArpts_arptNumb')
save ([saveFileName, '\CA_candArpts_hubType'], 'CA_candArpts_hubType')
save ([saveFileName, '\CA_candArpts_driveDist'], 'CA_candArpts_driveDist')
save ([saveFileName, '\CA_candArpts_driveTime'], 'CA_candArpts_driveTime')

return
---------------------------------------------------------------------------------------------------------
```

## 7.4 Pre-process County-level Calibration Travel Times and Costs

### *7.4.1 Aggregate Automobile and Commercial Air Travel Time and Cost (ignore demand between counties)*

**SubFuntions (**C2C_TT_TC_Aggt_noDemand_modular.m, select_CA_CandidateRoutes_modular.m**)**

----------------------------------------------------------------------------------------------------------------------------------

```
function call_C2C_TT_TC_Aggt_noDemand_modular()

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Calibration of Mixed Logit Model
%
% Create aggregate County-to-county Travel Time and Travel Cost Input
% Tables for Mixed Logit Model (tables are for Automobile and Commercial Airline)
% The data is segregate over distance [100:50:3000]
%
% Crearted By:   Senanu Ashiabor
% Date:          March 8, 2006
% Last Modified:
%
% Calling:   aggregate_C2C_TT_and_TC_noDemand_using_PT_Survey
% Called by: None
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

clear all; clc

% Name of reference directory
Install_Dir = 'C:\Program Files\TSAM\4.2\data\mode_choice\input';

%----------------------------
% Declare global variables
%----------------------------
% tables
global A2A_CA_BusinessClassFare_DB1B_2000 A2A_CA_CoachClassFare_Filled_DB1B_2000
A2A_CA_CoachFare_Index_DB1B_2000
global A2A_CA_Schedule_Delay_2way A2A_seatData A2A_CA_travelTime_with_SD_2way
A2A_CA_travelTime_no_SD_2way
global CA_Hub_Type CA_candArpts_driveTime CA_candArpts_driveDist CA_candArpts_arptNumb
C2A_GCDistance_popCent
global C2CDriveTime_Population_Centroids C2CDriveDist_Population_Centroids distA2A_CA
global distC2C_Population_Centroids numberOfIncomeGroups MinNumberOfLegs
global msaIndicatorData waitingAtOriginAirport_CA waitingAtDestntAirport_CA

% constants
```

```
global averageDaily_LogdgingCost additionalOvernightTime_perDay_auto
additionalOvernightTime_perDay_CA
global averageRoundTripTime_auto avgOccupancy_auto additionalDay_Auto costPerMile_auto
caseName
global avgOccupancy_Auto costPerMile_VLJ c2aData_VLJ distA2A_VLJ CostProfile ConstantVLJCost
global costPerMile_Auto drivingSpeed_Urban drivingSpeed_Rural FlightSpeedProfile
global maxRoundTripTime_CA_2 minRoundTripTime_CA maxRoundTripTime_CA
oneDayLodgingCost
global maxAccessTime_VLJ maxEgressTime_VLJ min_A2A_Distance_VLJ
global max_FSP_Distance_VLJ min_FSP_Distance_VLJ max_FSP_Speed_VLJ min_FSP_Speed_VLJ
global min_CP_Distance_VLJ min_CP_Cost_VLJ max_CP_Cost_VLJ scenarioName
scheduleDelay_VLJ
global maxDailyDriveTime_Auto waitingAtOrginAirport_VLJ waitingAtDestAirport_VLJ

%--------------
% Load Tables
%--------------
tic
load (strcat(Install_Dir, '\A2A_CA_travelTime_with_SD_2way')) % Inter-airporrt travel times in hours
(685x685)
load (strcat(Install_Dir, '\A2A_CA_travelTime_no_SD_2way'))   % Inter-airporrt flight times in hours
(443*443)
load (strcat(Install_Dir, '\A2A_CA_Schedule_Delay_2way'))     % Inter-airporrt schedule delay in hours
(443x443)

load (strcat(Install_Dir, '\A2A_CA_BusinessClassFare_DB1B_2000')) % Inter-airporrt business class
fares ($)
load (strcat(Install_Dir, '\A2A_CA_CoachClassFare_Filled_DB1B_2000'))    % Inter-airporrt coach class
fares ($)
load (strcat(Install_Dir, '\A2A_CA_CoachFare_Index_DB1B_2000'))

load (strcat(Install_Dir, '\C2A_GCDistance_popCent')) % County to Airport distance table (3091x419)

% Hub types: 1 -> Large hub; 2-> Medium hub; 3 -> Small hub; 4 -> Non-Hub
load (strcat(Install_Dir, '\CA_candArpts_arptNumb'))  % Index of CA candidate airports within 100mile
radius of each county
load (strcat(Install_Dir, '\CA_candArpts_driveDist')) % Drive distance to CA candidate airports within
100 mile radius of each county
load (strcat(Install_Dir, '\CA_candArpts_driveTime')) % Drive time to CA candidate airports in 100 mile
radius of each county

load (strcat(Install_Dir, '\distA2A_CA_443'))   % Airport to airport distance table (443x443)
load (strcat(Install_Dir, '\msaIndicatorData')) % List indicating if county is in an MSA. (1 -> MSA; 0 ->
nonMSA)

load (strcat(Install_Dir, '\stateNames'))    % List of states in US
load (strcat(Install_Dir, '\stateIndexing')) % Index showing where counties of each state begin and end in
the county to county lis
load (strcat(Install_Dir, '\avgPartySize_ATS_Buss'))    % Business party size from ATS [5x7]
load (strcat(Install_Dir, '\avgPartySize_ATS_NonBuss')) % Business party size from ATS [5x7]
load (strcat(Install_Dir, '\seats'))  % List of states in US
load (strcat(Install_Dir, '\CA_Hub_Type'))
load (strcat(Install_Dir, '\MinNumberOfLegs'))
A2A_seatData = sum(seats(1:443,1:443), 2);

% checking NaN travel time with SD
if sum(sum(isnan(A2A_CA_travelTime_with_SD_2way))) ||
size(find(A2A_CA_travelTime_with_SD_2way==0),1) >443
    disp('Error: Travel Time CA')
    pause
end % if sum(sum(isnan(A2A_CA_travelTime_with_SD_2way))) ||
size(find(A2A_CA_travelTime_with_SD_2way==0),1) >443

% checking NaN travel time without SD
if sum(sum(isnan(A2A_CA_travelTime_no_SD_2way))) ||
size(find(A2A_CA_travelTime_no_SD_2way==0),1) >443
    disp('Error: Travel Time CA')
    pause
end % if sum(sum(isnan(A2A_CA_travelTime_no_SD_2way))) ||
size(find(A2A_CA_travelTime_no_SD_2way==0),1) >443

% checking NaN SD
if sum(sum(isnan(A2A_CA_Schedule_Delay_2way))) ||
size(find(A2A_CA_Schedule_Delay_2way==0),1) >443
    disp('Error: Travel Time CA')
    pause
end % if sum(sum(isnan(A2A_CA_Schedule_Delay_2way))) ||
size(find(A2A_CA_Schedule_Delay_2way==0),1) >443

% checking NaN access time
if sum(sum(isnan(CA_candArpts100_driveTime))) || sum(sum(isnan(CA_candArpts200_driveTime)))
    disp('Error: Access Time CA')
    pause
end % if sum(sum(isnan(CA_candArpts100_driveTime))) ||
sum(sum(isnan(CA_candArpts200_driveTime)))

% checking zero access distance 1
if size(find(CA_candArpts100_driveDist<0 & CA_candArpts100_driveDist~=-9999),1)
    disp('Error in Access Distance 1')
end

% checking zero access distance 2
if size(find(CA_candArpts200_driveDist<0 & CA_candArpts200_driveDist~=-9999),1)
    disp('Error in Access Distance 2')
end

% checking NaN drive distance
if sum(sum(isnan(CA_candArpts100_driveDist))) || sum(sum(isnan(CA_candArpts200_driveDist)))
    disp('Error: Access Distance')
    pause
```

```
end % if sum(sum(isnan(CA_candArpts100_driveDist))) ||
sum(sum(isnan(CA_candArpts200_driveDist)))

% checking NaN fares
if sum(sum(isnan(A2A_CA_BusinessClass_Fares_2000))) ||
sum(sum(isnan(A2A_CA_CoachClass_Fares_2000)))
    disp('Error: CA Fares')
    pause
end % if sum(sum(isnan(CA_candArpts100_driveDist))) ||
sum(sum(isnan(CA_candArpts200_driveDist)))

% checking zero coach fares
if size(find(A2A_CA_CoachClass_Fares_2000==0),1) > 443
    disp('Error with fares from curve')
end

% checking NaN coach fare index
if sum(sum(sum(isnan(A2A_CA_CoachFare_Index_2000))))
    disp('Error with fares from curve')
end

disp(['Elasped Time: ', num2str(toc)])
%--------------------------------------------------------------------------------

% variables and switches
costPerMile_auto = 0.3;
Project_Dir      = 'D:\VLJ_Project';
StateNumber      = 0; %0 is for ALL the states.
saveDirName      = 'D:\Mode Choice\Calibration_February_07\';
scenarioName     = 'Nested Logit Travel Times and Costs\';
Year             = 2000;

%--------------------
% VLJ Options
%--------------------
Install_Dir_VLJ = 'C:\Program Files\TSAM\4.2\data';
VLJ_YesNo       = 'No';

maxDailyDriveTime_Auto_vb  = [8 10];
maxAccessEgressTime_VLJ_vb = [4.0 4.0];

VLJAirportsProcessingTime_vb(1,1) = 0.5;      % Origin hrs %45/60
VLJAirportsProcessingTime_vb(1,2) = 0.25;      % Destination hrs%35/60
costPerMile_Auto_vb  = 0.37;
costPerMile_VLJ_vb   = 1.75;
scheduleDelay_VLJ_vb = 1.0;
FlightSpeedProfileFilename = 'VLJ_RANGE_1100_MAXIMUMALTITUDE_400_PROFILE';
CostProfileFilename  = 'NONE';

drivingSpeed_Urban = 30; % mph
drivingSpeed_Rural = 60; % mph
```

```
AirportSetName     = 'FULL_With_OEP';

if strcmp(VLJ_YesNo, 'Yes')  == 1
    VLJSwitch = 2;
    maxAccessTime_VLJ  = maxAccessEgressTime_VLJ_vb(1);
    maxEgressTime_VLJ  = maxAccessEgressTime_VLJ_vb(2);
    min_A2A_Distance_VLJ = 75; % statute miles
    waitingAtOrginAirport_VLJ = VLJAirportsProcessingTime_vb(1);
    waitingAtDestAirport_VLJ  = VLJAirportsProcessingTime_vb(2);
    scheduleDelay_VLJ         = scheduleDelay_VLJ_vb; % hrs
    costPerMile_VLJ           = double(costPerMile_VLJ_vb);

    %FLight Speed Profile
    load (strcat(Install_Dir_VLJ, '\bada_library\', FlightSpeedProfileFilename, '.mat'));
    min_FSP_Distance_VLJ = FlightSpeedProfile(2, 1);
    max_FSP_Distance_VLJ = FlightSpeedProfile(2, end);
    min_FSP_Speed_VLJ = FlightSpeedProfile(1, 1);
    max_FSP_Speed_VLJ = FlightSpeedProfile(1, end);

    % Cost Profile
    if strcmp(CostProfileFilename, 'NONE') == 0
        CostProfile_Filename = fopen(strcat(Install_Dir_VLJ, '\cost_profiles\', CostProfileFilename, '.cp'),
'r');
        CostProfile = cell2mat(textscan(CostProfile_Filename, '%f %f %f', 'delimiter', ','));
        fclose(CostProfile_Filename);
        min_CP_Distance_VLJ = CostProfile(1, 2);
        max_CP_Distance_VLJ = CostProfile(end, 2);
        min_CP_Cost_VLJ = CostProfile(1, 3);
        max_CP_Cost_VLJ = CostProfile(end, 3);
        ConstantVLJCost = 0;
    else
        ConstantVLJCost = 1;
    end % if strcmp(CostProfileFilename, 'NONE') == 0

    %VLJ Airport Set Selection
    if strcmp(AirportSetName, 'FULL_Without_OEP') == 1 %FULL Without OEP
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\C2AData_FULL_Without_OEP.mat'));
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\distA2A_FULL_Without_OEP.mat'));  % Airport
to airport distance table for VLJ airports (3346x3346)
        c2aData_VLJ = C2AData_FULL_Without_OEP;
        distA2A_VLJ = distA2A_FULL_Without_OEP;
        clear C2AData_FULL_Without_OEP;
        clear distA2A_FULL_Without_OEP;
    elseif strcmp(AirportSetName, 'FULL_With_OEP') == 1 % FULL With OEP
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\C2AData_FULL_With_OEP.mat'));
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\distA2A_FULL_With_OEP.mat'));  % Airport to
airport distance table for VLJ airports (3346x3346)
        c2aData_VLJ = C2AData_FULL_With_OEP;
        distA2A_VLJ = distA2A_FULL_With_OEP;
        clear C2AData_FULL_With_OEP;
        clear distA2A_FULL_With_OEP;
    elseif strcmp(AirportSetName, 'ILS_Without_OEP') == 1 %ILS_Without_OEP Airport Set
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\C2AData_ILS_Without_OEP.mat'));
        load (strcat(Install_Dir_VLJ, '\mode_choice\input\distA2A_ILS_Without_OEP.mat')) % Airport to
airport distance table for VLJ airports (3346x3346)
        c2aData_VLJ = C2AData_ILS_Without_OEP;
        distA2A_VLJ = distA2A_ILS_Without_OEP;
        clear C2AData_ILS_Without_OEP;
        clear distA2A_ILS_Without_OEP;
    elseif strcmp(AirportSetName, 'LLM') == 1 %LLM Airport Set
        load (strcat(Project_Dir, '\mode_choice\input\LLM\', LLMAirportSetName, '\C2AData_LLM.mat'));
        load (strcat(Project_Dir, '\mode_choice\input\LLM\', LLMAirportSetName, '\distA2A_LLM.mat'))
% Airport to airport distance table for VLJ airports (3346x3346)
        c2aData_VLJ = C2AData_LLM;
        distA2A_VLJ = distA2A_LLM;
        clear C2AData_LLM;
        clear distA2A_LLM;
    elseif strcmp(AirportSetName, 'Custom') == 1 %Custom Airport Set
        load (strcat(Install_Dir_VLJ, '\custom_airport_sets\', CustomAirportSetName,
'\C2AData_Custom.mat'));
        load (strcat(Install_Dir_VLJ, '\custom_airport_sets\', CustomAirportSetName,
'\distA2A_Custom.mat')) % Airport to airport distance table for SABTechnology airports (3346x3346)
        c2aData_VLJ = C2AData_Custom;
        distA2A_VLJ = distA2A_Custom;
        clear distA2A_Custom;
        clear C2AData_Custom;
    end % if strcmp(AirportSetName, 'FULL_Without_OEP') == 1 %FULL Without OEP
else
    VLJSwitch = 1;
    FlightSpeedProfileVLJ     = [];
    min_FlightSpeedProfileVLJ = [];
    max_FlightSpeedProfileVLJ = [];
    min_A2A_Distance_VLJ      = [];
end % if strcmp(VLJ_YesNo, 'Yes')  == 1


%-------------------------
% Initialize Contants
%-------------------------
numberOfIncomeGroups      = size(avgPartySize_ATS_Buss, 1);
waitingAtOriginAirport_CA = [2.00 1.50 1.25 1.00]; % Waiting time in hours
waitingAtDestntAirport_CA = [0.75 0.75 0.50 0.50]; % Waiting time in hours
%-----------------------------------------------------------------------------

% Loop for trip purpose
for tripType = 1 : 2

    %-------------
    % constants
    %-------------
    if tripType == 1
        avgOccupancy_auto         = mean(avgPartySize_ATS_Buss, 2);
        averageRoundTripTime_auto = 16;
        costPerMile_auto          = 0.37;
        caseName                  = 'business\';
        maxRoundTripTime_CA       = 18;
        maxRoundTripTime_CA_2     = 21;
        minRoundTripTime_CA       = 14;
        maxDailyDriveTime_Auto    = maxDailyDriveTime_Auto_vb(1,1);
        oneDayLodgingCost         = [70 80 90 100 120];
    else
        avgOccupancy_auto         = mean(avgPartySize_ATS_NonBuss, 2);
        averageRoundTripTime_auto = 20;
        costPerMile_auto          = 0.15;
        caseName                  = 'nonBusiness\';
        maxRoundTripTime_CA       = 20;
        maxRoundTripTime_CA_2     = 22;
        minRoundTripTime_CA       = 18;
        maxDailyDriveTime_Auto    = maxDailyDriveTime_Auto_vb(1,2);
        oneDayLodgingCost         = [50 60 70 80 90];
    end % if tripPuroposeSwitch == 1;

    avgOccupancy_Auto = avgOccupancy_auto;
    costPerMile_Auto  = costPerMile_auto;

    additionalOvernightTime_perDay_CA   = 24 - minRoundTripTime_CA; % hrs
    additionalOvernightTime_perDay_auto = 24 - averageRoundTripTime_auto/2; % hrs
    averageDaily_LogdgingCost           = sum(oneDayLodgingCost)/5;
    disp(['Elasped Time: ', num2str(toc)])

    additionalDay_Auto = additionalOvernightTime_perDay_auto;

    inputProjDir = (strcat(Project_Dir, '\Trip_Distribution\Output\Y', num2str(Year)));
    mkdir (strcat(saveDirName, scenarioName, caseName));
    row_counter = 0;

    for stateCounter = 1 : 52
        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 3) == 0;
                disp(['Origin State: ', num2str(stateCounter), ' time: ', num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            stateName = char(stateNames(stateCounter, 1));
            starter   = stateIndexing(stateCounter, 1);

            startRow = stateIndexing(stateCounter, 1);
            endRow   = stateIndexing(stateCounter, 2);

            % County to County drive times (using mappoint)
            load (strcat(Install_Dir, '\input\C2CDriveTime_Population_Centroids'))
```

```
          C2CDriveTime_Population_Centroids =
C2CDriveTime_Population_Centroids(startRow:endRow, :);

          % County to County drive distance (using mappoint)
          load (strcat(Install_Dir, '\input\C2CDriveDist_Population_Centroids'))
          C2CDriveDist_Population_Centroids = C2CDriveDist_Population_Centroids(startRow:endRow,
:);

          % County to county distance table (Upper triangular) use when mapquest cannot compute drive
times
          load (strcat(Install_Dir, '\input\distC2C_Population_Centroids'))
          distC2C_Population_Centroids = distC2C_Population_Centroids(startRow:endRow, :);

          nRows = endRow - startRow + 1;

          if StateNumber == 0 %All States
             C2C_TT_TC_Aggt_noDemand_modular(VLJSwitch, starter, stateName, 1, 3091,
saveDirName, nRows, tripType);
          else % One State
             % Finding the starting and ending index of the state
             StateStartIndex = stateIndexing(StateNumber, 1);
             StateEndIndex   = stateIndexing(StateNumber, 2);
             if StateNumber == stateCounter % Selected State
                C2C_TT_TC_Aggt_noDemand_modular(starter, stateName, 1, 3091, saveDirName, nRows);
             else
                C2C_TT_TC_Aggt_noDemand_modular(starter, stateName, StateStartIndex, StateEndIndex,
saveDirName, nRows);
             end % if StateNumber == stateCounter % Selected State
          end % if StateNumber == 0 %All States
       end % if (stateCounter == 2 | stateCounter == 12)
   end % for stateCounter = 1 : 52

end % for tripType = 1 : 2

toc
call_C2C_TT_TC_Aggrt_demandBased_modular()

return
--------------------------------------------------------------------------------------------------------------
```

#### 7.4.1.1 Compute Automobile and Commercial Air Travel Times and Costs

```
-------------------------------------------------------------------------------------------------------------------------

function C2C_TT_TC_Aggt_noDemand_modular(VLJSwitch, starter, stateName, StateStartIndex,
StateEndIndex, saveDirName, numbRows, TripPurpose)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Pre-process CA and Auto travel time tables
% (Demand-based: only inter-city trips with demand)
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global C2CDriveDist_Population_Centroids caseName scenarioName

saveDirectoryName = [saveDirName scenarioName caseName stateName];

% Preprocess driving cost and time
income          = [25000 45000 80000 125000 170000]; % mean/median values of household incomes
[c2cAuto_TT_2way, c2cAuto_TC_2way, c2cAuto_meanTC_2way, c2cAuto_numbOfNights_2way] =
ComputeAutoCostAndTime(income, TripPurpose);
clear c2cAuto_numbOfNights_2way

save ([saveDirectoryName, '_c2cAuto_TT_2way'], 'c2cAuto_TT_2way')
% c2cAuto_TravelTime_2way -> total travel time including ovevernight stay
save ([saveDirectoryName, '_c2cAuto_TC_2way'], 'c2cAuto_TC_2way')
% TrvlCost_Auto_2way -> total travel cost including lodging cost
save ([saveDirectoryName, '_c2cAuto_meanTC_2way'], 'c2cAuto_meanTC_2way')
% c2cAuto_TravelCost_2way -> mean travel cost
% save ([saveDirectoryName, '_c2cAuto_numbOfNights_2way'], 'c2cAuto_numbOfNights_2way')
% % numberOfDays_2way -> number of days

%---------------------
% Table Structure
%---------------------
% c2cAuto_TT_2way(nRows,3091)            -> Automobile C2C travel time
% c2cAuto_numbOfNights_2way(nRows,3091)   -> Automobile C2C number of days
% c2cAuto_TC_noOccupancy_2way(nRows,3091) -> Automobile C2C travel cost without occupancy
% nRows is the number of counties in origin State

number_of_airports = 3;
% initial_index      = [1 2 4 5 3 7 3 6 8 9];
CA_index                     = zeros(number_of_airports, numbRows, StateEndIndex);
CA_Schedule_Delay_2way      = CA_index;
CA_travelTime_with_SD_2way  = CA_index;
CA_numberOfNights_2way      = CA_index;
CA_flightTime_no_SD_2way    = CA_index;

CA_businessFare_2way  = CA_index;
CA_CoachFare_2way     = CA_index;
CA_CoachFare_Index    = CA_index;
```

```
CA_accEgrsCost_2way   = CA_index;

initial_index(:,1)  = [1 2 4 5 3 7  6 8  9  0  0  0  0  0  0  0];
initial_index(:,3)  = [1 2 4 3 7 10 5 6  8  11 9  12 0  0  0  0];
initial_index(:,2)  = [1 2 5 6 3 9  4 7  10 8  11 12 0  0  0  0];
initial_index(:,4)  = [1 2 5 3 9 6  4 13 7  10 8  11 14 12 15 16];

load ([saveDirectoryName, '_c2cAuto_TT_2way'])
load ([saveDirectoryName, '_c2cAuto_TC_2way'])

%---------------------------------------------------
% Start Mode Split
%---------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
   orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct county number

   % keep track of location in run
   if mod(orgCountyReal, 30) == 0; disp(['Origin County: ', num2str(orgCountyReal), ' time: ',
num2str(toc) ]); end

   alaskaOrigin = ~isempty(find(orgCountyReal == 68:78));
   hawaiiOrigin = ~isempty(find(orgCountyReal == 527:530));

   if (alaskaOrigin == 1 || hawaiiOrigin == 1)
   else
      for desCounty =  StateStartIndex : StateEndIndex %1 : numbCols % loop for destination counties

         % if mod(desCounty, 300) == 0; disp(['Destination County: ', num2str(desCounty), ' of ',
num2str(numbRows), ' time: ', num2str(toc) ]); end

         alaskaDestnt = ~isempty(find(desCounty == 68:78));
         hawaiiDestnt = ~isempty(find(desCounty == 527:530));
         routeDistance = C2CDriveDist_Population_Centroids(orgCounty, desCounty);

         if alaskaDestnt == 1 || hawaiiDestnt == 1
         else
            if routeDistance >= 100 && routeDistance < 3500

               number_of_routes = 3;
               TrvlTime_Auto_2way = c2cAuto_TT_2way(orgCounty, desCounty);
               max_Auto_cost_2way = c2cAuto_meanTC_2way(1,orgCounty, desCounty);

               % Select candidate CA routes
               %----------------------------------------------------
               [CA_routeInfo, CA_travelTime, CA_travelCost, loopType] =
select_CA_CandidateRoutes_modular(orgCountyReal, desCounty, ...
                  TrvlTime_Auto_2way, max_Auto_cost_2way);
               %-> CA_routeInfo(:,1) = Origin County Airports
               %-> CA_routeInfo(:,2) = Destination County Airports
               %-> CA_routeInfo(:,3) = Position in Table
               %
```

```
%-> CA_travelTime(:,1) = Door-2-Door CA Travel Time (with Schedule Delay)
%-> CA_travelTime(:,2) = Schedule Delay (Round Trip)
%-> CA_travelTime(:,3) = Number of Nights (Round Trip)
%  door-to-door => AccessTime + WaitingTime_org + FlyingTime + ScheduleDelay +
%                          WaitingTime_dest + EgressTime)
%
%-> CA_travelCost(:,1) = Access Cost (without party size)
%-> CA_travelCost(:,2) = Egress Cost (without party size)
%-> CA_travelCost(:,3) = Average Business Class Fare
%-> CA_travelCost(:,4) = Average Coach Class Fare
%-> CA_travelCost(:,5) = Average Coach Fare Index
%-----------------------------------------------------

if sum(sum(CA_routeInfo)) == 0;
else
   selected_row    = [];
   fill_counter    = 1;
   airport_counter = 0;

   if loopType == 1
      initial_index_new = initial_index(1:9,loopType);
   elseif loopType == 2 || loopType == 3
      initial_index_new = initial_index(1:12,loopType);
   elseif loopType == 4
      initial_index_new = initial_index(:,loopType);
   end % if loopType == 1

   while fill_counter <= number_of_routes
      airport_counter = airport_counter + 1;
      airportPosition = initial_index_new(airport_counter);
      airportIndex    = find(CA_routeInfo(:,3) == airportPosition);
      if isempty(airportIndex)
      else
         selected_row(1,fill_counter) = airportIndex;
         fill_counter = fill_counter + 1;
      end % if isempty(find(CA_routeInfo(3,:) == airportPosition))
      if airport_counter == size(initial_index_new,1)
         break
      end % if airport_counter == 9
   end % while fill_counter <= number_of_routes
   number_of_routes = size(selected_row,2);

   %--------------------------------------------------------------------------------------------
   % airport index
   CA_index(1:number_of_routes, orgCounty, desCounty) = CA_routeInfo(selected_row,3);

   %CA ground times
   CA_Schedule_Delay_2way(1:number_of_routes, orgCounty, desCounty)     =
CA_travelTime(selected_row,1);
   CA_travelTime_with_SD_2way(1:number_of_routes, orgCounty, desCounty) =
CA_travelTime(selected_row,2);
```

```
                        CA_numberOfNights_2way(1:number_of_routes, orgCounty, desCounty)     =
CA_travelTime(selected_row,3);
                        CA_flightTime_no_SD_2way(1:number_of_routes, orgCounty, desCounty)   =
CA_travelTime(selected_row,4);

                        % CA flight info: cost
                        CA_accEgrsCost_2way(1:number_of_routes, orgCounty, desCounty)   = 2 *
sum(CA_travelCost(selected_row,1:2),2);
                        CA_businessFare_2way(1:number_of_routes, orgCounty, desCounty)  =
CA_travelCost(selected_row,3);
                        CA_CoachFare_2way(1:number_of_routes, orgCounty, desCounty)     =
CA_travelCost(selected_row,4);
                        CA_CoachFare_Index(1:number_of_routes, orgCounty, desCounty)    =
CA_travelCost(selected_row,5);
                        %-------------------------------------------------------------------------------------------
                    end % if sum(sum(CA_routeInfo)) = 0;
                end % if routeDistance >= 100
            end % if alaskaDestnt == 1 || hawaiiDestnt == 1
        end % for desCounty = StateStartIndex : StateEndIndex
    end % if (alaskaOrigin == 1 || hawaiiOrigin == 1)
end % for orgCounty = 1 : numbRows % loop for origin counties

save (strcat(saveDirectoryName, '_CA_index'), 'CA_index') % Index of airport routes

save (strcat(saveDirectoryName, '_CA_Schedule_Delay_2way'), 'CA_Schedule_Delay_2way') % 2-way
schedule delay
save (strcat(saveDirectoryName, '_CA_travelTime_with_SD_2way'), 'CA_travelTime_with_SD_2way')
% 2-way travel time without Number of nights
save (strcat(saveDirectoryName, '_CA_numberOfNights_2way'), 'CA_numberOfNights_2way') % 2-way
number of nights
save (strcat(saveDirectoryName, '_CA_flightTime_no_SD_2way'), 'CA_flightTime_no_SD_2way') % 2-
way number of nights

save (strcat(saveDirectoryName, '_CA_accEgrsCost_2way'), 'CA_accEgrsCost_2way') % 2-way access
egress cost
save (strcat(saveDirectoryName, '_CA_businessFare_2way'), 'CA_businessFare_2way') % business fare
save (strcat(saveDirectoryName, '_CA_CoachFare_2way'), 'CA_CoachFare_2way') % coach fare
save (strcat(saveDirectoryName, '_CA_CoachFare_Index'), 'CA_CoachFare_Index') % index of fares
from curves

%---------------------
% Table Structure
%---------------------
% CA_roundTrip_travelTime(3,5,nRows,3091) -> County-to-county Travel Time for shortest, and 2
cheapest routes for 5 income groups
% CA_travelCost(3,5,nRows,3091) -> County-to-county Travel Cost for shortest, and 2 cheapest routes
for 5 income groups
% nRows => Number of counties in origin State
```

---------------------------------------------------------------------------------------------------------

#### 7.4.1.2 Compute Travel Time for Commercial Air Routes

```
--------------------------------------------------------------------------------------------------------------

function [CA_airportInfo, CA_travelTime, CA_travelCost, loopType] = ...
   select_CA_CandidateRoutes_modular(orgCountyReal, desCounty, TrvlTime_Auto_2way,
max_Auto_cost_2way)

%---------------------------------------------------------------------------
% This script computes access and egress times and derives in-vehicle time
% for commercial aviation trips as
% In-vehicle-time = AccTime + EggressTime + LinehaulTime
%
% Assumptions: Waiting time is the same for all airports in US = 1.45hrs
%-----------------------------------------------------------------------

% initialize tables
global A2A_CA_BusinessClassFare_DB1B_2000 A2A_CA_CoachClassFare_Filled_DB1B_2000
A2A_CA_CoachFare_Index_DB1B_2000
global CA_candArpts_driveTime CA_candArpts_driveDist CA_candArpts_arptNumb
global A2A_CA_travelTime_with_SD_2way A2A_CA_travelTime_no_SD_2way
A2A_CA_Schedule_Delay_2way
global distA2A_CA msaIndicatorData MinNumberOfLegs waitingAtOriginAirport_CA
waitingAtDestntAirport_CA

% global CA_candArpts100_hubType CA_candArpts200_hubType

% initialize constants
global additionalOvernightTime_perDay_CA costPerMile_auto CA_LOOP minRoundTripTime_CA
global maxRoundTripTime_CA maxRoundTripTime_CA_2

% MSA to MSA
if msaIndicatorData(orgCountyReal, 2) == 1 && msaIndicatorData(desCounty, 2) == 1
   orgCA_candArpts_arptNumb   = CA_candArpts100_arptNumb;
   orgCA_candArpts_driveTime  = CA_candArpts100_driveTime;
   orgCA_candArpts_driveDist  = CA_candArpts100_driveDist;

   desCA_candArpts_arptNumb   = orgCA_candArpts_arptNumb;
   desCA_candArpts_driveTime  = orgCA_candArpts_driveTime;
   desCA_candArpts_driveDist  = orgCA_candArpts_driveDist;

   % MSA to NON-MSA
elseif msaIndicatorData(orgCountyReal, 2) == 1 && msaIndicatorData(desCounty, 2) == 0
   orgCA_candArpts_arptNumb   = CA_candArpts100_arptNumb;
   orgCA_candArpts_driveTime  = CA_candArpts100_driveTime;
   orgCA_candArpts_driveDist  = CA_candArpts100_driveDist;

   desCA_candArpts_arptNumb   = CA_candArpts200_arptNumb;
   desCA_candArpts_driveTime  = CA_candArpts200_driveTime;
   desCA_candArpts_driveDist  = CA_candArpts200_driveDist;
```

```
    % NON-MSA to MSA
elseif msaIndicatorData(orgCountyReal, 2) == 0 && msaIndicatorData(desCounty, 2) == 1
    orgCA_candArpts_arptNumb   = CA_candArpts200_arptNumb;
    orgCA_candArpts_driveTime  = CA_candArpts200_driveTime;
    orgCA_candArpts_driveDist  = CA_candArpts200_driveDist;

    desCA_candArpts_arptNumb   = CA_candArpts100_arptNumb;
    desCA_candArpts_driveTime  = CA_candArpts100_driveTime;
    desCA_candArpts_driveDist  = CA_candArpts100_driveDist;

    % NON-MSA to NON-MSA
elseif msaIndicatorData(orgCountyReal, 2) == 0 && msaIndicatorData(desCounty, 2) == 0
    orgCA_candArpts_arptNumb   = CA_candArpts200_arptNumb;
    orgCA_candArpts_driveTime  = CA_candArpts200_driveTime;
    orgCA_candArpts_driveDist  = CA_candArpts200_driveDist;

    desCA_candArpts_arptNumb   = orgCA_candArpts_arptNumb;
    desCA_candArpts_driveTime  = orgCA_candArpts_driveTime;
    desCA_candArpts_driveDist  = orgCA_candArpts_driveDist;
end % if msaIndicatorData(orgCountyReal, 2) == 1 && msaIndicatorData(desCounty, 2) == 1


%-----------------
% Start Analysis
%-----------------
ROW_COUNTER = 0;
CA_airportInfo = zeros(9,3);
CA_travelTime  = zeros(9,3);
CA_travelCost  = zeros(9,5);
CA_LOOP        = [1 2 3; 4 5 6; 7 8 9];
loopType       = 1;

org_arptNumbers = orgCA_candArpts_arptNumb(orgCountyReal, :);
des_arptNumbers = desCA_candArpts_arptNumb(desCounty, :);

nOrgArpt     = size(org_arptNumbers, 2);
nDesArpt     = size(des_arptNumbers, 2);

orgArpt_info = zeros(1,nOrgArpt);
desArpt_info = zeros(1,nDesArpt);

for k = 1 : nOrgArpt
    if org_arptNumbers(k) == -9999
    else
        orgArpt_info(1, k) = CA_Hub_Type(org_arptNumbers(k));
    end % if org_arptNumbers(k) == -9999
end % for k = 1 : nOrgArpt

for m = 1 : nDesArpt
    if des_arptNumbers(m) == -9999
    else
        desArpt_info(1, m) = CA_Hub_Type(des_arptNumbers(m));
    end % if des_arptNumbers(m) == -9999
end % for k = 1 : nOrgArpt

org_LH_index = find(orgArpt_info == 1); % check for presence of large hub
des_LH_index = find(desArpt_info == 1); % check for presence of large hub

for i = 1 : nOrgArpt

    for j = 1 : nDesArpt
        orgArptNumber  = orgCA_candArpts_arptNumb(orgCountyReal, i);
        desArptNumber  = desCA_candArpts_arptNumb(desCounty, j);

        if orgArptNumber == -9999 || desArptNumber == -9999
        else
            if orgArptNumber ~= desArptNumber
                orgArpt_hubType = CA_Hub_Type(orgArptNumber);
                desArpt_hubType = CA_Hub_Type(desArptNumber);
                A2A_distance    = distA2A_CA(orgArptNumber, desArptNumber);

                if A2A_distance > 75 || (orgArpt_hubType == 1 & desArpt_hubType == 1)
                    loop_value = 0;

                    % if there is a non-hub competing with large hub at origin
                    if orgArpt_hubType == 4 & sum(org_LH_index) > 0
                        minimumConnection = MinNumberOfLegs(orgArptNumber, desArptNumber);
                        curveIndex        = sum(A2A_CA_CoachFare_Index_2000(orgArptNumber,
desArptNumber,:));
                        % if non-hub has a direct flight OR its fare is from DB1B
                        if (minimumConnection == 1) || (curveIndex == 0)
                        else
                            loop_value = 1;
                        end % if (minimumConnection == 1) || (curveIndex == 0)
                    end % if orgArpt_hubType == 4 & sum(org_LH_index) > 0

                    if loop_value == 1
                    else
                        % if there is a non-hub competing with large hub at origin
                        if desArpt_hubType == 4 & sum(des_LH_index) > 0
                            minimumConnection = MinNumberOfLegs(desArptNumber, orgArptNumber);
                            curveIndex        = sum(A2A_CA_CoachFare_Index_2000(desArptNumber,
orgArptNumber,:));
                            % if non-hub has a direct flight OR at fare is from DB1B
                            if (minimumConnection == 1) || (curveIndex == 0)
                            else
                                loop_value = 1;
                            end % if (minimumConnection == 1) || (curveIndex == 0)
                        end % if orgArpt_hubType == 4 & sum(org_LH_index) > 0

                        if loop_value == 1
                        else
                            ROW_COUNTER = ROW_COUNTER + 1;

                            % a) Access-Egress Times
                            access_time = orgCA_candArpts_driveTime(orgCountyReal, i);
                            egress_time = desCA_candArpts_driveTime(desCounty, j);
                            roundTrip_groundTime = 2 * (access_time + egress_time);

                            % b) Processing Times
                            outbound_ProcTime   = waitingAtOriginAirport_CA(1, orgArpt_hubType) +
waitingAtDestntAirport_CA(1, desArpt_hubType);
                            inbound_ProcTime    = waitingAtOriginAirport_CA(1, desArpt_hubType) +
waitingAtDestntAirport_CA(1, orgArpt_hubType);
                            roundTrip_processingTimes  = outbound_ProcTime + inbound_ProcTime;

                            % c) Line-haul time
                            roundTrip_lineHaulTime_with_SD =
A2A_CA_travelTime_with_SD_2way(orgArptNumber, desArptNumber);
                            flightTime_no_SD              = A2A_CA_travelTime_no_SD_2way(orgArptNumber,
desArptNumber);
                            checkScheduleDelay            = A2A_CA_Schedule_Delay_2way(orgArptNumber,
desArptNumber);

                            % (a) + (b) + (c)
                            travelTime_CA_with_SD_2way = roundTrip_lineHaulTime_with_SD +
roundTrip_groundTime + roundTrip_processingTimes;


                            % Access Costs
                            access_distance       = orgCA_candArpts_driveDist(orgCountyReal, i);
                            egress_distance       = desCA_candArpts_driveDist(desCounty, j);

                            if (access_distance < 0) || (egress_distance < 0)
                                orgArptNumber
                                desArptNumber
                                disp('Error with access cost')
                                pause
                            end % if (accessCost_noPartySize == 0)

                            accessCost_noPartySize = access_distance * costPerMile_auto;
                            egressCost_noPartySize = egress_distance + 3; % assume taxi is used and taxi costs $3
+ $1/mile

                            businessClassFare = A2A_CA_BusinessClass_Fares_2000(orgArptNumber,
desArptNumber);
                            coachClassFare    = A2A_CA_CoachClass_Fares_2000(orgArptNumber,
desArptNumber);

                            if businessClassFare < coachClassFare
                                businessClassFare = coachClassFare;
                            end % if businessClassFare < coachClassFare
                            indexOfFaresFromCurve = A2A_CA_CoachFare_Index_2000(orgArptNumber,
desArptNumber);

                            % overnight time and costs
                            CA_numberOfNights = 0;
                            if travelTime_CA_with_SD_2way > minRoundTripTime_CA &&
travelTime_CA_with_SD_2way <= maxRoundTripTime_CA
                           rampWidth_x       = maxRoundTripTime_CA - minRoundTripTime_CA;
                           rampHeight_y      = 1;
                           travelTime_x      = travelTime_CA_with_SD_2way - minRoundTripTime_CA;
                           CA_numberOfNights = rampHeight_y * (travelTime_x / rampWidth_x);
                            elseif travelTime_CA_with_SD_2way > maxRoundTripTime_CA &&
travelTime_CA_with_SD_2way <= maxRoundTripTime_CA_2 % two night stay
                           rampWidth_x       = maxRoundTripTime_CA_2 - maxRoundTripTime_CA;
                           rampHeight_y      = 1;
                           travelTime_x      = travelTime_CA_with_SD_2way - maxRoundTripTime_CA;
                           CA_numberOfNights = rampHeight_y * (travelTime_x / rampWidth_x);
                           CA_numberOfNights = CA_numberOfNights + 1;
                            elseif travelTime_CA_with_SD_2way > maxRoundTripTime_CA_2
                           CA_numberOfNights = 2;
                            end % if CA_travelTime_roundTrip > minRoundTripTime_CA &&

                            % save data
                            CA_airportInfo(ROW_COUNTER, 1) = orgArptNumber;
                            CA_airportInfo(ROW_COUNTER, 2) = desArptNumber;
                            CA_airportInfo(ROW_COUNTER, 3) = CA_LOOP(i,j);

                            CA_travelTime(ROW_COUNTER, 1)  = checkScheduleDelay;
                            CA_travelTime(ROW_COUNTER, 2)  = travelTime_CA_with_SD_2way;
                            CA_travelTime(ROW_COUNTER, 3)  = CA_numberOfNights;
                            CA_travelTime(ROW_COUNTER, 4)  = flightTime_no_SD;

                            CA_travelCost(ROW_COUNTER, 1)  = accessCost_noPartySize;
                            CA_travelCost(ROW_COUNTER, 2)  = egressCost_noPartySize;
                            CA_travelCost(ROW_COUNTER, 3)  = businessClassFare;
                            CA_travelCost(ROW_COUNTER, 4)  = coachClassFare;
                            CA_travelCost(ROW_COUNTER, 5)  = indexOfFaresFromCurve;
                         end % if loop_value == 1
                      end % if loop_value == 1
                   end % if A2A_distance > 75 || (orgArpt_hubType == 1 & desArpt_hubType == 1)
                end % if orgArptNumber ~= desArptNumber
             end % if orgArptNumber == -9999 || desArptNumber == -9999
       end % for i = 1 : size(orgCntArptData, 1)
end % for i = 1 : size(destCntArptData, 1)

SUM_AIRPORTS      = sum(CA_airportInfo, 2);
invalid_Routes_Index = find(SUM_AIRPORTS == 0);

CA_airportInfo(invalid_Routes_Index, :) = [];
CA_travelTime(invalid_Routes_Index, :)  = [];
```

```
CA_travelCost(invalid_Routes_Index, :)  = [];

if ~isempty(CA_airportInfo)
   TrvlTime_CA_2way = CA_travelTime(:,2) + CA_travelTime(:,3) *
additionalOvernightTime_perDay_CA;
   TrvlTime_CA_2way = round(TrvlTime_CA_2way * 10) / 10;

   % Make sure there is at least one valid CA route
   Valid_CA_Routes  = find(TrvlTime_CA_2way <= TrvlTime_Auto_2way);

   scaleFactor = 1.3;
   while isempty(Valid_CA_Routes);
      Valid_CA_Routes  = find(TrvlTime_CA_2way <= TrvlTime_Auto_2way * scaleFactor);
      scaleFactor      = scaleFactor + 0.2;
   end % while isempty(candArptMatrix);

