FORECASTING MODEL FOR HIGH-SPEED RAIL IN THE UNITED STATES

Saloni Ramesh Chirania

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

Master of Science
in
Civil Engineering

Antonio A. Trani, Chair
Montasir M. Abbas
Antoine G. Hobeika

October 02, 2012
Blacksburg, Virginia

Keywords: High-Speed Rail, AMTRAK, Travel Demand Forecast, Box-Cox, American Travel Survey, TSAM

FORECASTING MODEL FOR HIGH-SPEED RAIL IN THE UNITED STATES

Saloni Ramesh Chirania

ABSTRACT

A tool to model both current rail and future high-speed rail (HSR) corridors has been presented in this work. The model is designed as an addition to the existing TSAM (Transportation System Analysis Model) capabilities of modeling commercial airline and automobile demand. TSAM is a nationwide county to county multimodal demand forecasting tool based on the classical four step process. A variation of the Box-Cox logit model is proposed to best capture the characteristic behavior of rail demand in US. The utility equation uses travel time and travel cost as the decision variables for each model. Additionally, a mode specific geographic constant is applied to the rail mode to model the North-East Corridor (NEC). NEC is of peculiar interest in modeling, as it accounts for most of the rail ridership. The coefficients are computed using Genetic Algorithms. A one county to one station assignment is employed for the station choice model. Modifications are made to the station choice model to replicate choices affected by the ease of access via driving and mass transit. The functions for time and cost inputs for the rail system were developed from the AMTRAK website. These changes and calibration coefficients are incorporated in TSAM. The TSAM model is executed for the present and future years and the predictions are discussed. Sensitivity analysis for cost and speed of the predicted HSR is shown. The model shows the market shift for different modes with the introduction of HSR. Limited data presents the most critical hindrance in improving the model further. The current validation process incorporates essential assumptions and approximations for transfer rates, short trip percentages, and access and egress distances. The challenges for the model posed by limited data are discussed in the model.

ACKNOWLEDGEMENTS

I would like to first thank Dr. Antonio Trani for giving me the opportunity to conduct this research and for his critical insights and critique. This work would not have been possible without his guidance. I would also like to thank my advisory committee Dr. Montasir Abbas and Dr. Antoine Hobeika for their constant support and valuable inputs.

I would like to extend a special thanks to Nicolas Hinze for assisting me through every step of the project. His experience and acumen have been crucial in development of the computer modeling code. I wish to extend my gratitude to Nelson Guerreiro for sharing the calibration codes, contributing greatly to this research. I am grateful to Jeff Viken, Samuel Dollyhigh and Teck-Seng Kwa from NASA for their constructive criticism and comments. A special thanks to Mr. Howard Swingle for his insightful pointers and for making the work at lab enjoyable.

I also want to mention all the new friends I have made in the Air Transportation Systems Lab ,the Civil Engineering Graduate Program and Blacksburg, thanks for all the good times and memories. Thanks Mr. Zou, it always helped to know you have not completed your presentation either. And finally, I wish to thank all my friends back home, for always being there.

ATTRIBUTION

Several colleagues have helped in the writing and development of this model. A brief description of their contributions is presented below.

Chapter 3: Forecasting Model for High-Speed Rail Demand in the United States

Vandyke, A. (Air Transportation Systems Lab Virginia Tech, Civil Engineering) is currently a transportation engineer at VDOT. His thesis work and mine have been combined and presented in the paper that was submitted to the TRB conference (Transportation Research Board). He developed the basic framework of the model, allowing for the calibrations presented in the paper

Guerreiro, N. (Analytical Mechanics Associates Inc.) is currently a research engineer at the NASA Langley Research Center. He was a co-author on the paper, and developed the code for the genetic algorithm calibration of the model, which has been a huge contribution to the project. Part of the code is presented in Appendix D.7

Hinze, N (Air Transportation Systems Lab Virginia Tech, Civil Engineering), is currently a Senior Research Associate at Virginia Tech. He is a co-author on the paper and was responsible for the development of the TSAM framework.

Trani, A. (Air Transportation Systems Lab Virginia Tech, Civil Engineering), is currently a professor in the Civil Engineering Department at Virginia Tech. Dr. Trani was a co-author on this paper, the principal investigator for the grants supporting this research, and for mentoring the project.

Dedication

To Ma, my first teacher
To Papa, for believing in me
To Bhaiya, for teaching me to dream

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 BACKGROUND AND MOTIVATION FOR RESEARCH

High-Speed rail (HSR) systems have been successfully deployed in countries like Japan, France, Spain, Germany, Taiwan, and Korea and more recently in China. The Next-Generation HSR service is planned to be introduced in the United States of America (US) with sustained maximum speeds of 220 mph (354 kph). The project is expected to be completed by 2040 in the North East Corridor (NEC). This service will connect major hub cities in the NEC like Boston, New York, Philadelphia along with smaller cities, airports and suburban hubs.

With the introduction of Acela, AMTRAK's market share between New York and Washington DC grew significantly. In the air-rail market, AMTRAK's share grew from 36% to 53%. (Clifford March 2005). If closer cities like Baltimore and Philadelphia are also included, the market share is about 75% for AMTRAK. In the New York – Boston air-rail market, AMTRAK's share grew from 18% - 40%. In the first quarter of FY 2004, around 4.767 million air/rail trips were made, out of which 2.928 million (61%) were on AMTRAK trains. More than 10% of AMTRAK's 24 million annual passengers' i.e. around 2 million passengers annually are served by the Acela Rail. About 25% of AMTRAK's annual ticket revenue is accounted by the Acela Express. Introducing rail with higher speeds, while maintaining reliability and safety, is expected to attract higher ridership. Due to comfortable work environments, business trips are expected to make up for a major part of the attracted trips. A comprehensive nationwide multimodal model is required to be able to predict the HSR demand in US.

This work describes a family of logit models developed to model rail demand along with commercial air and automobiles.

1.2 TSAM 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. A model called Transportation Systems Analysis Model (TSAM) was developed at the Air Transportation Systems Laboratory at Virginia Tech, to estimate demand for small aircraft used if on-demand services (Trani 2003). Over the years extensive work on TSAM has enabled it to additionally model nationwide commercial airlines and automobile demands (Ashiabor 2007) (Baik 2008).

TSAM is a nationwide multimodal logit software package capable of modeling intercity long-distance trips until 2040. Counties are used as activity zones, and 3091 counties are modeled in TSAM. Additionally, 443 commercial air airports have been modeled in TSAM, and the airport choice model facilitates airport – airport level demand forecasting. Business trips and Non-Business trips are modeled separately. The person trips are characterized by income groups (in 2000 dollars): < \$29K, \$29K - \$57K, \$57K - \$86K, \$86K - \$144K, and > \$144K. The model uses the American Travel Survey (ATS) data for the year 1995. The model also uses other date sets like the Woods & Poole socioeconomic data, the Airline Origin and Destination Survey DB1B etc.

The algorithm for TSAM is shown in the flowchart in Figure 1. The proposed additions are described in this paper.

1.3 PROBLEM DESCRIPTION

The purpose of this research is development of a mode choice model for rail travel in US. The main goals are to replicate the currently observed AMTRAK ridership, along with commercial air and automobile trips. The aim is to model a traveler's decision when a set of possible transportation alternatives are available between two counties. The mode choice model combined with a station choice model, gives the station to station ridership for rail and commercial airline trips. The model aims at developing probabilities of such decisions based on the utility maximization rationale. The framework of the model is based on a Box-Cox Logit model. Faced with a choice set of K alternatives, each individual n attaches a utility U_{nk} to the k^{th} alternative, based on the attributes of the alternative. A probability that an individual selects a specific alternative i can be written as P_{ni} = Prob ($U_{ni} > U_{nk} \forall\ k \neq i$). These probability functions can then be used along with the trip demand in TSAM to compute the demand for the selected model. The utility functions are developed based on data about individuals, their choices, their decision collected in surveys.