   TrvlTime_CA_2way  = TrvlTime_CA_2way(Valid_CA_Routes);
   CA_airportInfo    = CA_airportInfo(Valid_CA_Routes, :);
   CA_travelTime     = CA_travelTime(Valid_CA_Routes, :);
   CA_travelCost     = CA_travelCost(Valid_CA_Routes, :);

   if scaleFactor > 1.3

      CA_Auto_time_ratio = TrvlTime_CA_2way./TrvlTime_Auto_2way;
      meanFare_findzero  = find(CA_travelCost(:,3) > 0);
      meanFare_data      = CA_travelCost(meanFare_findzero,:);
      meanCost           = meanFare_data(:,1) + meanFare_data(:,2) + meanFare_data(:,3)*.25 +
meanFare_data(:,4)*.75;
      CA_Auto_cost_ratio = meanCost/max_Auto_cost_2way;
      deleteIndex = [];

      for k = 1 : length(CA_Auto_time_ratio)
         if CA_Auto_time_ratio(k) > 2 & CA_Auto_cost_ratio(k) > 2
            deleteIndex(k) = k;
         end % if CA_Auto_time_ratio(k) > 2 & CA_Auto_cost_ratio(k) > 2
      end % for k = 1 : length(CA_Auto_time_ratio)

      if isempty(deleteIndex)
      else
         CA_airportInfo(nonzeros(deleteIndex),:)  = [];
         CA_travelTime(nonzeros(deleteIndex),:)   = [];
         CA_travelCost(nonzeros(deleteIndex),:)   = [];
      end % if isempty(deleteIndex)
   end % if scaleFactor > 1.3
end % if isempty(CA_airportInfo)
```

---------------------------------------------------------------------------------------------------------------

### 7.4.2 *Aggregate County-level Travel Times and Costs to by State, Region and Distance (Weight by Person Trips between County pairs)*

Sub-Functions:
C2C_TT_TC_Aggrt_demandBased_modular_MSA_MSA
C2C_TT_TC_Aggrt_demandBased_modular_MSA_nonMSA
C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_MSA
C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_nonMSA

-------------------------------------------------------------------------------------------------------------------------------------------

```
function call_C2C_TT_TC_Aggrt_demandBased_modular()

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Viginia Tech. Air Transportation Systems Lab., Blacksburg, VA
%
% Crearted By:   Senanu Ashiabor
% Date:          November, 2005
% Last Modified:
%
% Calling:   c2c_TravelTimeAndCost_demandBased_byRegion_survey
% Called by: None
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tic; clear all; clc

% Declare global variables and constants
%--------------------------------------------
global Auto_averageOccupancy Auto_dailyOvernightTime
C2CDriveDist_Population_Centroids CA_dailyOvernightTime
global msaIndicatorData caseName scenarioName percentCoachTickets
percentFirstClassTickets oneDayLodgingCost
global countMSAs_all stateIndexing maxDist maxDistIndex

% Name of reference directory
Install_Dir    = 'C:\Program Files\TSAM\4.2\data\mode_choice';

% Load Tables
%--------------
load (strcat(Install_Dir, '\input\msaIndicatorData'))         % List
indicating if county is in an MSA. (1 -> MSA; 0 -> nonMSA)
load (strcat(Install_Dir, '\input\stateNames'))               % List of
states in US
load (strcat(Install_Dir, '\input\stateIndexing'))            % Index showing
where counties of each state begin and end in the county to county lis
load (strcat(Install_Dir, '\input\avgPartySize_ATS_Buss'))    % Business
party size from ATS [5x7]
load (strcat(Install_Dir, '\input\avgPartySize_ATS_NonBuss')) % Business
party size from ATS [5x7]
countMSAs_all = size(find(unique(msaIndicatorData(:,4))), 1);

disp(['Elasped Time: ', num2str(toc)])
```

```
% variables and switches
%-------------------------------------
Project_Dir          = 'D:\TSAM_Project';
StateNumber          = 0; %0 is for ALL the states.
Year                 = 2000;
saveDirName          = 'D:\Mode Choice\Calibration_February_07\';
scenarioName         = 'Nested Logit Travel Times and Costs\';

TICKET_TYPE_PERCENT = [0.25 0.40 0.50 0.60 0.70; 0.10 0.20 0.30 0.35 0.50];
TICKET_TYPE_PERCENT = [0.10 0.20 0.35 0.45 0.60; 0.02 0.05 0.10 0.15 0.25];

% Values are for 5 income groups
% TICKET_TYPE_PERCENT(1,:) -> Percent of Business Travelers buying Business
Class tickets
% TICKET_TYPE_PERCENT(2,:) -> Percent of Business Travelers buying Coach
Class tickets
% 1 - TICKET_TYPE_PERCENT = Gives percent values for Non-business travelers

VLJSwitch    = 1;
maxDist      = 3500;
maxDistIndex = floor(maxDist/50 - 1);

for tripType = 1 : 2

    if tripType == 1
        averageRoundTripTime_auto = 16;
        Auto_averageOccupancy     = mean(avgPartySize_ATS_Buss, 2);
        caseName                  = 'business\';
        minRoundTripTime_CA       = 10;
        oneDayLodgingCost         = [70 80 100 150 200];
        percentFirstClassTickets  = TICKET_TYPE_PERCENT(1,:);
        percentCoachTickets       = 1 - percentFirstClassTickets;
        tripPuroposeSwitch = 1;
    else
        averageRoundTripTime_auto = 20;
        Auto_averageOccupancy     = mean(avgPartySize_ATS_NonBuss, 2);
        caseName                  = 'nonBusiness\';
        minRoundTripTime_CA       = 10;
        oneDayLodgingCost         = [40 50 70 90 120];
        percentFirstClassTickets  = TICKET_TYPE_PERCENT(2,:);
        percentCoachTickets       = 1 - percentFirstClassTickets;
        tripPuroposeSwitch        = 2;
    end % if tripType == 1

    CA_dailyOvernightTime   = 24 - minRoundTripTime_CA; % hrs
    Auto_dailyOvernightTime = 24 - averageRoundTripTime_auto/2; % hrs

    inputProjDir  = (strcat(Project_Dir, '\Trip_Distribution\Output\Y',
num2str(Year)));
    disp(['Elasped Time: ', num2str(toc)])

    for stateCounter = 1 : 52
        stateCounter;
        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 5) == 0;
```

```
                disp(['Origin State: ', num2str(stateCounter), ' time: ',
num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            stateName = char(stateNames(stateCounter, 1));
            starter   = stateIndexing(stateCounter, 1);

            startRow = stateIndexing(stateCounter, 1);
            endRow   = stateIndexing(stateCounter, 2);

            joinDirName = [saveDirName scenarioName caseName stateName];
            load (strcat(joinDirName, '_CA_businessFare_2way'))
            load (strcat(joinDirName, '_CA_CoachFare_2way'))

            zeroBusiness                       = find(CA_businessFare_2way ==
0);
            CA_businessFare_2way(zeroBusiness) =
CA_CoachFare_2way(zeroBusiness);
            clear zeroBusiness
            for i = 1:5
                CA_AverageFare(i,:,:,:) = CA_businessFare_2way *
percentFirstClassTickets(i) + CA_CoachFare_2way * percentCoachTickets(i);
            end % for i = 1:5
            clear CA_businessFare_2way CA_CoachFare_2way

            % County to County drive distance (using mappoint)
            load (strcat(Install_Dir,
'\input\C2CDriveDist_Population_Centroids'))
            C2CDriveDist_Population_Centroids =
C2CDriveDist_Population_Centroids(startRow:endRow, :);

            if tripPuroposeSwitch == 1
                load (strcat(inputProjDir, '\CABuzzTripTable_', stateName,
'_', num2str(Year), '.mat'));
                CA_TripTable = CAbuzzTripTable; clear CAbuzzTripTable
            else
                load (strcat(inputProjDir, '\CAnonBuzzTripTable_', stateName,
'_', num2str(Year), '.mat'));
                CA_TripTable = CAnonBuzzTripTable; clear CAnonBuzzTripTable
            end % if tripPuroposeSwitch == 1

            load (strcat(joinDirName, '_c2cAuto_TT_2way'))
            load (strcat(joinDirName, '_c2cAuto_TC_2way'))

            load (strcat(joinDirName, '_CA_index'))
            load (strcat(joinDirName, '_CA_travelTime_with_SD_2way'))
            load (strcat(joinDirName, '_CA_numberOfNights_2way'))
            load (strcat(joinDirName, '_CA_flightTime_no_SD_2way'))

            load (strcat(joinDirName, '_CA_accEgrsCost_2way'))

            if VLJSwitch == 2
                load (strcat(joinDirName, '_VLJ_TrvlTime_2way'))
                load (strcat(joinDirName, '_VLJ_TrvlCost_2way'))
            else
                VLJ_TrvlTime_2way = [];
                VLJ_TrvlCost_2way = [];
            end % if VLJSwitch ==1

            nRows = endRow - startRow + 1;
            countiesInState           = msaIndicatorData(startRow:endRow,:);
            stateMSA_Counties_Index   = find(countiesInState(:, 2));
            stateMSA_Counties         =
countiesInState(stateMSA_Counties_Index,:);
            currentState_MSAs         = unique(stateMSA_Counties(:,4));
            currentState_MSAs_index   = find(currentState_MSAs);
            currentState_MSAs         =
currentState_MSAs(currentState_MSAs_index);
            currentState_NumberOfMSAs = size(currentState_MSAs, 1);

            if StateNumber == 0 %All States
                C2C_TT_TC_Aggrt_demandBased_modular_MSA_MSA(CA_TripTable,
starter, stateName, 1, ...
                    3091, saveDirName, nRows, c2cAuto_TT_2way, ...
                    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way,
CA_numberOfNights_2way, ...
                    CA_accEgrsCost_2way, CA_AverageFare, VLJ_TrvlTime_2way,
VLJ_TrvlCost_2way, ...
                    currentState_NumberOfMSAs, currentState_MSAs,
CA_flightTime_no_SD_2way);

                C2C_TT_TC_Aggrt_demandBased_modular_MSA_nonMSA(CA_TripTable,
starter, stateName, 1, ...
                    3091, saveDirName, nRows, c2cAuto_TT_2way, ...
                    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way,
CA_numberOfNights_2way, ...
                    CA_accEgrsCost_2way, CA_AverageFare, VLJ_TrvlTime_2way,
VLJ_TrvlCost_2way, ...
                    currentState_NumberOfMSAs, currentState_MSAs,
CA_flightTime_no_SD_2way);

                C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_MSA(CA_TripTable,
starter, stateName, 1, ...
                    3091, saveDirName, nRows, c2cAuto_TT_2way, ...
                    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way,
CA_numberOfNights_2way, ...
                    CA_accEgrsCost_2way, CA_AverageFare, VLJ_TrvlTime_2way,
VLJ_TrvlCost_2way, CA_flightTime_no_SD_2way);


C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_nonMSA(CA_TripTable, starter,
stateName, 1, ...
                    3091, saveDirName, nRows, c2cAuto_TT_2way, ...
                    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way,
CA_numberOfNights_2way, ...
                    CA_accEgrsCost_2way, CA_AverageFare, VLJ_TrvlTime_2way,
VLJ_TrvlCost_2way, CA_flightTime_no_SD_2way);
            else % One State
                % Finding the starting and ending index of the state
                StateStartIndex = stateIndexing(StateNumber, 1);
                StateEndIndex   = stateIndexing(StateNumber, 2);
                if StateNumber == stateCounter % Selected State
                    C2C_TT_TC_Aggrt_demandBased_modular(CA_TripTable,
starter, stateName, 1, ...
```

```
                        3091, saveDirName, nRows, c2cAuto_TT_2way, ...
                        c2cAuto_numbOfNights_2way,
c2cAuto_TC_noOccupancy_2way, CA_index, CA_Access_Time, ...
                        CA_Egress_Time, CA_Origin_ProcTime,
CA_Destn_ProcTime, CA_Schedule_Delay_2way, ...
                        CA_Flight_Time_2way, CA_roundTrip_travelTime,
CA_numberOfNights, CA_businessFare, ...
                        CA_CoachFare, CA_CoachFare_Index,
CA_accEgrsCost_2way);
                else
                    C2C_TT_TC_Aggrt_demandBased_modular(CA_TripTable,
starter, stateName, StateStartIndex, ...
                        StateEndIndex, saveDirName, nRows, c2cAuto_TT_2way,
...
                        c2cAuto_numbOfNights_2way,
c2cAuto_TC_noOccupancy_2way, CA_index, CA_Access_Time, ...
                        CA_Egress_Time, CA_Origin_ProcTime,
CA_Destn_ProcTime, CA_Schedule_Delay_2way, ...
                        CA_Flight_Time_with_SD_2way,
CA_Flight_Time_NO_SD_2way, CA_travelTime_with_SD_2way,
CA_travelTime_NO_SD_2way, ...
                        CA_numberOfNights, CA_businessFare, CA_CoachFare,
CA_CoachFare_Index, CA_accEgrsCost_2way);
                end % if StateNumber == stateCounter % Selected State
            end % if StateNumber == 0 %All States

            clear CA_AverageFare
        end % if (stateCounter == 2 | stateCounter == 12)
    end % for stateCounter = 1 : 52
end % for tripType = 1 : 2

aggregate_TT_TC_MSA_MSA_modular()

return
```

------------------------------------------------------------------------------------------------------------

#### 7.4.2.1 Aggregate Travel Times and Costs by State (Trips Starting and Ending in MSA's)

---

```
function C2C_TT_TC_Aggrt_demandBased_modular_MSA_MSA(CA_TripTable, starter,
stateName, StateStartIndex, ...
    StateEndIndex, saveDirName, numbRows, c2cAuto_TT_2way, ...
    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way, CA_numberOfNights,
...
    CA_accEgrsCost_2way, CA_AverageFare, c2c_VLJ_TrvlTime_2way,
c2c_VLJ_TrvlCost_2way, ...
    currentState_NumberOfMSAs, currentState_MSAs, CA_flightTime_no_SD_2way)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveVLJ
% Calling:    selectCandidateRoutes_CA
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global Auto_averageOccupancy C2CDriveDist_Population_Centroids
CA_dailyOvernightTime
global caseName countMSAs_all msaIndicatorData oneDayLodgingCost scenarioName
VLJSwitch

% Initialize trip tables
INCOME  = [25000 45000 80000 125000 170000]; % mean/median values of
household incomes
incomeIndex = size(INCOME, 2);

% initialize constants
route_TYPES    = 3;

Auto_travelTime_LodgingTime_2way  = zeros(currentState_NumberOfMSAs,
countMSAs_all);
Auto_travelCost_LodgingCost_2way  = zeros(incomeIndex,
currentState_NumberOfMSAs, countMSAs_all);
Auto_all_trips                    = Auto_travelCost_LodgingCost_2way;

CA_travelTime_SD_LodgingTime  = zeros(currentState_NumberOfMSAs,
countMSAs_all, route_TYPES);
CA_flightTime_noScheduleDelay = zeros(currentState_NumberOfMSAs,
countMSAs_all, route_TYPES);

CA_travelCost_LodgingCost     = zeros(incomeIndex, currentState_NumberOfMSAs,
countMSAs_all, route_TYPES);
CA_all_trips                  = zeros(incomeIndex, currentState_NumberOfMSAs,
countMSAs_all, route_TYPES);
CA_binRoutes                  = zeros(incomeIndex, currentState_NumberOfMSAs,
countMSAs_all, route_TYPES);

newValues   = [1 4 2];
route_TYPES = size(newValues, 2);
```

```
if VLJSwitch == 2
    VLJ_TravelTime = zeros(msa_TYPES, distance_INDEX);
    VLJ_TravelCost = zeros(msa_TYPES,  incomeIndex, distance_INDEX);
    VLJ_all_trips  = VLJ_TravelCost;
end % if VLJSwitch == 2

%-----------------------------------------------------
% Start Mode Split
%-----------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCounty;
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number

    % keep track of location in run
    if mod(orgCountyReal, 30) == 0; disp(['Origin County: ',
num2str(orgCountyReal), ' time: ', num2str(toc) ]); end

    if (msaIndicatorData(orgCountyReal,2) == 1)
        originFIPS = msaIndicatorData(orgCountyReal,1);

        alaskaOrigin = ~isempty(find(orgCountyReal == 68:78));
        hawaiiOrigin = ~isempty(find(orgCountyReal == 527:530));

        if (alaskaOrigin == 1 || hawaiiOrigin == 1)
        else
            for desCounty = StateStartIndex : StateEndIndex %1 : numbCols %
loop for destination counties
                desCounty;

                if (msaIndicatorData(desCounty,2) == 1)

                    alaskaDestnt = ~isempty(find(desCounty == 68:78));
                    hawaiiDestnt = ~isempty(find(desCounty == 527:530));

                    route_Distance =
C2CDriveDist_Population_Centroids(orgCounty, desCounty);

                    if alaskaDestnt == 1 || hawaiiDestnt == 1
                    else
                        % check if intercity trip
                        if route_Distance >= 100 && route_Distance < 3500

                            trips_1 = CA_TripTable(1).trips(orgCounty,
desCounty);
                            trips_2 = CA_TripTable(2).trips(orgCounty,
desCounty);
                            trips_3 = CA_TripTable(3).trips(orgCounty,
desCounty);
                            trips_4 = CA_TripTable(4).trips(orgCounty,
desCounty);
                            trips_5 = CA_TripTable(5).trips(orgCounty,
desCounty);
                            tot_Trips = [trips_1; trips_2; trips_3; trips_4;
trips_5];
                            sum_tot_trips = trips_1 + trips_2 + trips_3 +
trips_4 + trips_5;
```

```
                                % check if trips b/n O-D pair
                                if sum_tot_trips > 0

                                    % find the origin and destination MSA index
                                    orgMSA_Index_file =
msaIndicatorData(orgCountyReal, 4);
                                    orgMSA_Index = find(currentState_MSAs ==
orgMSA_Index_file);
                                    desMSA_Index = msaIndicatorData(desCounty,
4);
                                    if (orgMSA_Index == 0) | (desMSA_Index == 0)
| isempty(orgMSA_Index)
                                        disp('Error With MSAs')
                                        pause
                                    end % if (orgMSA_Index == 0) | (desMSA_Index
== 0)

                                    % AUTO TRAVEL TIME
                                    %extract Auto round-trip travel time from
table
                                    Auto_TravelTime_2way =
c2cAuto_TT_2way(orgCounty, desCounty);
                                    if Auto_TravelTime_2way < 0
                                        orgCounty
                                        disp('Zero Auto Travel Time?'); pause;
                                    end % if Auto_TravelTime_2way < 0

Auto_travelTime_LodgingTime_2way(orgMSA_Index, desMSA_Index) =
Auto_travelTime_LodgingTime_2way(orgMSA_Index, desMSA_Index) + ...
                                        Auto_TravelTime_2way * sum_tot_trips;

                                    % AUTO TRAVEL COST
                                    Auto_travelCost_LodgingCost_2way(:,
orgMSA_Index, desMSA_Index) = Auto_travelCost_LodgingCost_2way(:,
orgMSA_Index, desMSA_Index) +  ...
                                        c2cAuto_TC_2way(:, orgCounty, desCounty)
.* tot_Trips;
                                    Auto_all_trips(:, orgMSA_Index, desMSA_Index)
= Auto_all_trips(:, orgMSA_Index, desMSA_Index)  + tot_Trips;


                                    % airport index
                                    airport_row_Numbers =
CA_index(:,orgCounty,desCounty); %9x1

                                    % CA TRAVEL TIME
                                    travelTime_includingScheduleDelay  =
CA_travelTime_with_SD_2way(:,orgCounty,desCounty); % 3x1
                                    flightTime_noScheduleDelay         =
CA_flightTime_no_SD_2way(:,orgCounty,desCounty); % 3x1
                                    overnightTime_CA_2way              =
CA_numberOfNights(:,orgCounty,desCounty) * CA_dailyOvernightTime; % 3x1

                                    % CA TRAVEL COST
                                    accEgrsCost_CA_2way_1 =
CA_accEgrsCost_2way(:,orgCounty,desCounty);
```

```
                                accessRows = size(accEgrsCost_CA_2way_1,1);
                                accEgrsCost_CA_2way = zeros(5, accessRows);
                                for cntRt = 1 : accessRows
                                    accEgrsCost_CA_2way(:,cntRt) =
accEgrsCost_CA_2way_1(cntRt) ./ Auto_averageOccupancy'; %#ok<AGROW>
                                end % for cntRt = 1 :
size(accEgrsCost_CA_2way_1,1)

                                averageFare_CA_2way =
CA_AverageFare(:,:,orgCounty,desCounty); % 3x4
                                overnight_TC_CA     =
CA_numberOfNights(:,orgCounty,desCounty) * oneDayLodgingCost; % 3x4
                                overnight_TC_CA     = overnight_TC_CA';


                                % Delete CA routes without Information
                                zeroIndex = find(airport_row_Numbers == 0 );
                                if ~isempty(zeroIndex)
                                    airport_row_Numbers(zeroIndex)      = [];

travelTime_includingScheduleDelay(zeroIndex) = [];
                                    overnightTime_CA_2way(zeroIndex, :) = [];
                                    accEgrsCost_CA_2way(:, zeroIndex)   = [];
                                    averageFare_CA_2way(:, zeroIndex)   = [];
                                    overnight_TC_CA(:, zeroIndex)       = [];
                                    flightTime_noScheduleDelay(zeroIndex, :)
= [];

                                end % if ~isempty(zeroIndex)

                                if isempty(airport_row_Numbers)
                                else
                                    for route_type = 1 : route_TYPES
                                        airportIndexSelect =
newValues(route_type);

                                        row_Index = find(airportIndexSelect
== airport_row_Numbers);

                                        if isempty(row_Index)
                                        else
                                            check_TT_index =
find(travelTime_includingScheduleDelay == 0);
                                            if isempty(check_TT_index)

total_CA_travelTime_withLoding = travelTime_includingScheduleDelay +
overnightTime_CA_2way;

total_CA_travelCost_withLodging  = accEgrsCost_CA_2way(:,row_Index) +
averageFare_CA_2way(:,row_Index) + overnight_TC_CA(:,row_Index);
                                            else
                                                disp('Problem with Travel
Time')

                                                pause
                                            end % if isempty(check_TT_index)


                                            CA_pureFare_total_2way =
averageFare_CA_2way(row_Index, :);

                                            if CA_pureFare_total_2way > 0
```

```
CA_travelTime_SD_LodgingTime(orgMSA_Index, desMSA_Index, route_type)  =
CA_travelTime_SD_LodgingTime(orgMSA_Index, desMSA_Index, route_type)  +
total_CA_travelTime_withLoding(row_Index) * sum_tot_trips;

CA_flightTime_noScheduleDelay(orgMSA_Index, desMSA_Index, route_type)  =
CA_flightTime_noScheduleDelay(orgMSA_Index, desMSA_Index, route_type)  +
flightTime_noScheduleDelay(row_Index) * sum_tot_trips;

                                                CA_travelCost_LodgingCost(:,
orgMSA_Index, desMSA_Index, route_type)  = CA_travelCost_LodgingCost(:,
orgMSA_Index, desMSA_Index, route_type)  + total_CA_travelCost_withLodging .*
tot_Trips;
                                                CA_all_trips(:, orgMSA_Index,
desMSA_Index, route_type) = CA_all_trips(:, orgMSA_Index, desMSA_Index,
route_type) + tot_Trips;
                                                CA_binRoutes(:, orgMSA_Index,
desMSA_Index, route_type) = CA_binRoutes(:, orgMSA_Index, desMSA_Index,
route_type) + 1;

                                            end % if CA_pureFare_total_2way >
0
                                        end % if isempty(row_Index)
                                    end % for route_type = 1 : route_TYPES
                                end % if isempty(airport_row_Numbers)

                            end % if sum_trips_income > 0
                        end % if route_Distance >= 100 && route_Distance <
3050
                    end % if alaskaDestnt == 1 | hawaiiDestnt == 1

                end % if (msaIndicatorData(desCounty,2) == 1)
            end % for desCounty = StateStartIndex : StateEndIndex
        end % if alaskaOrigin == 1 | hawaiiOrigin == 1
    end % if (msaIndicatorData(orgCountyReal,2) == 1)
end % for orgCounty = 1 : numbRows % loop for origin counties

joinDirName = [saveDirName scenarioName caseName stateName];

save (strcat(joinDirName, '_Auto_travelTime_LodgingTime_2way'),
'Auto_travelTime_LodgingTime_2way')
save (strcat(joinDirName, '_Auto_travelCost_LodgingCost_2way'),
'Auto_travelCost_LodgingCost_2way')
save (strcat(joinDirName, '_Auto_all_trips'), 'Auto_all_trips')

save (strcat(joinDirName, '_CA_travelTime_SD_LodgingTime'),
'CA_travelTime_SD_LodgingTime')
save (strcat(joinDirName, '_CA_flightTime_noScheduleDelay'),
'CA_flightTime_noScheduleDelay')

save (strcat(joinDirName, '_CA_travelCost_LodgingCost'),
'CA_travelCost_LodgingCost')
save (strcat(joinDirName, '_CA_all_trips'), 'CA_all_trips')
save (strcat(joinDirName, '_CA_binRoutes'), 'CA_binRoutes')

return
```

---------------------------------------------------------------------------------------------------------------------------

#### 7.4.2.2 Aggregate Travel Times and Costs by State, and Distance (Trips from MSA's to non-MSA's)

```
------------------------------------------------------------------------------------------------------------------------------------
-----------

function C2C_TT_TC_Aggrt_demandBased_modular_MSA_nonMSA(CA_TripTable,
starter, stateName, StateStartIndex, ...
    StateEndIndex, saveDirName, numbRows, c2cAuto_TT_2way, ...
    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way, CA_numberOfNights,
...
    CA_accEgrsCost_2way, CA_AverageFare, c2c_VLJ_TrvlTime_2way,
c2c_VLJ_TrvlCost_2way, ...
    currentState_NumberOfMSAs, currentState_MSAs, CA_flightTime_no_SD_2way)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveVLJ
% Calling:    selectCandidateRoutes_CA
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global Auto_averageOccupancy C2CDriveDist_Population_Centroids
CA_dailyOvernightTime
global caseName msaIndicatorData oneDayLodgingCost scenarioName stateIndexing
VLJSwitch
global maxDist maxDistIndex

% Initialize trip tables
INCOME  = [25000 45000 80000 125000 170000]; % mean/median values of
household incomes
incomeIndex = size(INCOME, 2);

% initialize constants
route_TYPES = 3;


Auto_travelTime_LodgingTime_2way  = zeros(currentState_NumberOfMSAs, 52,
maxDistIndex);
Auto_travelCost_LodgingCost_2way  = zeros(incomeIndex,
currentState_NumberOfMSAs, 52, maxDistIndex);
Auto_all_trips                    = Auto_travelCost_LodgingCost_2way;

CA_travelTime_SD_LodgingTime  = zeros(currentState_NumberOfMSAs, 52,
maxDistIndex, route_TYPES);
CA_flightTime_noScheduleDelay = zeros(currentState_NumberOfMSAs, 52,
maxDistIndex, route_TYPES);
CA_travelCost_LodgingCost     = zeros(incomeIndex, currentState_NumberOfMSAs,
52, maxDistIndex, route_TYPES);
CA_all_trips                  = zeros(incomeIndex, currentState_NumberOfMSAs,
52, maxDistIndex, route_TYPES);
CA_binRoutes                  = zeros(incomeIndex, currentState_NumberOfMSAs,
52, maxDistIndex, route_TYPES);

newValues   = [1 4 2];
```

```
route_TYPES = size(newValues, 2);

if VLJSwitch == 2
    VLJ_TravelTime = zeros(msa_TYPES, distance_INDEX);
    VLJ_TravelCost = zeros(msa_TYPES,  incomeIndex, distance_INDEX);
    VLJ_all_trips  = VLJ_TravelCost;
end % if VLJSwitch == 2


%----------------------------------------------------
% Start Mode Split
%----------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCounty;
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number

    % keep track of location in run
    if mod(orgCountyReal, 30) == 0; disp(['Origin County: ',
num2str(orgCountyReal), ' time: ', num2str(toc) ]); end

    if (msaIndicatorData(orgCountyReal,2) == 1)
        originFIPS = msaIndicatorData(orgCountyReal,1);
        startDestStateCounter = 0;

        alaskaOrigin = ~isempty(find(orgCountyReal == 68:78));
        hawaiiOrigin = ~isempty(find(orgCountyReal == 527:530));

        if (alaskaOrigin == 1 || hawaiiOrigin == 1)
        else
            for desCounty = StateStartIndex : StateEndIndex %1 : numbCols %
loop for destination counties

                checkState = find(desCounty == stateIndexing(:, 1));
                if ~isempty(checkState)
                    startDestStateCounter = startDestStateCounter + 1;
                end % if ~isempty(checkState)

                if (msaIndicatorData(desCounty,2) ~= 1)

                    alaskaDestnt = ~isempty(find(desCounty == 68:78));
                    hawaiiDestnt = ~isempty(find(desCounty == 527:530));

                    route_Distance =
C2CDriveDist_Population_Centroids(orgCounty, desCounty);

                    if alaskaDestnt == 1 || hawaiiDestnt == 1
                    else
                        % check if intercity trip
                        if route_Distance >= 100 && route_Distance < maxDist

                            trips_1 = CA_TripTable(1).trips(orgCounty,
desCounty);
                            trips_2 = CA_TripTable(2).trips(orgCounty,
desCounty);
                            trips_3 = CA_TripTable(3).trips(orgCounty,
desCounty);
```

```
                            trips_4 = CA_TripTable(4).trips(orgCounty, 
desCounty);
                            trips_5 = CA_TripTable(5).trips(orgCounty, 
desCounty);
                            tot_Trips = [trips_1; trips_2; trips_3; trips_4; 
trips_5];
                            sum_tot_trips = trips_1 + trips_2 + trips_3 + 
trips_4 + trips_5;

                            % check if trips b/n O-D pair
                            if sum_tot_trips > 0

                                dist_index = floor(route_Distance/50 - 1);

                                % find the origin and destination MSA index
                                orgMSA_Index_file = 
msaIndicatorData(orgCountyReal, 4);
                                orgMSA_Index = find(currentState_MSAs == 
orgMSA_Index_file);
                                if (orgMSA_Index == 0) | 
isempty(orgMSA_Index)
                                    disp('Error With MSAs')
                                    pause
                                end % if (orgMSA_Index == 0) | (desMSA_Index 
== 0)


                                % AUTO TRAVEL TIME
                                %extract Auto round-trip travel time from 
table
                                Auto_TravelTime_2way = 
c2cAuto_TT_2way(orgCounty, desCounty);
                                if Auto_TravelTime_2way < 0
                                    orgCounty
                                    disp('Zero Auto Travel Time?'); pause;
                                end % if Auto_TravelTime_2way < 0

Auto_travelTime_LodgingTime_2way(orgMSA_Index, startDestStateCounter, 
dist_index) = Auto_travelTime_LodgingTime_2way(orgMSA_Index, 
startDestStateCounter, dist_index) + Auto_TravelTime_2way * sum_tot_trips;

                                % AUTO TRAVEL COST
                                Auto_travelCost_LodgingCost_2way(:, 
orgMSA_Index, startDestStateCounter, dist_index) = 
Auto_travelCost_LodgingCost_2way(:, orgMSA_Index, startDestStateCounter, 
dist_index) + ...
                                        c2cAuto_TC_2way(:, orgCounty, 
desCounty) .* tot_Trips;
                                Auto_all_trips(:, orgMSA_Index, 
startDestStateCounter,dist_index) = Auto_all_trips(:, orgMSA_Index, 
startDestStateCounter, dist_index) + tot_Trips;


                                % airport index
                                airport_row_Numbers = 
CA_index(:,orgCounty,desCounty); %9x1

                                % CA TRAVEL TIME
                                travelTime_includingScheduleDelay  =
CA_travelTime_with_SD_2way(:,orgCounty,desCounty); % 3x1
                                flightTime_noScheduleDelay         =
CA_flightTime_no_SD_2way(:,orgCounty,desCounty); % 3x1
                                overnightTime_CA_2way              =
CA_numberOfNights(:,orgCounty,desCounty) * CA_dailyOvernightTime; % 3x1

                                % CA TRAVEL COST
                                accEgrsCost_CA_2way_1 =
CA_accEgrsCost_2way(:,orgCounty,desCounty);
                                accessRows = size(accEgrsCost_CA_2way_1, 1);
                                accEgrsCost_CA_2way = zeros(5, accessRows);
                                for cntRt = 1 : accessRows
                                    accEgrsCost_CA_2way(:, cntRt) =
accEgrsCost_CA_2way_1(cntRt) ./ Auto_averageOccupancy'; %#ok<AGROW>
                                end % for cntRt = 1 :
size(accEgrsCost_CA_2way_1,1)

                                averageFare_CA_2way =
CA_AverageFare(:,:,orgCounty,desCounty); % 3x4
                                overnight_TC_CA     =
CA_numberOfNights(:,orgCounty,desCounty) * oneDayLodgingCost; % 3x4
                                overnight_TC_CA     = overnight_TC_CA';


                                % Delete CA routes without Information
                                zeroIndex = find(airport_row_Numbers == 0 );
                                if ~isempty(zeroIndex)
                                    airport_row_Numbers(zeroIndex)      = [];

travelTime_includingScheduleDelay(zeroIndex) = [];
                                    overnightTime_CA_2way(zeroIndex, :) = [];
                                    accEgrsCost_CA_2way(:, zeroIndex)   = [];
                                    averageFare_CA_2way(:, zeroIndex)   = [];
                                    overnight_TC_CA(:, zeroIndex)       = [];
                                    flightTime_noScheduleDelay(zeroIndex, :)
= [];
                                end % if ~isempty(zeroIndex)

                                if isempty(airport_row_Numbers)
                                else
                                    for route_type = 1 : route_TYPES
                                        airportIndexSelect =
newValues(route_type);
                                        row_Index = find(airportIndexSelect
== airport_row_Numbers);
                                        if isempty(row_Index)
                                        else
                                            CA_pureFare_total_2way =
averageFare_CA_2way(row_Index, :);
                                            if CA_pureFare_total_2way > 0
                                                check_TT_index =
find(travelTime_includingScheduleDelay == 0);
                                                if isempty(check_TT_index)

total_CA_travelTime_withLoding = travelTime_includingScheduleDelay +
overnightTime_CA_2way;
```

```
total_CA_travelCost_withLodging  = accEgrsCost_CA_2way(:,row_Index) +
averageFare_CA_2way(:,row_Index) + overnight_TC_CA(:,row_Index);
                                                else
                                                    disp('Problem with Travel
Time')

                                                    pause
                                                end % if
isempty(check_TT_index)


CA_travelTime_SD_LodgingTime(orgMSA_Index, startDestStateCounter, dist_index,
route_type) = ...

CA_travelTime_SD_LodgingTime(orgMSA_Index, startDestStateCounter, dist_index,
route_type)  + total_CA_travelTime_withLoding(row_Index) * sum_tot_trips;


CA_flightTime_noScheduleDelay(orgMSA_Index, startDestStateCounter,
dist_index, route_type) = ...

CA_flightTime_noScheduleDelay(orgMSA_Index, startDestStateCounter,
dist_index, route_type)  + flightTime_noScheduleDelay(row_Index) *
sum_tot_trips;


                                                CA_travelCost_LodgingCost(:,
orgMSA_Index, startDestStateCounter, dist_index, route_type) =  ...

CA_travelCost_LodgingCost(:, orgMSA_Index, startDestStateCounter, dist_index,
route_type) + total_CA_travelCost_withLodging(row_Index) .* tot_Trips;

                                                CA_all_trips(:, orgMSA_Index,
startDestStateCounter, dist_index, route_type) = CA_all_trips(:,
orgMSA_Index, startDestStateCounter, dist_index, route_type) + tot_Trips;

                                                CA_binRoutes(:, orgMSA_Index,
startDestStateCounter, dist_index, route_type) = CA_binRoutes(:,
orgMSA_Index, startDestStateCounter, dist_index, route_type) + 1;

                                            end % if CA_pureFare_total_2way >
0

                                        end % if isempty(row_Index)
                                    end % for route_type = 1 : route_TYPES
                                end % if isempty(airport_row_Numbers)

                            end % if sum_trips_income > 0
                        end % if route_Distance >= 100 && route_Distance <
3050
                    end % if alaskaDestnt == 1 | hawaiiDestnt == 1

                end % if (msaIndicatorData(desCounty,2) == 1)
            end % for desCounty = StateStartIndex : StateEndIndex
        end % if alaskaOrigin == 1 | hawaiiOrigin == 1
    end % if (msaIndicatorData(orgCountyReal,2) == 1)
end % for orgCounty = 1 : numbRows % loop for origin counties
```

```
joinDirName = [saveDirName scenarioName caseName stateName];

save ([joinDirName, '_Auto_travelTime_LodgingTime_2way_MSA_nonMSA'],
'Auto_travelTime_LodgingTime_2way')
save ([joinDirName, '_Auto_travelCost_LodgingCost_2way_MSA_nonMSA'],
'Auto_travelCost_LodgingCost_2way')
save ([joinDirName, '_Auto_all_trips_MSA_nonMSA'], 'Auto_all_trips')

save ([joinDirName, '_CA_travelTime_SD_LodgingTime_MSA_nonMSA'],
'CA_travelTime_SD_LodgingTime')
save ([joinDirName, '_CA_flightTime_noScheduleDelay_MSA_nonMSA'],
'CA_flightTime_noScheduleDelay')

save ([joinDirName, '_CA_travelCost_LodgingCost_MSA_nonMSA'],
'CA_travelCost_LodgingCost')
save ([joinDirName, '_CA_all_trips_MSA_nonMSA'], 'CA_all_trips')
save ([joinDirName, '_CA_binRoutes_MSA_nonMSA'], 'CA_binRoutes')

return
```

-----------------------------------------------------------------------------------------------------------------------

### 7.4.2.3 Aggregate Travel Times and Costs by Distance (Trips from non-MSA's to MSA's)

-----------------------------------------------------------------------------------------------------------------------------

```
function C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_MSA(CA_TripTable, 
starter, stateName, StateStartIndex, ...
    StateEndIndex, saveDirName, numbRows, c2cAuto_TT_2way, ...
    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way, CA_numberOfNights, 
...
    CA_accEgrsCost_2way, CA_AverageFare, c2c_VLJ_TrvlTime_2way, 
c2c_VLJ_TrvlCost_2way, CA_flightTime_no_SD_2way)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveVLJ
% Calling:    selectCandidateRoutes_CA
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global Auto_averageOccupancy C2CDriveDist_Population_Centroids 
CA_dailyOvernightTime
global caseName msaIndicatorData oneDayLodgingCost scenarioName VLJSwitch
global maxDist maxDistIndex

% Initialize trip tables
INCOME  = [25000 45000 80000 125000 170000]; % mean/median values of 
household incomes
incomeIndex = size(INCOME, 2);

% initialize constants
route_TYPES = 3;


Auto_travelTime_LodgingTime_2way  = zeros(318, maxDistIndex);
Auto_travelCost_LodgingCost_2way  = zeros(incomeIndex, 318, maxDistIndex);
Auto_all_trips                    = Auto_travelCost_LodgingCost_2way;

CA_travelTime_SD_LodgingTime  = zeros(318, maxDistIndex, route_TYPES);
CA_flightTime_noScheduleDelay = zeros(318, maxDistIndex, route_TYPES);
CA_travelCost_LodgingCost     = zeros(incomeIndex, 318, maxDistIndex, 
route_TYPES);
CA_all_trips                  = zeros(incomeIndex, 318, maxDistIndex, 
route_TYPES);
CA_binRoutes                  = zeros(incomeIndex, 318, maxDistIndex, 
route_TYPES);

newValues   = [1 4 2];
route_TYPES = size(newValues, 2);

if VLJSwitch == 2
    VLJ_TravelTime = zeros(msa_TYPES, distance_INDEX);
    VLJ_TravelCost = zeros(msa_TYPES,  incomeIndex, distance_INDEX);
    VLJ_all_trips  = VLJ_TravelCost;
end % if VLJSwitch == 2
```

```
%---------------------------------------------------
% Start Mode Split
%---------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCounty;
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number

    % keep track of location in run
    if mod(orgCountyReal, 30) == 0; disp(['Origin County: ',
num2str(orgCountyReal), ' time: ', num2str(toc) ]); end

    if (msaIndicatorData(orgCountyReal,2) ~= 1)

        alaskaOrigin = ~isempty(find(orgCountyReal == 68:78));
        hawaiiOrigin = ~isempty(find(orgCountyReal == 527:530));

        if (alaskaOrigin == 1 || hawaiiOrigin == 1)
        else
            for desCounty = StateStartIndex : StateEndIndex %1 : numbCols %
loop for destination counties

                if (msaIndicatorData(desCounty,2) == 1)
                    destnFIPS = msaIndicatorData(orgCountyReal,1);
                    startDestStateCounter = 0;

                    alaskaDestnt = ~isempty(find(desCounty == 68:78));
                    hawaiiDestnt = ~isempty(find(desCounty == 527:530));

                    route_Distance =
C2CDriveDist_Population_Centroids(orgCounty, desCounty);

                    if alaskaDestnt == 1 || hawaiiDestnt == 1
                    else
                        % check if intercity trip
                        if route_Distance >= 100 && route_Distance < maxDist
                            desCounty;

                            trips_1 = CA_TripTable(1).trips(orgCounty,
desCounty);
                            trips_2 = CA_TripTable(2).trips(orgCounty,
desCounty);
                            trips_3 = CA_TripTable(3).trips(orgCounty,
desCounty);
                            trips_4 = CA_TripTable(4).trips(orgCounty,
desCounty);
                            trips_5 = CA_TripTable(5).trips(orgCounty,
desCounty);
                            tot_Trips = [trips_1; trips_2; trips_3; trips_4;
trips_5];
                            sum_tot_trips = trips_1 + trips_2 + trips_3 +
trips_4 + trips_5;

                            % check if trips b/n O-D pair
                            if sum_tot_trips > 0
```

```
                                dist_index = floor(route_Distance/50 - 1);

                                % find the origin and destination MSA index
                                desMSA_Index = msaIndicatorData(desCounty,
4);
                                if (desMSA_Index == 0) |
isempty(desMSA_Index)
                                    disp('Error With MSAs')
                                    pause
                                end % if (orgMSA_Index == 0) | (desMSA_Index
== 0)

                                % AUTO TRAVEL TIME
                                %extract Auto round-trip travel time from
table
                                Auto_TravelTime_2way =
c2cAuto_TT_2way(orgCounty, desCounty);
                                if Auto_TravelTime_2way < 0
                                    orgCounty
                                    disp('Zero Auto Travel Time?'); pause;
                                end % if Auto_TravelTime_2way < 0

Auto_travelTime_LodgingTime_2way(desMSA_Index, dist_index) =
Auto_travelTime_LodgingTime_2way(desMSA_Index, dist_index) +
Auto_TravelTime_2way * sum_tot_trips;

                                % AUTO TRAVEL COST
                                Auto_travelCost_LodgingCost_2way(:,
desMSA_Index, dist_index) = Auto_travelCost_LodgingCost_2way(:, desMSA_Index,
dist_index) + ...
                                        c2cAuto_TC_2way(:, orgCounty,
desCounty) .* tot_Trips;
                                Auto_all_trips(:, desMSA_Index, dist_index) =
Auto_all_trips(:, desMSA_Index, dist_index) + tot_Trips;


                                % airport index
                                airport_row_Numbers =
CA_index(:,orgCounty,desCounty); %9x1