TSAM is modeled using the ATS 1995 records. The ATS is a nationwide survey of travelers in the US conducted by the Census Bureau. One of the limitations of ATS is the limited geographical details of the data that is publicly available. Rail ridership in US is concentrated in specific corridors especially of the NEC and California. Acela rail significantly changed the market since 2000, and have to be given special consideration in modeling. The base data is from 1995, and the model aims at modeling the intercity long-distance rail demand till 2040. Many HSR lines are in the planning stage and the model aims at being able to calculate travel times and costs for both the present and planned lines.

The AMTRAK data used for validation has one way trips including short trips. Information about transfer rates, access distances and O-D pairs is not available. Lack of literature on these parameters, requires that the data be analyzed and parsed by approximation, to serve as a validation platform. The model also aims at estimating rail demand at the station level by employing station choice model. The aim of this research is to use the ATS and AMTRAK data to develop a comprehensive flexible tool, capable of credible long-distance intercity rail ridership estimation at the county and station levels in the future.

1.4 RELEVANCE OF RESEARCH

NASA uses TSAM to estimate the impact of aircraft operations on the National Airspace System (NAS). The proposed additions in TSAM will enable NASA to study the changes in NAS with the introduction of HSR in the US. The model can also be used by federal agencies like the U.S. Department of Transportation, Federal Highway Administration (FHWA), Federal Aviation Administration (FAA), and Federal Transit Administration (FTA) to study the intermodal behavior and effect of HSR. The model enables sensitivity analysis in terms of speed, cost, selective HSR corridors, year of operation and station processing times. It can be used as a decision making tool by AMTRAK, to study the effect of different operating policies on the rail demand. Commercial airlines can use the

model to study the loss of air market share due to HSR, and make strategic plans. The tool is thus of practical importance, and would prove to be a resourceful tool for decision makers.

1.5 OUTLINE OF THESIS

The document is outlined as follows. Chapter 2 presents the Literature Review discussing the various work done on logit models and calibrating High-Speed Rail. Chapter 3 presents a paper on the Forecasting of High-Speed Rail Demand in the United States. The paper describes a logit model to calibrate rail mode choice along with commercial air and automobile demand, using the ATS 1995 data. The paper represents the current state of the model as of August 2012.

Chapter 4 presents several attempted calibration models. A train station choice model and the modifications made to the existing train module in TSAM are discussed in Chapter 4. The validation and HSR forecast results are illustrated. Conclusions and Recommendations sum up the paper in Chapter 5. The appendices have Matlab codes developed to execute the logit models, station choice model and for the required preprocessing of the data.

2 LITERATURE REVIEW

2.1 HIGH-SPEED RAIL AROUND THE WORLD

The Japanese Shinkansen runs from Tokyo to Osaka with a current speed of 320kph (200mph) (Central Japan Railway Company n.d.). Trials with the Fastetch 360 test rail have shown that a speed of 360km/h (224 mph) is currently not feasible due to noise pollution and braking distance issues. Noise pollution concerns have been of major interest in Japan, considering the high population density. Shinkansen as a result is now regulated at noise levels of less than 70 dB in residential areas. Travelling Tokyo – Osaka by Shinkansen produces only 16% of CO of the equivalent of that produced by an automobile for the same journey. There are currently three types of services Tokaido Shinkansen services, the Nozomi, Hikari, and the Kodam; the Series 300, the series 700 and the Series N700 (N700 Shinaknsen, Japan n.d.).

The TGV (Rail à Grande Vitesse) is the world speed holder in train technology, and travels at a speed of 322 km/h (Tiller and Nottara n.d.). The project was launched in the late 1960's, and the first test was done with TGV 001 in the early 1970's. The TGV 001 was gas powered and set a world speed record of 318 km/hr. Today hundreds of trains depart Paris for Geneva, Bordeaux, Toulouse, Marseilles, Cannes, Dijon, Brussels, London, Amsterdam, Frankfurt, and even Milan. The SNCF (Societe Nationale des Chemins de Fer) , the National French Railroad Company guarantees an arrival within 30 minutes on HSR or a refund of one third of the ticket price (Lew n.d.).

The Intercity-Express is a system of HSR functioning in Germany and the neighboring countries. There are 259 train sets in five different version of ICE (High Speed Rail in Germany: Inter-city Planes are Grounded by Faster Trains, World News n.d.). Operating at a maximum speed of 300km/h (186 mph), it connects Germany to (Wien, Innsbruck), Belgium (Brussels, Liège), Denmark (Copenhagen, Arhus), France (Paris), the Netherlands (Arnhem, Utrecht, Amsterdam) and Switzerland (Zürich, Interlaken). The ICE rail are used by the around 65,000 people every day. The HSR has a 97% air-rail market share in between Cologne and Frankfurt.

The AVE (Alta Velocidad Española) is a family of HSR in Spain. It travels up to speeds of 300km/h (186mph) (History of the French TGV n.d.). The AVE train sets are made in France with some Spanish parts, and is a descendant of the French TGV Atlantique. The AVE track is similar to the German ICE. The trip between Madrid and Barcelona is completed in less than 3 hours. The AVE is established as Europe's most punctual rail service, with almost 100% on time arrivals. The operators have policies that refund 100% if the delays exceed 30 minutes.

China is a relatively new member in the field of HSR. It had a pretty poor market before it began speeding up and renovating the existing lines. This speed up campaign began in April 1997 and continued until April 2004 (Feng n.d.). Recently, China has laid a network of more than 10,000 kms. of HSR track, calling for investments of billions of Yuan. The building of HSR continues with an aim to build more than 13,000 kms. by 2012 and expanding it further to 16,000 kms. by 2020. The collision in Wenzhou in July, 2011, that killed 40 people, has slowed down the progress. The Beijing – Tianjin intercity railway is China's first official HSR that was inaugurated on August1,

2008. With a top speed of 350 km/h (217 mph), it travels a distance of about 120 kms at a price of about 8 – 15$ (US 2012 $) for the trip (Feng n.d.).

2.2 RAIL IN AMERICA

AMTRAK's Acela Express connecting the North-East Corridor from Boston to Washington DC, runs at an average speed of 68mph (109 km/h) ("High-Speed Rail Strategic Plan: Press Release & Highlights" 2009, April 16). In 2009, President Obama signed the American Recovery and Reinvestment Act and unveiled a plan to build a $13 billion network of high-speed, intercity passenger rail services connecting US metropolitan areas in 31 states with rail traveling at speeds of over 200 miles per hour (High Speed Rails in the United States n.d.). This service was expected to serve 80% Americans by 2035. $8 billion in stimulus funds were awarded to upgrade existing services and construct new lines ("High-Speed Rail Strategic Plan: Press Release & Highlights" 2009, April 16). California will probably have the first HSR investment in US, from Anaheim to San Francisco via Los Angeles and San Jose.

The Federal Railroad Administration (FRA) defines HSR as a service "that is time- competitive with air and/or auto for travel markets in the approximate range of 100 – 500 miles" (Department of Transportation 1997). The federal government plans on starting the construction of the initial line around September 2012. The report by the US Department of Transportation identifies ten potential HSR corridors: California, Pacific Northwest, South Central, Gulf Coast, Chicago Hub Network, Florida, Southeast, Keystone, Empire and Northern New England. It is envisioned to reduce the dependence on foreign oil, lower harmful emissions, foster economic development, make travel more convenient and easy by reducing dependence on cars and planes.

The HSR plans have been delayed due to several reasons. The federal government's financial support for lines outside NEC has not been adequate for purposes other than primary planning (Frittelli 2009). Moreover, there has not been substantial evidence to gauge the profit of HSR. These problems combined with high initial investments, and the long time before breaking the costs even have been the major reasons for postponing the HSR project. Currently there is only one line in the country with an attainable speed on over 110 mph (NEC) and four other corridors that can reach top speeds of more than 79mph (Table 1).

TABLE 1 High-Speed Rail Corridors in the United States

Corridor	Length (miles)	Current Top Speed (mph)	Current Average Speed (mph)
Los Angeles- San Diego, CA	130	90	55
Chicago, IL – Detroit/Pontiac, MI	304	95	53
New York City – Albany/ Schenectady, NY	158	110	56
Philadelphia – Harrisburg, PA	104	110	66
Northeast Corridor (NEC)	454		
Boston, MA – New York City, NY, segment	229	150	68
New York City, NY – Washington, DC, segment	225	135	82

2.3 EXISTING MODE CHOICE LOGIT MODELS

Logit models used are based on the principles of rational decision and utility maximization. The probability of choosing a certain mode over the others, given a set of conditions is computed using a set of utility equations. The purpose of calibration is to estimate the coefficients of the function to best suit the existing conditions. For the mode choice models, the utility of every mode under study is computed based on the explanatory variables which could be time, cost, frequency and others. The probability of choosing a mode i , with a utility value U_i is then given by Equation 1.

$$P(i) = \frac{e^{U_i}}{\sum_{i=1}^{i} e^{U_i}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{(1)}$$

Where:

$P(i)$ = probability of selecting mode i

$U(i)$ = Utility of mode i (usually expressed as a function of factors that affect mode split)

Various attempts at developing a multimodal logit models for the US have been made. The work by Stopher and Prashker in 1976 developed a multinomial logit model for automobile, air, bus and rail modes using 2,085 observations from the 1972 National Travel Survey (NTS) (Stopher 1976). Grayson in 1982 calibrated a model for trips starting and ending in Metropolitan Statistical Areas, using variables of cost, travel time, departure frequency and access time (Grayson 1981). Morrison and Winston in 1985 used a log-sum variable to hierarchically nest three models. The models were decision to rent a car, bus, rail and air. Both these models used the NTS 1977 database (Morrison 1985). Koppelman in 1990 extended Morrison's hierarchical approach to nest a nest a trip frequency, trip destination, mode choice and fare class models (F. Koppelman 1989). All these models had commercial air, automobile, bus and rail trips in their mode choices. The NTS is conducted by the Bureau of the Census and the Bureau of Transportation Statistics (BTS). The major constraint in improving the credibility of these models was the limited geographical details of the information in NTS. Koppelman et. al also showed that lack of service variables and low level of information on choice set in the NTS data were additional limitations for the models (F. G. Koppelman 1984) .

The work by Poorzahedy, Tabatabaee, Kermanshah, Aashtiani, and Toobaei, examined the calibration of the logit model with two methods: Maximum log likelihood and non-linear least squares regression (Poorzahedy 2004). The results showed that the non-linear regression has better accuracy in prediction but has a higher standard deviation than the maximum likelihood. The maximum likelihood is thus suited better when certain inputs or data might be missing.

2.4 HIGH-SPEED RAIL LOGIT MODELS

The work for HSR has mainly been done for the Asian and European countries. Yao, Morikawa, Kurauchi, and Tokida examined the demand for high-speed rail in Japan using the logit models (Yao 2002). The input data had stated preference (SP) and revealed preferences (RP). Stated preference though easy to collect is lesser accurate. RP

data, although more reliable, is very expensive to collect. As such, this work developed two logit models, one each for SP and RP inputs, and then combined the two to the maximum log likelihood. This model provided more reliable results as compared to models using SP data only.

Pagliara, Vassallo and Roman's work explains the calibration of a model of high-speed rail and air transportation (Pagliara 2012). The model uses SP survey data to calibrate a mode choice model for the Madrid-Barcelona corridor and uses variables other than travel time and cost. Three different models were proposed, based on the linear-in-the parameter specification for the utility.

$$V_{AIR} = \beta_{ASC-AIR} + \beta_{COST}.COST_{AIR} + \beta_{FREQ}.FREQ_{AIR} + \beta_{CHECK-IN}.CHECK-IN_{AIR} \ldots\ldots (2)$$

$$V_{AVE} = \beta_{COST}.COST_{AVE} + \beta_{FREQ}.FREQ_{AVE} + \beta_{PARKING}.PARKING_{AVE}.CAR$$

Where,
COST is travel cost in euros
FREQ is the service frequency measures in departures per hour
CHECK-IN is 1 if security control service and check-in are rapid, 0 otherwise
CAR is 1 if access is by car and 0 otherwise

The paper shows that apart from travel time, there are other crucial parameters like frequency and cost. It also introduces the concept of willingness to pay (WTP) which measures changes in utility in monetary terms. WTP in other words, is the marginal rate of substitution between travel cost and the corresponding utility. The first two models were analyzed considering Binomial Logit model, Mixed Logit model with fixed parameters. The third model analyzed random taste heterogeneity with more than two random parameters. The third model presented a better likelihood and goodness of fit.

Mandel, Gaudry, and Rothengatter propose the use of Box-Cox transformations to improve the logit model prediction (Mandel .B 1997). The demand market is usually assumed to have a linear function, and very little work has been done testing the non-linearity of the function. The linear assumption, if proven wrong introduces large errors in the models, which cannot be fixed by constants. The Box-Cox transformations overcome this drawback, and the work shows the differences in the results of a linear and non-linear model. The linear models rend to over predict ridership in shorter distances, while predict lower ridership for longer distances. The non-linear models gave a more realistic picture of the market. Linear utility functions are still used due to ease of computations. The formulation of the Box-Cox transformation is:

$$X_{kjn}^{(\lambda_{kj})} = \begin{cases} \frac{(X_{kjn}^{\lambda_{kj}} - 1)}{\lambda_{kj}} & \text{if} \quad \lambda_{kj} \neq 0, \\ \ln X_{kjn} & \text{if} \quad \lambda_{kj} = 0 \end{cases} \quad \ldots\ldots\ldots\ldots (3)$$

The utility equations for each of the alternatives are then formulated. The probability equation for a given mode, after applying the transformation is given by Equation 4. The transformation removes some of the linear assumptions and adds more degrees of freedom to the model, to give a better fitting model.

$$P(i)_n = \frac{\exp\left(\beta_i X_i + \sum_{k=1}^{K} \beta_{ki} x_{kin}^{\lambda_{kj}}\right)}{\sum_{j \in C_n} \exp\left(\beta_i X_i + \sum_{k=1}^{K} \beta_{ki} x_{kin}^{\lambda_{kj}}\right)} \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The proposed model is a variation of the box-cox logit model. The model is calibrated using methods such as genetic algorithms and maximum log likelihood. The model is modified to best capture the US trends for rail.

2.5 PRODUCTIVE TIME

A very limited work has been done to understand the concept of productive time during travel. The work Lyons et.al illustrates about the usage of travel time in Great Britain by rail passengers, based on a survey of 26,000 rail travelers (Lyons 2007). An analysis is presented based on the factors like gender, travel duration, class of travel and type of activity. Working or studying was found to be the most prevalent activity among the business travelers. It also points out that with longer travel times, the number of different activities that a person engages in increases, reflecting on the productivity of the individual. An important finding was that 33% of first class travelers considered their time used worthwhile compared to 23% of other passengers. This reflects on the effect of work environment on productivity.