                                % CA TRAVEL TIME
                                travelTime_includingScheduleDelay  =
CA_travelTime_with_SD_2way(:,orgCounty,desCounty); % 3x1
                                flightTime_noScheduleDelay         =
CA_flightTime_no_SD_2way(:,orgCounty,desCounty); % 3x1
                                overnightTime_CA_2way              =
CA_numberOfNights(:,orgCounty,desCounty) * CA_dailyOvernightTime; % 3x1

                                % CA TRAVEL COST
                                accEgrsCost_CA_2way_1 =
CA_accEgrsCost_2way(:,orgCounty,desCounty);
                                accessRows = size(accEgrsCost_CA_2way_1,1);
                                accEgrsCost_CA_2way = zeros(5, accessRows);
                                for cntRt = 1 : accessRows
                                    accEgrsCost_CA_2way(:, cntRt) =
accEgrsCost_CA_2way_1(cntRt) ./ Auto_averageOccupancy;
```

```
                                end % for cntRt = 1 :
size(accEgrsCost_CA_2way_1,1)

                                averageFare_CA_2way =
CA_AverageFare(:,:,orgCounty,desCounty); % 3x4
                                overnight_TC_CA     =
CA_numberOfNights(:,orgCounty,desCounty) * oneDayLodgingCost; % 3x4
                                overnight_TC_CA     = overnight_TC_CA';


                                % Delete CA routes without Information
                                zeroIndex = find(airport_row_Numbers == 0 );
                                if ~isempty(zeroIndex)
                                    airport_row_Numbers(zeroIndex)      = [];

travelTime_includingScheduleDelay(zeroIndex) = [];
                                    overnightTime_CA_2way(zeroIndex, :) = [];
                                    accEgrsCost_CA_2way(:, zeroIndex)   = [];
                                    averageFare_CA_2way(:, zeroIndex)   = [];
                                    overnight_TC_CA(:, zeroIndex)       = [];
                                    flightTime_noScheduleDelay(zeroIndex, :)
= [];
                                end % if ~isempty(zeroIndex)

                                if isempty(airport_row_Numbers)
                                else
                                    for route_type = 1 : route_TYPES
                                        airportIndexSelect =
newValues(route_type);

                                        row_Index = find(airportIndexSelect
== airport_row_Numbers);

                                        if isempty(row_Index)
                                        else
                                            check_TT_index =
find(travelTime_includingScheduleDelay == 0);
                                            if isempty(check_TT_index)

total_CA_travelTime_withLoding = travelTime_includingScheduleDelay +
overnightTime_CA_2way;

total_CA_travelCost_withLodging = accEgrsCost_CA_2way(:,row_Index) +
averageFare_CA_2way(:,row_Index) + overnight_TC_CA(:,row_Index);
                                            else
                                                disp('Problem with Travel
Time')

                                                pause
                                            end % if isempty(check_TT_index)


                                            CA_pureFare_total_2way =
averageFare_CA_2way(row_Index, :);
                                            if CA_pureFare_total_2way > 0


CA_travelTime_SD_LodgingTime(desMSA_Index, dist_index, route_type) =
CA_travelTime_SD_LodgingTime(desMSA_Index, dist_index, route_type) ...
```

```
                                                    +
total_CA_travelTime_withLoding(row_Index) * sum_tot_trips;


CA_flightTime_noScheduleDelay(desMSA_Index, dist_index, route_type) =
CA_flightTime_noScheduleDelay(desMSA_Index, dist_index, route_type) ...
                                                    +
flightTime_noScheduleDelay(row_Index) * sum_tot_trips;

                                                CA_travelCost_LodgingCost(:,
desMSA_Index, dist_index, route_type) = CA_travelCost_LodgingCost(:,
desMSA_Index, dist_index, route_type) ...
                                                    +
total_CA_travelCost_withLodging .* tot_Trips;

                                                CA_all_trips(:, desMSA_Index,
dist_index, route_type) = CA_all_trips(:, desMSA_Index, dist_index,
route_type) + tot_Trips;


                                                CA_binRoutes(:, desMSA_Index,
dist_index, route_type) = CA_binRoutes(:, desMSA_Index, dist_index,
route_type) + 1;
                                            end % if CA_pureFare_total_2way >
0
                                        end % if isempty(row_Index)
                                    end % for route_type = 1 : route_TYPES
                                end % if isempty(airport_row_Numbers)

                            end % if sum_trips_income > 0
                        end % if route_Distance >= 100 && route_Distance <
3050
                    end % if alaskaDestnt == 1 | hawaiiDestnt == 1

                end % if (msaIndicatorData(desCounty,2) == 1)
            end % for desCounty = StateStartIndex : StateEndIndex
        end % if alaskaOrigin == 1 | hawaiiOrigin == 1
    end % if (msaIndicatorData(orgCountyReal,2) == 1)
end % for orgCounty = 1 : numbRows % loop for origin counties

joinDirName = [saveDirName scenarioName caseName stateName];

save ([joinDirName, '_Auto_travelTime_LodgingTime_2way_nonMSA_MSA'],
'Auto_travelTime_LodgingTime_2way')
save ([joinDirName, '_Auto_travelCost_LodgingCost_2way_nonMSA_MSA'],
'Auto_travelCost_LodgingCost_2way')
save ([joinDirName, '_Auto_all_trips_nonMSA_MSA'], 'Auto_all_trips')

save ([joinDirName, '_CA_travelTime_SD_LodgingTime_nonMSA_MSA'],
'CA_travelTime_SD_LodgingTime')
save ([joinDirName, '_CA_flightTime_noScheduleDelay_nonMSA_MSA'],
'CA_flightTime_noScheduleDelay')

save ([joinDirName, '_CA_travelCost_LodgingCost_nonMSA_MSA'],
'CA_travelCost_LodgingCost')
save ([joinDirName, '_CA_all_trips_nonMSA_MSA'], 'CA_all_trips')
save ([joinDirName, '_CA_binRoutes_nonMSA_MSA'], 'CA_binRoutes')
```

```
return
```

---------------------------------------------------------------------------------------------------------------------------

#### 7.4.2.4 Aggregate Travel Times and Costs by State and Distance (Trips Starting and Ending in non-MSA's)

---

```
function C2C_TT_TC_Aggrt_demandBased_modular_nonMSA_nonMSA(CA_TripTable, 
starter, stateName, StateStartIndex, ...
    StateEndIndex, saveDirName, numbRows, c2cAuto_TT_2way, ...
    c2cAuto_TC_2way, CA_index, CA_travelTime_with_SD_2way, CA_numberOfNights, 
...
    CA_accEgrsCost_2way, CA_AverageFare, c2c_VLJ_TrvlTime_2way, 
c2c_VLJ_TrvlCost_2way, CA_flightTime_no_SD_2way)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveVLJ
% Calling:    selectCandidateRoutes_CA
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global Auto_averageOccupancy C2CDriveDist_Population_Centroids 
CA_dailyOvernightTime
global caseName msaIndicatorData oneDayLodgingCost scenarioName stateIndexing 
VLJSwitch
global maxDist maxDistIndex

% Initialize trip tables
INCOME  = [25000 45000 80000 125000 170000]; % mean/median values of 
household incomes
incomeIndex = size(INCOME, 2);

% initialize constants
route_TYPES    = 3;

Auto_travelTime_LodgingTime_2way  = zeros(52, maxDistIndex);
Auto_travelCost_LodgingCost_2way  = zeros(incomeIndex, 52, maxDistIndex);
Auto_all_trips                    = Auto_travelCost_LodgingCost_2way;

CA_travelTime_SD_LodgingTime  = zeros(52, maxDistIndex, route_TYPES);
CA_flightTime_noScheduleDelay = zeros(52, maxDistIndex, route_TYPES);

CA_travelCost_LodgingCost     = zeros(incomeIndex, 52, maxDistIndex, 
route_TYPES);
CA_all_trips                  = zeros(incomeIndex, 52, maxDistIndex, 
route_TYPES);
CA_binRoutes                  = zeros(incomeIndex, 52, maxDistIndex, 
route_TYPES);

newValues   = [1 4 2];
route_TYPES = size(newValues, 2);

if VLJSwitch == 2
    VLJ_TravelTime = zeros(msa_TYPES, distance_INDEX);
```

```
    VLJ_TravelCost = zeros(msa_TYPES,  incomeIndex, distance_INDEX);
    VLJ_all_trips  = VLJ_TravelCost;
end % if VLJSwitch == 2


%-----------------------------------------------------
% Start Mode Split
%-----------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCounty;
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number

    % keep track of location in run
    if mod(orgCountyReal, 30) == 0; disp(['Origin County: ',
num2str(orgCountyReal), ' time: ', num2str(toc) ]); end

    if (msaIndicatorData(orgCountyReal,2) ~= 1)
        originFIPS = msaIndicatorData(orgCountyReal,1);
        startDestStateCounter = 0;

        alaskaOrigin = ~isempty(find(orgCountyReal == 68:78));
        hawaiiOrigin = ~isempty(find(orgCountyReal == 527:530));

        if (alaskaOrigin == 1 || hawaiiOrigin == 1)
        else
            for desCounty = StateStartIndex : StateEndIndex %1 : numbCols %
loop for destination counties
                desCounty;

                checkState = find(desCounty == stateIndexing(:, 1));
                if ~isempty(checkState)
                    startDestStateCounter = startDestStateCounter + 1;
                end % if ~isempty(checkState)

                if (msaIndicatorData(desCounty,2) ~= 1)

                    alaskaDestnt = ~isempty(find(desCounty == 68:78));
                    hawaiiDestnt = ~isempty(find(desCounty == 527:530));

                    route_Distance =
C2CDriveDist_Population_Centroids(orgCounty, desCounty);

                    if alaskaDestnt == 1 || hawaiiDestnt == 1
                    else
                        % check if intercity trip
                        if route_Distance >= 100 && route_Distance < maxDist

                            trips_1 = CA_TripTable(1).trips(orgCounty,
desCounty);
                            trips_2 = CA_TripTable(2).trips(orgCounty,
desCounty);
                            trips_3 = CA_TripTable(3).trips(orgCounty,
desCounty);
                            trips_4 = CA_TripTable(4).trips(orgCounty,
desCounty);
```

```
                            trips_5 = CA_TripTable(5).trips(orgCounty,
desCounty);
                            tot_Trips = [trips_1; trips_2; trips_3; trips_4;
trips_5];
                            sum_tot_trips = trips_1 + trips_2 + trips_3 +
trips_4 + trips_5;

                            % check if trips b/n O-D pair
                            if sum_tot_trips > 0

                                dist_index = floor(route_Distance/50 - 1);

                                % AUTO TRAVEL TIME
                                %extract Auto round-trip travel time from
table
                                Auto_TravelTime_2way =
c2cAuto_TT_2way(orgCounty, desCounty);
                                if Auto_TravelTime_2way < 0
                                    orgCounty
                                    disp('Zero Auto Travel Time?'); pause;
                                end % if Auto_TravelTime_2way < 0

Auto_travelTime_LodgingTime_2way(startDestStateCounter, dist_index) =
Auto_travelTime_LodgingTime_2way(startDestStateCounter, dist_index) +
Auto_TravelTime_2way * sum_tot_trips;

                                % AUTO TRAVEL COST
                                Auto_travelCost_LodgingCost_2way(:,
startDestStateCounter, dist_index) = ...
                                    Auto_travelCost_LodgingCost_2way(:,
startDestStateCounter, dist_index) + c2cAuto_TC_2way(:, orgCounty, desCounty)
.* tot_Trips;
                                Auto_all_trips(:, startDestStateCounter,
dist_index) = Auto_all_trips(:, startDestStateCounter, dist_index) +
tot_Trips;



                                % airport index
                                airport_row_Numbers =
CA_index(:,orgCounty,desCounty); %9x1

                                % CA TRAVEL TIME
                                travelTime_includingScheduleDelay  =
CA_travelTime_with_SD_2way(:,orgCounty,desCounty); % 3x1
                                flightTime_noScheduleDelay         =
CA_flightTime_no_SD_2way(:,orgCounty,desCounty); % 3x1
                                overnightTime_CA_2way              =
CA_numberOfNights(:,orgCounty,desCounty) * CA_dailyOvernightTime; % 3x1

                                % CA TRAVEL COST
                                accEgrsCost_CA_2way_1 =
CA_accEgrsCost_2way(:,orgCounty,desCounty);
                                accessRows = size(accEgrsCost_CA_2way_1,1);
                                accEgrsCost_CA_2way = zeros(5, accessRows);
                                for cntRt = 1 : accessRows
```

```
                                    accEgrsCost_CA_2way(:, cntRt) =
accEgrsCost_CA_2way_1(cntRt) ./ Auto_averageOccupancy'; %#ok<AGROW>
                                end % for cntRt = 1 :
size(accEgrsCost_CA_2way_1,1)

                                averageFare_CA_2way =
CA_AverageFare(:,:,orgCounty,desCounty); % 3x4
                                overnight_TC_CA     =
CA_numberOfNights(:,orgCounty,desCounty) * oneDayLodgingCost; % 3x4
                                overnight_TC_CA     = overnight_TC_CA';

                                % Delete CA routes without Information
                                zeroIndex = find(airport_row_Numbers == 0 );
                                if ~isempty(zeroIndex)
                                    airport_row_Numbers(zeroIndex)      = [];

travelTime_includingScheduleDelay(zeroIndex) = [];
                                    overnightTime_CA_2way(zeroIndex, :) = [];
                                    accEgrsCost_CA_2way(:, zeroIndex)   = [];
                                    averageFare_CA_2way(:, zeroIndex)   = [];
                                    overnight_TC_CA(:, zeroIndex)       = [];
                                    flightTime_noScheduleDelay(zeroIndex, :)
= [];
                                end % if ~isempty(zeroIndex)

                                if isempty(airport_row_Numbers)
                                else
                                    for route_type = 1 : route_TYPES
                                        airportIndexSelect =
newValues(route_type);
                                        row_Index = find(airportIndexSelect
== airport_row_Numbers);
                                        if isempty(row_Index)
                                        else
                                            CA_pureFare_total_2way =
averageFare_CA_2way(row_Index, :);
                                            if CA_pureFare_total_2way > 0

                                                check_TT_index =
find(travelTime_includingScheduleDelay == 0);
                                                if isempty(check_TT_index)

total_CA_travelTime_withLoding = travelTime_includingScheduleDelay +
overnightTime_CA_2way;

total_CA_travelCost_withLodging = accEgrsCost_CA_2way(:,row_Index) +
averageFare_CA_2way(:,row_Index) + overnight_TC_CA(:,row_Index);
                                                else
                                                    disp('Problem with Travel
Time')
                                                    pause
                                                end % if
isempty(check_TT_index)


CA_travelTime_SD_LodgingTime(startDestStateCounter, dist_index, route_type) =
...
```

```
CA_travelTime_SD_LodgingTime(startDestStateCounter, dist_index, route_type)
+ total_CA_travelTime_withLoding(row_Index) * sum_tot_trips;


CA_flightTime_noScheduleDelay(startDestStateCounter, dist_index, route_type)
= ...

CA_flightTime_noScheduleDelay(startDestStateCounter, dist_index, route_type)
+ flightTime_noScheduleDelay(row_Index) * sum_tot_trips;


                                                CA_travelCost_LodgingCost(:,
startDestStateCounter, dist_index, route_type) = CA_travelCost_LodgingCost(:,
startDestStateCounter, dist_index, route_type) ...
                                                    +
total_CA_travelCost_withLodging(row_Index, :) .* tot_Trips;

                                                CA_all_trips(:,
startDestStateCounter, dist_index, route_type) = CA_all_trips(:,
startDestStateCounter, dist_index, route_type) + tot_Trips;

                                                CA_binRoutes(:,
startDestStateCounter, dist_index, route_type) = CA_binRoutes(:,
startDestStateCounter, dist_index, route_type) + 1;

                                            end % if CA_pureFare_total_2way >
0
                                        end % if isempty(row_Index)
                                    end % for route_type = 1 : route_TYPES
                                end % if isempty(airport_row_Numbers)

                            end % if sum_trips_income > 0
                        end % if route_Distance >= 100 && route_Distance <
3050
                    end % if alaskaDestnt == 1 | hawaiiDestnt == 1

                end % if (msaIndicatorData(desCounty,2) == 1)
            end % for desCounty = StateStartIndex : StateEndIndex
        end % if alaskaOrigin == 1 | hawaiiOrigin == 1
    end % if (msaIndicatorData(orgCountyReal,2) == 1)
end % for orgCounty = 1 : numbRows % loop for origin counties

joinDirName = [saveDirName scenarioName caseName stateName];

save (strcat(joinDirName, '_Auto_travelTime_LodgingTime_2way_nonMSA_nonMSA'),
'Auto_travelTime_LodgingTime_2way')
save (strcat(joinDirName, '_Auto_travelCost_LodgingCost_2way_nonMSA_nonMSA'),
'Auto_travelCost_LodgingCost_2way')
save (strcat(joinDirName, '_Auto_all_trips_nonMSA_nonMSA'), 'Auto_all_trips')

save (strcat(joinDirName, '_CA_travelTime_SD_LodgingTime_nonMSA_nonMSA'),
'CA_travelTime_SD_LodgingTime')
save (strcat(joinDirName, '_CA_flightTime_noScheduleDelay_nonMSA_nonMSA'),
'CA_flightTime_noScheduleDelay')
```

```
save (strcat(joinDirName, '_CA_travelCost_LodgingCost_nonMSA_nonMSA'),
'CA_travelCost_LodgingCost')
save (strcat(joinDirName, '_CA_all_trips_nonMSA_nonMSA'), 'CA_all_trips')
save (strcat(joinDirName, '_CA_binRoutes_nonMSA_nonMSA'), 'CA_binRoutes')

return
```

----------------------------------------------------------------------------------------------------------------------

### 7.4.3 Aggregate MSA to MSA data by Distance, Income, Region, and Commercial Air Route

```
-----------------------------------------------------------------------------------------------------------------------
-----------
function aggregate_TT_TC_MSA_MSA()

tic; clear all; clc

% Load Tables
%--------------
InstallDir = 'C:\Program Files\TSAM\4.2\data\mode_choice\';
load (strcat(InstallDir, 'input\stateNames'))    % List of states in US
load (strcat(InstallDir, 'input\stateIndexing')) % Index showing where
counties of each state begin and end in the county to county lis
load (strcat(InstallDir, '\input\msaIndicatorData'))         % List
indicating if county is in an MSA. (1 -> MSA; 0 -> nonMSA)

% Name of reference directory
%-----------------------------
LoadDirName  = 'D:\Mode Choice\Calibration_February_07_four\';
scenarioName = 'Nested Logit Travel Times and Costs\';
load ([LoadDirName,
'Create_Calibration_TravelTimeAndCost\MSA_2_MSA_DistanceBins'])

[rows cols] = size(MSA_2_MSA_RouteDistance_Unweighted);
MSA_2_MSA_DistanceBins = zeros(rows, cols);
for i = 1 : rows
    for k = 1 : cols
        distanceValue = MSA_2_MSA_RouteDistance_Unweighted(i,k);
        MSA_2_MSA_DistanceBins(i,k) = floor(distanceValue/50 - 1);
    end % for k = 1 : cols
end % for i = 1 : rows
save ([load_Dir,
'\Create_Calibration_TravelTimeAndCost\MSA_2_MSA_DistanceBins'],
'MSA_2_MSA_DistanceBins')
return

warning('off')
for tripType = 1 : 2

    disp(['Elasped Time: ', num2str(toc)])
    if tripType == 1;
        caseName  = 'business\';
        SAVENAME  = 'Business';
    else
        caseName  = 'nonBusiness\';
        SAVENAME  = 'NonBusiness';
    end % if tripType == 1

    upperDirName = [LoadDirName scenarioName caseName];

    load ([upperDirName, 'auto_TT'])
    load ([upperDirName, 'auto_TC'])

    AverageTravelTime_Auto_All = zeros(69,52);
    AverageTravelCost_Auto_All = zeros(69,5,52);
    for stateCounter = 1 : 52

        AverageTravelTime_Auto = zeros(69,1);
        AverageTravelCost_Auto = zeros(69,5);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
                disp(['Auto Origin State: ', num2str(stateCounter), ' time:
', num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            startRow = stateIndexing(stateCounter, 1);
            endRow   = stateIndexing(stateCounter, 2);
            nRows    = endRow - startRow + 1;

            countiesInState           = msaIndicatorData(startRow:endRow,:);
            stateMSA_Counties_Index   = find(countiesInState(:, 2));
            stateMSA_Counties         =
countiesInState(stateMSA_Counties_Index,:);
            currentState_MSAs         = unique(stateMSA_Counties(:,4));
            MSA_2_MSA_IndexDistance =
MSA_2_MSA_DistanceBins(currentState_MSAs,:);
            clear currentState_MSAs startRow endRow nRows countiesInState
stateMSA_Counties_Index stateMSA_Counties

            travelTime_auto = auto_TT(stateCounter).Data;
            nanIndex        = isnan(travelTime_auto);
            travelTime_auto(nanIndex) = 0; clear nanIndex

            [rowVariable colVariable] = size(travelTime_auto);
            [rowMSAs colMSAs] = size(MSA_2_MSA_IndexDistance);
            if (rowVariable == rowMSAs) && (colVariable == colMSAs)
            else
                disp('Error in MSA Info')
            end % if (rowVariable == rowMSAs) && (colVariable == colMSAs)
            clear rowVariable colVariable rowMSAs colMSAs

            travelCost_auto = auto_TC(stateCounter).Data;
            nanIndex        = isnan(travelCost_auto);
            travelCost_auto(nanIndex) = 0; clear nanIndex

            travelCost_auto = rearrangeTableDim_MSA_MSA_3(travelCost_auto);

            for cntDist = 1 : 69
                VariableIndex  = find(MSA_2_MSA_IndexDistance == cntDist);
                if ~isempty(VariableIndex)
                    AverageTravelTime_Auto(cntDist, 1) =
mean(nonzeros(travelTime_auto(VariableIndex)));

                    for i = 1 : 5
                        tableValues = travelCost_auto(:,:,i);
                        AverageTravelCost_Auto(cntDist, i) =
mean(nonzeros(tableValues(VariableIndex)));
                    end % for i = 1 : 5
                end % if ~isempty(VariableIndex)
            end % for cntDist = 1 : 69
            clear VariableIndex travelTime_auto travelCost_auto tableValues
timeValues
        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Auto_All(:,stateCounter) = AverageTravelTime_Auto;
        AverageTravelCost_Auto_All(:,:,stateCounter) =
AverageTravelCost_Auto;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Auto AverageTravelCost_Auto auto_TT auto_TC
    save (['AverageTravelTime_Auto_All_MSA_MSA_' SAVENAME],
'AverageTravelTime_Auto_All')
    save (['AverageTravelCost_Auto_All_MSA_MSA_' SAVENAME],
'AverageTravelCost_Auto_All')
    clear AverageTravelTime_Auto_All AverageTravelCost_Auto_All

    load ([upperDirName, 'CA_TT'])
    load ([upperDirName, 'CA_TC'])
    load ([upperDirName, 'CA_FT'])

    AverageTravelTime_Comm_All = zeros(69,3,52);
    AverageFlightTime_Comm_All = zeros(69,3,52);
    AverageTravelCost_Comm_All = zeros(69,5,3,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Comm = zeros(69,3);
        AverageFlightTime_Comm = zeros(69,3);
        AverageTravelCost_Comm = zeros(69,5,3);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
                disp(['CA Origin State: ', num2str(stateCounter), ' time: ',
num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            stateName = char(stateNames(stateCounter, 1));
            joinDirName = [LoadDirName scenarioName caseName stateName];

            startRow = stateIndexing(stateCounter, 1);
            endRow   = stateIndexing(stateCounter, 2);
            nRows    = endRow - startRow + 1;

            countiesInState           = msaIndicatorData(startRow:endRow,:);
            stateMSA_Counties_Index   = find(countiesInState(:, 2));
            stateMSA_Counties         =
countiesInState(stateMSA_Counties_Index,:);
            currentState_MSAs         = unique(stateMSA_Counties(:,4));
            MSA_2_MSA_IndexDistance =
MSA_2_MSA_DistanceBins(currentState_MSAs,:);
            clear currentState_MSAs startRow endRow nRows countiesInState
stateMSA_Counties_Index stateMSA_Counties

            travelTime_comm = CA_TT(stateCounter).Data;
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            nanIndex        = isnan(travelTime_comm);
            travelTime_comm(nanIndex) = 0; clear nanIndex

            flightTime_comm = CA_FT(stateCounter).Data;
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            nanIndex        = isnan(flightTime_comm);
            flightTime_comm(nanIndex) = 0; clear nanIndex


            travelCost_comm = CA_TC(stateCounter).Data;
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            nanIndex        = isnan(travelCost_comm);
            travelCost_comm(nanIndex) = 0; clear nanIndex


            for cntDist = 1 : 69
                VariableIndex  = find(MSA_2_MSA_IndexDistance == cntDist);

                if ~isempty(VariableIndex)
                    for i = 1: 3
                        tableValues_1 = travelTime_comm(:,:,i);
                        AverageTravelTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_1(VariableIndex)));
                        tableValues_2 = flightTime_comm(:,:,i);
                        AverageFlightTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_2(VariableIndex)));
                        for k = 1 : 5
                            tableValues_3 = travelCost_comm(:,:,k,i);
                            AverageTravelCost_Comm(cntDist, k, i) =
mean(nonzeros(tableValues_3(VariableIndex)));
                        end
                    end % for i = 1: 3
                end % if ~isempty(VariableIndex)
            end % for cntDist = 1 : 69

            clear tableValues_1 tableValues_2 tableValues_3
            clear travelTime_comm flightTime_comm travelCost_comm

        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Comm_All(:,:,stateCounter) =
AverageTravelTime_Comm;
        AverageFlightTime_Comm_All(:,:,stateCounter) =
AverageFlightTime_Comm;
        AverageTravelCost_Comm_All(:,:,:,stateCounter) =
AverageTravelCost_Comm;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Comm AverageFlightTime_Comm
AverageTravelCost_Comm CA_TT CA_FT CA_TC
    save (['AverageTravelTime_Comm_All_MSA_MSA_' SAVENAME],
'AverageTravelTime_Comm_All')
    save (['AverageFlightTime_Comm_All_MSA_MSA_' SAVENAME],
'AverageFlightTime_Comm_All')
    save (['AverageTravelCost_Comm_All_MSA_MSA_' SAVENAME],
'AverageTravelCost_Comm_All')
```

```
    clear AverageTravelCost_Comm_All AverageFlightTime_Comm_All
AverageTravelCost_Comm_All

end % for tripType = 1 : 2

aggregate_TT_TC_MSA_nonMSA()

return
```

-----------------------------------------------------------------------------------------------------------------------

### *7.4.4 Aggregate MSA to non-MSA data by Distance, Income, Region, and Commercial Air Route*

```
------------------------------------------------------------------------------------------------------------------------------
--------
function aggregate_TT_TC_MSA_nonMSA()

tic; clear all; clc

% Load Tables
%--------------
InstallDir = 'C:\Program Files\TSAM\4.2\data\mode_choice\';
load (strcat(InstallDir, 'input\stateNames'))    % List of states in US
load (strcat(InstallDir, 'input\stateIndexing')) % Index showing where
counties of each state begin and end in the county to county lis
load (strcat(InstallDir, '\input\msaIndicatorData'))         % List
indicating if county is in an MSA. (1 -> MSA; 0 -> nonMSA)

% Name of reference directory
%-----------------------------
LoadDirName  = 'D:\Mode Choice\Calibration_February_07_four\';
scenarioName   = 'Nested Logit Travel Times and Costs\';
load ([LoadDirName,
'Create_Calibration_TravelTimeAndCost\MSA_2_MSA_DistanceBins'])

warning('off')
for tripType = 1 : 2

    disp(['Elasped Time: ', num2str(toc)])
    if tripType == 1;
        caseName  = 'business\';
        SAVENAME  = 'Business';
    else
        caseName  = 'nonBusiness\';
        SAVENAME  = 'NonBusiness';
    end % if tripType == 1

    upperDirName = [LoadDirName scenarioName caseName];

    load ([upperDirName, 'auto_TT_MSA_nonMSA'])
    load ([upperDirName, 'auto_TC_MSA_nonMSA'])

    AverageTravelTime_Auto_All = zeros(69,52);
    AverageTravelCost_Auto_All = zeros(69,5,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Auto = zeros(69,1);
        AverageTravelCost_Auto = zeros(69,5);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
                disp(['Auto Origin State: ', num2str(stateCounter), ' time:
', num2str(toc) ]);
```

```
            end % if mod(stateCounter, 5) == 0;

            stateName   = char(stateNames(stateCounter, 1));
            joinDirName = [LoadDirName scenarioName caseName stateName];

            startRow = stateIndexing(stateCounter, 1);
            endRow   = stateIndexing(stateCounter, 2);
            nRows    = endRow - startRow + 1;

            countiesInState           = msaIndicatorData(startRow:endRow,:);
            stateMSA_Counties_Index   = find(countiesInState(:, 2));
            stateMSA_Counties         =
countiesInState(stateMSA_Counties_Index,:);
            currentState_MSAs         = unique(stateMSA_Counties(:,4));
            MSA_2_MSA_IndexDistance =
MSA_2_MSA_DistanceBins(currentState_MSAs,:);
            clear currentState_MSAs startRow endRow nRows countiesInState
stateMSA_Counties_Index stateMSA_Counties

            travelTime_auto = auto_TT(stateCounter).Data;
            nanIndex        = isnan(travelTime_auto);
            travelTime_auto(nanIndex) = 0; clear nanIndex
            travelTime_auto = rearrangeTableDim_MSA_MSA_3(travelTime_auto);

            [rowVariable colVariable index] = size(travelTime_auto);
            [rowMSAs colMSAs]         = size(MSA_2_MSA_IndexDistance);
            if (colVariable == rowMSAs)
            else
                disp('Error in MSA Info')
            end % if (rowVariable == rowMSAs) && (colVariable == colMSAs)
            clear rowVariable colVariable rowMSAs colMSAs

            travelCost_auto = auto_TC(stateCounter).Data;
            nanIndex        = isnan(travelCost_auto);
            travelCost_auto(nanIndex) = 0; clear nanIndex
            travelCost_auto = rearrangeTableDim_MSA_MSA_4(travelCost_auto);
            travelCost_auto = rearrangeTableDim_MSA_MSA_4(travelCost_auto);

            for cntDist = 1 : 69
                AverageTravelTime_Auto(cntDist, 1) =
mean(nonzeros(travelTime_auto(:,:,cntDist)));
                for i = 1 : 5
                    tableValues = travelCost_auto(:,:,i,:);
                    AverageTravelCost_Auto(cntDist, i) =
mean(nonzeros(tableValues(:,:,:,cntDist)));
                end % for i = 1 : 5
            end % for cntDist = 1 : 69
            clear travelTime_auto tableValues tableValues travelCost_auto
        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Auto_All(:,stateCounter) = AverageTravelTime_Auto;
        AverageTravelCost_Auto_All(:,:,stateCounter) =
AverageTravelCost_Auto;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Auto AverageTravelCost_Auto auto_TT auto_TC
    save (['AverageTravelTime_Auto_All_MSA_nonMSA_' SAVENAME],
'AverageTravelTime_Auto_All')
    save (['AverageTravelCost_Auto_All_MSA_nonMSA_' SAVENAME],
'AverageTravelCost_Auto_All')
    clear AverageTravelTime_Auto_All AverageTravelCost_Auto_All


    load ([upperDirName, 'CA_TT_MSA_nonMSA'])
    load ([upperDirName, 'CA_TC_MSA_nonMSA'])
    load ([upperDirName, 'CA_FT_MSA_nonMSA'])

    AverageTravelTime_Comm_All = zeros(69,3,52);
    AverageFlightTime_Comm_All = zeros(69,3,52);
    AverageTravelCost_Comm_All = zeros(69,5,3,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Comm = zeros(69,3);
        AverageFlightTime_Comm = zeros(69,3);
        AverageTravelCost_Comm = zeros(69,5,3);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
                disp(['CA Origin State: ', num2str(stateCounter), ' time: ',
num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            stateName = char(stateNames(stateCounter, 1));
            joinDirName = [LoadDirName scenarioName caseName stateName];

            travelTime_comm = CA_TT(stateCounter).Data;
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            nanIndex        = isnan(travelTime_comm);
            travelTime_comm(nanIndex) = 0; clear nanIndex

            flightTime_comm = CA_FT(stateCounter).Data;
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            nanIndex        = isnan(flightTime_comm);
            flightTime_comm(nanIndex) = 0; clear nanIndex

            travelCost_comm = CA_TC(stateCounter).Data;
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            nanIndex        = isnan(travelCost_comm);
            travelCost_comm(nanIndex) = 0; clear nanIndex

            for cntDist = 1 : 69
                for i = 1: 3
                    tableValues_1 = travelTime_comm(:,i,:);
                    AverageTravelTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_1(:,:,cntDist)));
                    tableValues_2 = flightTime_comm(:,i,:);
                    AverageFlightTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_2(:,:,cntDist)));
                    for k = 1 : 5
                        tableValues_3 = travelCost_comm(:,k,i,:);
                        AverageTravelCost_Comm(cntDist, k, i) =
mean(nonzeros(tableValues_3(:,:,:,cntDist)));
                    end
                end % for i = 1: 3
            end % for cntDist = 1 : 69
            clear tableValues_1 tableValues_2 tableValues_3
            clear travelTime_comm flightTime_comm travelCost_comm

        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Comm_All(:,:,stateCounter) =
AverageTravelTime_Comm;
        AverageFlightTime_Comm_All(:,:,stateCounter) =
AverageFlightTime_Comm;
        AverageTravelCost_Comm_All(:,:,:,stateCounter) =
AverageTravelCost_Comm;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Comm AverageFlightTime_Comm
AverageTravelCost_Comm CA_TT CA_FT CA_TC
    save (['AverageTravelTime_Comm_All_MSA_nonMSA_' SAVENAME],
'AverageTravelTime_Comm_All')
    save (['AverageFlightTime_Comm_All_MSA_nonMSA_' SAVENAME],
'AverageFlightTime_Comm_All')
    save (['AverageTravelCost_Comm_All_MSA_nonMSA_' SAVENAME],
'AverageTravelCost_Comm_All')
    clear AverageTravelCost_Comm_All AverageFlightTime_Comm_All
AverageTravelCost_Comm_All

end % for tripType = 1 : 2


aggregate_TT_TC_nonMSA_MSA()

return
```

-----------------------------------------------------------------------------------------------------------------------

### 7.4.5 *Aggregate non-MSA to MSA data by Distance, Income, Region, and Commercial Air Route*

```
--------------------------------------------------------------------------------------------------------------------------------
function aggregate_TT_TC_nonMSA_MSA()

tic; clear all; clc

% Load Tables
%--------------
InstallDir = 'C:\Program Files\TSAM\4.2\data\mode_choice\';
load (strcat(InstallDir, 'input\stateNames'))    % List of states in US
load (strcat(InstallDir, 'input\stateIndexing')) % Index showing where
counties of each state begin and end in the county to county lis

% Name of reference directory
%-----------------------------
LoadDirName  = 'D:\Mode Choice\Calibration_February_07_four\';
scenarioName   = 'Nested Logit Travel Times and Costs\';
load ([LoadDirName,
'Create_Calibration_TravelTimeAndCost\MSA_2_MSA_DistanceBins'])


warning('off')
for tripType = 1 : 2

    disp(['Elasped Time: ', num2str(toc)])
    if tripType == 1;
        caseName  = 'business\';
        SAVENAME  = 'Business';
    else
        caseName  = 'nonBusiness\';
        SAVENAME  = 'NonBusiness';
    end % if tripType == 1

    upperDirName = [LoadDirName scenarioName caseName];

    load ([upperDirName, 'auto_TT_nonMSA_MSA'])
    load ([upperDirName, 'auto_TC_nonMSA_MSA'])

    AverageTravelTime_Auto_All = zeros(69,52);
    AverageTravelCost_Auto_All = zeros(69,5,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Auto = zeros(69,1);
        AverageTravelCost_Auto = zeros(69,5);

        if (stateCounter == 2 || stateCounter == 12)
        else
             if mod(stateCounter, 10) == 0;
                disp(['Auto Origin State: ', num2str(stateCounter), ' time:
', num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;
            travelTime_auto = auto_TT(stateCounter).Data;
            nanIndex        = isnan(travelTime_auto);
            travelTime_auto(nanIndex) = 0; clear nanIndex

            travelCost_auto = auto_TC(stateCounter).Data;
            travelCost_auto = rearrangeTableDim_MSA_MSA_3(travelCost_auto);
            travelCost_auto = rearrangeTableDim_MSA_MSA_3(travelCost_auto);
            nanIndex        = isnan(travelCost_auto);
            travelCost_auto(nanIndex) = 0; clear nanIndex

            for cntDist = 1 : 69
                AverageTravelTime_Auto(cntDist, 1) =
mean(nonzeros(travelTime_auto(:,cntDist)));
                for i = 1 : 5
                    tableValues = travelCost_auto(:,i,:);
                    AverageTravelCost_Auto(cntDist, i) =
mean(nonzeros(tableValues(:,:,cntDist)));
                end % for i = 1 : 5
            end % for cntDist = 1 : 69
            clear travelTime_auto tableValues tableValues travelCost_auto
        end % if (stateCounter == 2 || stateCounter == 12)

         AverageTravelTime_Auto_All(:,stateCounter) = AverageTravelTime_Auto;
        AverageTravelCost_Auto_All(:,:,stateCounter) =
AverageTravelCost_Auto;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Auto AverageTravelCost_Auto auto_TT auto_TC
    save (['AverageTravelTime_Auto_All_nonMSA_MSA_' SAVENAME],
'AverageTravelTime_Auto_All')
    save (['AverageTravelCost_Auto_All_nonMSA_MSA_' SAVENAME],
'AverageTravelCost_Auto_All')
    clear AverageTravelTime_Auto_All AverageTravelCost_Auto_All


    load ([upperDirName, 'CA_TT_nonMSA_MSA'])
    load ([upperDirName, 'CA_TC_nonMSA_MSA'])
    load ([upperDirName, 'CA_FT_nonMSA_MSA'])

    AverageTravelTime_Comm_All = zeros(69,3,52);
    AverageFlightTime_Comm_All = zeros(69,3,52);
    AverageTravelCost_Comm_All = zeros(69,5,3,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Comm = zeros(69,3);
        AverageFlightTime_Comm = zeros(69,3);
        AverageTravelCost_Comm = zeros(69,5,3);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
                disp(['CA Origin State: ', num2str(stateCounter), ' time: ',
num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            travelTime_comm = CA_TT(stateCounter).Data;
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            nanIndex        = isnan(travelTime_comm);
            travelTime_comm(nanIndex) = 0; clear nanIndex

            flightTime_comm = CA_FT(stateCounter).Data;
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            nanIndex        = isnan(flightTime_comm);
            flightTime_comm(nanIndex) = 0; clear nanIndex

            travelCost_comm = CA_TC(stateCounter).Data;
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            nanIndex        = isnan(travelCost_comm);
            travelCost_comm(nanIndex) = 0; clear nanIndex

            for cntDist = 1 : 69
                for i = 1: 3
                    tableValues_1 = travelTime_comm(:,i,:);
                    AverageTravelTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_1(:,:,cntDist)));
                    tableValues_2 = flightTime_comm(:,i,:);
                    AverageFlightTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_2(:,:,cntDist)));
                    for k = 1 : 5
                        tableValues_3 = travelCost_comm(:,k,i,:);
                        AverageTravelCost_Comm(cntDist, k, i) =
mean(nonzeros(tableValues_3(:,:,:,cntDist)));
                    end
                end % for i = 1: 3
            end % for cntDist = 1 : 69
            clear tableValues_1 tableValues_2 tableValues_3
            clear travelTime_comm flightTime_comm travelCost_comm

        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Comm_All(:,:,stateCounter) =
AverageTravelTime_Comm;
        AverageFlightTime_Comm_All(:,:,stateCounter) =
AverageFlightTime_Comm;
        AverageTravelCost_Comm_All(:,:,:,stateCounter) =
AverageTravelCost_Comm;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Comm AverageFlightTime_Comm
AverageTravelCost_Comm CA_TT CA_FT CA_TC
    save (['AverageTravelTime_Comm_All_nonMSA_MSA_' SAVENAME],
'AverageTravelTime_Comm_All')
    save (['AverageFlightTime_Comm_All_nonMSA_MSA_' SAVENAME],
'AverageFlightTime_Comm_All')
    save (['AverageTravelCost_Comm_All_nonMSA_MSA_' SAVENAME],
'AverageTravelCost_Comm_All')
    clear AverageTravelCost_Comm_All AverageFlightTime_Comm_All
AverageTravelCost_Comm_All
```

```
end % for tripType = 1 : 2

% aggregate_TT_TC_nonMSA_nonMSA()

return
```

----------------------------------------------------------------------------------------------------------------------------

### 7.4.6 Aggregate non-MSA to non-MSA data by Distance, Income, Region, and Commercial Air Route

```
---------------------------------------------------------------------------------------------------------------------------
-------
function aggregate_TT_TC_nonMSA_nonMSA()

tic; clear all; clc

% Load Tables
%--------------
InstallDir = 'C:\Program Files\TSAM\4.2\data\mode_choice\';
load (strcat(InstallDir, 'input\stateNames'))    % List of states in US
load (strcat(InstallDir, 'input\stateIndexing')) % Index showing where
counties of each state begin and end in the county to county lis


% Name of reference directory
%-----------------------------
LoadDirName  = 'D:\Mode Choice\Calibration_February_07_four\';
saveDirName    = 'D:\Mode Choice\Calibration_February_07_four\';
scenarioName   = 'Nested Logit Travel Times and Costs\';


warning('off')
for tripType = 1 : 2

    disp(['Elasped Time: ', num2str(toc)])
    if tripType == 1;
        caseName  = 'business\';
        SAVENAME  = 'Business';
    else
        caseName  = 'nonBusiness\';
        SAVENAME  = 'NonBusiness';
    end % if tripType == 1

    upperDirName = [LoadDirName scenarioName caseName];

    load ([upperDirName, 'auto_TT_nonMSA_nonMSA'])
    load ([upperDirName, 'auto_TC_nonMSA_nonMSA'])

    AverageTravelTime_Auto_All = zeros(69,52);
    AverageTravelCost_Auto_All = zeros(69,5,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Auto = zeros(69,1);
        AverageTravelCost_Auto = zeros(69,5);

        if (stateCounter == 2 || stateCounter == 12)
        else
             if mod(stateCounter, 10) == 0;
                disp(['Auto Origin State: ', num2str(stateCounter), ' time:
', num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;
            stateName   = char(stateNames(stateCounter, 1));
            joinDirName = [saveDirName scenarioName caseName stateName];

            travelTime_auto = auto_TT(stateCounter).Data;
            nanIndex        = isnan(travelTime_auto);
            travelTime_auto(nanIndex) = 0; clear nanIndex

            travelCost_auto = auto_TC(stateCounter).Data;
            travelCost_auto = rearrangeTableDim_MSA_MSA_3(travelCost_auto);
            travelCost_auto = rearrangeTableDim_MSA_MSA_3(travelCost_auto);
            nanIndex        = isnan(travelCost_auto);
            travelCost_auto(nanIndex) = 0; clear nanIndex

            for cntDist = 1 : 69
                AverageTravelTime_Auto(cntDist, 1) =
mean(nonzeros(travelTime_auto(:,cntDist)));
                for i = 1 : 5
                    tableValues = travelCost_auto(:,i,:);
                    AverageTravelCost_Auto(cntDist, i) =
mean(nonzeros(tableValues(:,:,cntDist)));
                end % for i = 1 : 5
            end % for cntDist = 1 : 69
            clear travelTime_auto tableValues tableValues travelCost_auto
        end % if (stateCounter == 2 || stateCounter == 12)