Another study by MacDonald analyzes the duration of travel time spent by rail business travelers working (R Fickling, et al. 2008). For rail journeys of 1 -3 hours in length, 69% of business travelers are found to use their time productively. For all the rail journeys, 43% of the business travelers were found to do some work. Across all the business travelers, 30% of the journey time was time spent working. It also compared the productivity of the work done and the rail productivity was significantly comparable to the office productivity as shown in Table 2. The productive time on return trips was found to be lesser than the outward journey.

TABLE 2 Time Weighted Average of Individual Productivity Factors

Journey time band (JTBand_2)	Mean
Less than 45 mins	98%
45-89 mins	97%
90-149 mins	98%
150 mins or more	96%
Total	97%
Sample count = 1190	

A survey for 60 businessmen at Netherlands, who commute using commercial air regularly, is presented in the work by Breure and Meel (Breure A 2003). The sample set is relatively small, and the travelers were found to work more than 60% of the usable time at airports.

3 FORECASTING MODEL FOR HIGH-SPEED RAIL DEMAND IN THE UNITED STATES.

Submission date: 1st August, 2012
Number of words: 3866 +1750 (7 figures) + 1750 (7 tables) = 7366

First and corresponding author: Chirania Saloni
Affiliation: Graduate Research Assistant, Virginia tech
Address: Blacksburg, VA 24060
Phone: +1 940-293-5195
E-mail: saloni@vt.edu

Second author: Vandyke Alex
Affiliation: VDOT
Address: 1596 Deborah Lane Salem, VA 24153
Phone: +1 540-375-0122
E-mail: alex.vandyke@vdot.virginia.gov

Third author: Guerreiro Nelson
Affiliation: Analytical Mechanics Associates, Inc.,
Address: Hampton, VA, 23681-2199
Phone: +1 757-864-2083
E-mail: nelson.m.guerreiro@nasa.gov

Fourth author: Hinze Nicolas
Affiliation: Senior Research Associate, Virginia Tech
Address: Hampton, VA, 23681-2199
Phone: +1 757-864-2083
E-mail: nhinze@vt.edu

Fifth author: Trani Antonio
Affiliation: Professor, Virginia Tech
Address: Blacksburg, VA 24060
Phone: +1 540-231-4418
E-mail: vuela@vt.edu

ABSTRACT

A family of logit models is developed to predict county to county rail trips at the station level for 464 rail stations and 3091 counties across the United States of America. High-Speed Rail (HSR) forecasts for long-distance annual ridership are modeled for twelve proposed high-speed corridors. Business and Non-Business trips are modeled separately. Travel choices are further divided into five income groups. The model developed is calibrated using a variation of the Box-Cox logit model with data from the 1995 American Travel Survey. Travel time and travel cost are employed as the explanatory variables. Rail frequencies, overnight travel costs, travel times including Acela rail performance functions are considered to develop the explanatory variables. Lodging time, access and egress times, mass transit access in selected cities have also been included. Limited, publicly-available AMTRAK data is used for validation and analysis. Potential uses of the model are forecasting HSR demand and performing sensitivity analyses for different operating HSR policies and network configurations. Similarly, the model can be employed to study the effects of HSR on commercial and ground mode ridership. The model developed has been incorporated into a software framework called the Transportation Systems Analysis Model (TSAM), which is a nationwide, long-distance travel demand forecast model. The proposed model will allow TSAM to predict rail trips along with existing ridership estimates for commercial airline and automobile trips.

3.1 INTRODUCTION

High-Speed rail (HSR) systems have been successfully deployed in countries like Japan, France, Spain, Germany, Taiwan, Korea and more recently in China. United States is exploring the idea of implementing HSR in selected corridors. To capture the effect of HSRs on market, a multimodal intercity model is required.

The Transportation System Analysis Model (TSAM) is a multi-mode, national-level, transportation framework developed by Virginia Tech using the classical four-step planning process *(1-3)*. TSAM considers commercial air, automobile and air taxi trips. TSAM estimates county to county, nationwide demand using county-level socio-economic data forecast until the year 2040. The American Travel Survey (ATS) 1995 defines long trips as trips which have a one way door to door distance of greater than 100 miles *(4)*. While TSAM has a commuter travel module, the scope of the model presented in this paper is only for long-distance trips. Business purpose trips and Non-Business trips are modeled separately. Person trips are further segregated based on five income groups (in 2000 dollars): < $29K, $29K - $57K, $57K - $86K, $86K - $144K, and > $144K.

There are 3091 counties and 464 rail stations modeled in TSAM, along with 443 commercial service airports. The mode choice model uses a 3091 x 3091 county-county, person trip origin-destination (OD) matrix developed from the trip distribution output *(1)*. A new rail station choice model is incorporated in the mode choice model. The model is formulated such that sensitivity analysis can be performed for rail cost, speed and frequencies. The National Aeronautics and Space Administration (NASA) is using TSAM to forecast commercial air travel demands and to analyze the impacts of next-generation air transportation system investments in the overall mobility of the country. As new HSR corridors are created in the U.S., they could be expected to attract demand from other modes of transportation and could potentially relieve congestion at busy airport facilities and interstate highways. Thus, a model that synergistically considers multiple modes of transportation is of practical importance and an important tool for decision makers. A family of logit models has been developed to include rail mode as a choice in TSAM to facilitate HSR ridership forecasts. Figure 1shows the modeling framework used in TSAM with additional modules for rail analysis.