         AverageTravelTime_Auto_All(:,stateCounter) = AverageTravelTime_Auto;
        AverageTravelCost_Auto_All(:,:,stateCounter) =
AverageTravelCost_Auto;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Auto AverageTravelCost_Auto auto_TT auto_TC
    save (['AverageTravelTime_Auto_All_nonMSA_nonMSA_' SAVENAME],
'AverageTravelTime_Auto_All')
    save (['AverageTravelCost_Auto_All_nonMSA_nonMSA_' SAVENAME],
'AverageTravelCost_Auto_All')
    clear AverageTravelTime_Auto_All AverageTravelCost_Auto_All


    load ([upperDirName, 'CA_TT_nonMSA_nonMSA'])
    load ([upperDirName, 'CA_TC_nonMSA_nonMSA'])
    load ([upperDirName, 'CA_FT_nonMSA_nonMSA'])

    AverageTravelTime_Comm_All = zeros(69,3,52);
    AverageFlightTime_Comm_All = zeros(69,3,52);
    AverageTravelCost_Comm_All = zeros(69,5,3,52);

    for stateCounter = 1 : 52

        AverageTravelTime_Comm = zeros(69,3);
        AverageFlightTime_Comm = zeros(69,3);
        AverageTravelCost_Comm = zeros(69,5,3);

        if (stateCounter == 2 || stateCounter == 12)
        else
            if mod(stateCounter, 10) == 0;
```

```
                disp(['CA Origin State: ', num2str(stateCounter), ' time: ',
num2str(toc) ]);
            end % if mod(stateCounter, 5) == 0;

            travelTime_comm = CA_TT(stateCounter).Data;
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            travelTime_comm = rearrangeTableDim_MSA_MSA_3(travelTime_comm);
            nanIndex        = isnan(travelTime_comm);
            travelTime_comm(nanIndex) = 0; clear nanIndex

            flightTime_comm = CA_FT(stateCounter).Data;
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            flightTime_comm = rearrangeTableDim_MSA_MSA_3(flightTime_comm);
            nanIndex        = isnan(flightTime_comm);
            flightTime_comm(nanIndex) = 0; clear nanIndex

            travelCost_comm = CA_TC(stateCounter).Data;
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            travelCost_comm = rearrangeTableDim_MSA_MSA_4(travelCost_comm);
            nanIndex        = isnan(travelCost_comm);
            travelCost_comm(nanIndex) = 0; clear nanIndex

            for cntDist = 1 : 69
                for i = 1: 3
                    tableValues_1 = travelTime_comm(:,i,:);
                    AverageTravelTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_1(:,:,cntDist)));
                    tableValues_2 = flightTime_comm(:,i,:);
                    AverageFlightTime_Comm(cntDist, i) =
mean(nonzeros(tableValues_2(:,:,cntDist)));
                    for k = 1 : 5
                        tableValues_3 = travelCost_comm(:,k,i,:);
                        AverageTravelCost_Comm(cntDist, k, i) =
mean(nonzeros(tableValues_3(:,:,:,cntDist)));
                    end
                end % for i = 1: 3
            end % for cntDist = 1 : 69
            clear tableValues_1 tableValues_2 tableValues_3
            clear travelTime_comm flightTime_comm travelCost_comm

        end % if (stateCounter == 2 || stateCounter == 12)

        AverageTravelTime_Comm_All(:,:,stateCounter) =
AverageTravelTime_Comm;
        AverageFlightTime_Comm_All(:,:,stateCounter) =
AverageFlightTime_Comm;
        AverageTravelCost_Comm_All(:,:,:,stateCounter) =
AverageTravelCost_Comm;
    end % for stateCounter = 1 : 52

    clear AverageTravelTime_Comm AverageFlightTime_Comm
AverageTravelCost_Comm CA_TT CA_FT CA_TC
    save (['AverageTravelTime_Comm_All_nonMSA_nonMSA_' SAVENAME],
'AverageTravelTime_Comm_All')
    save (['AverageFlightTime_Comm_All_nonMSA_nonMSA_' SAVENAME],
'AverageFlightTime_Comm_All')
```

```
    save (['AverageTravelCost_Comm_All_nonMSA_nonMSA_' SAVENAME],
'AverageTravelCost_Comm_All')
    clear AverageTravelCost_Comm_All AverageFlightTime_Comm_All
AverageTravelCost_Comm_All

end % for tripType = 1 : 2

return
```

------------------------------------------------------------------------------------------------------------------------

## 7.5 SUMMARIZE 1995 ATS TRIP DISTRIBUTION OUTPUT

```
-----------------------------------------------------------------------------------------------------------------------
-------
function aggregate_TT_TC_nonMSA_nonMSA()



function summarize_2000_tripDistribution()

%------------------------------------------
% Load Trip Distribtion Data and Summarize
%------------------------------------------

clear all; clc

bintIndex_start  = 100 : 50 :  2950;
bintIndex_end    = 150 : 50 :  3000;

% distPopCent      = [];
% load ('C:\Program
Files\TSAM\4.2\data\mode_choice\input\C2CDriveDist_Population_Centroids')
% for cntDist = 1 : size(bintIndex_start, 2)
%     [rows cols k] = find(C2CDriveDist_Population_Centroids >=
bintIndex_start(cntDist) & C2CDriveDist_Population_Centroids <
bintIndex_end(cntDist));
%     distPopCent(cntDist).row = rows;
%     distPopCent(cntDist).col = cols;
% end % for cntDist = 1 : size(bintIndex_start, 2)
% save ('C:\Program Files\TSAM\4.2\data\mode_choice\input\distPopCent',
'distPopCent')
%
% return

load ('C:\Program Files\TSAM\4.2\data\mode_choice\input\distPopCent')

fileDir = 'D:\TSAM_Project\Trip_Distribution\Output\Y2000';

tic
trips_county     = zeros(size(bintIndex_start, 2),5,2);
tripTable_county = zeros(3091,5,3091);

for cntTripDist = 1 : 2

    if cntTripDist == 1
        load ([fileDir, '\businessTripTable_county_less30K_2000'])
        tripTable_county(:,1,:) =
businessTripTable_county_less30K(1:3091,1:3091);
        clear businessTripTable_county_less30K
        load ([fileDir, '\businessTripTable_county_less60K_2000'])
        tripTable_county(:,2,:)  =
businessTripTable_county_less60K(1:3091,1:3091);
        clear businessTripTable_county_less60K
        load ([fileDir, '\businessTripTable_county_less100K_2000'])
        tripTable_county(:,3,:) =
businessTripTable_county_less100K(1:3091,1:3091);
```

```
        clear businessTripTable_county_less100K
        load ([fileDir, '\businessTripTable_county_less150K_2000'])
        tripTable_county(:,4,:) =
businessTripTable_county_less150K(1:3091,1:3091);
        clear businessTripTable_county_less150K
        load ([fileDir, '\businessTripTable_county_more150K_2000'])
        tripTable_county(:,5,:) =
businessTripTable_county_more150K(1:3091,1:3091);
        clear businessTripTable_county_more150K
    else
        load ([fileDir, '\nonBusinessTripTable_county_less30K_2000'])
        tripTable_county(:,1,:)  =
nonBusinessTripTable_county_less30K(1:3091,1:3091);
        clear nonBusinessTripTable_county_less30K
        load ([fileDir, '\nonBusinessTripTable_county_less60K_2000'])
        tripTable_county(:,2,:)  =
nonBusinessTripTable_county_less60K(1:3091,1:3091);
        clear nonBusinessTripTable_county_less60K
        load ([fileDir, '\nonBusinessTripTable_county_less100K_2000'])
        tripTable_county(:,3,:) =
nonBusinessTripTable_county_less100K(1:3091,1:3091);
        clear nonBusinessTripTable_county_less100K
        load ([fileDir, '\nonBusinessTripTable_county_less150K_2000'])
        tripTable_county(:,4,:) =
nonBusinessTripTable_county_less150K(1:3091,1:3091);
        clear nonBusinessTripTable_county_less150K
        load ([fileDir, '\nonBusinessTripTable_county_more150K_2000'])
        tripTable_county(:,5,:) =
nonBusinessTripTable_county_more150K(1:3091,1:3091);
        clear nonBusinessTripTable_county_more150K
    end % if cntTripDist == 1

    for cntLoop = 1 : size(bintIndex_start, 2)

        if mod(cntLoop, 5) == 0
            disp(['Counter: ', num2str(cntLoop), ' Time: ', num2str(toc/60),
' Minutes'])
        end % if mod(cntLoop, 5) == 0

        rowValues = distPopCent(cntLoop).row;
        colValues = distPopCent(cntLoop).col;
        for t = 1 : size(rowValues, 1)
            trips_county(cntLoop,:,cntTripDist) =
trips_county(cntLoop,:,cntTripDist) +
tripTable_county(rowValues(t),:,colValues(t));
        end % for k = 1 : 5
    end % for cntLoop = 1 : size(bintIndex_start, 2)

end % for cntTripDist = 1 : 2

save ([fileDir, '\trips_county'],'trips_county')

return

--------------------------------------------------------------------------------
-------
```

## 7.6 Create Smoothed American Travel Survey Curves

-----------------------------------------------------------------------------------------------------------------------------------

```
%--------------------------------------------------
% plot smoothed ATS curves (from originLab software
%--------------------------------------------------

clear all; clc

dirName = 'D:\Mode Choice\marketShareCurves_ATS\captive';

smooth_ATS_bin_centers = [125:50:3000]';

for i =1 : 2

    if i == 1
        marketShare(:, 1:2) = csvread([dirName,
'\business\business_inc_1_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 3:4) = csvread([dirName,
'\business\business_inc_2_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 5:6) = csvread([dirName,
'\business\business_inc_3_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 7:8) = csvread([dirName,
'\business\business_inc_4_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 9:10) = csvread([dirName,
'\business\business_inc_5_smoothcurveonly.csv'], 1, 0);
    else
        marketShare(:, 1:2) = csvread([dirName,
'\nonbusiness\nonbusiness_inc_1_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 3:4) = csvread([dirName,
'\nonbusiness\nonbusiness_inc_2_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 5:6) = csvread([dirName,
'\nonbusiness\nonbusiness_inc_3_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 7:8) = csvread([dirName,
'\nonbusiness\nonbusiness_inc_4_smoothcurveonly.csv'], 1, 0);
        marketShare(:, 9:10) = csvread([dirName,
'\nonbusiness\nonbusiness_inc_5_smoothcurveonly.csv'], 1, 0);
    end % if i == 1

    % re-zero marke share values > 100
    n = [0 1:4];
    max_index       = marketShare(1:2, [2+n*2]);
    MS_grt_than_100 = find(max_index > 100);
    max_index(MS_grt_than_100) = 100;
    marketShare(1:2, [2+n*2]) = max_index;

    % interpolate to create market share curves from 0 to 2500
    interpolate_MS_income(:, 1) = interp1(marketShare(:,1), marketShare(:,2),
smooth_ATS_bin_centers);
    interpolate_MS_income(:, 2) = interp1(marketShare(:,3), marketShare(:,4),
smooth_ATS_bin_centers);
    interpolate_MS_income(:, 3) = interp1(marketShare(:,5), marketShare(:,6),
smooth_ATS_bin_centers);
    interpolate_MS_income(:, 4) = interp1(marketShare(:,7), marketShare(:,8),
smooth_ATS_bin_centers);
```

```
    interpolate_MS_income(:, 5) = interp1(marketShare(:,9),
marketShare(:,10), smooth_ATS_bin_centers);

    % interpolate_MS_income(:,1) -> Market Share for Income < 30K
    % interpolate_MS_income(:,2) -> Market Share for Income 30K to 60K
    % interpolate_MS_income(:,3) -> Market Share for Income 60K to 100K
    % interpolate_MS_income(:,4) -> Market Share for Income 100K to 150K
    % interpolate_MS_income(:,5) -> Market Share for Income > 50K

    figure; hold on; grid on
    plot(marketShare(:,1), marketShare(:,2), '-r', smooth_ATS_bin_centers,
interpolate_MS_income(:, 1), '.r');
    plot(marketShare(:,3), marketShare(:,4), 'k', smooth_ATS_bin_centers,
interpolate_MS_income(:, 2), '.k');
    plot(marketShare(:,5), marketShare(:,6), 'g', smooth_ATS_bin_centers,
interpolate_MS_income(:, 3), '.g');
    plot(marketShare(:,7), marketShare(:,8), 'm', smooth_ATS_bin_centers,
interpolate_MS_income(:, 4), '.m');
    plot(marketShare(:,9), marketShare(:,10), 'b', smooth_ATS_bin_centers,
interpolate_MS_income(:, 5), '.b');
    grid on; xlabel('Distance (sm)'); ylabel('Auto Market Share %');
    title('Auto Market Share versus Distance')

    axis([100 3000 0 100])

    interpolate_MS_income(:, 6:10) = 100 - interpolate_MS_income;

    interpolate_MS_income(:,11) = smooth_ATS_bin_centers;
    if i == 1
        save ([dirName, '\business\interpolate_MS_income_Business'],
'interpolate_MS_income')
        saveas (gcf, [dirName, '\business\ATS_MS_Business.tif'])
        saveas (gcf, [dirName, '\business\ATS_MS_Business.fig'])
    else
        save ([dirName, '\nonbusiness\interpolate_MS_income_nonBusiness'],
'interpolate_MS_income')
        saveas (gcf, [dirName, '\nonbusiness\ATS_MS_NonBusiness.tif'])
        saveas (gcf, [dirName, '\nonbusiness\ATS_MS_NonBusiness.fig'])
    end % if i == 1

    interpolate_MS_income = [];

end % for i = 1 : 2
------------------------------------------------------------------------------------------------------------------------
-------
```

## 7.7 Create SAS Calibration Input File Using Smoothed ATS Market Share Curves and Trip Distribution Table

-----------------------------------------------------------------------------------------------------------------------

```
function createSAS_InputFile_Aggregate()

clear all; clc; tic

%-----------------------------------------------
% Create and Input File to Calibrate Model
%-----------------------------------------------
dirName  = 'D:\Mode Choice\marketShareCurves_ATS\captive';
fileDir  = 'D:\TSAM_Project\Trip_Distribution\Output\Y2000';

%-----------------------------------------------
% Load Trip Distribution Data
%-----------------------------------------------
load ([fileDir, '\trips_county'])

trips_county_Business   = trips_county(:,:,1);
Weighted_trips_Business = round(trips_county_Business * 80000 ./
sum(sum(trips_county_Business)));

trips_county_NonBusiness   = trips_county(:,:,2);
Weighted_trips_NonBusiness = round(trips_county_NonBusiness * 80000 ./
sum(sum(trips_county_NonBusiness)));
clear fileDir trips_county trips_county_Business trips_county_NonBusiness

binDistance_start  = 100:50:3500;
binDistance_end    = 150:50:3550;
binDistance_center = (binDistance_start + binDistance_end)/2;

region_dummy = [4 4 4];

for tripType = 1 : 2

    disp(['Elasped Time: ', num2str(toc)])
    if tripType == 1;
        SAVENAME  = 'Business';
        routePercentage = [0.7159 0.7954 1];
        weightedTrips = Weighted_trips_Business; clear
Weighted_trips_Business
    else
        SAVENAME  = 'NonBusiness';
        routePercentage    = [0.6932 0.8296 1];
        weightedTrips = Weighted_trips_NonBusiness; clear
Weighted_trips_NonBusiness
    end % if tripType == 1

    dataFileForLogitModel = fopen([SAVENAME,
'_logitModel_SASFile_Aggregate.txt'], 'w');
    sumTrips  = sum(sum(weightedTrips));
    SAVE_DATA = zeros(sumTrips, 17);

    AverageTravelTime_Auto_Region = zeros(69,4);
    AverageTravelCost_Auto_Region = zeros(69,5,4);
    AverageTravelTime_Comm_Region = zeros(69,3,4);
    AverageTravelCost_Comm_Region = zeros(69,5,3,4);

    %-----------------------------------------------
    % Load Travel time and Cost Data
    %-----------------------------------------------
    for cnt_Region = 1 : 4
        if cnt_Region == 1
            regionName = 'MSA_MSA_';
        elseif cnt_Region == 2
            regionName = 'MSA_nonMSA_';
        elseif cnt_Region == 3
            regionName = 'nonMSA_MSA_';
        elseif cnt_Region == 4
            regionName = 'nonMSA_nonMSA_';
        end % if cnt_Region == 1

        load ([regionName, SAVENAME, '_AverageTravelTime_Auto'])
        load ([regionName, SAVENAME, '_AverageTravelCost_Auto'])
        load ([regionName, SAVENAME, '_AverageTravelTime_Comm'])
        load ([regionName, SAVENAME, '_AverageTravelCost_Comm'])

        AverageTravelTime_Auto_Region(:,cnt_Region)     =
AverageTravelTime_Auto;
        AverageTravelCost_Auto_Region(:,:,cnt_Region)   =
AverageTravelCost_Auto;
        AverageTravelTime_Comm_Region(:,:,cnt_Region)   =
AverageTravelTime_Comm;
        AverageTravelCost_Comm_Region(:,:,:,cnt_Region) =
AverageTravelCost_Comm;

        clear AverageTravelTime_Auto AverageTravelCost_Auto
AverageTravelTime_Comm
        clear AverageTravelCost_Comm regionName
    end % for cnt_Region = 1 : 4


    %-----------------------------------------------
    % Load Market Share Data
    %-----------------------------------------------
    if tripType == 1
        load ([dirName, '\business\interpolate_MS_income_Business'])
    else
        load ([dirName, '\nonbusiness\interpolate_MS_income_nonBusiness'])
    end % if tripType == 1
    % interpolate_MS_income(:,1) -> Market Share for Income < 30K
    % interpolate_MS_income(:,2) -> Market Share for Income 30K to 60K
    % interpolate_MS_income(:,3) -> Market Share for Income 60K to 100K
    % interpolate_MS_income(:,4) -> Market Share for Income 100K to 150K
    % interpolate_MS_income(:,5) -> Market Share for Income > 50K

    commAirMarketShare = interpolate_MS_income(:,6:10);
    if tripType == 2;
        commAirMarketShare(53:end,:) = [];
    end
    rows = size(commAirMarketShare,1);
    %---------------------------------
    % (1) Plot Market Share Curves
    %---------------------------------
    figure;
    subplot(2,2,1); plot(binDistance_center(1:rows), commAirMarketShare(:,1),
'.-r', ...
         binDistance_center(1:rows), commAirMarketShare(:,2), '.-b', ...
         binDistance_center(1:rows), commAirMarketShare(:,3), '.-k', ...
         binDistance_center(1:rows), commAirMarketShare(:,4), '.-g', ...
         binDistance_center(1:rows), commAirMarketShare(:,5), '.-m'); grid on

    %---------------------------------
    % (2) Straighten out Crossing Portion
    % of Market Share Curves and re-plot
    %---------------------------------
    income_slope = [];
    if tripType == 1

        startSlope_x = 2400+25;
        endSlope_x   = 3000;
        xDistance    = endSlope_x - startSlope_x;
        slopeIndex_x = find(binDistance_center == startSlope_x);
        startSlope_y = commAirMarketShare(slopeIndex_x,:);

        endSlope_y   = sort(commAirMarketShare(end,:));

        income_slope(:,1) = interp1([startSlope_x endSlope_x],
[startSlope_y(1) endSlope_y(1)], 2425:50:3000);
        income_slope(:,2) = interp1([startSlope_x endSlope_x],
[startSlope_y(2) endSlope_y(2)], 2425:50:3000);
        income_slope(:,3) = interp1([startSlope_x endSlope_x],
[startSlope_y(3) endSlope_y(3)], 2425:50:3000);
        income_slope(:,4) = interp1([startSlope_x endSlope_x],
[startSlope_y(4) endSlope_y(4)], 2425:50:3000);
        income_slope(:,5) = interp1([startSlope_x endSlope_x],
[startSlope_y(5) endSlope_y(5)], 2425:50:3000);

        commAirMarketShare(slopeIndex_x:end,:) = income_slope;
    else
        startSlope_x = 2525;
        endSlope_x   = 3000;

        slopeIndex_x = find(binDistance_center == startSlope_x);
        startSlope_y = commAirMarketShare(slopeIndex_x,:);
        endSlope_y   = [87 88 92.5 99 100];

        income_slope(:,1) = interp1([startSlope_x endSlope_x],
[startSlope_y(1) endSlope_y(1)], 2525:50:3000);
        income_slope(:,2) = interp1([startSlope_x endSlope_x],
[startSlope_y(2) endSlope_y(2)], 2525:50:3000);
        income_slope(:,3) = interp1([startSlope_x endSlope_x],
[startSlope_y(3) endSlope_y(3)], 2525:50:3000);
        income_slope(:,4) = interp1([startSlope_x endSlope_x],
[startSlope_y(4) endSlope_y(4)], 2525:50:3000);
        income_slope(:,5) = interp1([startSlope_x endSlope_x],
[startSlope_y(5) endSlope_y(5)], 2525:50:3000);
```

```
        commAirMarketShare(49:end,:) = [];
        commAirMarketShare = [commAirMarketShare; income_slope];
        rows = size(commAirMarketShare,1);
    end % if tripType == 1


    subplot(2,2,2); plot(binDistance_center(1:rows), commAirMarketShare(:,1),
'.-r', ...
         binDistance_center(1:rows), commAirMarketShare(:,2), '.-b', ...
         binDistance_center(1:rows), commAirMarketShare(:,3), '.-k', ...
         binDistance_center(1:rows), commAirMarketShare(:,4), '.-g', ...
         binDistance_center(1:rows), commAirMarketShare(:,5), '.-m'); grid on

    %---------------------------------
    % (3) Use Trip Distribution to
    % Create new Set of Curves
    %---------------------------------
    commAirTrips_Estimate = round(commAirMarketShare/100 .* weightedTrips);
    autoTrips_Estimate    = weightedTrips - commAirTrips_Estimate;
    newCommAirMarketShare = commAirTrips_Estimate * 100 ./ weightedTrips;

    subplot(2,2,3); plot(binDistance_center(1:rows),
newCommAirMarketShare(:,1), '.-r', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,2), '.-b', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,3), '.-k', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,4), '.-g', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,5), '.-m'); grid
on

    %---------------------------------
    % (3) For Business Plots, Correct
    % Overlapping portion of high income curve
    %---------------------------------
    for i = 1 : size(newCommAirMarketShare,1)
        if (newCommAirMarketShare(i,4) > newCommAirMarketShare(i,5))
            keepValue_5 = newCommAirMarketShare(i,5);
            newCommAirMarketShare(i,5) = newCommAirMarketShare(i,4);
            newCommAirMarketShare(i,4) = keepValue_5;
        end
    end
    subplot(2,2,4); plot(binDistance_center(1:rows),
newCommAirMarketShare(:,1), '.-r', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,2), '.-b', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,3), '.-k', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,4), '.-g', ...
         binDistance_center(1:rows), newCommAirMarketShare(:,5), '.-m'); grid
on

    %---------------------------------------------
    % Create Input File
    %---------------------------------------------
    counterIndex = 0;
    for cntTrips = 1 : 49 % rows

        numberOfTrips = weightedTrips(cntTrips,:);
        tripDist_1way = binDistance_center(cntTrips);
        if tripDist_1way > 200; shortDummy = 0;
        else;                   shortDummy = 1;
        end % if cntTrips > 5; shortDummy = 0;

        for cntIncome = 1 : 5

            if tripType == 1;
                if cntIncome == 1
                    region_dummy = [1 0 0 0];
                elseif cntIncome == 2
                    region_dummy = [0 1 0 0];
                elseif cntIncome == 3
                    region_dummy = [0 0 1 0];
                else
                    region_dummy = [0 0 0 1];
                end % if income == 4 || income == 5
            else
                if cntIncome == 1
                    region_dummy = [1 0 0 0];
                elseif cntIncome == 2
                    region_dummy = [0 1 0 0];
                elseif cntIncome == 3
                    region_dummy = [0 0 1 0];
                else
                    region_dummy = [0 0 0 1];
                end % if income == 4 || income == 5

%                 if cntIncome == 1
%                     region_dummy = [1 0 0 0];
%                 elseif cntIncome == 2
%                     region_dummy = [0 1 0 0];
%                 else
%                     region_dummy = [0 0 0 1];
%                 end % if income == 4 || income == 5
            end % if tripType == 1

            numberOfTrips_Income  = numberOfTrips(cntIncome);
            marketShareValue_Comm =
newCommAirMarketShare(cntTrips,cntIncome);
            numberOfTrips_Comm    = marketShareValue_Comm *
numberOfTrips_Income / 100;
            numberOfTrips_Auto    = numberOfTrips_Income -
numberOfTrips_Comm;


            auto_TT_value =
mean(nonzeros(AverageTravelTime_Auto_Region(cntTrips,:)));
            auto_TC_value =
mean(nonzeros(AverageTravelCost_Auto_Region(cntTrips,cntIncome,:)));

            comm_TT_value(1) =
mean(nonzeros(AverageTravelTime_Comm_Region(cntTrips,1,:)));
            comm_TT_value(2) =
mean(nonzeros(AverageTravelTime_Comm_Region(cntTrips,2,:)));
            comm_TT_value(3) =
mean(nonzeros(AverageTravelTime_Comm_Region(cntTrips,3,:)));
```

```
            comm_TC_value(1) =
mean(nonzeros(AverageTravelCost_Comm_Region(cntTrips,cntIncome,1,:)));
            comm_TC_value(2) =
mean(nonzeros(AverageTravelCost_Comm_Region(cntTrips,cntIncome,2,:)));
            comm_TC_value(3) =
mean(nonzeros(AverageTravelCost_Comm_Region(cntTrips,cntIncome,3,:)));

            for cntLoop = 1 : numberOfTrips_Income

                counterIndex = counterIndex + 1;
                if mod(counterIndex, 10000) == 0;
                    disp([num2str(counterIndex), ' / ' num2str(sumTrips), ...
                        ' Trips, TripPurpose: = ', num2str(tripType), '
Time:= ', num2str(toc/60), ' Minutes']);
                end % if mod(totalRecords, 1000) == 0

                numberOfTrips_Auto = numberOfTrips_Auto - 1;
                if numberOfTrips_Auto > 0
                    mode = 1;
                else
                    mode         = 2;
                    rand_value_1 = rand(1,1);
                    if rand_value_1 <= routePercentage(1,1)
                    elseif (rand_value_1 > routePercentage(1,1) &
rand_value_1 <= routePercentage(1,2));
                        mode = mode + 1;
                    elseif rand_value_1 > routePercentage(1,2)
                        mode = mode + 2;
                    end % if rand_value_1 <= routePercentage(1,1)
                end % if numberOfTrips_Auto > 0

                SAVE_DATA(counterIndex, :) = [counterIndex mode  cntIncome
auto_TT_value auto_TC_value ...
                    comm_TT_value comm_TC_value tripDist_1way shortDummy
region_dummy];

                %-------------------------
                % Write Data to Text File
                %-------------------------
                for j = 1 : 4
                    costIncome = zeros(1,5);
                    costIncome(cntIncome) = 1;
                    if j == 1
                        auto_TC_value_1 = costIncome * auto_TC_value;
                        fprintf(dataFileForLogitModel, '%d %d %d %6.2f %6.2f
%6.2f %6.2f %6.2f %6.2f %d %d %d %d %d %d', ...
                            counterIndex, tripDist_1way, j, auto_TT_value,
auto_TC_value_1, shortDummy, region_dummy);

%                         auto_TC_value_1 = costIncome * auto_TC_value;
%                         fprintf(dataFileForLogitModel, '%d %d %d %6.2f
%6.2f %6.2f %6.2f %6.2f %6.2f %d %d %d %d %d %d', ...
%                             counterIndex, tripDist_1way, j, auto_TT_value,
auto_TC_value_1, 0, 0, 0, 0, 0);
                    elseif j == 2
                        comm_TC_value_1 = costIncome * comm_TC_value(1);
                        fprintf(dataFileForLogitModel, '%d %d %d %6.2f %6.2f
%6.2f %6.2f %6.2f %6.2f %d %d %d %d %d %d', ...
                            counterIndex, tripDist_1way, j, comm_TT_value(1),
comm_TC_value_1, shortDummy, region_dummy);
                    elseif j == 3
                        comm_TC_value_2 = costIncome * comm_TC_value(2);
                        fprintf(dataFileForLogitModel, '%d %d %d %6.2f %6.2f
%6.2f %6.2f %6.2f %6.2f %d %d %d %d %d %d', ...
                            counterIndex, tripDist_1way, j, comm_TT_value(2),
comm_TC_value_2, shortDummy, region_dummy);
                    elseif j == 4
                        comm_TC_value_3 = costIncome * comm_TC_value(3);
                        fprintf(dataFileForLogitModel, '%d %d %d %6.2f %6.2f
%6.2f %6.2f %6.2f %6.2f %d %d %d %d %d %d', ...
                            counterIndex, tripDist_1way, j, comm_TT_value(3),
comm_TC_value_3, shortDummy, region_dummy);
                    end % if j == 1

                    if j == mode
                        fprintf(dataFileForLogitModel, ' %d\n', 1);
                    else
                        fprintf(dataFileForLogitModel, ' %d\n', 0);
                    end % if mode == 1
                end % for j = 1 : 4

            end % for cntLoop = 1 : numberOfTrips_Income
        end % for cntIncome = 1 : 5

    end % for cntTrips = 1 : 69

    save ([SAVENAME, '_SAVE_DATA'], 'SAVE_DATA')
    %save ([SAVENAME, '_SAVE_DATA_2'], 'SAVE_DATA')
    fclose('all')

    SAVE_DATA = SAVE_DATA(:,[2 3 12]);

    autoIndex      = find(SAVE_DATA(:,1)==1);
    SAVE_DATA_auto = SAVE_DATA(autoIndex,:);

    commAirIndex   = find(SAVE_DATA(:,1)>1);
    SAVE_DATA_comm = SAVE_DATA(commAirIndex,:);

    for k = 1 : 5
        autoIncIndex      = find(SAVE_DATA_auto(:,2) == k);
        SAVE_DATA_autoInc = SAVE_DATA_auto(autoIncIndex,:);

        commAirIncIndex   = find(SAVE_DATA_comm(:,2) == k);
        SAVE_DATA_commInc = SAVE_DATA_comm(commAirIncIndex,:);

        for m = 1 : rows
            findAutoTrips   = find(SAVE_DATA_autoInc(:,3) >
binDistance_start(m) & SAVE_DATA_autoInc(:,3) <= binDistance_end(m));
            TripInfo(m,k,1) = size(findAutoTrips,1);

            findCommTrips   = find(SAVE_DATA_commInc(:,3) >
binDistance_start(m) & SAVE_DATA_commInc(:,3) <= binDistance_end(m));
            TripInfo(m,k,2) = size(findCommTrips,1);
        end % for m = 1 : rows
    end % for k = 1 : 5

    marketShareTest = TripInfo(:,:,2) * 100 ./ sum(TripInfo,3);
    figure;
    plot(binDistance_center(1:rows), marketShareTest(:,1), '.-r', ...
         binDistance_center(1:rows), marketShareTest(:,2), '.-b', ...
         binDistance_center(1:rows), marketShareTest(:,3), '.-k', ...
         binDistance_center(1:rows), marketShareTest(:,4), '.-g', ...
         binDistance_center(1:rows), marketShareTest(:,5), '.-m');
         grid on

end % for tripType = 1 : 2

return
```

---------------------------------------------------------------------------------------------------------------------

## 7.8 Calibrate data with SAS and Extract Parameters

### *7.8.1 SAS Business Nested Logit Calibration Script*

```
-------------------------------------------------------------------------------------------------------------------
-------
/* Load Data File */
DATA modeChoice;
INFILE 'D:\Mode Choice\Aggregate Travel Times and
Cost\Business_logitModel_SASFile_Aggregate.txt';
INPUT pid tripDist mode ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5 shortTripDummy regionDummy_1 regionDummy_2
regionDummy_3 regionDummy decision @@;
datalines;
run;

/*=============================================================================
==========================*/
/* Initialize SAS parameter output file */
proc printto print='D:\Mode Choice\Aggregate Travel Times and
Cost\NL_Business_Par_TT_TC_SHT_MODE_DUM.txt' new;
run;
/* Run PROC MDC */
proc mdc data=modeChoice type=nlogit;
      model decision = ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5/ samescale maxiter = 300
            choice = (mode 1 2 3 4);
      id pid;
      utility u(1, ) = ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5;
      nest level(1) = (1 @ 1, 2 3 4 @ 2),
           level(2) = (1 2 @ 1);
      output out = probdata pred=p;
run;
/* Close SAS parameter output file */
proc printto;
run;
/* Save output data into a text file */
PROC EXPORT DATA= WORK.Probdata
       OUTFILE= "D:\Mode Choice\Aggregate Travel Times and
Cost\NL_Business_Output_TT_TC_SHT_MODE_DUM.txt"
     DBMS=TAB REPLACE;
RUN;
/*END OF FILE*/
/*=============================================================================
===========================================================================*/

-------------------------------------------------------------------------------------------------------------------
```

### 7.8.2 SAS Business Mixed Logit Calibration Script

```
--------------------------------------------------------------------------------------------------------------------
-------
/* Load Data File */
DATA modeChoice;
INFILE 'D:\Mode Choice\Aggregate Travel Times and
Cost\Business_logitModel_SASFile_Aggregate.txt';
INPUT pid tripDist mode ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5 shortTripDummy regionDummy_1 regionDummy_2
regionDummy_3 regionDummy decision @@;
datalines;
run;
/*=============================================================================
==========================*/
/* Initialize SAS parameter output file */
proc printto print='D:\Mode Choice\Aggregate Travel Times and
Cost\ML_Business_Par_TT_TC_SHT_MODE_DUM.txt' new;
run;
/* Run PROC MDC */
proc mdc data=modeChoice type=mixedlogit;
        model decision=ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5/ nchoice=4
                        randnum=pseudo
                mixed=(normalparm=tcost_1 tcost_2 tcost_3 tcost_4 tcost_5);
        id pid;
    output out = probdata pred=p;
run;
/* Close SAS parameter output file */
proc printto;
run;
/* Save output data into a text file */
PROC EXPORT DATA= WORK.Probdata
       OUTFILE= "D:\Mode Choice\Aggregate Travel Times and
Cost\ML_Business_Output_TT_TC_SHT_MODE_DUM.txt"
     DBMS=TAB REPLACE;
RUN;
/*END OF FILE*/
/*=============================================================================
=============================================================================*/

--------------------------------------------------------------------------------------------------------------------
```

### 7.8.3 SAS Non-Business Nested Calibration Script

```
-------------------------------------------------------------------------------------------------------------------
-------
/* Load Data File */
DATA modeChoice;
INFILE 'D:\Mode Choice\Aggregate Travel Times and
Cost\NonBusiness_logitModel_SASFile_Aggregate.txt';
INPUT pid tripDist mode ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5 shortTripDummy regionDummy_1 regionDummy_2
regionDummy_3 regionDummy decision @@;
datalines;
run;

/*============================================================================
==========================*/
/* Initialize SAS parameter output file */
proc printto print='D:\Mode Choice\Aggregate Travel Times and
Cost\NL_NonBusiness_Par_TT_TC_SHT_MODE_DUM.txt' new;
run;
/* Run PROC MDC */
proc mdc data=modeChoice type=nlogit;
      model decision = ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5/ samescale maxiter = 300
            choice = (mode 1 2 3 4);
      id pid;
      utility u(1, ) = ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1
tcost_2 tcost_3 tcost_4 tcost_5;
      nest level(1) = (1 @ 1, 2 3 4 @ 2),
           level(2) = (1 2 @ 1);
      output out = probdata pred=p;
run;

/* Close SAS parameter output file */
proc printto;
run;
/* Save output data into a text file */
PROC EXPORT DATA= WORK.Probdata
       OUTFILE= "D:\Mode Choice\Aggregate Travel Times and
Cost\NL_NonBusiness_Output_TT_TC_SHT_MODE_DUM.txt"
     DBMS=TAB REPLACE;
RUN;
/*END OF FILE*/
/*============================================================================
==========================================================================*/
-------------------------------------------------------------------------------------------------------------------
```

### 7.8.4 *SAS Non-Business Mixed Logit Calibration Script*

```
---------------------------------------------------------------------------------------------------------------
-------
/* Load Data File */
DATA modeChoice;
INFILE 'D:\Mode Choice\Aggregate Travel Times and 
Cost\NonBusiness_logitModel_SASFile_Aggregate.txt';
INPUT pid tripDist mode ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1 
tcost_2 tcost_3 tcost_4 tcost_5 shortTripDummy regionDummy_1 regionDummy_2 
regionDummy_3 regionDummy decision @@;
datalines;
run;
/*=============================================================================
==========================*/
/* Initialize SAS parameter output file */
proc printto print='D:\Mode Choice\Aggregate Travel Times and 
Cost\ML_NonBusiness_Par_TT_TC_SHT_MODE_DUM.txt' new;
run;
/* Run PROC MDC */
proc mdc data=modeChoice type=mixedlogit;
        model decision=ttime_1 ttime_2 ttime_3 ttime_4 ttime_5 tcost_1 
tcost_2 tcost_3 tcost_4 tcost_5/ nchoice=4
                        randnum=pseudo
                mixed=(normalparm=tcost_1 tcost_2 tcost_3 tcost_4 tcost_5);
        id pid;
    output out = probdata pred=p;
run;
/* Close SAS parameter output file */
proc printto;
run;
/* Save output data into a text file */
PROC EXPORT DATA= WORK.Probdata
       OUTFILE= "D:\Mode Choice\Aggregate Travel Times and 
Cost\ML_NonBusiness_Output_TT_TC_SHT_MODE_DUM.txt"
     DBMS=TAB REPLACE;
RUN;
/*END OF FILE*/
/*=============================================================================
=============================================================================*/
---------------------------------------------------------------------------------------------------------------
```

### *7.8.5 Read Calibrated Coefficientss into Matlab*

--------------------------------------------------------------------------------------------------------------------------------
-------

```
function read_SAS_parameters_main()

%------------------------------------------
% Call m-file to extract SAS coefficients
% from output text file
%------------------------------------------
clear all; clc;

dir_name = 'D:\Mode Choice\Aggregate Travel Times and Cost';

for cnt_tripPurpose = 1 : 2
    if cnt_tripPurpose == 1
        TP_NAME = 'Business';
    else
        TP_NAME = 'NonBusiness';
    end % if cnt_tripPurpose == 1

    for cnt_dummyType = 1 : 1
        if cnt_dummyType == 1
            DUM_NAME = 'MODE_DUM';
        else
            DUM_NAME = 'ODNRY_DUM';
        end % if cnt_dummyType == 1

        for cnt_modelType = 1 : 2

            ParameterEstimates = zeros(20,2,8);
            if cnt_modelType == 1
                MODEL_NAME = 'NL_';
                SAVE_NAME  = 'ParameterEstimate_NL_';
                setNumberOfCoeff = [7 8 11 10 9 11 11 12];
                stepsize   = 0;
            else
                MODEL_NAME = 'ML_';
                SAVE_NAME  = 'ParameterEstimate_ML_';
                setNumberOfCoeff = [7 8 15 10 9 11 11 12];
                stepsize   = 0;
            end %if cnt_modelType == 1

            for cnt_variableTypes  = 3 % 1:8
                if cnt_variableTypes == 1
                    VAR_NAME    = '_TT_TC_';
                elseif cnt_variableTypes == 2
                    VAR_NAME = '_TT_TC_INC_';
                elseif cnt_variableTypes == 3
                    VAR_NAME = '_TT_TC_SHT_';
                elseif cnt_variableTypes == 4
                    VAR_NAME = '_TT_TC_REG_';
                elseif cnt_variableTypes == 5
```

```
                    VAR_NAME = '_TT_TC_INC_SHT_';
                elseif cnt_variableTypes == 6
                    VAR_NAME = '_TT_TC_INC_REG_';
                elseif cnt_variableTypes == 7
                    VAR_NAME = '_TT_TC_SHT_REG_';
                elseif cnt_variableTypes == 8
                    VAR_NAME = '_TT_TC_INC_SHT_REG_';
                end % if cnt_variableTypes == 1

                fileName    = [MODEL_NAME TP_NAME, '_Par', VAR_NAME
DUM_NAME];
                saveDir     = [SAVE_NAME TP_NAME , VAR_NAME DUM_NAME
'_MatFile'];
                numbOfCoeff = stepsize + setNumberOfCoeff(cnt_variableTypes);
                read_SAS_parameters(fileName, saveDir, numbOfCoeff);
            end % for cnt_variableTypes  = 1 : 8

        end % for cnt_modelType = 1 : 2

    end % for cnt_dummyType = 1 : 2

end % for cnt_tripPurpose = 1 : 2
```
-----------------------------------------------------------------------------------------------------------------------

7.8.5.1 Read Coefficients from SAS text File into Matlab

------------------------------------------------------------------------------------------------------------------------

```
function [parameterEstimates] = read_SAS_parameters(caseFileName, saveDir,
numbOfCoeff)

%-----------------------------------------------------------------------
% This function opens SAS output text files and extarcts the
% calibrated model coefficients and r-squared values
%
% By: Senanu
% Date: October, 2004
%-----------------------------------------------------------------------


fid = fopen([caseFileName '.txt'], 'r');

format long;
% start search
while 1
    read_line = fgets(fid);
    fileLine  = sscanf(read_line, '%s');

    % R-squared values
    logValue = strncmp(fileLine, 'LogLikelihood', size('LogLikelihood', 2));
    if logValue == 1
        parameterEstimates(1,1) = sscanf(read_line, '%*s %*s %f');
    end % if logValue == 1

    % R-squared values
    textValue = strncmp(fileLine, 'AdjustedEstrella',
size('AdjustedEstrella', 2));
    if textValue == 1
        parameterEstimates(2,1) = sscanf(read_line, '%*s %*s %f');
        for i = 1 : 2
            read_line = fgets(fid);
            if i ==1
                parameterEstimates(3,1) = sscanf(read_line, '%*s %*s %f');
            else
                parameterEstimates(4,1) = sscanf(read_line, '%*s %f');
            end
        end % for i = 1 :2
    end % if textValue == 1

    % first coefficient values
    numericValue  = strncmp(fileLine, 'ttime', size('ttime', 2));
    if numericValue == 1
        for i = 1 : numbOfCoeff
            coeffValue              = sscanf(read_line', '%*s  %f %f %f %f %c
%f')';
            parameterEstimates(4+i,1) = coeffValue(2); % read coefficient
value
            maxsize = size(coeffValue,2);
            if maxsize < 6
            else
                parameterEstimates(4+i,2) = coeffValue(6); % read coefficient
value
            end % if isempty(coeffValue(6))
            read_line = fgets(fid);
        end % for i == 1 : numbOfCoeff
        break
    end % if numericValue == 1

end % while 1

fclose(fid);

save (saveDir, 'parameterEstimates')
```

-----------------------------------------------------------------------------------------------------------------------------

### 7.8.6 *Read Output SAS data into Matlab and Plot Market Share Curve*

------------------------------------------------------------------------------------------------------------------
------------

```
function read_SAS_output_February()

%----------------------------------------
% Call m-file to extract SAS coefficients
% from output text file
%----------------------------------------
clear all; clc;

dir_name = 'D:\Mode
Choice\Calibration_February_07\Calibration_Input_Files_SAS\captive\';