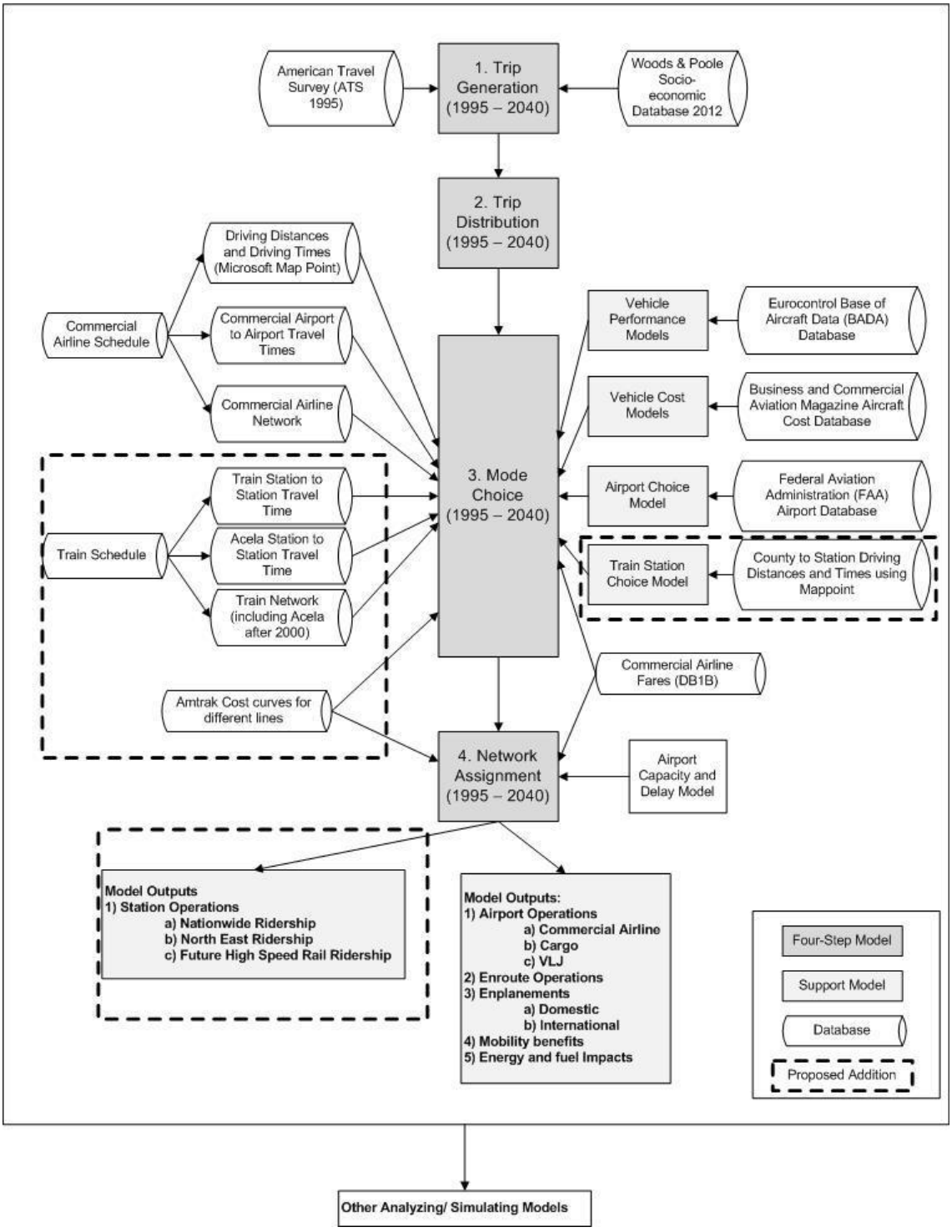


FIGURE 1 TSAM Model Flowchart.

3.2 LITERATURE REVIEW

The introduction of Acela rail in the Northeast corridor (NEC) induced changes in the air-rail market in the NEC. The rail share between Washington, DC (DC) and New York (NY) increased from 36% - 53%, while the airlines dropped from 64% to 47% *(5)* (Clifford March 2005). In a closer OD pair like Baltimore and Philadelphia, AMTRAK share increased to about 75%. More than 10% of AMTRAK's passengers annually are served by the Acela rail. The multimodal model developed should be able to capture these market shifts, and predict HSR ridership with other level of service improvements.

Various attempts at developing a multimodal logit models for the US have been made. The work by Stopher and Prashker in 1976, Grayson in 1977, Morrison and Winston in 1985 and Koppelman is of particular interest *(6-9)* (Stopher 1976) (Grayson 1981) (Morrison 1985) (F. Koppelman 1989). All these models had commercial air, automobile, bus and rail trips in their mode choices. The explanatory variables of these models are summarized in Table 3. Most of these models were calibrated using the 1977 National Travel Survey (NTS) database. The NTS was conducted by the Bureau of the Census and the Bureau of Transportation Statistics (BTS). The major consrailt at improving the credibility of these models was the limited geographical details of the information contained in NTS. Koppelman et. al also showed that lack of service variables and low level of information on choice sets in the NTS data were additional limitations for the models *(10)* (F. G. Koppelman 1984). It is interesting to note that similar limitations exist in the more recent national-level surveys including the 1995 ATS.

TABLE 3 Review of Multimodal Logit Model Variables

Authors	Variables in the Utility Function
Stopher and Prashker (1976)	Relative time, relative distance, relative cost, relative access–egress distance, departure frequency
Alan Grayson (1982)	Travel time, travel cost, access time, and departure frequency
Morrison and Winston (1985)	Travel time, cost, party size, average time between departures
Koppelman (1990)	Travel time, cost, departure frequency, distance between city pairs, household income

HSR calibrations have been attempted majorly in countries of Asia and Europe. The models are based on the fundamental principle of utility maximization and rational choice. Albalate and Bel reviewed and analyzed information from different HSR projects over the world, and presented the obstacles and benefits for the US *(11)* (Albalate n.d.). Pagliara, Vassallo and Roman calibrated a mode choice model for Madrid-Barcelona HSR corridor that uses variables such as cost, frequency and check-in time spent *(12)* (Pagliara 2012). López-Pita and Robusté studied the impact of HSR on high frequency air service *(13)* (López-Pita 2005). Another model by Gunn, Bradley and Hensher, analyses HSR market share for the rail corridor connecting Sydney, Canberra and Melbourne in Australia. *(14)* (Gunn 1992) One common attribute of these models is their limited mode choice focus between a few cities.

In most models, the utility function is generally assumed to have a linear form. Linear utility functions are used to help in the computational effort and time involved in model calibration. A disaggregate box-cox logit model developed for the HSR in Germany showed that linear utility functions tend to over predict demand for shorter distances *(15)* (Mandel .B 1997). The asymmetry introduced by the nonlinear models was shown to have a stronger relation to the consumer behavior than the linear models. For these reasons, we employ a variation of the Box-Cox logit function to model the commercial air, automobile and rail demand.

3.3 CALIBRATION DATABASE

The American Travel Survey (ATS), performed jointly by Census Bureau and the Bureau of Transportation Statistics (BTS) in 1995, has 556,026 person-trip records and 348 variables for each record that help explain travel behavior *(4)* (American Travel Survey: Technical Documentation 1995). To date, ATS is the best data known to be publicly available for determining long-distance travel. The most recent National Household Travel Surveys (NHTS) do not have enough long-distance trip records for all modes of transportation to help calibrate a nationwide model *(16)* (National Household Travel Survey 2001). It is important to note that, of the 348 variables available for each ATS record, key variables such as travel time and cost were not sampled. As previously mentioned travel time and cost are the primary explanatory variables to explain mode choice behavior. Measures of travel time and travel cost are estimated using publicly available data sets for existing air, automobile and rail networks. For example, commercial air trip costs are derived from a combination of twelve million records of fares published by BTS every year. Similarly, rail cost is derived using published AMTRAK fares.

3.3.1 *ATS DATA ANALYSIS*

The ATS data used in TSAM is filtered for commercial air, automobile and rail trips, summarized in Table 4. It is noted that less than 1%, of the data corresponds to rail data.

TABLE 4 ATS Person Trip Record Analysis

Data description	Total Records	Auto	CA	Rail
Business	92,622	66,187	25,593	842
		71.5 %	27.6%	0.9 %
Non Business	211,126	186,038	23,832	1256
		88.12 %	11.29%	0.59%

Business rail trips have 842 records, 674 are from the Metropolitan Statistical Areas (MSA) in NEC. The model aims at capturing the high demand seen in NEC. Although the California Corridor also attracts significant AMTRAK ridership, the ATS has only 53 rail records for the California Corridor. Hence, the records are not sufficient to model the corridors separately.

3.4 RAIL STATION CHOICE MODEL

A simple station choice model is built into the mode choice model. Each rail station is associated with a set of neighboring counties. The outputs are thus in the form of station-to-station demand, as well as county to county demand. Travel distances and times are determined from the county population centroids to each station, using Microsoft Map Point Software *(17)* (MapPoint: Business Mapping and Data Visualization Software 2004). A station is assigned to each county based on the shortest driving distance and congestion travelling time. For cities with a high number of commuter rail demands, the mass transit network is also considered in the station assignment.

3.5 MODE CHOICE MODEL

3.5.1 *MODEL CALIBRATION*

The proposed model was calibrated using a Genetic Algorithm (GA) approach. GA was chosen because of its suitability to reach global minima determination. A large population size was used, to allow for a vast search of the parameter space (in general, greater than 500 - 600 times the number of parameters was used as the population size). In addition, each calibration was repeated at least 10 times – due to the random nature of the GA – selecting the best minima. The ten best solutions of the GA were followed by a local optimizer to further refine the minima solution. The logit models are based on the utility maximization rationale. The basic form of any logit model is,

$$P_i = \frac{e^{V_i}}{\sum_i e^{V_i}} \qquad (5)$$

where: P_i is the probability of choosing alternative i, given the utility value of V_i. The utility values are computed for each of the variables in the model. Calibrating the model gives the best estimate of the coefficients to compute the utilities.

3.5.2 *ATTEMPTED MODELS*

The calibration trials are done with a population size of 6000. Filtering data to better capture the NEC demand resulted in inadequate records for a good calibration. Business travelers are seen to prefer rail travel for their work friendly environment. Efforts to incorporate the productive time during travel for business trips resulted in a bad calibration due to the high correlation between the productive time and travel time. Correction factor trials by replacing travel time with factors like productive time, unproductive time, productive time to travel time ratio, unproductive time to productive time, did not seem to improve the calibration significantly. Different combinations of adding a mode specific constant to the rail was experimented with, and the utility structure shown in Equations 6 - 9 was selected.

3.5.3 *PROPOSED MODEL*

The proposed model is a variation of the Box-Cox logit model. The Business and Non-Business trips are represented by the equations below.

$$V_{\text{Auto}} = \alpha_{\text{TT}} \frac{(\text{TT}_{\text{Auto}}^{\lambda_{\text{TT Auto}}} - 1)}{\lambda_{\text{TT Auto}}} + \alpha_{\text{TC}} \frac{(\text{TC}_{\text{Auto}}^{\lambda_{\text{TC Auto}}} - 1)}{\lambda_{\text{TC Auto}}} \quad (6)$$

$$V_{\text{Air}} = \alpha_{\text{TT}} \frac{(\text{TT}_{\text{Air}}^{\lambda_{\text{TT Air}}} - 1)}{\lambda_{\text{TT Air}}} + \alpha_{\text{TC}} \frac{(\text{TC}_{\text{Air}}^{\lambda_{\text{TC Air}}} - 1)}{\lambda_{\text{TC Air}}} \quad (7)$$

$$V_{\text{Rail-bus}} = \gamma \times (0/1)_{\text{NEC}} + \alpha_{\text{TT}} \frac{(\text{TT}_{\text{Rail}}^{\lambda_{\text{TT Rail}}} - 1)}{\lambda_{\text{TT Rail}}} + \alpha_{\text{TC}} \frac{(\text{TC}_{\text{Rail}}^{\lambda_{\text{TC Rail}}} - 1)}{\lambda_{\text{TC Rail}}} \quad (8)$$

$$V_{\text{Rail-Non-bus}} = \gamma_{\text{a}} \times (0/1)_{\text{NY}} + \gamma_{\text{b}\times}(0/1)_{\text{DC}} + \alpha_{\text{TT}} \frac{(\text{TT}_{\text{Rail}}^{\lambda_{\text{TT Rail}}} - 1)}{\lambda_{\text{TT Rail}}} + \alpha_{\text{TC}} \frac{(\text{TC}_{\text{Rail}}^{\lambda_{\text{TC Rail}}} - 1)}{\lambda_{\text{TC Rail}}} \quad (9)$$

Where:

TT_{i} = Travel Time of mode i for given trip
TC_{i} = Travel Cost of mode i for given trip
α_{TT} = Travel Time Coefficient
α_{TC} = Travel Cost Coefficient
$\lambda_{\text{TT}_{\text{i}}}$ = Box − Cox Travel Time (TT) Coefficient specific to mode i
$\lambda_{\text{TC}_{\text{i}}}$ = Box − Cox Travel Cost (TC) Coefficient specific to mode i
γ = Mode specific constant for the rail mode

The Box-Cox model selected has one extra parameter for business trips, and two for non-business trips. For the business trips, the model has a mode specific constant for rail to capture the mode inertia for the trips that originate or terminate in the NEC. The Non-Business trips have two geographic constants, one each for NY and DC. This can be justified by the ATS record analysis presented in the OD matrix in Table 5. The probability of the mode selection is then calculated using Equation 1.

TABLE 5 ATS Non-Business Rail Record Analysis

O/D	CT	DC	DE	MA	MD	NJ	NY	PA	RI	VT	Total
CT		16	2	2	5	2	2	3		3	35
DC	6		6	6		7	17	3	2	4	51
DE	2	4			2	1	1		2		12
MA	7	11			1	7	10	2			38
MD	4	2				8	4	3	2		23
NJ	9	36		5	8		1	1	6	1	67
NY	61	109	35	14	30	3	49	6	41	11	**359**
PA	8	40		1	2	2	5	11	4	4	77
RI							1	2			3
Total	97	**218**	43	28	48	30	90	31	57	23	665

3.6 MODEL COMPONENTS

The following input matrices were developed and used as inputs for nationwide calibration and forecast.

3.6.1 *AMTRAK TIME AND COST*

A nationwide time and cost model is used based on functions developed from publicly available records and AMTRAK, through queries of random future dates. Time and cost functions are also developed for selected corridors. The Acela performance is captured by giving a weight of 1/3 to the Acela cost and time. This is based on the analysis of ridership of Acela and the Northeast regional service *(18)* (Monthly Performance Report January 2011). Weighted cost and time functions are used for the stations with Acela service after the year 2000. The cost function developed is shown in Equation 10, and has an R^2 value of 0.987.

$$C = -1^{-06}s^3 + 0.0006s^2 + 0.1654s + 10.304 \quad (10)$$

The weighted time function for the North-East Corridor post 2000 is given by Equation 11 with an R^2 value of 0.9942

$$T = -2^{-06} + 0.0012s^2 + 0.8418s - 13.929 \quad (11)$$

Where:
C = One way Cost for NEC after 2000, in 2010$
T = One way weighted travel time NEC, post 2000, in minutes
S = One way station to station distance in miles

3.6.2 *NORTH-EAST CORRIDOR*

NEC for modeling purposes is defined as counties within 150 miles from a NEC station. The trips originating or ending in this buffer area are considered to have access to the rail. The ATS record analysis of driving distances to stations in NEC forms the basis of this assumption and is shown in Table 6. A buffer of 150 miles while being slightly high captures most of the trips in the NEC.

TABLE 6 ATS's Reported Driving Distance to Station - Northeast Corridor Only

Air		Rail	
Distance (miles)	% of Records	Distance (miles)	% of Records
100	91.89	100	92.33
150	95.95	150	96
200	97.35	200	97.48

3.6.3 *SCHEDULE DELAY*

Schedule delay (SD) for airlines is defined as the time difference between the travelers' desired and actual time of departure *(19)* (Teodorovic 1998).The same SD is assumed to be applicable for rail. Rail frequency for the stations is computed from the AMTRAK schedules. Assuming the service period is 16 hours, SD for all the OD pairs is computed using Equation 12.

$$S.D. = \frac{T}{4f} \quad (12)$$

where

S.D. = schedule delay (hours)
T = daily service period (hours)
f = daily service frequency

3.6.4 *PROCESSING TIME*

The processing times at the origin station is assumed to be twenty minutes. A ten minute processing at the destination station is assumed for rail. The processing time includes time spent in purchasing a ticket, checking in, boarding/un-boarding, security check points, etc. In addition, a waiting time of ten minutes is assigned to both the stations. The waiting time at the origin station is to denote the passengers who come early to ensure that they do not miss the rail. For the destination station, the waiting time denotes the time spent in waiting for a connecting trip; another rail, bus, cab, and time spent exiting the station. These assumptions will need to be checked periodically to ensure that they are realistic.