VectorSize = 15;
records    = [73162 266577];

for cnt_tripPurpose = 1 : 2
    if cnt_tripPurpose == 1
        TP_NAME = 'business';
    else
        TP_NAME = 'nonBusiness';
    end % if cnt_tripPurpose == 1

    file_dir = [dir_name TP_NAME '\'];
    for cnt_dummyType = 1 : 1
        if cnt_dummyType == 1
            DUM_NAME = 'MODE_DUM';
        else
            DUM_NAME = 'ODNRY_DUM';
        end % if cnt_dummyType == 1

        for cnt_modelType = 1 : 1
            if cnt_modelType == 1
                MODEL_NAME = 'NL_Output_';
                stepsize   = 0;
            else
                MODEL_NAME = 'ML_Output_';
                stepsize   = 0;
            end % if cnt_modelType == 1

            for cnt_variableTypes  = 3
                if cnt_variableTypes == 1
                    VAR_NAME    = '_TT_TC_';
                elseif cnt_variableTypes == 2
                    VAR_NAME = '_TT_TC_INC_';
                elseif cnt_variableTypes == 3
                    VAR_NAME = '_TT_TC_SHT_';
                elseif cnt_variableTypes == 4
                    VAR_NAME = '_TT_TC_REG_';
                elseif cnt_variableTypes == 5
                    VAR_NAME = '_TT_TC_INC_SHT_';
                elseif cnt_variableTypes == 6
```

```
                    VAR_NAME = '_TT_TC_INC_REG_';
                elseif cnt_variableTypes == 7
                    VAR_NAME = '_TT_TC_SHT_REG_';
                elseif cnt_variableTypes == 8
                    VAR_NAME = '_TT_TC_INC_SHT_REG_';
                end % if cnt_variableTypes == 1

                fileName    = [file_dir MODEL_NAME TP_NAME VAR_NAME DUM_NAME
'.txt'];
                saveDir     = [file_dir MODEL_NAME TP_NAME VAR_NAME DUM_NAME
'_MatFile'];
                read_SAS_output(fileName, saveDir, VectorSize,
records(cnt_tripPurpose));
            end % for cnt_variableTypes  = 1 : 8

        end % for cnt_modelType = 1 : 2
    end % for cnt_dummyType = 1 : 2
end % for cnt_tripPurpose = 1 : 2
%
% return

bintIndex_start  = 100 : 50 :  2950;
bintIndex_end    = 150 : 50 :  3000;
bintIndex_center = 0.5 * (bintIndex_start + bintIndex_end);

for cnt_dummyType = 1 : 1
    if cnt_dummyType == 1
        DUM_NAME = 'MODE_DUM';
    else
        DUM_NAME = 'ODNRY_DUM';
    end % if cnt_dummyType == 1

    for cnt_modelType = 1 : 1
        if cnt_modelType == 1
            MODEL_NAME = 'NL_Output_';
            stepsize   = 0;
        else
            MODEL_NAME = 'ML_Output_';
            stepsize   = 0;
        end % if cnt_modelType == 1

        for cnt_variableTypes  = 3
            if cnt_variableTypes == 1
                VAR_NAME    = '_TT_TC_';
            elseif cnt_variableTypes == 2
                VAR_NAME = '_TT_TC_INC_';
            elseif cnt_variableTypes == 3
                VAR_NAME = '_TT_TC_SHT_';
            elseif cnt_variableTypes == 4
                VAR_NAME = '_TT_TC_REG_';
            elseif cnt_variableTypes == 5
                VAR_NAME = '_TT_TC_INC_SHT_';
            elseif cnt_variableTypes == 6
                VAR_NAME = '_TT_TC_INC_REG_';
            elseif cnt_variableTypes == 7
                VAR_NAME = '_TT_TC_SHT_REG_';
            elseif cnt_variableTypes == 8
```

```
                VAR_NAME = '_TT_TC_INC_SHT_REG_';
            end % if cnt_variableTypes == 1

            sum_all_trips_Model = zeros(1,2,2);
            sum_all_trips_ATS   = zeros(1,2,2);
            for cnt_tripPurpose = 1 : 2
                if cnt_tripPurpose == 1
                    TP_NAME = 'business';
                else
                    TP_NAME = 'nonBusiness';
                end % if cnt_tripPurpose == 1

                file_dir = [dir_name TP_NAME '\'];
                saveDir = [file_dir MODEL_NAME TP_NAME VAR_NAME DUM_NAME
'_MatFile'];
                load (saveDir, 'MS_dataFile')
                % MS_dataFile(:,1) -> Trips Index
                % MS_dataFile(:,2) -> ATS trip route distance
                % MS_dataFile(:,3) -> Household Income
                % MS_dataFile(:,4) -> Region Index
                % MS_dataFile(:,5) -> Mode selected in ATS
                % MS_dataFile(:,6) -> Auto Probability from SAS

                for i = 1 : size(bintIndex_start,2)
                    startBin = bintIndex_start(i);
                    endBin   = bintIndex_end(i);
                    selectIndex = find(MS_dataFile(:,2) >= startBin &
MS_dataFile(:,2) < endBin);
                    selectTrips = MS_dataFile(selectIndex,:);

                    autoSelect_ATS_index = find(selectTrips(:,5) == 1);
                    MS_ATS(i,1) = size(autoSelect_ATS_index,1) * 100 /
size(selectTrips,1);
                    MS_ATS(i,2) = 100 - MS_ATS(i,1);

                    COUNT_ATS(i,1) = size(autoSelect_ATS_index,1);
                    COUNT_ATS(i,2) = size(selectTrips,1) -
size(autoSelect_ATS_index,1);

                    MS_Model(i,1) = mean(nonzeros(selectTrips(:,6))) * 100;
                    MS_Model(i,2) = 100 - MS_Model(i,1);

                    for k = 1 : 5
                        incomeIndex = find(selectTrips(:,3) == k);
                        selectTripsIncome = selectTrips(incomeIndex,:);

                        autoSelect_ATS_index_income =
find(selectTripsIncome(:,5) == 1);
                        MS_ATS_income(i,k,1) =
size(autoSelect_ATS_index_income,1) * 100 / size(selectTripsIncome,1);
                        MS_ATS_income(i,k,2) = 100 - MS_ATS_income(i,k,1);

                        COUNT_ATS_INCOME(i,k,1) =
size(autoSelect_ATS_index_income,1);
                        COUNT_ATS_INCOME(i,k,2) = size(selectTripsIncome,1) -
size(autoSelect_ATS_index_income,1);
```

```
                        MS_Model_income(i,k,1) =
mean(nonzeros(selectTripsIncome(:,6))) * 100;
                        MS_Model_income(i,k,2) = 100 -
MS_Model_income(i,k,1);
                    end % for k = 1 : 5
                end % for i = 1 : size(bintIndex_start,2)

                %                  figure;
                %                  plot(bintIndex_center, MS_ATS(:,1), '.-r',
bintIndex_center, MS_ATS(:,2), '.-b');
                %                  text(bintIndex_center, MS_ATS(:,1),
num2str(COUNT_ATS(:,1)))
                %                  text(bintIndex_center, MS_ATS(:,2),
num2str(COUNT_ATS(:,2)))
                %                  xlabel('Distance (sm)'); ylabel('Market
Share (%)')
                %                  title('ATS MARKET SHARE PLOT');
legend('AUTO','COMM AIR'); grid on

                figure;
                plot(bintIndex_center, MS_ATS(:,2), '-r', bintIndex_center,
MS_Model(:,2), '.-k');
                xlabel('Distance (sm)'); ylabel('Market Share (%)');
                title('ATS VERSUS MODEL MARKET SHARE PLOT');
legend('ATS','Aggregate Model', 'location', 'southeast'); grid on

                %                  figure;
                %                  subplot(2,1,1); plot(bintIndex_center,
MS_ATS_income(:,1,2), '.-r', bintIndex_center, MS_ATS_income(:,2,2), '*-b',
...
                %                      bintIndex_center, MS_ATS_income(:,3,2),
'-g', bintIndex_center, MS_ATS_income(:,4,2), '-*k', ...
                %                      bintIndex_center, MS_ATS_income(:,5,2),
'-.m');
                %                  title('UNSMOOTHED ATS MARKET SHARE
CURVES'); xlabel('Distance in Statute Miles'); ylabel('Market Share %')
                %                  legend('Less than $30K','$30 to 60K','$60
to 100K','$100 to 150K','Greater than 150K'); grid on

                figure;
                subplot(3,2,1); plot(bintIndex_center, MS_ATS_income(:,1,2),
'-r', bintIndex_center, MS_Model_income(:,1,2), '.-k');
                title('Market Share for Income < 30K'); xlabel('Distance
(statute miles)'); ylabel('Market Share %'); grid on
                subplot(3,2,2); plot(bintIndex_center, MS_ATS_income(:,2,2),
'-r', bintIndex_center, MS_Model_income(:,2,2), '.-k');
                title('Market Share for Income 30 to 60K'); xlabel('Distance
(statute miles)'); ylabel('Market Share %'); grid on
                subplot(3,2,3); plot(bintIndex_center, MS_ATS_income(:,3,2),
'-r', bintIndex_center, MS_Model_income(:,3,2), '.-k');
                title('Market Share for Income 60 to 100K'); xlabel('Distance
(statute miles)'); ylabel('Market Share %'); grid on
                subplot(3,2,4); plot(bintIndex_center, MS_ATS_income(:,4,2),
'-r', bintIndex_center, MS_Model_income(:,4,2), '.-k');
                title('Market Share for Income 100 to 150K');
xlabel('Distance (statute miles)'); ylabel('Market Share %');  grid on
```

```
                subplot(3,2,5); plot(bintIndex_center, MS_ATS_income(:,5,2),
'-r', bintIndex_center, MS_Model_income(:,5,2), '.-k');
                title('Market Share for Income > 150K'); xlabel('Distance
(statute miles)'); ylabel('Market Share %');
                legend('Unsmoothed ATS','Model Estimates', 'Location',
'southeast'); grid on
            end % for cnt_tripPurpose = 1 : 2

        end % for cnt_variableTypes  = 3; %1 : 8
    end % for cnt_modelType = 1 : 1
end % for cnt_dummyType = 1 : 2

return
```

------------------------------------------------------------------------------------------------------------------------
------------

7.8.6.1 Read SAS output text file into Matlab

------------------------------------------------------------------------------------------------------------------------------------

```
function read_SAS_output(caseFileName, saveDir, VectorSize, fileSize)

tic;
rowCounter    = 0;
MS_dataFile   = zeros(fileSize,6);
recordCounter = 0;

fid = fopen(caseFileName, 'r');
read_line   = fgets(fid);

% start search
while ~feof(fid)
    tripCounter   = 1;
    fileRecords   = zeros(VectorSize,4);
    while tripCounter < 5
        recordCounter = recordCounter + 1;
        read_line = fgets(fid);
        fileRecords(:,tripCounter) = sscanf(read_line, '%f %d %d %d %f %f %f
%f %f %f %d %d %d %d %d');
        tripCounter = tripCounter + 1;
    end % if logValue == 1
    fileRecords = fileRecords';

    sumCol = sum(fileRecords(:,4));
    if sumCol == 10
        rowCounter  = rowCounter + 1;
        if mod(rowCounter, 10000) == 0
            disp(['Records: ', num2str(rowCounter), ' Time: ', num2str(toc),
' secs'])
        end % if mod(rowCounter, 5000) == 0
%         if rowCounter == 36926
%             if mod(rowCounter, 1) == 0
%                 disp(['Records: ', num2str(rowCounter), ' Time: ',
num2str(toc), ' secs'])
%             end % if mod(rowCounter, 5000) == 0
%         end
        tripIndex   = fileRecords(1,2);
        tripdist    = fileRecords(1,3);
        hhIncome    = find(fileRecords(2,6:10) > 0);
        regionIndex = find(fileRecords(2,12:14) > 0);
        if isempty(regionIndex)
            regionIndex = 4;
        end % if isempty(regionIndex)
        modeSelected = find(fileRecords(:,VectorSize));
        autoProb     = fileRecords(1,1);
        MS_dataFile(rowCounter,:) = [tripIndex tripdist hhIncome regionIndex
modeSelected autoProb];
    else
        fileRecords(1,2)
        disp('Error in reading file')
        pause
    end % if sumCol == 10

end % while ~(feof)

save (saveDir, 'MS_dataFile')
% MS_dataFile(:,1) -> Trips Index
% MS_dataFile(:,2) -> ATS trip route distance
% MS_dataFile(:,3) -> Household Income
% MS_dataFile(:,4) -> Region Index
% MS_dataFile(:,5) -> Mode selected in ATS
% MS_dataFile(:,6) -> Auto Probability from SAS
return
```

------------------------------------------------------------------------------------------------------------------------------------------

## 7.9 Compare Model and ATS

### *7.9.1 Compare Using Market Share Share Plots*

```
-----------------------------------------------------------------------------------------------------------------------
------------
function useATS_validateLogitCoefficients_captive_all()

%----------------------------------------------------------------------------
--------
% Extract American Travel Survey (ATS) data and creates input SAS dataset
% (used in calibrating Logit model)
%
% Calling: createGA_weigthedTimeAndCost.m
%          writeLogitDataToFile.m
%
% Programed by: Hojong Baik (2/ /03)
% Modified by:  Hojong Baik (5/28/03)
% Modified by:  Senanu Ashiabor (3/04)
% Modified by:  Senanu Ashiabor (5/30/06)
%
% Output: dataFileForLogitModel.txt
% The output is used in 'compareCoeffToATS_usingMShareCurves.m'
%----------------------------------------------------------------------------
--------

clear all; clc; tic;
global model_coefficients

load_Dir    = 'D:\Mode Choice\Calibration_February_07';

bintIndex_start  = 100 : 50 :  2950;
bintIndex_end    = 150 : 50 :  3000;
bintIndex_center = 0.5 * (bintIndex_start + bintIndex_end);

%---------------------------------------------
% % Loop for 'Business' and 'Non-Business' Trips
% %---------------------------------------------
for bussTripsSwitch = 2 : 2

    % bussTripsSwitch = 1 => trips made for business purpose, 2 => non-
business trips
    if bussTripsSwitch == 1
        % business trips are indexed 1, 2 & 3 in the ATS (see ATS
documentation)
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\Calibration_ValidationFiles\business'];
        load SAVE_DATA
        load ([load_Dir,
'\Calibration_Input_Files_SAS\captive\business\ParameterEstimate_NL_business_
TT_TC_SHT_MODE_DUM_MatFile']);
    elseif bussTripsSwitch == 2
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\Calibration_ValidationFiles\nonBusiness
'];
```

```
        load SAVE_DATA
        load ([load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\ParameterEstimate_NL_nonBus
iness_TT_TC_SHT_MODE_DUM_MatFile']);
    end % if travelerTypeSwitch == 1 & bussTripsSwitch == 1
    model_coefficients = parameterEstimates(5:12,1);

    findzeros = find(SAVE_DATA(:,1) == 0);
    SAVE_DATA(findzeros,:) = [];
    tripIndex           = 0;
    totSampleSize       = size(SAVE_DATA, 1);
    saveTripData_income = zeros(totSampleSize, 6);

    % SAVE_DATA(:,1)  -> Index
    % SAVE_DATA(:,2)  -> Mode selected
    % SAVE_DATA(:,3)  -> Household Income Index
    % SAVE_DATA(:,4)  -> Auto Travel Time 2-way
    % SAVE_DATA(:,5)  -> Auto Travel Cost 2-way
    % SAVE_DATA(:,6)  -> CA Travel Time (Route 1) 2-way
    % SAVE_DATA(:,7)  -> CA Travel Cost (Route 1) 2-way
    % SAVE_DATA(:,8) -> CA Travel Time (Route 2) 2-way
    % SAVE_DATA(:,9) -> CA Travel Cost (Route 2) 2-way
    % SAVE_DATA(:,10) -> CA Travel Time (Route 3) 2-way
    % SAVE_DATA(:,11) -> CA Travel Cost (Route 3) 2-way
    % SAVE_DATA(:,12) -> 1-way Route Distance
    % SAVE_DATA(:,13) -> Short trip dummy

    while(tripIndex < totSampleSize)
        tripIndex = 1 + tripIndex;
        if mod(tripIndex, 10000) == 0;
            disp(['Trips Processed: ', num2str(tripIndex), ' of ',
num2str(totSampleSize), ...
                ' -> TripPurpose: = ', num2str(bussTripsSwitch), ' Time:= ',
num2str(toc)]);
        end % if mod(totalRecords, 1000) == 0

        mode          = SAVE_DATA(tripIndex,2);
        incomeGrp     = SAVE_DATA(tripIndex,3);
        tTime_Auto    = SAVE_DATA(tripIndex,4);
        cost_Auto     = SAVE_DATA(tripIndex,5);
        tTime_comm    = SAVE_DATA(tripIndex,6:8)';
        cost_comm     = SAVE_DATA(tripIndex,9:11)';
        tripDist_1way = SAVE_DATA(tripIndex,12);
        shortDummy    = SAVE_DATA(tripIndex,13);

        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm, ...
            cost_Auto, cost_comm, incomeGrp, shortDummy);

        saveTripData_income(tripIndex,:)  = [tripIndex mode P_Auto_NL
P_CommAir_NL incomeGrp tripDist_1way];
        % saveTripData(:,1) => index value
        % saveTripData(:,2) => Mode assinged in ATS
        % saveTripData(:,3) => 1-way route distance
        % saveTripData(:,4) => Auto Probability in Logit model
        % saveTripData(:,5) => CA Probability in Logit model
        % saveTripData(:,6) => Income index
        % saveTripData(:,7) => Region index

    end %while(totalRecords < totSampleSize)

    save (strcat(OutputDir_case,
'\saveTripData_income'),'saveTripData_income')

end % for bussTripsSwitch = 1 : 2 % if bussTripsSwitch = 1, indicates trips
made for business purpose, 2

return

load_Dir    = 'D:\Mode Choice\Calibration_February_07';
for bussTripsSwitch = 2 : 2

    % bussTripsSwitch = 1 => trips made for business purpose, 2 => non-
business trips
    if bussTripsSwitch == 1
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\business'];
    elseif bussTripsSwitch == 2
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness'];
    end % if bussTripsSwitch == 1
    load (strcat(OutputDir_case, '\saveTripData_income'))
    saveTripData_current = saveTripData_income;
    clear saveTripData_income

    for i = 1 : size(bintIndex_start,2)
        startBin = bintIndex_start(i);
        endBin   = bintIndex_end(i);
        selectIndex = find(saveTripData_current(:,6) >= startBin &
saveTripData_current(:,6) < endBin);
        selectTrips = saveTripData_current(selectIndex,:);

        autoSelect_ATS_index = find(selectTrips(:,2) == 1);
        MS_ATS(i,1) = size(autoSelect_ATS_index,1) * 100 /
size(selectTrips,1);
        MS_ATS(i,2) = 100 - MS_ATS(i,1);

        COUNT_ATS(i,1) = size(autoSelect_ATS_index,1);
        COUNT_ATS(i,2) = size(selectTrips(:,2),1) -
size(autoSelect_ATS_index,1);

        MS_Model(i,1) = mean(nonzeros(selectTrips(:,3))) * 100;
        MS_Model(i,2) = mean(nonzeros(selectTrips(:,4))) * 100;

        for k = 1 : 5
            incomeIndex = find(selectTrips(:,5) == k);
            selectTripsIncome = selectTrips(incomeIndex,:);

            autoSelect_ATS_index_income = find(selectTripsIncome(:,2) == 1);
            MS_ATS_income(i,k,1) = size(autoSelect_ATS_index_income,1) * 100
/ size(selectTripsIncome,1); %#ok<AGROW>
            MS_ATS_income(i,k,2) = 100 - MS_ATS_income(i,k,1);

            COUNT_ATS_INCOME(i,k,1) = size(autoSelect_ATS_index_income,1);
            COUNT_ATS_INCOME(i,k,2) = size(selectTripsIncome(:,2),1) -
size(autoSelect_ATS_index_income,1);

            MS_Model_income(i,k,1) = mean(nonzeros(selectTripsIncome(:,3))) *
100;
            MS_Model_income(i,k,2) = mean(nonzeros(selectTripsIncome(:,4))) *
100;
        end % for k = 1 : 5
    end % for i = 1 : size(bintIndex_start,2)

    figure;
    plot(bintIndex_center, MS_ATS(:,1), '.-r', bintIndex_center, MS_ATS(:,2),
'.-b');
    text(bintIndex_center, MS_ATS(:,1), num2str(COUNT_ATS(:,1)))
    text(bintIndex_center, MS_ATS(:,2), num2str(COUNT_ATS(:,2)))
    xlabel('Distance (sm)'); ylabel('Market Share (%)')
    title('ATS MARKET SHARE PLOT'); legend('AUTO','COMM AIR'); grid on

    figure;
    plot(bintIndex_center, MS_ATS(:,2), '-r', bintIndex_center,
MS_Model(:,2), '.-k');
    xlabel('Distance (sm)'); ylabel('Market Share (%)')
    %     title('ATS VERSUS MODEL MARKET SHARE PLOT');
    legend('ATS Commercial Air','Aggregate Model','Location', 'southeast');
grid on

    %     figure;
    %     subplot(2,1,1); plot(bintIndex_center, MS_ATS_income(:,1,2), '.-r',
bintIndex_center, MS_ATS_income(:,2,2), '*-b', ...
    %         bintIndex_center, MS_ATS_income(:,3,2), '-g', bintIndex_center,
MS_ATS_income(:,4,2), '-*k', ...
    %         bintIndex_center, MS_ATS_income(:,5,2), '-.m');
    %     title('UNSMOOTHED ATS MARKET SHARE CURVES'); xlabel('Distance in
Statute Miles'); ylabel('Market Share %')
    %     legend('Less than $30K','$30 to 60K','$60 to 100K','$100 to
150K','Greater than 150K'); grid on

    figure;
    subplot(3,2,1); plot(bintIndex_center, MS_ATS_income(:,1,2), '-r',
bintIndex_center, MS_Model_income(:,1,2), '.-k');
    title('Market Share for Income < 30K'); xlabel('Distance (statute
miles)'); ylabel('Market Share %'); grid on
    subplot(3,2,2); plot(bintIndex_center, MS_ATS_income(:,2,2), '-r',
bintIndex_center, MS_Model_income(:,2,2), '.-k');
    title('Market Share for Income 30 to 60K'); xlabel('Distance (statute
miles)'); ylabel('Market Share %'); grid on
    subplot(3,2,3); plot(bintIndex_center, MS_ATS_income(:,3,2), '-r',
bintIndex_center, MS_Model_income(:,3,2), '.-k');
    title('Market Share for Income 60 to 100K'); xlabel('Distance (statute
miles)'); ylabel('Market Share %'); grid on
    subplot(3,2,4); plot(bintIndex_center, MS_ATS_income(:,4,2), '-r',
bintIndex_center, MS_Model_income(:,4,2), '.-k');
    title('Market Share for Income 100 to 150K'); xlabel('Distance (statute
miles)'); ylabel('Market Share %');  grid on
    subplot(3,2,5); plot(bintIndex_center, MS_ATS_income(:,5,2), '-r',
bintIndex_center, MS_Model_income(:,5,2), '.-k');
```

```
    title('Market Share for Income > 150K'); xlabel('Distance (statute
miles)'); ylabel('Market Share %');
    legend('Unsmoothed ATS','Model Estimates', 'Location', 'southeast'); grid
on

end % for bussTripsSwitch = 1 : 1

return
```

------------------------------------------------------------------------------------------------------------------------------------
------------

### 7.9.2 *Sensitivity Analysis*

------------------------------------------------------------------------------------------------------------------------------------

```
function useATS_validateLogitCoefficients_captive_sensitivity_Feb()

%----------------------------------------------------------------------------
--------
% Extract American Travel Survey (ATS) data and creates input SAS dataset
% (used in calibrating Logit model)
%
% Calling: createGA_weigthedTimeAndCost.m
%          writeLogitDataToFile.m
%
% Modified by:  Senanu Ashiabor (5/30/06)
%
% Output: dataFileForLogitModel.txt
% The output is used in 'compareCoeffToATS_usingMShareCurves.m'
%----------------------------------------------------------------------------
--------

clear all; clc; tic;
global model_coefficients

load_Dir = 'D:\Mode Choice\Calibration_February_07';

bintIndex_start  = 100 : 50 :  2950;
bintIndex_end    = 150 : 50 :  3000;
bintIndex_center = 0.5 * (bintIndex_start + bintIndex_end);

%-----------------------------------------------
% % Loop for 'Business' and 'Non-Business' Trips
% %-----------------------------------------------
for bussTripsSwitch = 1 : 2

    % bussTripsSwitch = 1 => trips made for business purpose, 2 => non-
business trips
    if bussTripsSwitch == 1
        % business trips are indexed 1, 2 & 3 in the ATS (see ATS
documentation)
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\business'];
        load ([OutputDir_case, '\SAVE_DATA'])
        load ('D:\Mode
Choice\Calibration_February_07\Calibration_Input_Files_SAS\captive\business\P
arameterEstimate_NL_business_TT_TC_SHT_MODE_DUM_MatFile');
    elseif bussTripsSwitch == 2
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness'];
        load ([OutputDir_case, '\SAVE_DATA'])
        load ('D:\Mode
Choice\Calibration_February_07\Calibration_Input_Files_SAS\captive\nonBusines
s\ParameterEstimate_NL_nonBusiness_TT_TC_SHT_MODE_DUM_MatFile');
    end % if travelerTypeSwitch == 1 & bussTripsSwitch == 1
    model_coefficients = parameterEstimates(5:12,1);

    findzeros = find(SAVE_DATA(:,1) == 0);
    SAVE_DATA(findzeros,:) = [];
    tripIndex = 0;
    totSampleSize = size(SAVE_DATA, 1);
    saveTripData_income = zeros(totSampleSize, 14);
    saveTripData_income_double_CA_TT = zeros(totSampleSize, 6);
    saveTripData_income_half_CA_TT   = zeros(totSampleSize, 6);
    saveTripData_income_double_CA_TC = zeros(totSampleSize, 6);
    saveTripData_income_half_CA_TC   = zeros(totSampleSize, 6);
    saveTripData_income_oneHour_TT = zeros(totSampleSize, 6);

    % SAVE_DATA(:,1)  -> Index
    % SAVE_DATA(:,2)  -> Mode selected
    % SAVE_DATA(:,3)  -> Household Income Index
    % SAVE_DATA(:,4)  -> Auto Travel Time 2-way
    % SAVE_DATA(:,5)  -> Auto Travel Cost 2-way
    % SAVE_DATA(:,6)  -> CA Travel Time (Route 1) 2-way
    % SAVE_DATA(:,7)  -> CA Travel Cost (Route 1) 2-way
    % SAVE_DATA(:,8) -> CA Travel Time (Route 2) 2-way
    % SAVE_DATA(:,9) -> CA Travel Cost (Route 2) 2-way
    % SAVE_DATA(:,10) -> CA Travel Time (Route 3) 2-way
    % SAVE_DATA(:,11) -> CA Travel Cost (Route 3) 2-way
    % SAVE_DATA(:,12) -> 1-way Route Distance
    % SAVE_DATA(:,13) -> Short trip dummy

    while(tripIndex < totSampleSize)
        tripIndex = 1 + tripIndex;
        if mod(tripIndex, 10000) == 0;
            disp(['Trips Processed: ', num2str(tripIndex), ' of ',
num2str(totSampleSize), ...
                ' -> TripPurpose: = ', num2str(bussTripsSwitch), ' Time:= ',
num2str(toc)]);
        end % if mod(totalRecords, 1000) == 0

        mode          = SAVE_DATA(tripIndex,2);
        incomeGrp     = SAVE_DATA(tripIndex,3);
        tTime_Auto    = SAVE_DATA(tripIndex,4);
        cost_Auto     = SAVE_DATA(tripIndex,5);
        tTime_comm    = SAVE_DATA(tripIndex,6:8)';
        cost_comm     = SAVE_DATA(tripIndex,9:11)';
        tripDist_1way = SAVE_DATA(tripIndex,12);
        shortDummy    = SAVE_DATA(tripIndex,13);

        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm, ...
            cost_Auto, cost_comm, incomeGrp, shortDummy);
        saveTripData_income(tripIndex,:)  = [tripIndex mode P_Auto_NL
P_CommAir_NL incomeGrp tripDist_1way tTime_Auto tTime_comm' ...
            cost_Auto cost_comm'];
        % saveTripData(:,1) => index value
        % saveTripData(:,2) => Mode assinged in ATS
        % saveTripData(:,3) => 1-way route distance
        % saveTripData(:,4) => Auto Probability in Logit model
        % saveTripData(:,5) => CA Probability in Logit model
        % saveTripData(:,6) => Income index
        % saveTripData(:,7) => Region index

        % double travel time
        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm*2, ...
            cost_Auto, cost_comm, incomeGrp, shortDummy);
        saveTripData_income_double_CA_TT(tripIndex,:)  = [tripIndex mode
P_Auto_NL P_CommAir_NL incomeGrp tripDist_1way];

        % half travel time
        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm*.5, ...
            cost_Auto, cost_comm, incomeGrp, shortDummy);
        saveTripData_income_half_CA_TT(tripIndex,:)  = [tripIndex mode
P_Auto_NL P_CommAir_NL incomeGrp tripDist_1way];


        % double travel cost
        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm, ...
            cost_Auto, cost_comm*2, incomeGrp, shortDummy);
        saveTripData_income_double_CA_TC(tripIndex,:)  = [tripIndex mode
P_Auto_NL P_CommAir_NL incomeGrp tripDist_1way];

        % half travel cost
        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm, ...
            cost_Auto, cost_comm*.5, incomeGrp, shortDummy);
        saveTripData_income_half_CA_TC(tripIndex,:)  = [tripIndex mode
P_Auto_NL P_CommAir_NL incomeGrp tripDist_1way];

        % half travel time
        [P_Auto_NL, P_CommAir_NL, P_CommAirRoute_NL, P_all_NL] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm+10, ...
            cost_Auto, cost_comm, incomeGrp, shortDummy);
        saveTripData_income_oneHour_TT(tripIndex,:)  = [tripIndex mode
P_Auto_NL P_CommAir_NL incomeGrp tripDist_1way];

    end %while(totalRecords < totSampleSize)

    save (strcat(OutputDir_case,
'\saveTripData_income'),'saveTripData_income')
    save (strcat(OutputDir_case,
'\saveTripData_income_double_CA_TT'),'saveTripData_income_double_CA_TT')
    save (strcat(OutputDir_case,
'\saveTripData_income_half_CA_TT'),'saveTripData_income_half_CA_TT')
    save (strcat(OutputDir_case,
'\saveTripData_income_double_CA_TC'),'saveTripData_income_double_CA_TC')
    save (strcat(OutputDir_case,
'\saveTripData_income_half_CA_TC'),'saveTripData_income_half_CA_TC')
    save (strcat(OutputDir_case,
'\saveTripData_income_oneHour_TT'),'saveTripData_income_oneHour_TT')

end % for bussTripsSwitch = 1 : 2 % if bussTripsSwitch = 1, indicates trips
made for business purpose, 2

% return
```

```
load_Dir     = 'D:\Mode Choice\Calibration_February_07';
for bussTripsSwitch = 1 : 2

    % bussTripsSwitch = 1 => trips made for business purpose, 2 => non-
business trips
    if bussTripsSwitch == 1
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\business'];
    elseif bussTripsSwitch == 2
        OutputDir_case = [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness'];
    end % if bussTripsSwitch == 1

    for cntFile = 1 : 6

        if cntFile == 1
            load (strcat(OutputDir_case, '\saveTripData_income'))
            avg_autoTravelTime = zeros(size(bintIndex_start,2),1);
            avg_commTravelTime = zeros(size(bintIndex_start,2),3);
            avg_autoTravelCost = zeros(size(bintIndex_start,2),1);
            avg_commTravelCost = zeros(size(bintIndex_start,2),3);
        elseif cntFile == 2
            load (strcat(OutputDir_case,
'\saveTripData_income_double_CA_TT'))
            saveTripData_income = saveTripData_income_double_CA_TT;
        elseif cntFile == 3
            load (strcat(OutputDir_case, '\saveTripData_income_half_CA_TT'))
            saveTripData_income = saveTripData_income_half_CA_TT;
        elseif cntFile == 4
            load (strcat(OutputDir_case,
'\saveTripData_income_double_CA_TC'))
            saveTripData_income = saveTripData_income_double_CA_TC;
        elseif cntFile == 5
            load (strcat(OutputDir_case, '\saveTripData_income_half_CA_TC'))
            saveTripData_income = saveTripData_income_half_CA_TC;
        elseif cntFile == 6
            load (strcat(OutputDir_case, '\saveTripData_income_oneHour_TT'))
            saveTripData_income = saveTripData_income_oneHour_TT;
        end % if cntFile == 1

        for i = 1 : size(bintIndex_start,2)
            startBin = bintIndex_start(i);
            endBin   = bintIndex_end(i);
            selectIndex = find(saveTripData_income(:,6) >= startBin &
saveTripData_income(:,6) < endBin);
            selectTrips = saveTripData_income(selectIndex,:);

            autoSelect_ATS_index = find(selectTrips(:,2) == 1);
            MS_ATS(i,1) = size(autoSelect_ATS_index,1) * 100 /
size(selectTrips,1);
            MS_ATS(i,2) = 100 - MS_ATS(i,1);

            COUNT_ATS(i,1) = size(autoSelect_ATS_index,1);
            COUNT_ATS(i,2) = size(selectTrips(:,2),1) -
size(autoSelect_ATS_index,1);
```

```
            MS_Model(i,1) = mean(nonzeros(selectTrips(:,3))) * 100;
            MS_Model(i,2) = mean(nonzeros(selectTrips(:,4))) * 100;

            if cntFile == 1
                autoTravelTime = selectTrips(:,7);
                commTravelTime = selectTrips(:,8:10);
                autoTravelCost = selectTrips(:,11);
                commTravelCost = selectTrips(:,12:14);

                avg_autoTravelTime(i,1) = mean(nonzeros(autoTravelTime));
                avg_commTravelTime(i,1) =
mean(nonzeros(commTravelTime(:,1)));
                avg_commTravelTime(i,2) =
mean(nonzeros(commTravelTime(:,2)));
                avg_commTravelTime(i,3) =
mean(nonzeros(commTravelTime(:,3)));
                avg_autoTravelCost(i,1) = mean(nonzeros(autoTravelCost));
                avg_commTravelCost(i,1) =
mean(nonzeros(commTravelCost(:,1)));
                avg_commTravelCost(i,2) =
mean(nonzeros(commTravelCost(:,2)));
                avg_commTravelCost(i,3) =
mean(nonzeros(commTravelCost(:,3)));
            end % if cntFile == 1

            for k = 1 : 5
                incomeIndex = find(selectTrips(:,5) == k);
                selectTripsIncome = selectTrips(incomeIndex,:);

                autoSelect_ATS_index_income = find(selectTripsIncome(:,2) ==
1);
                MS_ATS_income(i,k,1) = size(autoSelect_ATS_index_income,1) *
100 / size(selectTripsIncome,1); %#ok<AGROW>
                MS_ATS_income(i,k,2) = 100 - MS_ATS_income(i,k,1);

                COUNT_ATS_INCOME(i,k,1) =
size(autoSelect_ATS_index_income,1);
                COUNT_ATS_INCOME(i,k,2) = size(selectTripsIncome(:,2),1) -
size(autoSelect_ATS_index_income,1);

                MS_Model_income(i,k,1) =
mean(nonzeros(selectTripsIncome(:,3))) * 100;
                MS_Model_income(i,k,2) =
mean(nonzeros(selectTripsIncome(:,4))) * 100;
            end % for k = 1 : 5
        end % for i = 1 : size(bintIndex_start,2)

        if cntFile == 1
            MS_Model_base = MS_Model;
            MS_Model_base_income = MS_Model_income;
        elseif cntFile == 2
            MS_Model_double_CA_TT = MS_Model;
            MS_Model_double_CA_TT_income = MS_Model_income;
        elseif cntFile == 3
            MS_Model_half_CA_TT = MS_Model;
            MS_Model_half_CA_TT_income = MS_Model_income;
        elseif cntFile == 4
```

```
            MS_Model_double_CA_TC = MS_Model;
            MS_Model_double_CA_TC_income = MS_Model_income;
        elseif cntFile == 5
            MS_Model_half_CA_TC = MS_Model;
            MS_Model_half_CA_TC_income = MS_Model_income;
        elseif cntFile == 6
            MS_Model_delta_TT = MS_Model;
        end % if cntFile == 1

    end % for cntFile = 1 : 5

    figure;
    plot(bintIndex_center, MS_ATS(:,2), '-r', bintIndex_center,
MS_Model_base(:,2), '-k', ...
         bintIndex_center, MS_Model_double_CA_TT(:,2), '--b',
bintIndex_center, MS_Model_half_CA_TT(:,2), '-.b');
    xlabel('Distance (sm)'); ylabel('Market Share (%)')
    if bussTripsSwitch == 1
        title('BUSINESS: Sensitivity: Double and Half Commercial Air Travel
Time');
        legend('ATS', 'Base Model', 'Model Double Travel Time','Model half
Travel Time', 'Location', 'SouthEast'); grid on
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TT_Sensitivity_Buss.fig'])
    elseif bussTripsSwitch == 2
        title('NON-BUSINESS: Sensitivity: Double and Half Commercial Air
Travel Time');
        legend('ATS', 'Base Model', 'Model Double Travel Time','Model half
Travel Time', 'Location', 'SouthEast'); grid on
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TT_Sensitivity_NonBuss.fig'
])
    end % if bussTripsSwitch == 1

    figure;
    plot(bintIndex_center, MS_ATS(:,2), '-r', bintIndex_center,
MS_Model_base(:,2), '-k', ...
        bintIndex_center, MS_Model_double_CA_TC(:,2), '--b',
bintIndex_center, MS_Model_half_CA_TC(:,2), '-.b');
    xlabel('Distance (sm)'); ylabel('Market Share (%)')
    if bussTripsSwitch == 1
        title('BUSINESS: Sensitivity: Double and Half Commercial Air Travel
Cost');
        legend('ATS', 'Base Model', 'Model Double Travel Time','Model half
Travel Cost', 'Location', 'SouthEast'); grid on
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TC_Sensitivity_Buss.fig'])
    elseif bussTripsSwitch == 2
        title('NON-BUSINESS: Sensitivity: Double and Half Commercial Air
Travel Cost');
        legend('ATS', 'Base Model','Model Double Travel Time','Model half
Travel Cost', 'Location', 'SouthEast'); grid on
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TC_Sensitivity_NonBuss.fig'
])
    end % if bussTripsSwitch == 1
    figure;
    subplot(3,2,1); plot(bintIndex_center, MS_Model_base_income(:,1,2), '.-
r'); hold on
    subplot(3,2,1); plot(bintIndex_center,
MS_Model_double_CA_TT_income(:,1,2), '--b', bintIndex_center,
MS_Model_half_CA_TT_income(:,1,2), '--k')
    title('Travel Time Sensitivity: Income <30K'); grid on

    subplot(3,2,2); plot(bintIndex_center, MS_Model_base_income(:,2,2), '.-
r'); hold on
    subplot(3,2,2); plot(bintIndex_center,
MS_Model_double_CA_TT_income(:,2,2), '--b', bintIndex_center,
MS_Model_half_CA_TT_income(:,2,2), '--k'); grid on
    title('Travel Time Sensitivity: Income 30 to 60K')

    subplot(3,2,3); plot(bintIndex_center, MS_Model_base_income(:,3,2), '.-
r'); hold on
    subplot(3,2,3); plot(bintIndex_center,
MS_Model_double_CA_TT_income(:,3,2), '--b', bintIndex_center,
MS_Model_half_CA_TT_income(:,3,2), '--k'); grid on
    title('Travel Time Sensitivity: Income 60 to 100K')

    subplot(3,2,4); plot(bintIndex_center, MS_Model_base_income(:,4,2), '.-
r'); hold on
    subplot(3,2,4); plot(bintIndex_center,
MS_Model_double_CA_TT_income(:,4,2), '--b', bintIndex_center,
MS_Model_half_CA_TT_income(:,4,2), '--k'); grid on
    title('Travel Time Sensitivity: Income 100 to 150K')

    subplot(3,2,5); plot(bintIndex_center, MS_Model_base_income(:,5,2), '.-
r'); hold on
    subplot(3,2,5); plot(bintIndex_center,
MS_Model_double_CA_TT_income(:,5,2), '--b', bintIndex_center,
MS_Model_half_CA_TT_income(:,5,2), '--k'); grid on
    title('Travel Time Sensitivity: Income >150K')
    legend('Base Model', 'Double Travel Time', 'Half Travel Time' ,
'Location', 'SouthEast');

    if bussTripsSwitch == 1
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TT_Sensitivity_Buss_Income.fig
'])
    elseif bussTripsSwitch == 2
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TT_Sensitivity_NonBuss_Inco
me.fig'])
    end % if bussTripsSwitch == 1



    figure;
    subplot(3,2,1); plot(bintIndex_center, MS_Model_base_income(:,1,2), '-
r'); hold on
    subplot(3,2,1); plot(bintIndex_center,
MS_Model_double_CA_TC_income(:,1,2), '--b', bintIndex_center,
MS_Model_half_CA_TC_income(:,1,2), '--k'); grid on
    title('Travel Cost Sensitivity: Income <30K')
    subplot(3,2,2); plot(bintIndex_center, MS_Model_base_income(:,2,2), '-
r'); hold on
    subplot(3,2,2); plot(bintIndex_center,
MS_Model_double_CA_TC_income(:,2,2), '--b', bintIndex_center,
MS_Model_half_CA_TC_income(:,2,2), '--k'); grid on
    title('Travel Cost Sensitivity: Income 30 to 60K')

    subplot(3,2,3); plot(bintIndex_center, MS_Model_base_income(:,3,2), '-
r'); hold on
    subplot(3,2,3); plot(bintIndex_center,
MS_Model_double_CA_TC_income(:,3,2), '--b', bintIndex_center,
MS_Model_half_CA_TC_income(:,3,2), '--k'); grid on
    title('Travel Cost Sensitivity: Income 60 to 100K')

    subplot(3,2,4); plot(bintIndex_center, MS_Model_base_income(:,4,2), '-
r'); hold on
    subplot(3,2,4); plot(bintIndex_center,
MS_Model_double_CA_TC_income(:,4,2), '--b', bintIndex_center,
MS_Model_half_CA_TC_income(:,4,2), '--k'); grid on
    title('Travel Cost Sensitivity: Income 100 to 150K')

    subplot(3,2,5); plot(bintIndex_center, MS_Model_base_income(:,5,2), '-
r'); hold on
    subplot(3,2,5); plot(bintIndex_center,
MS_Model_double_CA_TC_income(:,5,2), '--b', bintIndex_center,
MS_Model_half_CA_TC_income(:,5,2), '--k'); grid on
    title('Travel Cost Sensitivity: Income >150K')
    legend('Base Model', 'Double Travel Cost', 'Half Travel Cost' ,
'Location', 'SouthEast');

    if bussTripsSwitch == 1
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TC_Sensitivity_Buss_Income.fig
'])
    elseif bussTripsSwitch == 2
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TC_Sensitivity_NonBuss_Inco
me.fig'])
    end % if bussTripsSwitch == 1


    figure;
    plot(bintIndex_center, avg_autoTravelTime(:,1), '-r'); hold on
    plot(bintIndex_center, avg_commTravelTime(:,1), '-k', bintIndex_center,
avg_commTravelTime(:,2), '-b', ...
          bintIndex_center, avg_commTravelTime(:,3), '-g'); grid on
    legend('Auto Travel Time','Comm Travel Time 1','Comm Travel Time 2','Comm
Travel Time 3')
    if bussTripsSwitch == 1
        title('BUSINESS: Travel Time Profile');
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TT_Profile_Buss.fig'])
    elseif bussTripsSwitch == 2
        title('NON-BUSINESS: Travel Time Profile');
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TT_Profile_NonBuss.fig'])
```

```
    end % if bussTripsSwitch == 1

    figure;
    plot(bintIndex_center, avg_autoTravelCost(:,1), '-r'); hold on
    plot(bintIndex_center, avg_commTravelCost(:,1), '-k', bintIndex_center,
avg_commTravelCost(:,2), '-b', ...
         bintIndex_center, avg_commTravelCost(:,3), '-g'); grid on
    legend('Auto Travel Cost','Comm Travel Cost 1','Comm Travel Cost 2','Comm
Travel Cost 3')
    if bussTripsSwitch == 1
        title('BUSINESS: Travel Cost Profile');
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\business\TC_Profile_Buss.fig'])
    elseif bussTripsSwitch == 2
        title('NON-BUSINESS: Travel Cost Profile');
        saveas(gcf, [load_Dir,
'\Calibration_Input_Files_SAS\captive\nonBusiness\TC_Profile_NonBuss.fig'])
    end % if bussTripsSwitch == 1

end % for bussTripsSwitch = 1 : 1

return


------------------------------------------------------------------------------------------------------------------------------------
------------
```

## 7.10 Pre-process Data for Mode Choice Application

### *7.10.1 Split Commercial Air Fare Data by Income Group*

```
-------------------------------------------------------------------------------------------------------------------
------------

function CA_Fares_PreProcess()

% This m file preprocesses the fare table for the mode choice in order to speed up calculations

clear all; clc

Install_Dir = 'C:\Program Files\TSAM\4.2\data\mode_choice\input';

load (strcat(Install_Dir, '\A2A_CA_BusinessClassFare_DB1B_2000')); % A2A Business class fare table
(443x443)
load (strcat(Install_Dir, '\A2A_CA_CoachClassFare_DB1B_Filled_2000'));    % A2A Coach class fare
table (443x443)
load (strcat(Install_Dir, '\distA2A_CA_443'));

% Initialize constants
incomeTripPurpose = [0.10 0.20 0.35 0.45 0.60; 0.02 0.05 0.10 0.15 0.25];


for tripTypeConstant = 1:2
    averageFare_CA = zeros(5, 443, 443);

    if tripTypeConstant == 1 %Business
        perctCoachClassTicketsByIncomeGrp   = 1-incomeTripPurpose(1,:);
        perctOfFirstClassTicketsByIncomeGrp = incomeTripPurpose(1,:);
    else %Non Business
        perctCoachClassTicketsByIncomeGrp   = 1-incomeTripPurpose(2,:);
        perctOfFirstClassTicketsByIncomeGrp = incomeTripPurpose(2,:);
    end % if tripTypeConstant == 1

    for cnt_orgApt = 1:443
        for cnt_desApt = 1:443
            if cnt_orgApt ~= cnt_desApt
                bussClassFare_2way  = A2A_CA_BusinessClassFare_DB1B_2000(cnt_orgApt, cnt_desApt);
                coachClassFare_2way = A2A_CA_CoachClassFare_DB1B_Filled_2000(cnt_orgApt,
cnt_desApt);
                if bussClassFare_2way <= 0
                    % if no business class fare use coach class fare
                    averageFare_CA(:, cnt_orgApt, cnt_desApt) = ones(1, 5) * coachClassFare_2way;
                else
                    averageFare_CA(:, cnt_orgApt, cnt_desApt) = (coachClassFare_2way *
perctCoachClassTicketsByIncomeGrp + ...
                        bussClassFare_2way * perctOfFirstClassTicketsByIncomeGrp);
                end % if bussClassFare_2way == 0
            end % if cnt_orgApt ~= cnt_desApt
        end % for cnt_desApt = 1:443
    end % for cnt_orgApt = 1:443

    averageFare_CA = round(averageFare_CA);

    if tripTypeConstant == 1
        save (strcat(Install_Dir, '\averageFare_CA_Business'), 'averageFare_CA');
    else
        save (strcat(Install_Dir, '\averageFare_CA_NonBusiness'), 'averageFare_CA');
    end % if tripTypeConstant == 1
end % for tripTypeConstant = 1:2
```
---------------------------------------------------------------------------------------------------------------------
------------

### *7.10.2 Pre-process Commerical Air Travel Times for TSAM (Main File)*

------------------------------------------------------------------------------------------------------------------------------------------

```
function ModeChoiceCaptive_AllToAll_SATS()

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Viginia Tech. Air Transportation Systems Lab., Blacksburg, VA
%
% Mode Choice Model (Including Commerical Aviation Airport Assignment)
%
% Main File: Converts the trip distribution output made up of [3091x3091]
inter-county person-trip tables
% to [3091x3091] inter-county person-trips by mode. Also creates inter-
airport trip tables for
% commerecial airline airpors and SATS airports
%
% Crearted By:   Senanu Ashiabor
% Date:          March, 2003
% Last Modified: August, 2004
%
% Calling:   createModeChoiceTable
% Called by: None
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% function ModeChoiceCaptive_AllToAll_SATS(Install_Dir, Project_Dir,
ModeChoiceType, CaseFolder, CaseName, StateNumber, SATSCase_YesNo, Year,
ProfileFilename, tripPuroposeSwitch, costPerMile_auto_vb,
costPerMile_SATS_vb, airportSetSwitch, LLMAirportSetName)
clc; clear all; tic

%-----------------------------
% Declare global variables
%-----------------------------
% tables
global A2A_CA_CoachFare_Index_DB1B_2000 A2A_CA_Schedule_Delay_2way
A2A_CA_travelTime_no_SD_2way
global CA_Hub_Type CA_candArpts_driveTime CA_candArpts_driveDist
CA_candArpts_arptNumb
global distA2A_CA MinNumberOfLegs C2CDriveDist_Population_Centroids

Install_Dir    = 'C:\Program Files\TSAM\4.2';
ModeChoiceType = 'ALL2ALL';
SATSCase_YesNo = 'No';
StateNumber    = 0; %0 is for ALL the states.

load (strcat(Install_Dir,
'\data\mode_choice\input\A2A_CA_CoachFare_Index_DB1B_2000'))  % Inter-
airporrt travel times in hours (685x685)

load (strcat(Install_Dir, '\data\mode_choice\input\avgPartySize_ATS_Buss'))
% Business party size from ATS [5x7]
```

```
load (strcat(Install_Dir,
'\data\mode_choice\input\avgPartySize_ATS_NonBuss'))          % Business
party size from ATS [5x7]

load (strcat(Install_Dir, '\data\mode_choice\input\CA_Hub_Type'))

load (strcat(Install_Dir,
'\data\mode_choice\input\A2A_CA_travelTime_no_SD_2way'))      % Inter-
airporrt travel times in hours (685x685)
load (strcat(Install_Dir,
'\data\mode_choice\input\A2A_CA_Schedule_Delay_2way'))        % Inter-
airporrt schedule delay in hours (685x685)

load (strcat(Install_Dir, '\data\mode_choice\input\C2A_GCDistance_popCent'))
% County to Airport distance table (3091x419)

load (strcat(Install_Dir, '\data\mode_choice\input\CA_candArpts_arptNumb'))
% Index of CA candidate airports within 100mile radius of each county
load (strcat(Install_Dir, '\data\mode_choice\input\CA_candArpts_driveDist'))
% Drive distance to CA candidate airports within 100mile radius of each
county
load (strcat(Install_Dir, '\data\mode_choice\input\CA_candArpts_driveTime'))
% Drive distance to CA candidate airports in 100mile radius of each county
% Hub types: 1 -> Large hub; 2-> Medium hub; 3 -> Small hub; 4 -> Non-Hub

load (strcat(Install_Dir, '\data\mode_choice\input\distA2A_CA_443'))
% Airport to airport distance table (443x443)

load (strcat(Install_Dir, '\data\mode_choice\input\MinNumberOfLegs'))
% Inter-airporrt travel times in hours (685x685)
load (strcat(Install_Dir, '\data\mode_choice\input\msaIndicatorData'))
% List indicating if county is in an MSA. (1 -> MSA; 0 -> nonMSA)

load (strcat(Install_Dir, '\data\mode_choice\input\stateNames'))
% List of states in US
load (strcat(Install_Dir, '\data\mode_choice\input\stateIndexing'))
% Index showing where counties of each state begin and end in the county to
county lis


for stateCounter = 1 : 52

    if mod(stateCounter, 5) == 0
        disp(['Origin State: ', num2str(stateCounter), ' Time: ',
num2str(toc)]);
    end % if mod(stateCounter, 5) == 0

    stateName  = char(stateNames(stateCounter, 1));
    starter    = stateIndexing(stateCounter, 1);

    startRow = stateIndexing(stateCounter, 1);
    endRow   = stateIndexing(stateCounter, 2);

    load (strcat(Install_Dir,
'\data\mode_choice\input\C2CDriveDist_Population_Centroids'))  % County to
County drive distance (using mappoint)
```

```
    C2CDriveDist_Population_Centroids = 
C2CDriveDist_Population_Centroids(startRow:endRow, :);
    load (strcat(Install_Dir, 
'\data\mode_choice\input\distC2C_Population_Centroids'))        % County to 
county distance table (Upper triangular) use when mapquest cannot compute 
drive times
    distC2C_Population_Centroids = 
distC2C_Population_Centroids(startRow:endRow, :);

    createModeChoiceTable(Install_Dir, starter, stateName, 1, 3091);

end % for stateCounter = 1 : 51

clear all;
```

--------------------------------------------------------------------------------------------------------------------------------------------

### *7.10.3 Compute Commerical Air Travel Times for TSAM*

------------------------------------------------------------------------------------------------------------------------------------

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveSATS
% Calling:    selectCandidateAirports_CA
%             computeSATS_TimeCost
%             nestedLogitFunction_SATS
%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function createModeChoiceTable(Install_Dir, starter, stateName,
StateStartIndex, StateEndIndex)

% initialize global variables
global C2CDriveDist_Population_Centroids

[numbRows numbCols]  = size(C2CDriveDist_Population_Centroids);
Selected_CA_Routes = zeros(9, 8, numbRows, numbCols);

%---------------------------------------------------
% Start Mode Split
%---------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCounty;
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number

    % keep track of location in run
    if mod(orgCountyReal, 50) == 0
        disp(['Origin County: ', num2str(orgCountyReal), ' Time: ',
num2str(toc)]);
    end % if mod(orgCountyReal, 50) == 0

    alaskaOrigin = ~isempty(find(orgCountyReal == [68:78]));
    hawaiiOrigin = ~isempty(find(orgCountyReal == [527:530]));

    if alaskaOrigin == 1 || hawaiiOrigin == 1
    else

        for desCounty = StateStartIndex : StateEndIndex %1 : numbCols % loop
for destination counties
            desCounty;
            alaskaDestnt = ~isempty(find(desCounty == [68:78]));
            hawaiiDestnt = ~isempty(find(desCounty == [527:530]));

            if alaskaDestnt == 1 || hawaiiDestnt == 1
            else
                routeDistance = C2CDriveDist_Population_Centroids(orgCounty,
desCounty);
```

```
                if routeDistance >= 100 % skip computation if not an
intercity trip

                    % Select candidate CA airports
                    %-------------------------------
                    candArptMatrix = [];
                    [candArptMatrix] = Select_CA_Routes(orgCountyReal,
desCounty);
                    %-> candArptMatrix(1, :) = Origin County Airports
                    %-> candArptMatrix(2, :) = Destination County Airports
                    %-> candArptMatrix(3, :) = Round trip Airport to Airport
travel time
                    %-> candArptMatrix(4, :) = Round trip Airport to Airport
Schedule delay
                    %-> candArptMatrix(5, :) = Access distance
                    %-> candArptMatrix(6, :) = Access Time (hours)
                    %-> candArptMatrix(7, :) = Egress distance
                    %-> candArptMatrix(8, :) = Egress Time (hours)
                    %-------------------------------------------------------

                    for i = 1:size(candArptMatrix, 2)
                        Selected_CA_Routes(i, :, orgCounty, desCounty)=
candArptMatrix(:, i);
                    end % for i = 1:size(candArptMatrix, 2)

                end %if distC2C_symm_popCentd(orgCounty_symm, desCounty_symm)
>= 100 % skip computation if not an intercity trip
            end % if alaskaDestnt == 1 | hawaiiDestnt == 1
        end % for desCounty = 1 : numbCols % loop for destination counties
    end % if alaskaOrigin == 1 | hawaiiOrigin == 1
end % for orgCounty = 1 : orgCntfor

save ([Install_Dir,
'\data\mode_choice\input\Selected_CA_Routes\Selected_CA_Routes_', stateName,
'.mat'], 'Selected_CA_Routes');
```

----------------------------------------------------------------------------------------------------------------------------------------
------------

### *7.10.4 Compute Commerical Air Travel Times for Routes between Counties*

```
------------------------------------------------------------------------------------------------------------
------------

function [candArptMatrix] = Select_CA_Routes(orgCountyReal, desCounty);

%----------------------------------------------------------------------------
--
% This script computes access and egress times and derives in-vehicle time
% for commercial aviation trips as
% In-vehicle-time = AccTime + EggressTime + LinehaulTime
%
% Assumptions: Waiting time is the same for all airports in US = 1.45hrs
%-------------------------------------------------------------------------

global A2A_CA_CoachFare_Index_DB1B_2000 CA_Hub_Type
A2A_CA_travelTime_no_SD_2way A2A_CA_Schedule_Delay_2way
global CA_candArpts_driveTime CA_candArpts_driveDist CA_candArpts_arptNumb
distA2A_CA MinNumberOfLegs

%-----------------
% Start Analysis
%-----------------
colCounter = 1;
candArptMatrix = [];

org_arptNumbers = CA_candArpts_arptNumb(orgCountyReal, :);
des_arptNumbers = CA_candArpts_arptNumb(desCounty, :);

nOrgArpt     = size(org_arptNumbers,2);
nDesArpt     = size(des_arptNumbers,2);
orgArpt_info = zeros(1,nOrgArpt);
desArpt_info = zeros(1,nDesArpt);

for k = 1 : nOrgArpt
    if org_arptNumbers(k) == -9999
    else
        orgArpt_info(1, k) = CA_Hub_Type(org_arptNumbers(k));
    end % if org_arptNumbers(k) == -9999
end % for k = 1 : nOrgArpt
for m = 1 : nDesArpt
    if des_arptNumbers(m) == -9999
    else
        desArpt_info(1, m) = CA_Hub_Type(des_arptNumbers(m));
    end % if des_arptNumbers(m) == -9999
end % for k = 1 : nOrgArpt

org_LH_index = find(orgArpt_info == 1);
des_LH_index = find(desArpt_info == 1);

for i = 1 : nOrgArpt
    orgArptNumber = org_arptNumbers(i);

    for j = 1 : nDesArpt
```

```
        desArptNumber = des_arptNumbers(j);

        if orgArptNumber == -9999 || desArptNumber == -9999
        else
            if orgArptNumber ~= desArptNumber
                orgArpt_hubType = CA_Hub_Type(orgArptNumber);
                desArpt_hubType = CA_Hub_Type(desArptNumber);
                A2A_distance    = distA2A_CA(orgArptNumber, desArptNumber);

                if A2A_distance > 75 || (orgArpt_hubType == 1 &
desArpt_hubType == 1)
                    loop_value = 0;

                    % if there is a non-hub competing with large hub at
origin
                    if orgArpt_hubType == 4 & sum(org_LH_index) > 0
                        minimumConnection = MinNumberOfLegs(orgArptNumber,
desArptNumber);
                        curveIndex =
sum(A2A_CA_CoachFare_Index_DB1B_2000(orgArptNumber, desArptNumber,:));
                        % if non-hub has a direct flight OR at fare is from
DB1B
                        if (minimumConnection == 1) || (curveIndex == 0)
                        else
                            loop_value = 1;
                        end % if (minimumConnection == 1) || (curveIndex ==
0)
                    end % if orgArpt_hubType == 4 & sum(org_LH_index) > 0

                    if loop_value == 1
                    else
                        % if there is a non-hub competing with large hub at
origin
                        if desArpt_hubType == 4 & sum(des_LH_index) > 0
                            minimumConnection =
MinNumberOfLegs(desArptNumber, orgArptNumber);
                            curveIndex =
sum(A2A_CA_CoachFare_Index_DB1B_2000(desArptNumber, orgArptNumber,:));
                            % if non-hub has a direct flight OR at fare is
from DB1B
                            if (minimumConnection == 1) || (curveIndex == 0)
                            else
                                loop_value = 1;
                            end % if (minimumConnection == 1) || (curveIndex
== 0)
                        end % if orgArpt_hubType == 4 & sum(org_LH_index) > 0

                        if loop_value == 1
                        else
                            % ACCESS TIME
                            accessTime_hours =
CA_candArpts_driveTime(orgCountyReal, i);
                            egressTime_hours =
CA_candArpts_driveTime(desCounty, j);

                            % ACCESS DISTANCE
                            accessDist =
CA_candArpts_driveDist(orgCountyReal, i);
                            egressDist = CA_candArpts_driveDist(desCounty,
j);

                            lineHaulTime_2way  =
A2A_CA_travelTime_no_SD_2way(orgArptNumber, desArptNumber);
                            scheduleDelay_2way =
A2A_CA_Schedule_Delay_2way(orgArptNumber, desArptNumber);

                            candArptMatrix(1, colCounter)  = orgArptNumber;
                            candArptMatrix(2, colCounter)  = desArptNumber;
                            candArptMatrix(3, colCounter)  =
round(lineHaulTime_2way*10)/10;
                            candArptMatrix(4, colCounter)  =
round(scheduleDelay_2way*100)/100;
                            candArptMatrix(5, colCounter)  =
round(accessDist);
                            candArptMatrix(6, colCounter)  =
round(accessTime_hours*10)/10;
                            candArptMatrix(7, colCounter)  =
round(egressDist);
                            candArptMatrix(8, colCounter)  =
round(egressTime_hours*10)/10;
                            colCounter = colCounter + 1;
                        end % if loop_value == 1
                    end % if loop_value == 1
                end % if loop_value == 1

            end % if A2A_distance > 75 || (orgArpt_hubType == 1 &
desArpt_hubType == 1)

        end % if orgArptNumber == -9999 | desArptNumber == -9999

    end % for i = 1 : size(orgCntArptData, 1)
end % for i = 1 : size(destCntArptData, 1)
```

-------------------------------------------------------------------------------------------------------------------------------------
------------

## 7.11 Model Application to Estimate National Level Demand (Using TSAM)

### *7.11.1 Main File: Loop by State and Summarize Mode Choice Output*

--------------------------------------------------------------------------------------------------------------------------------------

```
% function ModeChoiceCaptive_AllToAll(Install_Dir, Project_Dir,
ModeChoiceType, CaseFolder, CaseName, SelectedStateNumber, AirTaxi_YesNo, ...
%     Year, FlightSpeedProfileFilename, BADAFileName, TripPurpose,
costPerMile_Auto_vb, CommercailAirportsProcessingTime_vb,
AirlineFares_ScalingFactor_vb, AirlineFlyingTime_ScalingFactor_vb, ...
%     AirTaxiAirportsProcessingTime_vb, costPerMile_AirTaxi_vb,
CostProfileFilename, scheduleDelay_AirTaxi_vb, AirportSetName,
CustomAirportSetName, LLMAirportSetName, ...
%     MCATS_iteration_switch, MCATS_AirTaxi_cost, MCATS_schedule_delay,
maxDailyDriveTime_Auto_vb, maxAccessEgressTime_AirTaxi_vb, ...
%     Train_YesNo, TrainStationProcessingTime_vb,
maxAccessEgressTime_Train_vb, TrainType)

function ModeChoiceCaptive_AllToAll()

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Viginia Tech. Air Transportation Systems Lab., Blacksburg, VA
%
% Mode Choice Model (Including Commerical Aviation Airport Assignment)
%
% Main File: Converts the trip distribution output made up of [3091x3091]
inter-county person-trip tables
% to [3091x3091] inter-county person-trips by mode. Also creates inter-
airport trip tables for
% commerecial airline airpors and AirTaxi airports
%
% Crearted By:   Senanu Ashiabor
% Date:          March, 2003
% Last Modified: August, 2004
%
% Calling:   createModeChoiceTable
% Called by: None
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%-----------------------------
% Declare global variables
%-----------------------------
clear global;

% tables
global model_coefficients averageFare_CA c2aData_AirTaxi
C2CDriveTime_Population_Centroids
global CA_Hub_Type CA_TripTable C2CDriveDist_Population_Centroids
C2SData_Train distA2A_AirTaxi
```

```
global msaIndicatorData numberOfIncomeGroups Selected_CA_Routes
S2S_Train_TravelCost
global S2S_Train_TravelTime S2S_Train_ScheduleDelay waitingAtOriginAirport_CA
waitingAtDestntAirport_CA

% constants
global additionalDay_AirTaxi additionalDay_Auto avgOccupancy_Auto
AirlineFlyingTime_ScalingFactor
global costPerMile_Auto costPerMile_AirTaxi drivingSpeed_Rural CostProfile
ConstantAirTaxiCost
global CA_dailyOvernightTime drivingSpeed_Urban FlightSpeedProfile
global max_CP_Distance_AirTaxi min_CP_Distance_AirTaxi max_CP_Cost_AirTaxi
min_CP_Cost_AirTaxi
global maxDailyDriveTime_Auto maxFlightHrsPerDay
global max_FSP_Speed_AirTaxi min_FSP_Speed_AirTaxi maxAccessTime_AirTaxi
maxEgressTime_AirTaxi
global maxAccessTime_Train maxEgressTime_Train min_A2A_Distance_AirTaxi
maxRoundTripTime_CA
global minRoundTripTime_CA oneDayLodgingCost scheduleDelay_AirTaxi
waitingAtDestStation_Train
global waitingAtOrginStation_Train waitingAtOrginAirport_AirTaxi
waitingAtDestAirport_AirTaxi

AirlineFares_ScalingFactor_vb      = 1.00;
AirlineFlyingTime_ScalingFactor_vb = 1.0;
AirportSetName                     = 'FULL_With_OEP';
AirTaxi_YesNo                      = 'No';
AirTaxi_Trip_Redistribution        = 'No';
AirTaxiAirportsProcessingTime_vb(1,1) = 0.5;        % Origin hrs %45/60
AirTaxiAirportsProcessingTime_vb(1,2) = 0.25;        % Destination hrs%35/60
BADAFileName               = 'VLJ.PTF';
CaseFolder                 = 'Test New Model 2'; %;
CaseName                   = 'Base Case - Business - Car Cost $0.37 per mile
- Year 2000'; %'BaseCase_CAR30';
CommercailAirportsProcessingTime_vb(:,1) = [2.00 1.50 1.25 1.00]; % Waiting
time in hours
CommercailAirportsProcessingTime_vb(:,2) = [0.75 0.75 0.50 0.50]; % Waiting
time in hours
costPerMile_Auto_vb        = 0.37;
costPerMile_AirTaxi_vb     = 1.75;
CostProfileFilename        = 'VLJ 1200 hr per year utilization';
CustomAirportSetName       = 'NONE';
FlightSpeedProfileFilename = 'VLJ_RANGE_1100_MAXIMUMALTITUDE_400_PROFILE';
Install_Dir                   = 'C:\Program Files\TSAM\4.2';
%'D:\Workspace\AirTaxi_GUI\vb_380'; %'C:\Workspace\AirTaxi_GUI\vb';
LLMAirportSetName             = 'GPA3_DT0_ACO0_SSF10';
MCATS_iteration_switch        = 0;
ModeChoiceType                = 'ALL2ALL';
%maxDailyDriveTime_Auto_vb   = [7.5 9];
maxDailyDriveTime_Auto_vb     = [8 10];
maxAccessEgressTime_AirTaxi_vb   = [4.0 4.0];
maxAccessEgressTime_Train_vb  = [2.0 2.0];
Project_Dir                   = 'D:\TSAM_Project';
scheduleDelay_AirTaxi_vb        = 1.0;
SelectedStateNumber           = 0; %0 is for ALL the states.
Train_YesNo                   = 'No';
TripPurpose                   = 'Business';
```

```
TrainStationProcessingTime_vb = [0.333, 0.167];
Year                          = 2000;


%--------------
% Load Tables
%--------------
load (strcat(Install_Dir, '\data\mode_choice\input\msaIndicatorData.mat'))
% List indicating if county is in an MSA. (1 -> MSA; 0 -> nonMSA)
load (strcat(Install_Dir, '\data\mode_choice\input\stateNames.mat'))
% List of states in US
load (strcat(Install_Dir, '\data\mode_choice\input\stateIndexing.mat'))
% Index showing where counties of each state begin and end in the county to
county lis
load (strcat(Install_Dir, '\data\mode_choice\input\CA_Hub_Type.mat'))

if MCATS_iteration_switch == 1
    load (strcat(Install_Dir, '\data\mode_choice\input\MCATS_states_ID.mat'))
% List of states (MCATS ordering)
    load (strcat(Install_Dir,
'\data\mode_choice\input\MCATS_states_region.mat'))
end % if MCATS_iteration_switch == 1

numberOfIncomeGroups = 5;

%-----------%
% constants %
%-----------%
if strcmp(TripPurpose, 'Business') == 1; % Business
    load (strcat(Install_Dir,
'\data\mode_choice\input\averageFare_CA_Business.mat'))  % Airport to airport
business class fare table (443x443)
    load (strcat(Install_Dir,
'\data\mode_choice\input\avgPartySize_ATS_Buss'))        % Business party
size from ATS [5x7]
    load (strcat(Install_Dir,
'\data\mode_choice\input\ParameterEstimate_NL_business_TT_TC_SHT_MODE_DUM_Mat
File'));

    avgOccupancy_Auto      = mean(avgPartySize_ATS_Buss');
    model_coefficients     = parameterEstimates(5:12,1);
    minRoundTripTime_CA    = 10; maxRoundTripTime_CA    = 16;
    maxDailyDriveTime_Auto = maxDailyDriveTime_Auto_vb(1);           % hrs
    oneDayLodgingCost      = [70 80 90 100 120]; % $
    clear avgPartySize_ATS_Buss parameters_buss
elseif strcmp(TripPurpose, 'Non-Business') == 1; % Non-Business
    load (strcat(Install_Dir,
'\data\mode_choice\input\averageFare_CA_NonBusiness.mat')) % Airport to
airport coach class fare table (419x419)
    load (strcat(Install_Dir,
'\data\mode_choice\input\avgPartySize_ATS_NonBuss'))            % Business
party size from ATS [5x7]
    load (strcat(Install_Dir,
'\data\mode_choice\input\ParameterEstimate_NL_nonBusiness_TT_TC_SHT_MODE_DUM_
MatFile'));

    avgOccupancy_Auto      = mean(avgPartySize_ATS_NonBuss');
    model_coefficients     = parameterEstimates(5:12,1);
    minRoundTripTime_CA    = 10; maxRoundTripTime_CA    = 16;
    maxDailyDriveTime_Auto = maxDailyDriveTime_Auto_vb(2);           % hrs
    oneDayLodgingCost      = [50 60 70 80 90]; % $
    clear avgPartySize_ATS_NonBuss parameters_nonBuss
end % if TripPurpose == 1

costPerMile_Auto   = double(costPerMile_Auto_vb);
averageFare_CA     = averageFare_CA * AirlineFares_ScalingFactor_vb;
%AirlineFares_ScalingFactor = 1.0 (Default)
additionalDay_Auto = 24 - maxDailyDriveTime_Auto; % hrs.
drivingSpeed_Urban = 30; % mph
drivingSpeed_Rural = 60; % mph
maxFlightHrsPerDay = maxDailyDriveTime_Auto; % hrs
waitingAtOriginAirport_CA = CommercailAirportsProcessingTime_vb(:,1)';
waitingAtDestntAirport_CA = CommercailAirportsProcessingTime_vb(:,2)';
CA_dailyOvernightTime     = 24 - minRoundTripTime_CA; % hrs
AirlineFlyingTime_ScalingFactor = AirlineFlyingTime_ScalingFactor_vb;

% AirTaxi Options
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    AirTaxiSwitch = 2;
    maxAccessTime_AirTaxi = maxAccessEgressTime_AirTaxi_vb(1);
    maxEgressTime_AirTaxi = maxAccessEgressTime_AirTaxi_vb(2);
    min_A2A_Distance_AirTaxi = 75; % statute miles
    waitingAtOrginAirport_AirTaxi = AirTaxiAirportsProcessingTime_vb(1);
    waitingAtDestAirport_AirTaxi = AirTaxiAirportsProcessingTime_vb(2);
    scheduleDelay_AirTaxi            = scheduleDelay_AirTaxi_vb; % hrs
    additionalDay_AirTaxi            = additionalDay_Auto;          % hour
    costPerMile_AirTaxi = double(costPerMile_AirTaxi_vb);

    %FLight Speed Profile
    load (strcat(Install_Dir, '\data\bada_library\',
FlightSpeedProfileFilename, '.mat'));
    min_FSP_Distance_AirTaxi    = FlightSpeedProfile(2, 1);
    max_FSP_Distance_AirTaxi    = FlightSpeedProfile(2, end);
    min_FSP_Speed_AirTaxi       = FlightSpeedProfile(1, 1);
    max_FSP_Speed_AirTaxi       = FlightSpeedProfile(1, end);

    % Cost Profile
    if strcmp(CostProfileFilename, 'NONE') == 0
        CostProfile_Filename = fopen(strcat(Install_Dir,
'\data\cost_profiles\', CostProfileFilename, '.cp'), 'r');
        CostProfile = cell2mat(textscan(CostProfile_Filename, '%f %f %f',
'delimiter', ','));
        fclose(CostProfile_Filename);
        min_CP_Distance_AirTaxi    = CostProfile(1, 2);
        max_CP_Distance_AirTaxi    = CostProfile(end, 2);
        min_CP_Cost_AirTaxi        = CostProfile(1, 3);
        max_CP_Cost_AirTaxi        = CostProfile(end, 3);
        ConstantAirTaxiCost = 0;
    else
        ConstantAirTaxiCost = 1;
    end % if strcmp(CostProfileFilename, 'NONE') == 0

    %AirTaxi Airport Set Selection
    if strcmp(AirportSetName, 'FULL_Without_OEP') == 1 %FULL Without OEP
```

```
        load (strcat(Install_Dir,
'\data\mode_choice\input\C2AData_FULL_Without_OEP.mat'));
        load (strcat(Install_Dir,
'\data\mode_choice\input\distA2A_FULL_Without_OEP.mat'));  % Airport to
airport distance table for AirTaxi airports (3346x3346)
        c2aData_AirTaxi = C2AData_FULL_Without_OEP;
        distA2A_AirTaxi = distA2A_FULL_Without_OEP;
        clear C2AData_FULL_Without_OEP;
        clear distA2A_FULL_Without_OEP;
    elseif strcmp(AirportSetName, 'FULL_With_OEP') == 1 % FULL With OEP
        load (strcat(Install_Dir,
'\data\mode_choice\input\C2AData_FULL_With_OEP.mat'));
        load (strcat(Install_Dir,
'\data\mode_choice\input\distA2A_FULL_With_OEP.mat'));  % Airport to airport
distance table for AirTaxi airports (3346x3346)
        c2aData_AirTaxi = C2AData_FULL_With_OEP;
        distA2A_AirTaxi = distA2A_FULL_With_OEP;
        clear C2AData_FULL_With_OEP;
        clear distA2A_FULL_With_OEP;
    elseif strcmp(AirportSetName, 'ILS_Without_OEP') == 1 %ILS_Without_OEP
Airport Set
        load (strcat(Install_Dir,
'\data\mode_choice\input\C2AData_ILS_Without_OEP.mat'));
        load (strcat(Install_Dir,
'\data\mode_choice\input\distA2A_ILS_Without_OEP.mat')) % Airport to airport
distance table for AirTaxi airports (3346x3346)
        c2aData_AirTaxi = C2AData_ILS_Without_OEP;
        distA2A_AirTaxi = distA2A_ILS_Without_OEP;
        clear C2AData_ILS_Without_OEP;
        clear distA2A_ILS_Without_OEP;
    elseif strcmp(AirportSetName, 'LLM') == 1 %LLM Airport Set
        load (strcat(Project_Dir, '\mode_choice\input\LLM\',
LLMAirportSetName, '\C2AData_LLM.mat'));
        load (strcat(Project_Dir, '\mode_choice\input\LLM\',
LLMAirportSetName, '\distA2A_LLM.mat')) % Airport to airport distance table
for AirTaxi airports (3346x3346)
        c2aData_AirTaxi = C2AData_LLM;
        distA2A_AirTaxi = distA2A_LLM;
        clear C2AData_LLM;
        clear distA2A_LLM;
    elseif strcmp(AirportSetName, 'Custom') == 1 %Custom Airport Set
        load (strcat(Install_Dir, '\data\custom_airport_sets\',
CustomAirportSetName, '\C2AData_Custom.mat'));
        load (strcat(Install_Dir, '\data\custom_airport_sets\',
CustomAirportSetName, '\distA2A_Custom.mat')) % Airport to airport distance
table for SABTechnology airports (3346x3346)
        c2aData_AirTaxi = C2AData_Custom;
        distA2A_AirTaxi = distA2A_Custom;
        clear distA2A_Custom;
        clear C2AData_Custom;
    end % if strcmp(AirportSetName, 'FULL_Without_OEP') == 1 %FULL Without
OEP

    % Adjust disA2A_AirTaxi with detour factor
    Adjust_AirTaxi_DetourFactor(Install_Dir, BADAFileName);

else
```

```
    AirTaxiSwitch = 1;
    FlightSpeedProfileAirTaxi     = [];
    min_FlightSpeedProfileAirTaxi = [];
    max_FlightSpeedProfileAirTaxi = [];
    min_A2A_Distance_AirTaxi      = [];
end % if strcmp(AirTaxi_YesNo, 'Yes')  == 1

% Train Mode loading
if strcmp(Train_YesNo, 'Yes') == 1
    TrainSwitch = 2;
    load (strcat(Install_Dir, '\data\mode_choice\input\C2SData_Train.mat'));
% access time and distances to stations
    load (strcat(Install_Dir,
'\data\mode_choice\input\S2S_Train_TravelCost_', TrainType, '.mat')); % Train
Travel Cost between stations
    load (strcat(Install_Dir,
'\data\mode_choice\input\S2S_Train_ScheduleDelay_', TrainType, '.mat')); %
Train Schedule Delay between stations
    load (strcat(Install_Dir,
'\data\mode_choice\input\S2S_Train_TravelTime_', TrainType, '.mat')); % Train
Travel Time between stations
    TrainString_TC = ['S2S_Train_TravelCost = S2S_Train_TravelCost_',
TrainType, '; clear S2S_Train_TravelCost_', TrainType, ';'];
    TrainString_SD = ['S2S_Train_ScheduleDelay = S2S_Train_ScheduleDelay_',
TrainType, '; clear S2S_Train_ScheduleDelay_', TrainType, ';'];
    TrainString_TT = ['S2S_Train_TravelTime = S2S_Train_TravelTime_',
TrainType, '; clear S2S_Train_TravelTime_', TrainType, ';'];
    eval(TrainString_TC);
    eval(TrainString_SD);
    eval(TrainString_TT);
    waitingAtOrginStation_Train = TrainStationProcessingTime_vb(1);
    waitingAtDestStation_Train = TrainStationProcessingTime_vb(2);
    maxAccessTime_Train = maxAccessEgressTime_Train_vb(1);
    maxEgressTime_Train = maxAccessEgressTime_Train_vb(2);
else
    TrainSwitch = 1;
end % if strcmp(Train_YesNo, 'Yes') == 1

%--------------------------
% Initialize Contants
%--------------------------
inputProjDir = (strcat(Project_Dir, '\Trip_Distribution\Output\Y',
num2str(Year)));
saveProjDir  = (strcat(Project_Dir, '\mode_choice\output\', ModeChoiceType,
'\', CaseFolder, '\', CaseName, '\'));

% Save Selected State ID if State 2 State Module is running
if SelectedStateNumber > 0
    % Adjust the state number because Texas has 2 parts
    if SelectedStateNumber >= 45 % State 44 is Texas Part 1, 45 is Texas Part
2. However in GUI State 45 is UTAH.
        SelectedStateNumber = SelectedStateNumber + 1;
    end
    SelectedStateID = char(stateNames(SelectedStateNumber, 2));
end % if SelectedStateNumber > 0

%--------------------------
```

```
% Run Model State by State
%--------------------------
for stateCounter = 1 : 52

%     if mod(stateCounter, 5) == 0
%         disp(['State:', num2str(stateCounter), ' Time: ', num2str(toc)])
%     end % if mod(stateCounter, 5) == 0

    stateName  = char(stateNames(stateCounter, 1));
    stateID    = char(stateNames(stateCounter, 2));

    if MCATS_iteration_switch == 1
        MCATS_state_Index  = strmatch(stateID, MCATS_states_ID);
        MCATS_region_type  = MCATS_states_region(MCATS_state_Index);
        costPerMile_AirTaxi   = MCATS_AirTaxi_cost(MCATS_region_type);
        scheduleDelay_AirTaxi = MCATS_schedule_delay(MCATS_region_type);
    end % if MCATS_iteration_switch == 1

    starter  = stateIndexing(stateCounter, 1);
    startRow = stateIndexing(stateCounter, 1);
    endRow   = stateIndexing(stateCounter, 2);

    C2CDriveTime_Population_Centroids = [];
    C2CDriveDist_Population_Centroids = [];
    Selected_CA_Routes = [];

    load (strcat(Install_Dir,
'\data\mode_choice\input\Selected_CA_Routes\Selected_CA_Routes_', stateName,
'.mat'));
    load (strcat(Install_Dir,
'\data\mode_choice\input\C2CDriveTime_Population_Centroids'))  % County to
County drive times (using mappoint)
    C2CDriveTime_Population_Centroids =
C2CDriveTime_Population_Centroids(startRow:endRow, :);
    load (strcat(Install_Dir,
'\data\mode_choice\input\C2CDriveDist_Population_Centroids'))  % County to
County drive distance (using mappoint)
    C2CDriveDist_Population_Centroids =
C2CDriveDist_Population_Centroids(startRow:endRow, :);

    CA_TripTable = [];
    if strcmp(TripPurpose, 'Business') == 1;
        CAbuzzTripTable = [];
        load (strcat(inputProjDir, '\CAbuzzTripTable_', stateName, '_',
num2str(Year), '.mat'));
        CA_TripTable = CAbuzzTripTable;
        clear CAbuzzTripTable;
    elseif strcmp(TripPurpose, 'Non-Business') == 1;
        CAnonBuzzTripTable = [];
        load (strcat(inputProjDir, '\CAnonBuzzTripTable_', stateName, '_',
num2str(Year), '.mat'));
        CA_TripTable = CAnonBuzzTripTable;
        clear CAnonBuzzTripTable;
    end % if TripPurpose == 1;

    if SelectedStateNumber == 0 %All States
```

```
        createModeChoiceTable(TripPurpose, starter, AirTaxiSwitch,
TrainSwitch, stateName, 1, 3091, saveProjDir);
    else % One State
        if strcmp(SelectedStateID, stateID) == 1 % Selected State
            createModeChoiceTable(TripPurpose, starter, AirTaxiSwitch,
TrainSwitch, stateName, 1, 3091, saveProjDir);
        else
            % Finding the starting and ending index of the state
            StateStartIndex = stateIndexing(SelectedStateNumber, 1);
            StateEndIndex = stateIndexing(SelectedStateNumber, 2);
            createModeChoiceTable(TripPurpose, starter, AirTaxiSwitch,
TrainSwitch, stateName, StateStartIndex, StateEndIndex, saveProjDir);
        end % if strcmp(SelectedStateID, stateID) == 1 % Selected State
    end % if SelectedStateNumber == 0 %All States

    progress(1 + stateCounter);

end % for stateCounter = 1 : 52

% --------------------------
% Clear global variables
% --------------------------
% tables
clear C2CDriveTime_Population_Centroids C2CDriveDist_Population_Centroids
clear alpha_time_cost averageFare_CA c2aData_AirTaxi
waitingAtOriginAirport_CA waitingAtDestntAirport_CA
clear distA2A_AirTaxi numberOfIncomeGroups msaIndicatorData CA_Hub_Type
clear CA_TripTable Selected_CA_Routes
% constants
clear additionalDay_AirTaxi additionalDay_Auto avgOccupancy_Auto
clear costPerMile_Auto costPerMile_AirTaxi drivingMilesPerDay
FlightSpeedProfile
clear maxDailyDriveTime_Auto maxFlightHrsPerDay maxFlightTimePerLeg
maxDrivingLength min_Distance_AirTaxi
clear max_Distance_AirTaxi min_Speed_AirTaxi max_Speed_AirTaxi
oneDayLodgingCost waitingAtOrginAirport_AirTaxi waitingAtDestAirport_AirTaxi
clear drivingSpeed_Urban drivingSpeed_Rural scheduleDelay_AirTaxi
maxAccessTime_AirTaxi maxEgressTime_AirTaxi

% --------------------------
% Collapsing and Summary of mode choice results
% --------------------------

% %==========================================================================
% % Grouping of the C2A Tables
% SummarizeModeChoiceOutput_All2All_C2A_CA(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
% if strcmp(AirTaxi_YesNo, 'Yes')  == 1
%     SummarizeModeChoiceOutput_All2All_C2A_AirTaxi(Install_Dir, saveProjDir,
stateNames);
% end % if ~strcmp('BaseCase', CaseName(1, 1:8))
% if strcmp(Train_YesNo, 'Yes')  == 1
%     SummarizeModeChoiceOutput_All2All_C2A_Train(Install_Dir, saveProjDir,
stateNames);
% end % if ~strcmp('BaseCase', CaseName(1, 1:8))
% %==========================================================================
% Grouping of the C2C Tables
SummarizeModeChoiceOutput_All2All_C2C_Auto(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
SummarizeModeChoiceOutput_All2All_C2C_CA(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_C2C_AirTaxi(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end % if ~strcmp('BaseCase', CaseName(1, 1:8))
if strcmp(Train_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_C2C_Train(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end % if ~strcmp('BaseCase', CaseName(1, 1:8))
progress(54);

% Grouping of the A2A Tables
SummarizeModeChoiceOutput_All2All_A2A_CA(Install_Dir, saveProjDir,
stateNames);
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_A2A_AirTaxi(Install_Dir, saveProjDir,
stateNames);
end % if ~strcmp('BaseCase', CaseName(1, 1:8))
if strcmp(Train_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_A2A_Train(Install_Dir, saveProjDir,
stateNames);
end % if ~strcmp('BaseCase', CaseName(1, 1:8))
progress(55);

% Summarize Travel Time Output
SummarizeModeChoiceOutput_All2All_THBM_Auto(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
SummarizeModeChoiceOutput_All2All_THBM_CA(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_THBM_AirTaxi(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end
if strcmp(Train_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_THBM_Train(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end
progress(56);

% Summarize Travel Cost Output
SummarizeModeChoiceOutput_All2All_TCBM_Auto(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
SummarizeModeChoiceOutput_All2All_TCBM_CA(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_TCBM_AirTaxi(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end
if strcmp(Train_YesNo, 'Yes')  == 1
    SummarizeModeChoiceOutput_All2All_TCBM_Train(Install_Dir, saveProjDir,
stateNames, SelectedStateNumber, stateIndexing);
end
progress(57);
```

```
% Marketshare
[MarketshareByModeByIncome] = BinModelOutput_All2All(Install_Dir,
saveProjDir, AirTaxi_YesNo, Train_YesNo,  SelectedStateNumber,
stateIndexing);
save (strcat(saveProjDir, '\MarketshareByModeByIncome.mat'),
'MarketshareByModeByIncome');
clear MarketshareByModeByIncome;
progress(58);

% --------------------------
% AirTaxi Trip Redistribution
% --------------------------
if strcmp(AirTaxi_YesNo, 'Yes')  == 1
    if strcmp(AirTaxi_Trip_Redistribution, 'Yes') == 1
        redistribute_AirTaxi_Output(Install_Dir, Project_Dir, saveProjDir,
AirportSetName, LLMAirportSetName, CustomAirportSetName)
    end
end % if strcmp(AirTaxi_YesNo, 'Yes')  == 1
progress(59);

clear all;

return;
```