3.6.5 *LODGING TIME AND COST*

If the round trip travel time exceeds a defined maximum daily travel time, the lodging time penalty and costs are included in the computations. The maximum travel time for the business trips is 8 hours, and for the non-business trips is 10 hours. Lodging cost is a function of the origin county, the destination county, and the party size for rail. It is assigned based on the trip purpose and income group.

The time penalty is assigned in the form of a ramp function for modeling purposes. The penalty ranges from 0-1, increasing with every half an hour of travelling time for the first 16 hours, after which the maximum penalty of 1 day is assumed.

3.6.6 *TRAVEL TIME AND COST*

The total Travel time and cost for round trips is computed using the above factors and are given by Equations 13 and 14.

$$T_t = 2*(A_t + E_t + W_{ot} + W_{dt} + S2S_{Tt} + SD + L_t) \quad (13)$$

$$T_c = 2*(A_c + E_c + S2S_{Tc} + L_c) \quad (14)$$

Where T = Travel, $_c$ = Cost, $_t$ =Time in hours, A = Access, E = Egress, W = Waiting, $_O$ = Origin, $_D$ = Destination, S2S = Station to station, SD = Schedule Delay, L = Lodging

3.7 MODEL ASSUMPTIONS

3.7.1 *TRANSFER RATES*

To convert TSAM person trips into round trips, the transfer rates for the major stations are estimated and presented in Table 7. The Train network was modeled, and station to station ridership from TSAM was then input in the network. Twenty transfer hubs are defined from the network and route intersections. These hubs are assigned ranks based on their annual ridership and are parsed by the shortest distance algorithm. Assuming a range of 10% of the shortest travel distance, the hub with the highest rank was selected for transfer.

TABLE 7 Estimated Transfer Rates

Station	**NYP**	**WAS**	**PHL**	**CHI**	**BOS**	**LAX**	**SAC**	**ALB**	**MTZ**	**STL**
Transfer	22.35%	28.71%	12.89%	70.38%	10.40%	44.35%	70.34%	36.35%	77.42%	61.45%
Station	SJC	NOL	CLT	KCY	VAN	GRO	SAS	EVR	SPI	FTW
Transfer	18.29%	79.81%	40.65%	45.26%	57.65%	16.96%	22.08%	12.84%	7.11%	47.20%

The estimation is based on the 2012 rail network and the ridership as modeled by TSAM. TSAM currently does not incorporate transfer time and the disutility effect of transfers in computing the ridership. However, it accounts for SD, processing times and lodging times, which have correlation with transfer times. Due to limited data and literature available, the credibility of these factors cannot be further explored. Accurate transfer rates are important to improve the model and to validate its results.

3.7.1 *SHORT DISTANCE TRIPS PERCENTAGE*

AMTRAK data used in the model calibration includes commuter trips (i.e., less than 100 miles route distance). These commuter trips are filtered to establish baseline comparisons with TSAM rail model. The publicly available AMTRAK Fact Sheet has origin passenger counts by station for the years 2004 – 2009. The same database includes the top destination stations for 527 stations *(20)* (Fact Sheets for all Amtrak Stations n.d.). It also has the ridership distribution over distance with a bin size of 100 miles for each of the stations.

According to our analysis, the short trips reported by AMTRAK (i.e., < 100 miles one way) constitute 13.3% of the nationwide rail trips. The AMTRAK ridership in 1995 is 20.7 million one way person trips *(21)* (Transportation Stastics Annual Report 1995). Assuming the percentage for short trips and transfers were the same in 1995, the AMTRAK ridership estimates to about 3.7 million originating round trips. TSAM models 4.6 million originating round trips for the base year, over predicting rail trips by approximately one million.

3.8 CALIBRATION RESULTS

The calibration coefficients for non-business and business trips are shown in Table 8 and Table 9. Figure 2 and Figure 3 show the corresponding market share plots for all five income levels and three modes of transportation. The calibration plots for the non-business trips show consistently good results over all income groups. The business trip records are fewer and perhaps not well sampled over distance, and thus the calibration is not as accurate as that for non-business trips. The spikes seen in some of the plots reflect the low sampling of long-distance, business travel rail records.

TABLE 8 Non-Business Trips Calibration Coefficients

Income	α_{TT}	α_{TC}	$\lambda_{TT\text{-}Auto}$	$\lambda_{TC\text{-}Auto}$	$\lambda_{TT\text{-}CA}$	$\lambda_{TC\text{-}CA}$	γ_a	γ_b	$\lambda_{TT\text{-}Rail}$	$\lambda_{TC\text{-}Rail}$
< $29K	-6.204	-0.600	-0.291	0.000	-0.321	0.162	0.695	1.108	-0.248	0.109
$29 K –$57 K	-4.520	-1.153	-0.294	0.021	-0.252	0.049	0.588	1.446	-0.188	0.058
$57 K –$86 K	-1.294	-4.009	-0.316	-0.095	0.211	-0.166	0.108	1.550	0.161	-0.118
$86K –$144 K	-3.912	-5.052	-0.408	-0.173	-0.152	-0.235	0.897	0.714	-0.017	-0.289
> $144 K	-5.455	-0.500	-0.185	-9.808	-0.436	0.128	0.352	1.192	-0.062	-19.310

TABLE 9 Business Trips Calibration Coefficients

Income	α_{TT}	α_{TC}	$\lambda_{TT\text{-}Auto}$	$\lambda_{TC\text{-}Auto}$	$\lambda_{TT\text{-}CA}$	$\lambda_{TC\text{-}CA}$	γ_{NEC}	$\lambda_{TT\text{-}Rail}$	$\lambda_{TC\text{-}Rail}$
< $29K	-3.577	-7.390	-0.229	-54.952	-0.477	-1.953	0.311	-0.344	-1.885
$29 K –$57 K	-3.383	-0.348	-0.083	-33.800	-0.359	0.208	1.989	-0.018	0.064
$57 K –$86 K	-2.882	-0.153	-0.034	-41.825	-0.168	0.278	2.767	-0.006	0.371
$86K –$144 K	-3.627	-3.654	-0.080	-20.127	-0.133	-1.947	4.167	0.056	-1.877
> $144 K	-4.560	-0.066	-0.186	-7.393	-0.374	0.458	3.229	-0.460	0.802