----------------------------------------------------------------------------------------------------------------------------------------

*7.11.2 Computes Travel Times and Costs, Computes Probabilities, and Uses Probability to Estimates Demand by Mode of Transportation*

```
-------------------------------------------------------------------------------------------------------------------------------
------------

function createModeChoiceTable(TripPurpose, starter, AirTaxiSwitch,
TrainSwitch, stateName, StateStartIndex, StateEndIndex, saveProjDir)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Create county to county and airport to airport person
% trip tables
%
% Called By:  modeChoiceCaptiveAirTaxi
% Calling:    selectCandidateAirports_CA
%             computeAirTaxi_TimeCost
%             nestedLogitFunction_AirTaxi
%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% initialize global variables
global averageFare_CA CA_TripTable C2CDriveDist_Population_Centroids
distA2A_AirTaxi
global waitingAtOriginAirport_CA waitingAtDestntAirport_CA Selected_CA_Routes
S2S_Train_TravelTime

% Initialize trip tables
income            = [25000 45000 80000 125000 170000]; % mean/median values
of household incomes
numbOfIncomeGrps  = size(income, 2);

% Preprocess driving cost and time
[c2cAuto_TravelTime_2way, c2cAuto_TravelCost_2W, TrvlCost_Auto_2way] =
ComputeAutoCostAndTime(income, TripPurpose);

[numbRows numbCols]  = size(CA_TripTable(1).trips);
c2cAuto_pTripTable   = zeros(numbRows, numbCols, numbOfIncomeGrps);
c2cCA_pTripTable     = zeros(numbRows, numbCols, numbOfIncomeGrps);

c2cCA_MinTravelTime_1W   = zeros(numbRows, numbCols);
c2cCA_MinTravelCost_2W   = zeros(numbRows, numbCols);
c2cCA_MaxTravelTime_1W   = zeros(numbRows, numbCols);
c2cCA_MaxTravelCost_2W   = zeros(numbRows, numbCols);

numbRows_CA      = size(averageFare_CA, 2);
numbCols_CA      = size(averageFare_CA, 3);
a2aCA_pTripTable = zeros(numbRows_CA, numbCols_CA, numbOfIncomeGrps);
a2aCA_RevenueTable = zeros(numbRows_CA, numbCols_CA, numbOfIncomeGrps);

totalHoursByMode_Auto_c2c = zeros(numbRows, numbCols, numbOfIncomeGrps);
totalHoursByMode_CA_c2c   = zeros(numbRows, numbCols, numbOfIncomeGrps);
totalCostByMode_Auto_c2c  = zeros(numbRows, numbCols, numbOfIncomeGrps);
totalCostByMode_CA_c2c    = zeros(numbRows, numbCols, numbOfIncomeGrps);
c2cCA_RevenueTable        = zeros(numbRows, numbCols, numbOfIncomeGrps);
```

```
if AirTaxiSwitch ~= 1 % AirTaxi = Yes
    c2cAirTaxi_pTripTable             = zeros(numbRows, numbCols,
numbOfIncomeGrps);
    [numbRows_AirTaxi numbCols_AirTaxi]  = size(distA2A_AirTaxi);
    a2aAirTaxi_pTripTable_Inc1        = zeros(numbRows_AirTaxi,
numbRows_AirTaxi);
    a2aAirTaxi_pTripTable_Inc2        = zeros(numbRows_AirTaxi,
numbRows_AirTaxi);
    a2aAirTaxi_pTripTable_Inc3        = zeros(numbRows_AirTaxi,
numbRows_AirTaxi);
    a2aAirTaxi_pTripTable_Inc4        = zeros(numbRows_AirTaxi,
numbRows_AirTaxi);
    a2aAirTaxi_pTripTable_Inc5        = zeros(numbRows_AirTaxi,
numbRows_AirTaxi);
    c2cAirTaxi_TravelTime_1W          = zeros(numbRows, numbCols);
    c2cAirTaxi_TravelCost_2W          = zeros(numbRows, numbCols);
    totalHoursByMode_AirTaxi_c2c      = zeros(numbRows, numbCols,
numbOfIncomeGrps);
    totalCostByMode_AirTaxi_c2c       = zeros(numbRows, numbCols,
numbOfIncomeGrps);
elseif TrainSwitch ~= 1 % Train = Yes
    c2cTrain_pTripTable             = zeros(numbRows, numbCols,
numbOfIncomeGrps);
    [numbRows_Train numbCols_Train] = size(S2S_Train_TravelTime);
    s2sTrain_pTripTable             = zeros(numbRows_Train, numbRows_Train,
numbOfIncomeGrps);
    c2cTrain_TravelTime_1W          = zeros(numbRows, numbCols);
    c2cTrain_TravelCost_2W          = zeros(numbRows, numbCols);
    totalHoursByMode_Train_c2c      = zeros(numbRows, numbCols,
numbOfIncomeGrps);
    totalCostByMode_Train_c2c      = zeros(numbRows, numbCols,
numbOfIncomeGrps);
else % No AirTaxi or Train
    a2aAirTaxi_pTripTable_Inc1 = [];
    a2aAirTaxi_pTripTable_Inc2 = [];
    a2aAirTaxi_pTripTable_Inc3 = [];
    a2aAirTaxi_pTripTable_Inc4 = [];
    a2aAirTaxi_pTripTable_Inc5 = [];
    s2sTrain_pTripTable     = [];
    c2cAirTaxi_TravelTime_1W   = [];
    c2cAirTaxi_TravelCost_2W   = [];
    c2cAirTaxi_pTripTable      = [];
    c2cTrain_TravelTime_1W  = [];
    c2cTrain_TravelCost_2W  = [];
    c2cTrain_pTripTable     = [];
    totalHoursByMode_AirTaxi_c2c = [];
    totalCostByMode_AirTaxi_c2c = [];
    totalHoursByMode_Train_c2c = [];
    totalCostByMode_Train_c2c = [];
end % if AirTaxiSwitch ~=1

%-----------------------------------------------------
% Start Mode Split
%-----------------------------------------------------
for orgCounty = 1 : numbRows % loop for origin counties
    orgCountyReal = orgCounty + starter - 1; % Add 'starter' to give correct
county number
```

```
    %     if mod(orgCountyReal, 50) == 0
    %         disp(['Origin county:', num2str(orgCountyReal), ' Time: ',
num2str(toc)])
    %     end % if mod(orgCountyReal, 5) == 0

    alaskaOrigin = isempty(find(orgCountyReal == 68:78));
    hawaiiOrigin = isempty(find(orgCountyReal == 527:530));

    if alaskaOrigin == 0 || hawaiiOrigin == 0
    else

        for desCounty = StateStartIndex : StateEndIndex %1 : numbCols % loop
for destination counties

            alaskaDestnt = isempty(find(desCounty == 68:78));
            hawaiiDestnt = isempty(find(desCounty == 527:530));
            forceProbCA  = 0;

            if alaskaDestnt == 0 || hawaiiDestnt == 0
            else
                % Extract annual county to county trips for each income group
[1x4]
                numbOfTripsByIncomeGrp = [CA_TripTable(1).trips(orgCounty,
desCounty) CA_TripTable(2).trips(orgCounty, desCounty) ...
                    CA_TripTable(3).trips(orgCounty, desCounty)
CA_TripTable(4).trips(orgCounty, desCounty) CA_TripTable(5).trips(orgCounty,
desCounty)];

                routeDistance = C2CDriveDist_Population_Centroids(orgCounty,
desCounty);

                if routeDistance >= 100 % skip computation if not an
intercity trip

                    TrvlTime_Auto_2way = c2cAuto_TravelTime_2way(orgCounty,
desCounty);
                    [max_Auto_cost_2way autoCostIndex] =
max(TrvlCost_Auto_2way(orgCounty, desCounty, :));

                    %---------------------------------
                    % Commercial Air: Travel Time and Cost
                    %---------------------------------
                    candArptMatrix = Selected_CA_Routes(:, :, orgCounty,
desCounty);
                    no_CA_Airport = 0;

                    if sum(sum(candArptMatrix)) == 0
                        no_CA_Airport = 1;
                    else
                        sum_candArptMatrix = sum(candArptMatrix,2);
                        zero_values = find(sum_candArptMatrix == 0);
                        candArptMatrix(zero_values,:) = [];

                        [candArptMatrix, TrvlTime_CA_2way, TrvlCost_CA_2way,
CAFareCost_CA, NumberOfValidRoutes] = ComputeCACostAndTime(candArptMatrix,
waitingAtOriginAirport_CA, ...
```

```
                            waitingAtDestntAirport_CA, TrvlTime_Auto_2way,
numbOfIncomeGrps, max_Auto_cost_2way, autoCostIndex);

                        %---------------------------------------------------
------------------
                        %-> candArptMatrix(1, :) = Origin County Airports
                        %-> candArptMatrix(2, :) = Destination County
Airports
                        %-> candArptMatrix(3, :) = Round trip Airport to
Airport travel time
                        %-> candArptMatrix(4, :) = Round trip Airport to
Airport Schedule delay
                        %-> candArptMatrix(5, :) = Access distance
                        %-> candArptMatrix(6, :) = Access Time (hours)
                        %-> candArptMatrix(7, :) = Egress distance
                        %-> candArptMatrix(8, :) = Egress Time (hours)
                        %---------------------------------------------------
------------------

                        if isempty(candArptMatrix)
                            no_CA_Airport = 1;
                        else
                            c2cCA_MinTravelTime_1W(orgCounty, desCounty) =
min(TrvlTime_CA_2way) / 2;
                            c2cCA_MaxTravelTime_1W(orgCounty, desCounty) =
max(TrvlTime_CA_2way) / 2;

                            c2cCA_MinTravelCost_2W(orgCounty, desCounty) =
min(min(TrvlCost_CA_2way));
                            c2cCA_MaxTravelCost_2W(orgCounty, desCounty) =
max(max(TrvlCost_CA_2way));
                        end % if isempty(candArptMatrix)
                    end % if sum(sum(candArptMatrix)) == 0

                    %---------------------------------
                    % AirTaxi: Travel Time and Cost
                    %---------------------------------
                    if AirTaxiSwitch ~= 1
                        [TrvlTime_AirTaxi_2way, TrvlCost_AirTaxi_2way,
AirTaxiValidMode, orgArptAirTaxi, destArptAirTaxi] =
ComputeAirTaxiCostAndTime(orgCountyReal, desCounty);
                        if AirTaxiValidMode == 1
                            c2cAirTaxi_TravelTime_1W(orgCounty, desCounty) =
(TrvlTime_AirTaxi_2way) / 2;
                            c2cAirTaxi_TravelCost_2W(orgCounty, desCounty) =
sum(TrvlCost_AirTaxi_2way)/numbOfIncomeGrps;
                        else
                            c2cAirTaxi_TravelTime_1W(orgCounty, desCounty) =
-999;
                            c2cAirTaxi_TravelCost_2W(orgCounty, desCounty) =
-999;
                        end % if AirTaxiValidMode == 1
                    end % if AirTaxiSwitch ==1

                    %---------------------------------
                    % TRAIN: Travel Time and Cost
                    %---------------------------------
                    if TrainSwitch ~= 1
                        [TrvlTime_Train_2way, TrvlCost_Train_2way,
orgStationTrain, destStationTrain, TrainValidMode] =
ComputeTrain_TimeCost(orgCountyReal, desCounty);
                        if TrainValidMode == 1
                            c2cTrain_TravelTime_1W(orgCounty, desCounty) =
(TrvlTime_Train_2way) / 2;
                            c2cTrain_TravelCost_2W(orgCounty, desCounty) =
mean(TrvlCost_Train_2way);
                        else
                            c2cTrain_TravelTime_1W(orgCounty, desCounty) = -
999;
                            c2cTrain_TravelCost_2W(orgCounty, desCounty) = -
999;
                        end % if TrainValidMode == 1
                    end % if TrainSwitch ~= 1
                    %--------------------------------

                    %--------------
                    % MODE CHOICE
                    %--------------
                    if sum(numbOfTripsByIncomeGrp) > 0 % Skip computation if
there are no trips

                        checkCA_trips       = [];
                        P_AirTaxi              = [];   P_Train
= [];
                        commAvTrips_Rt      = [];   commAvTrips_Rt_base   =
[];
                        modeTrips           = [];   modeTrips_base        =
[];
                        P_all               = [];   P_all_base            =
[];
                        P_Auto              = [];   P_Auto_base           =
[];
                        P_CA                = [];   P_CA_base             =
[];
                        P_CommAirRoute      = [];   P_CommAirRoute_base   =
[];  negativeDiff_Index_train = [];
                        tripDifference      = [];   tripDifference_AirTaxi
= [];  tripDifference_Train     = [];
                        negativeDiff_Index  = [];   negativeDiff_Index_ca =
[];  negativeDiff_Index_AirTaxi  = [];

                        %---------------------------------
                        % Nested Logit Function (Probability Calculations)
                        %---------------------------------
                        minDist_2 = 350;
                        if routeDistance < minDist_2
                            dist_Dummy = 1;
                        else
                            dist_Dummy = 0;
                        end % if routeDistance <= minDist_2

                        for j = 1 : numbOfIncomeGrps % delete inf

                            if numbOfTripsByIncomeGrp(j) ~= 0
```

```
                                if AirTaxiSwitch ~= 1 && AirTaxiValidMode ==
1
%                                         P_Auto = 1;
%                                         P_AirTaxi = 0;
%                                         P_CA    = 0;
%                                         modeTrips =
[numbOfTripsByIncomeGrp(j) 0 0];
%                                         modeTrips_base =
[numbOfTripsByIncomeGrp(j) 0];

                                        %changed
                                        cost_Auto =
TrvlCost_Auto_2way(orgCounty, desCounty, j);
                                        cost_CA    = TrvlCost_CA_2way(:, j);
                                        cost_AirTaxi =
TrvlCost_AirTaxi_2way(j);

                                        [P_Auto_base, P_CA_base,
P_CommAirRoute_base, P_all_base] = ...

nestedLogitFunction_AutoCA(TrvlTime_Auto_2way, TrvlTime_CA_2way, cost_Auto,
cost_CA, j, dist_Dummy); %, income_Dummy_file, income_dist_Dummy_file);
                                        [modeTrips_base, commAvTrips_Rt_base]
= roundTripsFnctn_AutoCA(numbOfTripsByIncomeGrp(j), P_Auto_base, P_CA_base,
P_CommAirRoute_base, forceProbCA);

                                        [P_Auto, P_CA, P_AirTaxi,
P_CommAirRoute, P_all] = ...

nestedLogitFunction_AutoCAAirTaxi(TrvlTime_Auto_2way, TrvlTime_CA_2way,
TrvlTime_AirTaxi_2way, cost_Auto, cost_CA, cost_AirTaxi, j, dist_Dummy); %,
income_Dummy_file, income_dist_Dummy_file);

                                        if P_AirTaxi == 0
                                            P_Auto          = P_Auto_base;
                                            P_CA            = P_CA_base;
                                            P_CommAirRoute =
P_CommAirRoute_base;
                                            P_all           = P_all_base;
                                            modeTrips       = [modeTrips_base
0];
                                            commAvTrips_Rt =
commAvTrips_Rt_base;
                                        else
                                            [modeTrips, commAvTrips_Rt] =
roundTripsFnctn_AutoCAAirTaxi(numbOfTripsByIncomeGrp(j), P_Auto, P_CA,
P_CommAirRoute, P_AirTaxi, forceProbCA, modeTrips_base, commAvTrips_Rt_base);
                                        end % if P_AirTaxi == 0

                                        % compare Base Case to Current Case
                                        tripDifference     = modeTrips_base -
modeTrips(1:2);
                                        negativeDiff_Index =
find(tripDifference < 0);
```

```
                                        % if trips more trips have been
assigned to modes that were present in the base case
                                        % force values back to original
                                        while ~isempty(negativeDiff_Index)
                                            % loopCounter(orgCounty, j) =
loopCounter(orgCounty, j) + 1;
                                            modeTrips                 =
modeTrips + [tripDifference 0];
                                            tripDifference            =
modeTrips_base - modeTrips(:, 1:2);
                                            negativeDiff_Index        =
find(tripDifference < 0);
                                        end % if isempty(negativeDifference)

                                        tripDiffernce_ca =
commAvTrips_Rt_base - commAvTrips_Rt;
                                        negativeDiff_Index_ca =
find(tripDiffernce_ca < 0);
                                        checkCA_trips = modeTrips_base(2) -
modeTrips(2);

                                        if ((checkCA_trips == 0) &&
~isempty(negativeDiff_Index_ca))
                                            commAvTrips_Rt = commAvTrips_Rt +
tripDiffernce_ca;
                                            negativeDiff_Index_ca =
find(tripDiffernce_ca < 0);
                                        end % if ((checkCA_trips == 0) &
~isempty(negativeDiff_Index_ca))

                                        negativeDiff_Index = 0;
                                        negativeDiff_Index_ca = 0;

                                        % check to make sure people switching
to AirTaxi are not losing travel time
                                        tripDifference_AirTaxi     =
modeTrips - [modeTrips_base 0];
                                        tripDifference          = -
tripDifference_AirTaxi;
                                        meanTT_CA               =
sum(TrvlTime_CA_2way) / NumberOfValidRoutes;
                                        ttAll                   =
[TrvlTime_Auto_2way meanTT_CA];
                                        ttDifference            = ttAll -
TrvlTime_AirTaxi_2way;
                                        negativeDiff_Index_AirTaxi =
find(ttDifference < 0);

                                        if ~isempty(tripDifference_AirTaxi)
                                            for checkIndex = 1 :
size(negativeDiff_Index_AirTaxi, 2)
                                                indexModeTrips =
negativeDiff_Index_AirTaxi(checkIndex);
                                                modeTrips(indexModeTrips) =
modeTrips(indexModeTrips) + tripDifference(indexModeTrips);
                                                modeTrips(3) = modeTrips(3) +
tripDifference_AirTaxi(indexModeTrips);
```

```
                                                end % for checkIndex = 1 :
size(negativeDiff_Index_AirTaxi, 2)
                                            end % if
~isempty(tripDifference_AirTaxi)

                                    c2cAirTaxi_pTripTable(orgCounty,
desCounty, j) = modeTrips(1, 3);  % AirTaxi trips
                                    totalHoursByMode_AirTaxi_c2c(orgCounty,
desCounty, j) = modeTrips(1, 3) * TrvlTime_AirTaxi_2way;
                                    totalCostByMode_AirTaxi_c2c(orgCounty,
desCounty, j) = modeTrips(1, 3) * TrvlCost_AirTaxi_2way(j);

                                    if j == 1

a2aAirTaxi_pTripTable_Inc1(orgArptAirTaxi, destArptAirTaxi) =
a2aAirTaxi_pTripTable_Inc1(orgArptAirTaxi, destArptAirTaxi) + modeTrips(1,
3);
                                    elseif j == 2

a2aAirTaxi_pTripTable_Inc2(orgArptAirTaxi, destArptAirTaxi) =
a2aAirTaxi_pTripTable_Inc2(orgArptAirTaxi, destArptAirTaxi) + modeTrips(1,
3);
                                    elseif j == 3

a2aAirTaxi_pTripTable_Inc3(orgArptAirTaxi, destArptAirTaxi) =
a2aAirTaxi_pTripTable_Inc3(orgArptAirTaxi, destArptAirTaxi) + modeTrips(1,
3);
                                    elseif j == 4

a2aAirTaxi_pTripTable_Inc4(orgArptAirTaxi, destArptAirTaxi) =
a2aAirTaxi_pTripTable_Inc4(orgArptAirTaxi, destArptAirTaxi) + modeTrips(1,
3);
                                    elseif j == 5

a2aAirTaxi_pTripTable_Inc5(orgArptAirTaxi, destArptAirTaxi) =
a2aAirTaxi_pTripTable_Inc5(orgArptAirTaxi, destArptAirTaxi) + modeTrips(1,
3);
                                    end % if j == 1
                                elseif TrainSwitch ~= 1 && TrainValidMode ==
1
                                    %changed
                                    cost_Auto = TrvlCost_Auto_2way(orgCounty,
desCounty, j);
                                    cost_CA    = TrvlCost_CA_2way(:, j);
                                    cost_Train = TrvlCost_Train_2way(j);

                                    [P_Auto, P_CA, P_Train, P_CommAirRoute,
P_all] = ...

nestedLogitFunction_AutoCATrain(TrvlTime_Auto_2way, TrvlTime_CA_2way,
TrvlTime_Train_2way, cost_Auto, cost_CA, cost_Train, j, routeDistance);

                                    [modeTrips, commAvTrips_Rt] =
roundTripsFnctn_AutoCATrain(numbOfTripsByIncomeGrp(j), P_Auto, P_CA,
P_CommAirRoute, P_Train, forceProbCA);
```

```
                                    c2cTrain_pTripTable(orgCounty, desCounty,
j) = modeTrips(1, 3);  % Train trips
                                    totalHoursByMode_Train_c2c(orgCounty,
desCounty, j) = modeTrips(1, 3) * TrvlTime_Train_2way;
                                    totalCostByMode_Train_c2c(orgCounty,
desCounty, j) = modeTrips(1, 3) * TrvlCost_Train_2way(j);
                                    s2sTrain_pTripTable(orgStationTrain,
destStationTrain, j) = s2sTrain_pTripTable(orgStationTrain, destStationTrain,
j) + modeTrips(1, 3);
                                else
                                    if no_CA_Airport == 1
                                        P_Auto = 1;
                                        P_AirTaxi = 0;
                                        P_CA    = 0;
                                        modeTrips =
[numbOfTripsByIncomeGrp(j) 0];
                                    else
                                        %Filter for short CA trips
                                        cost_Auto =
TrvlCost_Auto_2way(orgCounty, desCounty, j);
                                        cost_CA = TrvlCost_CA_2way(:, j);
                                        P_AirTaxi = 0;

                                        [P_Auto, P_CA, P_CommAirRoute, P_all]
= ...

nestedLogitFunction_AutoCA(TrvlTime_Auto_2way, TrvlTime_CA_2way, cost_Auto,
cost_CA, j, dist_Dummy); %, income_Dummy_file, income_dist_Dummy_file);

                                        [modeTrips, commAvTrips_Rt] =
roundTripsFnctn_AutoCA(numbOfTripsByIncomeGrp(j), P_Auto, P_CA,
P_CommAirRoute, forceProbCA);
                                    end % if no_CA_Airport == 1
                                end % if AirTaxiSwitch ==1

                                c2cAuto_pTripTable(orgCounty, desCounty, j) =
modeTrips(1, 1); % Auto trips
                                c2cCA_pTripTable(orgCounty, desCounty, j)    =
modeTrips(1, 2); % CA trips

                                %---------------------------------------------
-----------------
                                % Assign Total Hours Travelled and Airport to
Airport Tables
                                %---------------------------------------------
-----------------
                                totalHoursByMode_Auto_c2c(orgCounty,
desCounty, j) = modeTrips(1, 1) * TrvlTime_Auto_2way;
                                totalCostByMode_Auto_c2c(orgCounty,
desCounty, j) = modeTrips(1, 1) * TrvlCost_Auto_2way(orgCounty, desCounty,
j);

                                if P_CA ~= 0
                                    a2aTrips_CA = [];
                                    a2aTrips_CA(:,1) = commAvTrips_Rt(1,:); %
Transpose
```

```
                                    totalHoursByMode_CA_c2c(orgCounty,
desCounty, j) = sum(a2aTrips_CA .* TrvlTime_CA_2way);
                                    totalCostByMode_CA_c2c(orgCounty,
desCounty, j)  = sum(a2aTrips_CA .* TrvlCost_CA_2way(:, j));
                                    c2cCA_RevenueTable(orgCounty, desCounty,
j)  = sum(a2aTrips_CA .* CAFareCost_CA(:, j));

                                    for k = 1 : NumberOfValidRoutes
                                        orgArprt = candArptMatrix(k, 1);
                                        destArprt = candArptMatrix(k, 2);
                                        %if isnan(a2aTrips_CA(k));
orgCountyReal; desCounty; pause; end
                                        a2aCA_pTripTable(orgArprt, destArprt,
j) = a2aCA_pTripTable(orgArprt, destArprt, j) + a2aTrips_CA(k);
                                        a2aCA_RevenueTable(orgArprt,
destArprt, j) = a2aCA_RevenueTable(orgArprt, destArprt, j) +
averageFare_CA(j, orgArprt, destArprt) * a2aTrips_CA(k);
                                    end % for k = 1 : size(airportMatrix, 2)
                                end % if P_CA == 0;
                            end %if numbOfTripsByIncomeGrp(j) ~= 0
                        end % for j = 1 : numbOfIncomeGrps % delete inf
values
                    end %if sum(numbOfTripsByIncomeGrp) > 0
                end %if distC2C_symm_popCentd(orgCounty_symm, desCounty_symm)
>= 100 % skip computation if not an intercity trip
            end % if alaskaDestnt == 1 | hawaiiDestnt == 1
        end % for desCounty = 1 : numbCols % loop for destination counties
    end % if alaskaOrigin == 1 | hawaiiOrigin == 1
end % for orgCounty = 1 : orgCntfor

c2cAuto_TravelTime_1W = c2cAuto_TravelTime_2way/2;

save (strcat(saveProjDir, '\c2cAuto_pTripTable_', stateName, '.mat'),
'c2cAuto_pTripTable');
save (strcat(saveProjDir, '\c2cCA_pTripTable_', stateName, '.mat'),
'c2cCA_pTripTable');
save (strcat(saveProjDir, '\a2aCA_pTripTable_', stateName, '.mat'),
'a2aCA_pTripTable');
save (strcat(saveProjDir, '\a2aCA_RevenueTable_', stateName, '.mat'),
'a2aCA_RevenueTable');
save (strcat(saveProjDir, '\totalHoursByMode_Auto_c2c_', stateName, '.mat'),
'totalHoursByMode_Auto_c2c');
save (strcat(saveProjDir, '\totalCostByMode_Auto_c2c_', stateName, '.mat'),
'totalCostByMode_Auto_c2c');
save (strcat(saveProjDir, '\totalHoursByMode_CA_c2c_', stateName, '.mat'),
'totalHoursByMode_CA_c2c');
save (strcat(saveProjDir, '\totalCostByMode_CA_c2c_', stateName, '.mat'),
'totalCostByMode_CA_c2c');
save (strcat(saveProjDir, '\c2cCA_RevenueTable_', stateName, '.mat'),
'c2cCA_RevenueTable');
save (strcat(saveProjDir, '\c2cAuto_TravelTime_1W_', stateName, '.mat'),
'c2cAuto_TravelTime_1W');
save (strcat(saveProjDir, '\c2cAuto_TravelCost_2W_', stateName, '.mat'),
'c2cAuto_TravelCost_2W');
save (strcat(saveProjDir, '\c2cCA_MinTravelTime_1W_', stateName, '.mat'),
'c2cCA_MinTravelTime_1W');
```

```
save (strcat(saveProjDir, '\c2cCA_MinTravelCost_2W_', stateName, '.mat'),
'c2cCA_MinTravelCost_2W');
save (strcat(saveProjDir, '\c2cCA_MaxTravelTime_1W_', stateName, '.mat'),
'c2cCA_MaxTravelTime_1W');
save (strcat(saveProjDir, '\c2cCA_MaxTravelCost_2W_', stateName, '.mat'),
'c2cCA_MaxTravelCost_2W');
clear c2cAuto_pTripTable;
clear c2cCA_pTripTable;
clear a2aCA_pTripTable;
clear totalHoursByMode_Auto_c2c;
clear totalCostByMode_Auto_c2c;
clear totalHoursByMode_CA_c2c;
clear totalCostByMode_CA_c2c;
clear c2cCA_RevenueTable;
clear c2cAuto_TravelTime_1W;
clear c2cAuto_TravelCost_2W;
clear c2cCA_MinTravelTime_1W;
clear c2cCA_MinTravelCost_2W;
clear c2cCA_MaxTravelTime_1W;
clear c2cCA_MaxTravelCost_2W;
clear CA_TripTable;

if AirTaxiSwitch ~= 1
    save (strcat(saveProjDir, '\c2cAirTaxi_pTripTable_', stateName, '.mat'),
'c2cAirTaxi_pTripTable');
    save (strcat(saveProjDir, '\a2aAirTaxi_pTripTable_Inc1_', stateName,
'.mat'), 'a2aAirTaxi_pTripTable_Inc1');
    save (strcat(saveProjDir, '\a2aAirTaxi_pTripTable_Inc2_', stateName,
'.mat'), 'a2aAirTaxi_pTripTable_Inc2');
    save (strcat(saveProjDir, '\a2aAirTaxi_pTripTable_Inc3_', stateName,
'.mat'), 'a2aAirTaxi_pTripTable_Inc3');
    save (strcat(saveProjDir, '\a2aAirTaxi_pTripTable_Inc4_', stateName,
'.mat'), 'a2aAirTaxi_pTripTable_Inc4');
    save (strcat(saveProjDir, '\a2aAirTaxi_pTripTable_Inc5_', stateName,
'.mat'), 'a2aAirTaxi_pTripTable_Inc5');
    save (strcat(saveProjDir, '\totalHoursByMode_AirTaxi_c2c_', stateName,
'.mat'), 'totalHoursByMode_AirTaxi_c2c');
    save (strcat(saveProjDir, '\totalCostByMode_AirTaxi_c2c_', stateName,
'.mat'), 'totalCostByMode_AirTaxi_c2c');
    save (strcat(saveProjDir, '\c2cAirTaxi_TravelTime_1W_', stateName,
'.mat'), 'c2cAirTaxi_TravelTime_1W');
    save (strcat(saveProjDir, '\c2cAirTaxi_TravelCost_2W_', stateName,
'.mat'), 'c2cAirTaxi_TravelCost_2W');
    clear c2cAirTaxi_pTripTable;
    clear a2aAirTaxi_pTripTable_Inc1;
    clear a2aAirTaxi_pTripTable_Inc2;
    clear a2aAirTaxi_pTripTable_Inc3;
    clear a2aAirTaxi_pTripTable_Inc4;
    clear a2aAirTaxi_pTripTable_Inc5;
    clear totalHoursByMode_AirTaxi_c2c;
    clear totalCostByMode_AirTaxi_c2c;
    clear c2cAirTaxi_TravelTime_1W;
    clear c2cAirTaxi_TravelCost_2W;
end % strcmp(AirTaxiCase_YesNo, 'Yes')  == 1

if TrainSwitch ~= 1
```

```
    save (strcat(saveProjDir, '\c2cTrain_pTripTable_', stateName, '.mat'), 
'c2cTrain_pTripTable');
    save (strcat(saveProjDir, '\s2sTrain_pTripTable_', stateName, '.mat'), 
's2sTrain_pTripTable');
    save (strcat(saveProjDir, '\totalHoursByMode_Train_c2c_', stateName, 
'.mat'), 'totalHoursByMode_Train_c2c');
    save (strcat(saveProjDir, '\totalCostByMode_Train_c2c_', stateName, 
'.mat'), 'totalCostByMode_Train_c2c');
    save (strcat(saveProjDir, '\c2cTrain_TravelTime_1W_', stateName, '.mat'), 
'c2cTrain_TravelTime_1W');
    save (strcat(saveProjDir, '\c2cTrain_TravelCost_2W_', stateName, '.mat'), 
'c2cTrain_TravelCost_2W');
    clear c2cTrain_pTripTable;
    clear a2aTrain_pTripTable;
    clear totalHoursByMode_Train_c2c;
    clear totalCostByMode_Train_c2c;
    clear c2cTrain_TravelTime_1W;
    clear c2cTrain_TravelCost_2W;
end % strcmp(AirTaxiCase_YesNo, 'Yes')  == 1

return;
```

---------------------------------------------------------------------------------------------------------------------------------------------

#### 7.11.2.1 Sub-Function to Compute Auto Cost and Time

----------------------------------------------------------------------------------------------------------------------------------------

```
% This m-file pre processes the car cost and time

function [c2cAuto_TravelTime_2way, c2cAuto_TravelCost_2way,
TrvlCost_Auto_2way] = ComputeAutoCostAndTime(income, TripPurpose)

global maxDailyDriveTime_Auto additionalDay_Auto costPerMile_Auto
avgOccupancy_Auto oneDayLodgingCost C2CDriveTime_Population_Centroids
C2CDriveDist_Population_Centroids

% Auto DRIVE TIME
%-----------------
% numberOfDays         = floor(C2CDriveTime_Population_Centroids /
% maxDailyDriveTime_Auto);
numberOfDays_2way       = max(0, (C2CDriveTime_Population_Centroids * 2 /
maxDailyDriveTime_Auto) - 1);
additionalDays_2way     = numberOfDays_2way * additionalDay_Auto;
c2cAuto_TravelTime_2way = C2CDriveTime_Population_Centroids * 2 +
additionalDays_2way;

AutoTravelCost_2way(:,:,1) = (C2CDriveDist_Population_Centroids * 2 *
costPerMile_Auto) / avgOccupancy_Auto(1);
AutoTravelCost_2way(:,:,2) = (C2CDriveDist_Population_Centroids * 2 *
costPerMile_Auto) / avgOccupancy_Auto(2);
AutoTravelCost_2way(:,:,3) = (C2CDriveDist_Population_Centroids * 2 *
costPerMile_Auto) / avgOccupancy_Auto(3);
AutoTravelCost_2way(:,:,4) = (C2CDriveDist_Population_Centroids * 2 *
costPerMile_Auto) / avgOccupancy_Auto(4);
AutoTravelCost_2way(:,:,5) = (C2CDriveDist_Population_Centroids * 2 *
costPerMile_Auto) / avgOccupancy_Auto(5);

if strcmp(TripPurpose, 'Business') == 1
    lodgingCost(:,:,1)  = numberOfDays_2way * oneDayLodgingCost(1);
    lodgingCost(:,:,2)  = numberOfDays_2way * oneDayLodgingCost(2);
    lodgingCost(:,:,3)  = numberOfDays_2way * oneDayLodgingCost(3);
    lodgingCost(:,:,4)  = numberOfDays_2way * oneDayLodgingCost(4);
    lodgingCost(:,:,5)  = numberOfDays_2way * oneDayLodgingCost(5);
else
    lodgingCost(:,:,1)  = numberOfDays_2way * oneDayLodgingCost(1) /
avgOccupancy_Auto(1);
    lodgingCost(:,:,2)  = numberOfDays_2way * oneDayLodgingCost(2) /
avgOccupancy_Auto(2);
    lodgingCost(:,:,3)  = numberOfDays_2way * oneDayLodgingCost(3) /
avgOccupancy_Auto(3);
    lodgingCost(:,:,4)  = numberOfDays_2way * oneDayLodgingCost(4) /
avgOccupancy_Auto(4);
    lodgingCost(:,:,5)  = numberOfDays_2way * oneDayLodgingCost(5) /
avgOccupancy_Auto(5);
end % if TripPurpose == 1
```

```
TrvlCost_Auto_2way  = AutoTravelCost_2way + lodgingCost;  % round trip cost
in dollars

% relTrvlCost_Auto(:,:,1) = TrvlCost_Auto_2way(:,:,1) * 100 / income(1); %
scale up by 100
% relTrvlCost_Auto(:,:,2) = TrvlCost_Auto_2way(:,:,2) * 100 / income(2); %
scale up by 100
% relTrvlCost_Auto(:,:,3) = TrvlCost_Auto_2way(:,:,3) * 100 / income(3); %
scale up by 100
% relTrvlCost_Auto(:,:,4) = TrvlCost_Auto_2way(:,:,4) * 100 / income(4); %
scale up by 100
% relTrvlCost_Auto(:,:,5) = TrvlCost_Auto_2way(:,:,5) * 100 / income(5); %
scale up by 100

c2cAuto_TravelCost_2way = mean(TrvlCost_Auto_2way, 3);
```

--------------------------------------------------------------------------------------------------------------------------------------------
------------

#### 7.11.2.2 Compute Commercial Air Travel Time and Costs (all Routes)

------------------------------------------------------------------------------------------------------------------------
------------

```
% This function computer the Commercial Airline Travel Time and Cost

function [candArptMatrix, TrvlTime_CA_2way, tripCost_CA, CAFareCost_CA,
NumberOfValidRoutes] = ComputeCACostAndTime(candArptMatrix,
waitingAtOriginAirport_CA, ...
    waitingAtDestntAirport_CA, TrvlTime_Auto_2way, numbOfIncomeGrps,
max_Auto_cost_2way, autoCostIndex)

global averageFare_CA CA_Hub_Type avgOccupancy_Auto costPerMile_Auto
AirlineFlyingTime_ScalingFactor
global minRoundTripTime_CA maxRoundTripTime_CA CA_dailyOvernightTime
oneDayLodgingCost

%-------------------------------------------------------------------
%-> candArptMatrix(1, :) = Origin County Airports
%-> candArptMatrix(2, :) = Destination County Airports
%-> candArptMatrix(3, :) = Round trip Airport to Airport travel time
%-> candArptMatrix(4, :) = Round trip Airport to Airport Schedule delay
%-> candArptMatrix(5, :) = Access distance
%-> candArptMatrix(6, :) = Access Time (hours)
%-> candArptMatrix(7, :) = Egress distance
%-> candArptMatrix(8, :) = Egress Time (hours)
%-------------------------------------------------------------------

NumberOfRoutes = size(candArptMatrix, 1);

TrvlTime_CA_2way = zeros(NumberOfRoutes,1);
tripCost_CA      = zeros(NumberOfRoutes,numbOfIncomeGrps);
CAFareCost_CA    = zeros(NumberOfRoutes,numbOfIncomeGrps);

% Calculate full CA travel time
for i = 1 : NumberOfRoutes

    CA_Origin_Airport      = candArptMatrix(i, 1);
    CA_Destination_Airport = candArptMatrix(i, 2);

    if CA_Origin_Airport == CA_Destination_Airport
    else
        %Outbound processing time
        Outbound_Processing_Time =
waitingAtOriginAirport_CA(CA_Hub_Type(CA_Origin_Airport)) +
waitingAtDestntAirport_CA(CA_Hub_Type(CA_Destination_Airport));
        Return_Processing_Time   =
waitingAtOriginAirport_CA(CA_Hub_Type(CA_Destination_Airport)) +
waitingAtDestntAirport_CA(CA_Hub_Type(CA_Origin_Airport));
        Access_Egress_Time_1Way  = candArptMatrix(i, 6) + candArptMatrix(i,
8);
        travelTime_CA_with_SD_2way = Access_Egress_Time_1Way * 2  +
Outbound_Processing_Time + candArptMatrix(i, 4) + candArptMatrix(i,3) *
AirlineFlyingTime_ScalingFactor + Return_Processing_Time;
```

```
        % overnight time and costs
        CA_numberOfNights = 0;

        if travelTime_CA_with_SD_2way > minRoundTripTime_CA &&
travelTime_CA_with_SD_2way <= maxRoundTripTime_CA
            rampWidth_x       = maxRoundTripTime_CA - minRoundTripTime_CA;
            rampHeight_y      = 1;
            travelTime_x      = travelTime_CA_with_SD_2way -
minRoundTripTime_CA;
            CA_numberOfNights = rampHeight_y * (travelTime_x / rampWidth_x);
        elseif travelTime_CA_with_SD_2way > maxRoundTripTime_CA &&
travelTime_CA_with_SD_2way < 20 % two night stay
            rampWidth_x       = 20 - maxRoundTripTime_CA;
            rampHeight_y      = 1;
            travelTime_x      = travelTime_CA_with_SD_2way -
maxRoundTripTime_CA;
            CA_numberOfNights = rampHeight_y * (travelTime_x / rampWidth_x);
            CA_numberOfNights = CA_numberOfNights + 1;
        elseif travelTime_CA_with_SD_2way > 20
            CA_numberOfNights = 2;
        end % if CA_travelTime_roundTrip > minRoundTripTime_CA &&

        TrvlTime_CA_2way(i, 1) = travelTime_CA_with_SD_2way +
CA_numberOfNights * CA_dailyOvernightTime;

        %Calculate Access and Egress Costs
        accessCost_1way = candArptMatrix(i, 5) * costPerMile_Auto ./
avgOccupancy_Auto;
        if candArptMatrix(i, 8) <= 1 % Less than 1 hour
            egressCost_1way = (3 + candArptMatrix(i, 7)) ./
avgOccupancy_Auto; % assume taxi is used and taxi costs $3 + $1/mile
        else
            egressCost_1way = 50 ./ avgOccupancy_Auto; % Assume car rental
        end
        accessEgressCost_2way = 2 * (accessCost_1way + egressCost_1way);

        CA_Origin_Airport      = candArptMatrix(i, 1);
        CA_Destination_Airport = candArptMatrix(i, 2);
        CA_costPerNight = CA_numberOfNights * oneDayLodgingCost;
        % based on the weigthed fares compute travel cost
        for j = 1:numbOfIncomeGrps
            tripCost_CA(i, j)   = CA_costPerNight(j) + averageFare_CA(j,
CA_Origin_Airport, CA_Destination_Airport) + accessEgressCost_2way(j);
            CAFareCost_CA(i, j) = averageFare_CA(j, CA_Origin_Airport,
CA_Destination_Airport);
        end % for j = 1:numbOfIncomeGrps
    end % if CA_Origin_Airport == CA_Destination_Airport

end % for i = 1 : NumberOfRoutes

% nonZeroIndex      = find(TrvlTime_CA_2way);
% TrvlTime_CA_2way  = TrvlTime_CA_2way(nonZeroIndex);
% candArptMatrix    = candArptMatrix(nonZeroIndex, :);
% tripCost_CA       = tripCost_CA(nonZeroIndex, :);
% CAFareCost_CA     = CAFareCost_CA(nonZeroIndex, :);
% Make sure there is at least one valid CA route
TrvlTime_CA_2way = round(TrvlTime_CA_2way * 10) / 10;
Valid_CA_Routes  = find(TrvlTime_CA_2way <= TrvlTime_Auto_2way);

scaleFactor      = 1.3;
while isempty(Valid_CA_Routes);
    Valid_CA_Routes  = find(TrvlTime_CA_2way <= TrvlTime_Auto_2way *
scaleFactor);
    scaleFactor      = scaleFactor + 0.2;
end % while isempty(candArptMatrix);
TrvlTime_CA_2way  = TrvlTime_CA_2way(Valid_CA_Routes);
candArptMatrix    = candArptMatrix(Valid_CA_Routes, :);
tripCost_CA       = tripCost_CA(Valid_CA_Routes, :);
CAFareCost_CA     = CAFareCost_CA(Valid_CA_Routes, :);

NumberOfValidRoutes = length(Valid_CA_Routes);

if scaleFactor > 1.3
    CA_Auto_time_ratio = TrvlTime_CA_2way./TrvlTime_Auto_2way;
    CA_Auto_cost_ratio = tripCost_CA(:,autoCostIndex)./max_Auto_cost_2way;
    deleteIndex = [];
    for k = 1 : length(CA_Auto_time_ratio)
        if CA_Auto_time_ratio(k) > 2 & CA_Auto_cost_ratio(k) > 2
            deleteIndex(k) = k;
        end % if CA_Auto_time_ratio(k) > 2 & CA_Auto_cost_ratio(k) > 2
    end % for k = 1 : length(CA_Auto_time_ratio)
    if isempty(deleteIndex)
    else
        TrvlTime_CA_2way(nonzeros(deleteIndex))  = [];
        candArptMatrix(nonzeros(deleteIndex),:)  = [];
        tripCost_CA(nonzeros(deleteIndex),:)     = [];
        CAFareCost_CA(nonzeros(deleteIndex),:)   = [];
    end % if isempty(deleteIndex)
    NumberOfValidRoutes = length(TrvlTime_CA_2way);
end % if scaleFactor > 1.3

-----------------------------------------------------------------------------------------------------------------
-----------
```

### 7.11.2.3 Sub-Function to Compute Air Taxi Travel Time and Costs

------------------------------------------------------------------------------------------------------------------------------------

```
% This function computer the AirTaxi Travel Time and Cost

function [TrvlTime_AirTaxi_2way, AirTaxiCost_2way, AirTaxiValidMode,
orgArptAirTaxi, destArptAirTaxi] = ComputeAirTaxiCostAndTime(orgCountyReal,
desCounty)

global avgOccupancy_Auto costPerMile_Auto costPerMile_AirTaxi c2aData_AirTaxi
distA2A_AirTaxi
global drivingSpeed_Urban drivingSpeed_Rural maxAccessTime_AirTaxi
maxEgressTime_AirTaxi min_A2A_Distance_AirTaxi
global msaIndicatorData waitingAtOrginAirport_AirTaxi
waitingAtDestAirport_AirTaxi  scheduleDelay_AirTaxi
global max_FSP_Distance_AirTaxi min_FSP_Distance_AirTaxi
max_FSP_Speed_AirTaxi min_FSP_Speed_AirTaxi FlightSpeedProfile
global min_CP_Distance_AirTaxi max_CP_Distance_AirTaxi min_CP_Cost_AirTaxi
max_CP_Cost_AirTaxi CostProfile ConstantAirTaxiCost

%------------------------------------------------
%1) create business trip data for logit model
%------------------------------------------------
orgArptAirTaxi  = c2aData_AirTaxi(orgCountyReal, 2);
destArptAirTaxi = c2aData_AirTaxi(desCounty, 2);
%ACCESS, EGRESS TIME
%----------------------
% read access & eggress time from table
accessTime  = c2aData_AirTaxi(orgCountyReal, 4);  %hr
egressTime  = c2aData_AirTaxi(desCounty, 4); %hr
Dist_Linehaul_AirTaxi = distA2A_AirTaxi(orgArptAirTaxi, destArptAirTaxi);
%tripDist1way

AirTaxiValidMode = 1; %1 = AirTaxi is a Valid Mode
%AirTaxi not valid if the access and egresss time violate these conditions
if orgArptAirTaxi == destArptAirTaxi || accessTime > maxAccessTime_AirTaxi ||
egressTime > maxEgressTime_AirTaxi || Dist_Linehaul_AirTaxi <=
min_A2A_Distance_AirTaxi %Not valid if the org and dest airport are the same
    AirTaxiValidMode      = 0;
    TrvlTime_AirTaxi_2way = [];
    AirTaxiCost_2way      = [];
    orgArptAirTaxi        = [];
    destArptAirTaxi       = [];
else
    % MSA to MSA
    if msaIndicatorData(orgCountyReal, 2) == 1 && msaIndicatorData(desCounty,
2) == 1
        drivingSpeed_access        = drivingSpeed_Urban;
        drivingSpeed_egress        = drivingSpeed_Urban;
        % MSA to NON-MSA
    elseif (msaIndicatorData(orgCountyReal, 2) == 1 &&
msaIndicatorData(desCounty, 2) == 0)
        drivingSpeed_access        = drivingSpeed_Urban;
        drivingSpeed_egress        = drivingSpeed_Rural;
        % NON-MSA to MSA
    elseif (msaIndicatorData(orgCountyReal, 2) == 0 &&
msaIndicatorData(desCounty, 2) == 1)
        drivingSpeed_access        = drivingSpeed_Rural;
        drivingSpeed_egress        = drivingSpeed_Urban;
        % NON-MSA to NON-MSA
    elseif (msaIndicatorData(orgCountyReal, 2) == 0 &&
msaIndicatorData(desCounty, 2) == 0)
        drivingSpeed_access        = drivingSpeed_Rural;
        drivingSpeed_egress        = drivingSpeed_Rural;
    end % if msaIndicatorData(orgCounty_symm) = 1 &
msaIndicatorData(desCounty_symm) == 1


    %Synthesize travel cost and time
    %---------------------------------
    % 1way Travel Time AirTaxi
    %-------------------------

    % IN-VEHICLE TIME
    %---------------------
    if Dist_Linehaul_AirTaxi <= min_FSP_Distance_AirTaxi
%min(FlightSpeedProfile(2, :))
        TSpeed  = min_Speed_AirTaxi; %min(FlightSpeedProfile(1, :));
    elseif Dist_Linehaul_AirTaxi >= max_FSP_Distance_AirTaxi
%max(FlightSpeedProfile(2, :))
        TSpeed  = max_FSP_Speed_AirTaxi; %max(FlightSpeedProfile(1, :));
    else
        A = find(FlightSpeedProfile(2,:) >= Dist_Linehaul_AirTaxi);
        X1 = FlightSpeedProfile(1,A(1)-1);  %Speed
        X2 = FlightSpeedProfile(1,A(1));    %Speed
        Y1 = FlightSpeedProfile(2,A(1)-1);  %Distance
        Y2 = FlightSpeedProfile(2,A(1));    %Distance
        TSpeed = (X2 - X1)/(Y2 - Y1)*(Dist_Linehaul_AirTaxi - Y1) + X1;
    end % if Dist_Linehaul_AirTaxi <= min_Distance_AirTaxi
%min(FlightSpeedProfile(2, :))

    if  (orgArptAirTaxi == destArptAirTaxi) && TSpeed == 0
        TT_Linehaul_AirTaxi = 15; %Increase AirTaxi travel time irrationally
    else
        TT_Linehaul_AirTaxi   = Dist_Linehaul_AirTaxi / TSpeed; %hour
    end % if  (orgArptAirTaxi ~= destArptAirTaxi) & TSpeed == 0

    if accessTime < 0
        accessTime  = c2aData_AirTaxi(orgCountyReal, 3) * 1.2 /
drivingSpeed_access;
    end % if accessTime_hours < 0

    if egressTime < 0
        egressTime  = c2aData_AirTaxi(desCounty, 3) * 1.2 /
drivingSpeed_egress;
    end % if egressTime_hours < 0

    accessEgressTime_1way = accessTime + egressTime;
    AirTaxiWaitingTime_1way  = waitingAtOrginAirport_AirTaxi +
waitingAtDestAirport_AirTaxi;
```

```
    TrvlTime_AirTaxi_2way        = 2 * (accessEgressTime_1way +
AirTaxiWaitingTime_1way + scheduleDelay_AirTaxi + TT_Linehaul_AirTaxi); %hr


    %-----------------------------
    % 1way Travel Cost AirTaxi
    %-----------------------------
    % In addition, if one way travel time is more than 4hr, we assume that
the traveler
    % cannot make the trip in a day. Thus we add one day logding at the
destination (6/5/03, Hojong)
    if ConstantAirTaxiCost == 1
        AirTaxiTravelCost_1way = Dist_Linehaul_AirTaxi * costPerMile_AirTaxi;
%Unit: Dollar
    else
        if Dist_Linehaul_AirTaxi <= min_CP_Distance_AirTaxi
%min(FlightSpeedProfile(2, :))
            AirTaxiTravelCost_1way  = min_CP_Cost_AirTaxi *
Dist_Linehaul_AirTaxi; %min(FlightSpeedProfile(1, :));
        elseif Dist_Linehaul_AirTaxi >= max_FSP_Distance_AirTaxi
%max(FlightSpeedProfile(2, :))
            AirTaxiTravelCost_1way  = max_CP_Cost_AirTaxi *
Dist_Linehaul_AirTaxi; %max(FlightSpeedProfile(1, :));
        else
            A = find(CostProfile(:, 2) >= Dist_Linehaul_AirTaxi);
            X1 = CostProfile(A(1)-1, 3);  %Cost
            X2 = CostProfile(A(1), 3);    %Cost
            Y1 = CostProfile(A(1)-1, 2);  %Distance
            Y2 = CostProfile(A(1), 2);    %Distance
            AirTaxiTravelCost_1way = ((X2 - X1)/(Y2 -
Y1)*(Dist_Linehaul_AirTaxi - Y1) + X1) * Dist_Linehaul_AirTaxi;
        end % if Dist_Linehaul_AirTaxi <= min_Distance_AirTaxi
%min(FlightSpeedProfile(2, :))
    end

    % read access & eggress cost from table
    accessCost         = c2aData_AirTaxi(orgCountyReal, 3) * costPerMile_Auto
./ avgOccupancy_Auto;  % Distance x cost/mile
    if egressTime <= 1
        egressCost       = (3 + c2aData_AirTaxi(desCounty, 3)) ./
avgOccupancy_Auto;  %Taxi
    else
        egressCost         = 50 ./ avgOccupancy_Auto; % Unit: Dollar % Rented
Car
    end
    accessEgressCost   = accessCost + egressCost;

    AirTaxiCost_2way      = 2 * (accessEgressCost + AirTaxiTravelCost_1way);
% Unit: Dollar

end % if orgArptAirTaxi == destArptAirTaxi %Not valid if the org and dest
airport are the same

return;
```

------------------------------------------------------------------------------------------------------------------------------------
------------

#### 7.11.2.4 Sub-Function to Compute Train Travel Time and Costs

---------------------------------------------------------------------------------------------------------------------------------------------------------------

```
function [TrainTime_2way, TrainCost_2way, orgStation, destStation,
TrainValidMode] = computeTrain_TimeCost(orgCountyReal, desCounty)

% Global variables
global additionalDay_Train avgOccupancy_Auto maxAccessTime_Train
maxEgressTime_Train
global costPerMile_Auto costPerMile_Train oneDayLodgingCost C2SData_Train
global drivingSpeed_Urban drivingSpeed_Rural msaIndicatorData
global waitingAtOrginStation_Train waitingAtDestStation_Train
global S2S_Train_TravelCost S2S_Train_TravelTime S2S_Train_ScheduleDelay

% MSA to MSA
if msaIndicatorData(orgCountyReal, 2) == 1 & msaIndicatorData(desCounty, 2)
== 1
    drivingSpeed_access        = drivingSpeed_Urban;
    drivingSpeed_egress        = drivingSpeed_Urban;
% MSA to NON-MSA
elseif (msaIndicatorData(orgCountyReal, 2) == 1 & msaIndicatorData(desCounty,
2) == 0)
    drivingSpeed_access        = drivingSpeed_Urban;
    drivingSpeed_egress        = drivingSpeed_Rural;
% NON-MSA to MSA
elseif (msaIndicatorData(orgCountyReal, 2) == 0 & msaIndicatorData(desCounty,
2) == 1)
    drivingSpeed_access        = drivingSpeed_Rural;
    drivingSpeed_egress        = drivingSpeed_Urban;
% NON-MSA to NON-MSA
elseif (msaIndicatorData(orgCountyReal, 2) == 0 & msaIndicatorData(desCounty,
2) == 0)
    drivingSpeed_access        = drivingSpeed_Rural;
    drivingSpeed_egress        = drivingSpeed_Rural;
end % if msaIndicatorData(orgCounty_symm) = 1 &
msaIndicatorData(desCounty_symm) == 1


% Find origin and destination stations
orgStation  = C2SData_Train(orgCountyReal, 2);
destStation = C2SData_Train(desCounty, 2);

% Find travel time and travle cost between stations
Station2Station_TravelCost_2way = S2S_Train_TravelCost(orgStation,
destStation) + S2S_Train_TravelCost(destStation, orgStation);
Station2Station_TravelTime_2way = S2S_Train_TravelTime(orgStation,
destStation) + S2S_Train_TravelTime(destStation, orgStation);
Station2Station_ScheduleDelay_2way = S2S_Train_ScheduleDelay(orgStation,
destStation)+ S2S_Train_ScheduleDelay(destStation, orgStation);

%ACCESS, EGRESS TIME
%----------------------
% read access & eggress time from table
```

```
accessTime  = C2SData_Train(orgCountyReal, 4);  %hr
egressTime  = C2SData_Train(desCounty, 4); %hr
accessDistance = C2SData_Train(orgCountyReal, 3);
egressDistance = C2SData_Train(desCounty, 3);

if accessTime < 0
    accessTime  = C2SData_Train(orgCountyReal, 3) * 1.2 /
drivingSpeed_access;
end % if accessTime_hours < 0

if egressTime < 0
    egressTime  = C2SData_Train(desCounty, 3) * 1.2 / drivingSpeed_egress;
end % if egressTime_hours < 0

accessEgressTime_1way = accessTime + egressTime;
TrainWaitingTime_1way  = waitingAtOrginStation_Train +
waitingAtDestStation_Train;

TrainTime_2way        = 2 * (accessEgressTime_1way + TrainWaitingTime_1way) +
Station2Station_TravelTime_2way + Station2Station_ScheduleDelay_2way; %hr

%----------------------------
% 1way Travel Cost Train
%----------------------------
% In addition, if one way travel time is more than 4hr, we assume that the
traveler
% cannot make the trip in a day. Thus we add one day logding at the
destination (6/5/03, Hojong)

% read access & eggress cost from table
accessCost         = C2SData_Train(orgCountyReal, 3) * costPerMile_Auto ./
avgOccupancy_Auto;  % Distance x cost/mile
if egressTime <= 1
    egressCost       = (3 + C2SData_Train(desCounty, 3)) ./
avgOccupancy_Auto;  %Taxi
else
    egressCost         = 50 ./ avgOccupancy_Auto; % Unit: Dollar % Rented Car
end
accessEgressCost   = accessCost + egressCost;

TrainCost_2way      = 2 * accessEgressCost + Station2Station_TravelCost_2way;
% Unit: Dollar

% Test if Train is a valid mode
TrainValidMode = 1; %1 = SAB Technology is a Valid Mode
if accessTime > maxAccessTime_Train || egressTime > maxEgressTime_Train
    TrainValidMode = 0;
end
if orgStation == destStation
    TrainValidMode = 0;
end
```

---------------------------------------------------------------------------------------------------------------------------------------------

7.11.2.5 Sub-Function to Compute Probability of Choosing Automobile & Commercial Air

```
--------------------------------------------------------------------------------------------------------------------
------------

function [P_Auto, P_CommAir, P_CommAirRoute, P_all] =
nestedLogitFunction_AutoCA(tTime_Auto, tTime_comm,  ...
    rCost_Auto, rCost_comm, incomeGrp, dist_Dummy)

% called by createmodechoicetable
% global bestAlphaTime bestAlphaCost  %coeff for  travel time  and relative
cost at level 1, group 2
global model_coefficients  % coefficients for  travel time  and relative cost

bestAlphaTime     = model_coefficients(1);
bestAlphaCost     = model_coefficients(incomeGrp + 1, 1);

% income_coeff      = model_coefficients(7, 1);
% dist_coeff        = model_coefficients(8, 1);
% INC_L2G1C1        = model_coefficients(9, 1); % inclusion value at level 2,
group 1 choice 1
% INC_L2G1C2        = INC_L2G1C1; % 0.7404; % inclusion value at level 2,
group 1 choice 2

income_coeff      = 0;
dist_coeff        = model_coefficients(7, 1);
INC_L2G1C1        = model_coefficients(8, 1); % inclusion value at level 2,
group 1 choice 1
INC_L2G1C2        = INC_L2G1C1; % 0.7404; % inclusion value at level 2, group
1 choice 2


%Step 1: compute utilities
U_Auto    = bestAlphaTime * tTime_Auto + bestAlphaCost * rCost_Auto;
U_airline = bestAlphaTime .* tTime_comm + bestAlphaCost .* rCost_comm +
dist_coeff * dist_Dummy;

%Step 2: Inclusive value for each nest in level 2
incValue_ground  = U_Auto; %log(exp(U_Auto));
exp_U_airline = exp(U_airline);
incValue_CommAir = log( sum( exp_U_airline ) );

%Step 3: Compute probability for each mode in level 1
exp_INC_L2G1C1_incValue_ground  = exp(INC_L2G1C1 * incValue_ground);
exp_INC_L2G1C2_incValue_CommAir = exp(INC_L2G1C2 * incValue_CommAir);
denominator = (exp_INC_L2G1C1_incValue_ground +
exp_INC_L2G1C2_incValue_CommAir);
P_Auto      = exp_INC_L2G1C1_incValue_ground  / denominator;
P_CommAir   = exp_INC_L2G1C2_incValue_CommAir / denominator;

%Step 4: Compute the conditional proabability for each mode in level 1:
SumExpU_Airline = sum(exp(U_airline));
for k = 1 : size(tTime_comm, 1)
   if SumExpU_Airline ~= 0
```

```
       P_CommAirRoute(k) = P_CommAir * exp_U_airline(k)/SumExpU_Airline; %
P_CommAirRoute(k) = P_CommAir * exp(U_airline(k))/SumExpU_Airline;
   else
       P_CommAirRoute(k) = 0;
   end
end

P_Auto           = round(P_Auto*10000)/10000;
P_CommAirRoute   = round(P_CommAirRoute * 10000)/10000;

sum_P_CommAirRoute = sum(P_CommAirRoute);
if(P_CommAir - sum_P_CommAirRoute >  0 || P_CommAir - sum_P_CommAirRoute < 0)
    P_CommAirRoute(1) = P_CommAirRoute(1) + (P_CommAir - sum_P_CommAirRoute);
end

P_CommAir = round(P_CommAir*10000)/10000;

P_all     = P_Auto + P_CommAir;
```

------------------------------------------------------------------------------------------------------------------------------------------------------

### 7.11.2.6 Sub-Function to round-up demand estimates to Integer values (Automobile & Commercial Air)

------------------------------------------------------------------------------------------------------------------------------------

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Using the probability for each mode compute number
% of trips and round of to integer values
%
% Called By: createmodechoicetable
% Calling:   None
%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function [modeTrips, commAvTrips_Rt] = roundTripsFnctn_AutoCA(intialTrips,
P_auto, P_CA, P_CommAirRoute, forceProbCA)

probList       = [P_auto P_CA];               % Probabilities
modeTrips      = zeros(1, size(probList, 2));        % initialize number of
modes
commAvTrips_Rt = zeros(1, size(P_CommAirRoute, 2));  % initialize number of
commercial air routes

if intialTrips == 1
    % if there is only 1 trip assign it to the mode with the highest
    % probability
    [valPrb indxPrb]      = max(probList); %
    modeTrips(1, indxPrb) = 1;

    if indxPrb == 2
        [valRt indxRt ] = max(P_CommAirRoute);
        commAvTrips_Rt(1, indxRt) = 1;
    end % if indxPrb == 2
else
    if forceProbCA == 1
        modeTrips = [0 intialTrips];
    else
        modeTrips  = [round(intialTrips * P_auto) round(intialTrips * P_CA)];

        sumTrips         = sum(modeTrips);
        diffTrips        = intialTrips - sumTrips; % difference between trips
assigned and initial trips

        % Keep distributing trips until there are no more trips to be
distriburted
        [valModeTrips indxModeTrips] = sort(modeTrips, 'descend');

        cntIndx_mode = 1;
        if abs(diffTrips) == 1
            modeTrips(1, indxModeTrips(cntIndx_mode)) = modeTrips(1,
indxModeTrips(cntIndx_mode)) + diffTrips;
        elseif abs(diffTrips) > 1
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference greater than 1'])
```

```
            pause
        elseif abs(diffTrips) < 0
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference is negative'])
            pause
        end % if abs(diffTrips) == 1


    end % if forceProbCA == 1

    % distribute CA trips
    caTrips_ini    = modeTrips(1,2);

    if P_CA > 0

        routeProb      = P_CommAirRoute./P_CA;

        if caTrips_ini == 0
            commAvTrips_Rt = zeros(1, size(routeProb, 2));
        else
            caTrips_round     = round(caTrips_ini .* routeProb);
            sum_caTrips_round = sum(caTrips_round);
            diffTrips_ca      = caTrips_ini - sum_caTrips_round;
            % diffTrips_ca > 0 => we have less trips than we need, so add
some trips
            % diffTrips_ca < 0 => we have more trips than we started with, so
delete some trips


            [valRoutes_ascend indxRoutes_ascend]   = sort(routeProb,
'ascend');
            [valRoutes_descend indxRoutes_descend] = sort(routeProb,
'descend');

            cntIndx = 1;
            while diffTrips_ca ~= 0

                % we have less trips than we need, so add some trips
                if diffTrips_ca > 0

                    tripIndex_more                   =
indxRoutes_descend(cntIndx);
                    caTrips_round(1, tripIndex_more) = caTrips_round(1,
tripIndex_more) + 1;
                    diffTrips_ca                     = diffTrips_ca - 1;

                % we have more trips than we started with, so delete some
trips
                elseif diffTrips_ca < 0

                    tripIndex_less                   =
indxRoutes_ascend(cntIndx);
                    if caTrips_round(1, tripIndex_less) > 0
                        caTrips_round(1, tripIndex_less) = caTrips_round(1,
tripIndex_less) - 1;
                        diffTrips_ca                     = diffTrips_ca + 1;
                    end % if caTrips_round(1, tripIndex_less) > 0
                end % if diffTrips_ca > 0

                if cntIndx == size(routeProb, 2)
                    cntIndx = 1;
                else
                    cntIndx = cntIndx + 1;
                end % if cntIndx == size(routeProb, 2)
            end % while diffTrips_ca ~= 0

            commAvTrips_Rt = caTrips_round;
        end % if caTrips_ini == 0
    end %if P_CA > 0

end % if intialTrips == 1
```

-----------------------------------------------------------------------------------------------------------------------------------

7.11.2.7 Sub-Function to Compute Probability of Choosing Automobile, Commercial Air, or Air Taxi

------------------------------------------------------------------------------------------------------------------------------------

```
function [P_Auto, P_CommAir, P_AirTaxi, P_CommAirRoute, P_all] =
nestedLogitFunction_AutoCAAirTaxi(tTime_Auto, tTime_comm, tTime_AirTaxi, ...
    rCost_Auto, rCost_comm, rCost_AirTaxi, incomeGrp, dist_Dummy); %,
income_Dummy, income_dist_Dummy)

% called by createmodechoicetable
% global bestAlphaTime bestAlphaCost  %coeff for  travel time  and relative
cost at level 1, group 2
global model_coefficients  % coefficients for  travel time  and relative cost

bestAlphaTime      = model_coefficients(1);
bestAlphaCost      = model_coefficients(incomeGrp + 1, 1);
% income_coeff       = 0;
dist_coeff         = model_coefficients(7, 1);

INC_L2G1C1    = model_coefficients(8, 1); % inclusion value at level 2, group
1 choice 1
INC_L2G1C2    = model_coefficients(8, 1); % 0.7404; % inclusion value at
level 2, group 1 choice 2
INC_L2G1C3    = model_coefficients(8, 1); % inclusion value at level 2, group
1 choice 3

%Step 1: compute utilities
U_Auto    = bestAlphaTime * tTime_Auto + bestAlphaCost * rCost_Auto;
U_airline = bestAlphaTime .* tTime_comm + bestAlphaCost .* rCost_comm +
dist_coeff * dist_Dummy; % + income_coeff * income_Dummy + dist_coeff *
income_dist_Dummy;
U_AirTaxi    = bestAlphaTime * tTime_AirTaxi + bestAlphaCost * rCost_AirTaxi;
% + income_coeff * income_Dummy + dist_coeff * income_dist_Dummy;

%Step 2: Inclusive value for each nest in level 2
incValue_ground  = log(exp(U_Auto));
incValue_CommAir = log( sum( exp(U_airline) ) );
incValue_AirTaxi    = log(exp(U_AirTaxi));

%Step 3: Compute probability for each mode in level 1
denominator = (exp(INC_L2G1C1 * incValue_ground) + exp(INC_L2G1C2 *
incValue_CommAir) + exp(INC_L2G1C3 * incValue_AirTaxi));
P_Auto      = exp(INC_L2G1C1 * incValue_ground)  / denominator;
P_CommAir   = exp(INC_L2G1C2 * incValue_CommAir) / denominator;

%Step 4: Compute the conditional proabability for each mode in level 1:
SumExpU_Airline = sum(exp(U_airline));
for k = 1 : size(tTime_comm, 1)
   if SumExpU_Airline ~= 0
       P_CommAirRoute(k) = P_CommAir * exp(U_airline(k))/SumExpU_Airline;
   else
       P_CommAirRoute(k) = 0;
   end
end
```

```
P_Auto         = round(P_Auto*10000)/10000;
P_CommAirRoute = round(P_CommAirRoute * 10000)/10000;

sum_P_CommAirRoute = sum(P_CommAirRoute);
if(P_CommAir - sum_P_CommAirRoute >  0 || P_CommAir - sum_P_CommAirRoute < 0)
    P_CommAirRoute(1) = P_CommAirRoute(1) + (P_CommAir - sum_P_CommAirRoute);
end

P_CommAir = round(P_CommAir*10000)/10000;
P_AirTaxi = 1 - P_Auto - P_CommAir;

P_all  = P_Auto + P_CommAir + P_AirTaxi;

return;
```

---------------------------------------------------------------------------------------------------------------------------------------------

#### 7.11.2.8 Sub-Function to round-up demand estimates to Integer values (Automobile & Commercial Air and Air Taxi)

------------------------------------------------------------------------------------------------------------------------------------

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Using the probability for each mode compute number
% of trips and round of to integer values
%
% Called By: createmodechoicetable
% Calling:   None
%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function [modeTrips, commAvTrips_Rt] =
roundTripsFnctn_AutoCAAirTaxi(intialTrips, P_auto, P_CA, P_CommAirRoute,
P_AirTaxi, forceProbCA, modeTrips_base, commAvTrips_Rt_base)

probList       = [P_auto P_CA P_AirTaxi];               % Probabilities
modeTrips      = zeros(1, size(probList, 2));        % initialize number of
modes
commAvTrips_Rt = zeros(1, size(P_CommAirRoute, 2));  % initialize number of
commercial air routes

if intialTrips == 1
    % if there is only 1 trip assign it to the mode with the highest
    % probability
    [valPrb indxPrb]      = max(probList); %
    modeTrips(1, indxPrb) = 1;

    if indxPrb == 2
        [valRt indxRt ] = max(P_CommAirRoute);
        commAvTrips_Rt(1, indxRt) = 1;
    end % if indxPrb == 2
else
    modeTrips  = [round(intialTrips * P_auto) round(intialTrips * P_CA)
round(intialTrips * P_AirTaxi)];

    differenceInTrips_mode = modeTrips(1:2) - modeTrips_base;
    increaseInTripsIndex_mode  = find(differenceInTrips_mode > 0);
    if isempty(increaseInTripsIndex_mode)
    else
        modeTrips(increaseInTripsIndex_mode) =
modeTrips_base(increaseInTripsIndex_mode);
    end % if isempty(increaseInTripsIndex)

    sumTrips   = sum(modeTrips);
    diffTrips  = intialTrips - sumTrips; % difference between trips assigned
and initial trips

    % Keep distributing trips until there are no more trips to be
distriburted
    [valModeTrips_descend indxModeTrips_descend] = sort(modeTrips,
'descend');
```

```
    [valModeTrips_ascend  indxModeTrips_ascend] = sort(modeTrips, 'ascend');

    modeIndex_all = 1;
    if forceProbCA == 1
        if abs(diffTrips) == 1
            modeTrips(1, indxModeTrips(modeIndex_all)) = modeTrips(1,
indxModeTrips(modeIndex_all)) + diffTrips;
        elseif abs(diffTrips) > 1
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference greater than 1'])
            pause
        elseif abs(diffTrips) < 0
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference is negative'])
            pause
        end % if abs(diffTrips) == 1
    else
        while diffTrips ~= 0

            % if diffTrips > 0 => we have less trips than needed, so add
trips
            if diffTrips > 0
                selectedModeIndex = indxModeTrips_descend(modeIndex_all);
                % make sure number of Auto and CA trips not greater than the
base case
                if selectedModeIndex ~= 3
                    current_tripsAll = modeTrips(1, selectedModeIndex);
                    old_tripsAll     = modeTrips_base(1, selectedModeIndex);
                    if current_tripsAll < old_tripsAll
                        modeTrips(1, selectedModeIndex) = modeTrips(1,
selectedModeIndex) + 1;
                        diffTrips = diffTrips - 1;
                    end % if current_tripsAll < old_tripsAll
                % Check not needed for AirTaxi
                else
                    modeTrips(1, selectedModeIndex) = modeTrips(1,
selectedModeIndex) + 1;
                    diffTrips = diffTrips - 1;
                end

            % if diffTrips < 0 => we have less trips than needed, so add
trips
            elseif diffTrips < 0
                selectedModeIndex = indxModeTrips_ascend(modeIndex_all);
                if modeTrips(1, selectedModeIndex) > 0
                    modeTrips(1, selectedModeIndex) = modeTrips(1,
selectedModeIndex) - 1;
                    diffTrips = diffTrips + 1;
                end % if modeTrips(1, selectedModeIndex) > 0
            end % if diffTrips < 0

            if modeIndex_all == size(probList, 2)
                modeIndex_all = 1;
            else
                modeIndex_all = modeIndex_all + 1;
            end % if modeIndex_all == size(probList, 2)
        end % while diffTrips ~= 0
    end % if forceProbCA == 1


    % distribute CA trips by route
    caTrips_ini    = modeTrips(1,2);

    if P_CA > 0
        routeProb      = P_CommAirRoute./P_CA;

        if caTrips_ini == 0
            commAvTrips_Rt = zeros(1, size(routeProb, 2));
        else
            caTrips_round     = round(caTrips_ini .* routeProb);

            differenceInTrips_route = caTrips_round - commAvTrips_Rt_base;
            increaseInTripsIndex_route  = find(differenceInTrips_route > 0);
            if isempty(increaseInTripsIndex_route)
            else
                caTrips_round(increaseInTripsIndex_route) =
commAvTrips_Rt_base(increaseInTripsIndex_route);
            end % if isempty(increaseInTripsIndex)

            sum_caTrips_round = sum(caTrips_round);
            diffTrips_ca      = caTrips_ini - sum_caTrips_round;

            [valRoutes_ascend indxRoutes_ascend]   = sort(routeProb,
'ascend');
            [valRoutes_descend indxRoutes_descend] = sort(routeProb,
'descend');

            modeIndex_route = 1;
            while diffTrips_ca ~= 0

                % diffTrips_ca > 0 => we have less trips than we need, so add
trips
                if diffTrips_ca > 0
                    tripIndex_more     = indxRoutes_descend(modeIndex_route);
                    old_RouteTrips     = commAvTrips_Rt_base(tripIndex_more);
                    cuurent_RouteTrips = caTrips_round(1, tripIndex_more);

                    if cuurent_RouteTrips < old_RouteTrips
                        caTrips_round(1, tripIndex_more) = caTrips_round(1,
tripIndex_more) + 1;
                        diffTrips_ca                     = diffTrips_ca - 1;
                    else
                    end % if cuurent_RouteTrips < old_RouteTrips

                % diffTrips_ca < 0 => we have more trips than we need, so
delete trips
                elseif diffTrips_ca < 0
                    tripIndex_less                   =
indxRoutes_ascend(modeIndex_route);
                    if caTrips_round(1, tripIndex_less) > 0
                        caTrips_round(1, tripIndex_less) = caTrips_round(1,
tripIndex_less) - 1;
                        diffTrips_ca                     = diffTrips_ca + 1;
                    end % if caTrips_round(1, tripIndex_less) > 0
                end % if diffTrips_ca > 0

                if modeIndex_route == size(routeProb, 2)
                    modeIndex_route = 1;
                else
                    modeIndex_route = modeIndex_route + 1;
                end % if modeIndex_route == size(routeProb, 2)
            end % while diffTrips_ca ~= 0

            commAvTrips_Rt = caTrips_round;
        end % if caTrips_ini == 0
    end % if P_CA > 0

end % if intialTrips == 1ÿ
```

---------------------------------------------------------------------------------------------------------------------------------------------

#### 7.11.2.9 Sub-Function to Compute Probability of Choosing Automobile, Commercial Air, or Train

------------------------------------------------------------------------------------------------------------------------------------------------

```
function [P_Auto, P_CommAir, P_Train, P_CommAirRoute, P_all] =
nestedLogitFunction_AutoCATrain(tTime_Auto, tTime_comm, tTime_Train, ...

rCost_Auto, rCost_comm, rCost_Train, incomeGrp, rtDistance)

% called by createmodechoicetable
% global bestAlphaTime bestAlphaCost  %coeff for  travel time  and relative
cost at level 1, group 2
global alpha_time_cost  % coefficients for  travel time  and relative cost

bestAlphaTime     = alpha_time_cost(1);
bestAlphaCost     = alpha_time_cost(incomeGrp + 1, 1);

INC_L2G1C1        = alpha_time_cost(7, 1); % inclusion value at level 2,
group 1 choice 1
INC_L2G1C2        = alpha_time_cost(8, 1); % 0.7404; % inclusion value at
level 2, group 1 choice 2
INC_L2G1C3        = alpha_time_cost(9, 1); % inclusion value at level 2,
group 1 choice 3

% INC_L2G1C1  = 0.6; % 0.3916; % inclusion value at level 2, group 1 choice 1
% INC_L2G1C2  = 0.9; % 0.7404; % inclusion value at level 2, group 1 choice 2
% INC_L2G1C3  = 0.95; % 0.7404; % inclusion value at level 2, group 1 choice
2

%Step 1: compute utilities
U_Auto    = bestAlphaTime * tTime_Auto + bestAlphaCost * rCost_Auto;
U_airline = bestAlphaTime .* tTime_comm + bestAlphaCost .* rCost_comm;
U_Train    = bestAlphaTime * tTime_Train + bestAlphaCost * rCost_Train;

%Step 2: Inclusive value for each nest in level 2
incValue_ground  = log(exp(U_Auto));
incValue_CommAir = log( sum( exp(U_airline) ) );
incValue_Train    = log(exp(U_Train));

%Step 3: Compute probability for each mode in level 1
denominator = (exp(INC_L2G1C1 * incValue_ground) + exp(INC_L2G1C2 *
incValue_CommAir) + exp(INC_L2G1C3 * incValue_Train));
P_Auto      = exp(INC_L2G1C1 * incValue_ground)  / denominator;
P_CommAir   = exp(INC_L2G1C2 * incValue_CommAir) / denominator;
%P_Train       = exp(INC_L2G1C3 * incValue_Train) / denominator;


%Step 4: Compute the conditional proabability for each mode in level 1:
for k = 1 : size(tTime_comm, 1)
   SumExpU_Airline = sum(exp(U_airline));
   if SumExpU_Airline ~= 0
       P_CommAirRoute(k) = P_CommAir * exp(U_airline(k))/SumExpU_Airline;
   else
       P_CommAirRoute(k) = 0;
   end
end

P_Auto         = round(P_Auto*10000)/10000;
P_CommAirRoute = round(P_CommAirRoute * 10000)/10000;

sum_P_CommAirRoute = sum(P_CommAirRoute);
if(P_CommAir - sum_P_CommAirRoute >  0 || P_CommAir - sum_P_CommAirRoute < 0)
    P_CommAirRoute(1) = P_CommAirRoute(1) + (P_CommAir - sum_P_CommAirRoute);
end

P_CommAir = round(P_CommAir*10000)/10000;
P_Train = 1 - P_Auto - P_CommAir;

P_all  = P_Auto + P_CommAir + P_Train;
```

----------------------------------------------------------------------------------------------------------------------------------------------

7.11.2.10 Sub-Function to to round-up demand estimates to Integer values (Automobile & Commercial Air and Train)

------------------------------------------------------------------------------------------------------------------------------------

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Using the probability for each mode compute number
% of trips and round of to integer values
%
% Called By: createmodechoicetable
% Calling:   None
%
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function [modeTrips, commAvTrips_Rt] =
roundTripsFnctn_AutoCATrain(intialTrips, P_auto, P_CA, P_CommAirRoute,
P_Train, forceProbCA)

probList       = [P_auto P_CA P_Train];                % Probabilities
modeTrips      = zeros(1, size(probList, 2));         % initialize number of
modes
commAvTrips_Rt = zeros(1, size(P_CommAirRoute, 2));  % initialize number of
commercial air routes

if intialTrips == 1
    % if there is only 1 trip assign it to the mode with the highest
    % probability
    [valPrb indxPrb]      = max(probList); %
    modeTrips(1, indxPrb) = 1;

    if indxPrb == 2
        [valRt indxRt ] = max(P_CommAirRoute);
        commAvTrips_Rt(1, indxRt) = 1;
    end % if indxPrb == 2
else
    modeTrips  = [round(intialTrips * P_auto) round(intialTrips * P_CA)
round(intialTrips * P_Train)];

    sumTrips   = sum(modeTrips);
    diffTrips  = intialTrips - sumTrips; % difference between trips assigned
and initial trips

    % Keep distributing trips until there are no more trips to be
distriburted
    [valModeTrips indxModeTrips] = sort(modeTrips, 'descend');

    cntIndx_mode = 1;
    if forceProbCA == 1
        if abs(diffTrips) == 1
            modeTrips(1, indxModeTrips(cntIndx_mode)) = modeTrips(1,
indxModeTrips(cntIndx_mode)) + diffTrips;
        elseif abs(diffTrips) > 1
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference greater than 1'])
```

```
            pause
        elseif abs(diffTrips) < 0
            disp(['Warning Error in Mode choice code: ROUND TRIP FUNCTION'])
            disp(['difference is negative'])
            pause
        end % if abs(diffTrips) == 1

    else
        while diffTrips ~= 0
            if diffTrips > 0
                if modeTrips(1, indxModeTrips(cntIndx_mode)) > 0
                    modeTrips(1, indxModeTrips(cntIndx_mode)) = modeTrips(1,
indxModeTrips(cntIndx_mode)) + 1;
                    diffTrips = diffTrips - 1;
                end % if modeTrips(1, indxModeTrips(cntIndx_mode)) > 0
            else
                diffTrips = diffTrips + 1;
                modeTrips(1, indxModeTrips(cntIndx_mode)) = modeTrips(1,
indxModeTrips(cntIndx_mode)) - 1;
            end % if diffTrips < 0

            if cntIndx_mode == size(probList, 2)
                cntIndx_mode = 1;
            else
                cntIndx_mode = cntIndx_mode + 1;
            end % if cntIndx == size(routeProb, 2)
        end % while diffTrips ~= 0
    end % if forceProbCA == 1


    % distribute CA trips by route
    caTrips_ini    = modeTrips(1,2);

    if P_CA > 0

        routeProb      = P_CommAirRoute./P_CA;

        if caTrips_ini == 0
            commAvTrips_Rt = zeros(1, size(routeProb, 2));
        else
            caTrips_round     = round(caTrips_ini .* routeProb);
            sum_caTrips_round = sum(caTrips_round);
            diffTrips_ca      = caTrips_ini - sum_caTrips_round;
            % diffTrips_ca < 0 => we have more trips than we started with, so
delete some trips
            % diffTrips_ca > 0 => we have less trips than we need, so add
some trips

            [valRoutes_ascend indxRoutes_ascend]   = sort(routeProb,
'ascend');
            [valRoutes_descend indxRoutes_descend] = sort(routeProb,
'descend');

            cntIndx = 1;
            while diffTrips_ca ~= 0

                % we have less trips than we need, so add some trips
                if diffTrips_ca > 0

                    tripIndex_more                     =
indxRoutes_descend(cntIndx);
                    caTrips_round(1, tripIndex_more) = caTrips_round(1,
tripIndex_more) + 1;
                    diffTrips_ca                     = diffTrips_ca - 1;

                % We have less trips than we need
                elseif diffTrips_ca < 0

                    tripIndex_less                     =
indxRoutes_ascend(cntIndx);
                    if caTrips_round(1, tripIndex_less) > 0
                        caTrips_round(1, tripIndex_less) = caTrips_round(1,
tripIndex_less) - 1;
                        diffTrips_ca                     = diffTrips_ca + 1;
                    end % if caTrips_round(1, tripIndex_less) > 0

                end % if diffTrips_ca > 0

                if cntIndx == size(routeProb, 2)
                    cntIndx = 1;
                else
                    cntIndx = cntIndx + 1;
                end % if cntIndx == size(routeProb, 2)
            end % while diffTrips_ca ~= 0

            commAvTrips_Rt = caTrips_round;
        end % if caTrips_ini == 0
    end % if P_CA > 0

end % if intialTrips == 1

-------------------------------------------------------------------------------
------------
```

# Matlab and SAS Code to (a) Calibrate Logit Models, and (b) Estimate Demand in Transportation Systems Analysis Model

# 8 CURRICULUM VITA

**Senanu Ashiabor**

## EDUCATION

**Ph.D. in Civil Engineering,** May, 2007
Virginia Tech, Blacksburg, Virginia.
Dissertation: *Modeling Intercity Mode Choice and Airport Choice in the United States.*

**M.S. in Civil Engineering,** December, 2002
Virginia Tech, Blacksburg, Virginia.
Thesis*: Airport Choice Model for General Aviation Operations in the US.*

**B.S. in Civil Engineering**, June 1997
University of Science and Technology, Kumasi, Ghana.
*Award for Best Civil Engineering Student Design Project in Graduating Class.*

## RESEARCH INTERESTS

Travel Demand Modeling

Travel Behavior Modeling

Aviation Systems Planning

Transportation Economics

## PROFESSIONAL AFFILIATIONS AND ACTIVITIES

Member and Webmaster: Aviation Systems Planning Committee, Transportation Research Board of National Academies (TRB) – January 2003 to date.

Panel Member: TRB Airport Cooperative Research Program Panel on 'Guidebook for Airport-User Survey Methodology – June 2006 to date.

Chartered Member: Transportation Development Institute of American Society of Civil Engineers – 2004 to date

Vice President: Virginia Tech Student Chapter of Institute of Transportation Engineers – 2003 - 2005)

## PEER-REVIEWED PUBLICATIONS AND PRESENTATIONS

Ashiabor. S, Baik. H, Trani. A, 2006. Logit Models to Forecast Nationwide Intercity Travel Demand in the United States, to be published in upcoming Transportation Research Record.

Baik. H, Ashiabor. S, Trani. A, 2006. Development of an Airport Choice Model for General Aviation Operations, Transportation Research Record 1951.

Baik. H, Trani. A, Ashiabor. S, et. al., 2006. Large-Scale Transportation Systems Analysis Model to Predict County-to-County Demand, Accepted for presentation at 2007 Transportation Research Board 2007 Conference.

Viken. J, Trani. A, Baik. H, Ashiabor. S, et. al., 2006. Utilizing Traveler Demand Modeling to Predict Future Commercial Flight Schedules in the NAS, 11th AIAA Conference, Portsmouth.

Trani. A, Baik. H, Hinze. N, Ashiabor. S, et. al., 2005. Integrating Air Transportation System Demand Predictions in Preliminary Aircraft Design, AIAA 5th Aviation Technology Integration and Operations Conference.

Trani. A, Baik. H, Ashiabor. S, 2003. An Integrated Model to Studying Small Aircraft Transportation System, Transportation Research Record 1850.