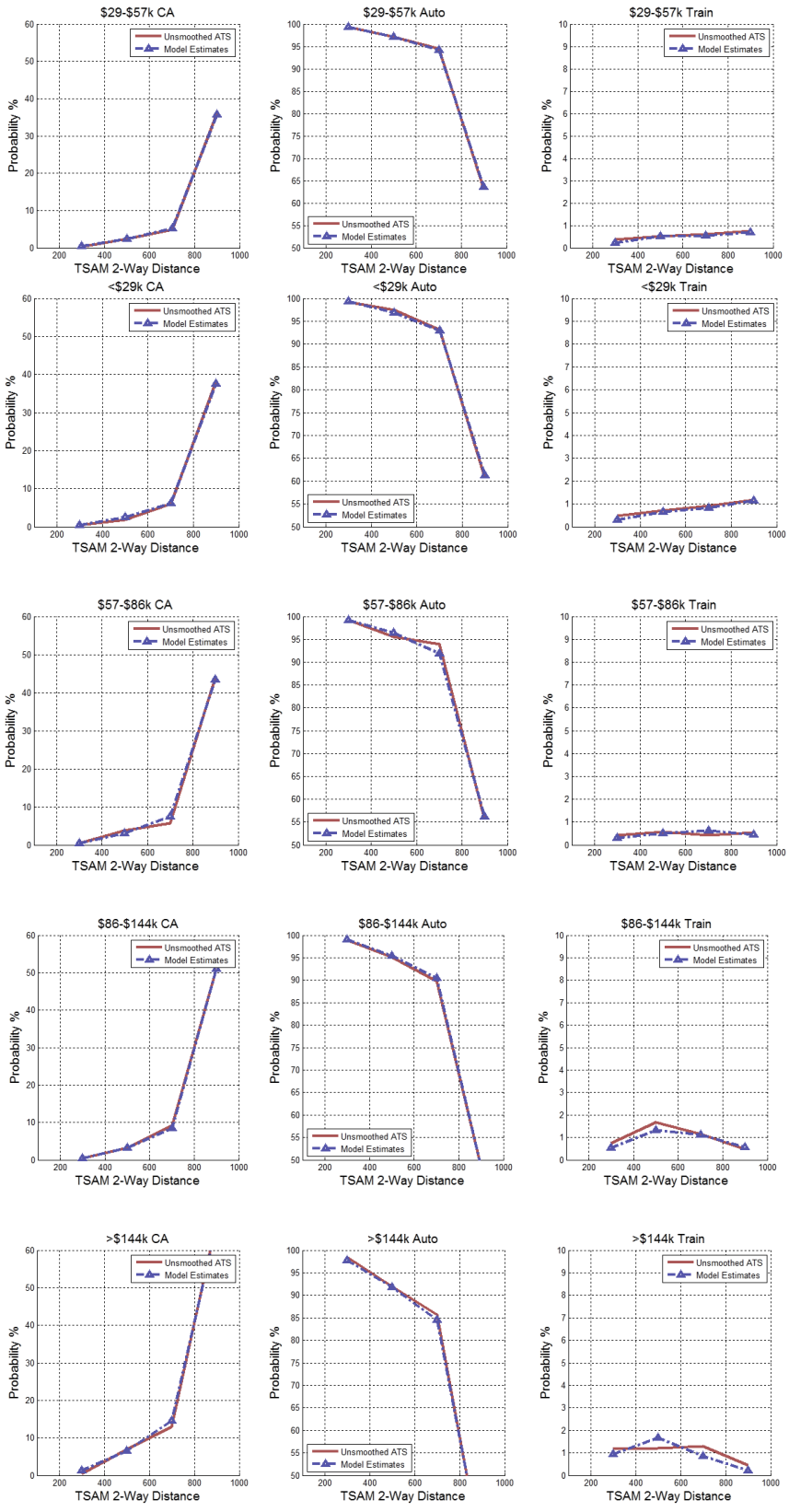


FIGURE 2 NON-BUSINESS ATS TRIPS CALIBRATION PLOTS.

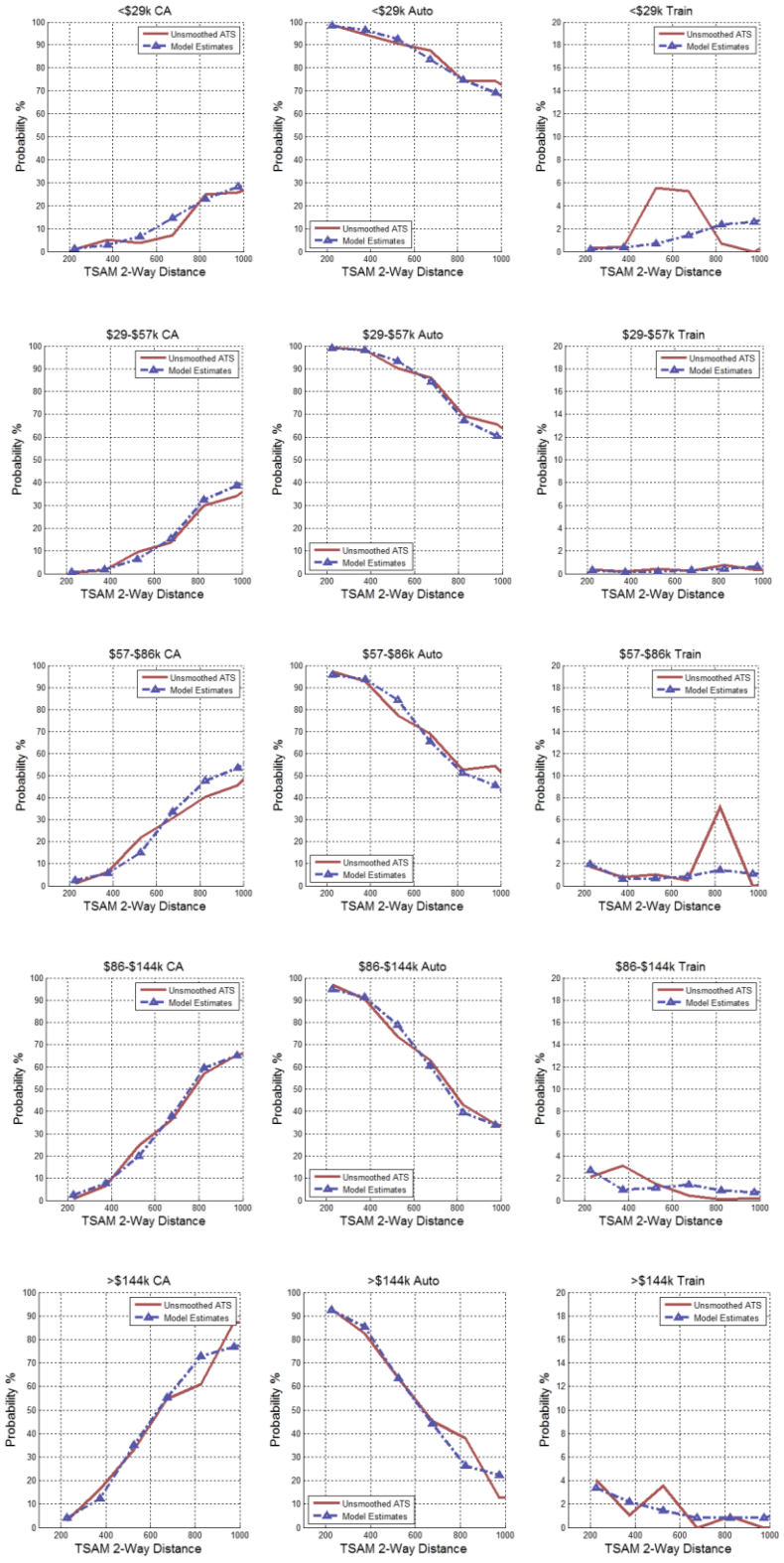


FIGURE 3 BUSINESS ATS TRIPS CALIBRATION PLOTS

3.9 VALIDATION

The model output is compared to the filtered AMTRAK data for year 2009. The total rail round trip ridership predicted by the model is 8.5 million. The ridership distribution across stations is illustrated in Figure 4 and Figure 5. According to AMTRAK data, the number of round trips in 2009 was estimated at 7.1 million. The variations from the estimated AMTRAK ridership at the station level are within the range of +/- 170K, as shown in Figure 5. Most of the ridership estimation deficit observed is in the NEC and California corridors.

The ridership distribution over distance comparisons for a few major stations is shown in Figure 6. A Chi-square test is performed to check the consistency of the model ridership predictions as a function of distance compared to AMTRAK published data, at a 0.05 significance level, with a critical value of 16.919 (9 degrees of freedom). The distribution of the predicted vs. estimated AMTRAK trips for Penn station and Philadelphia can be seen to be fairly consistent, with low $\Sigma\chi^2$ values. The trends for Boston and Los Angeles show larger variations, especially in the 400 – 600 mile range. Although there are some discrepancies observed at the station level, the model can be used satisfactorily at the national level.

FIGURE 4 ESTIMATED ROUND TRIP RAIL RIDERSHIP IN 2009 BY TSAM.

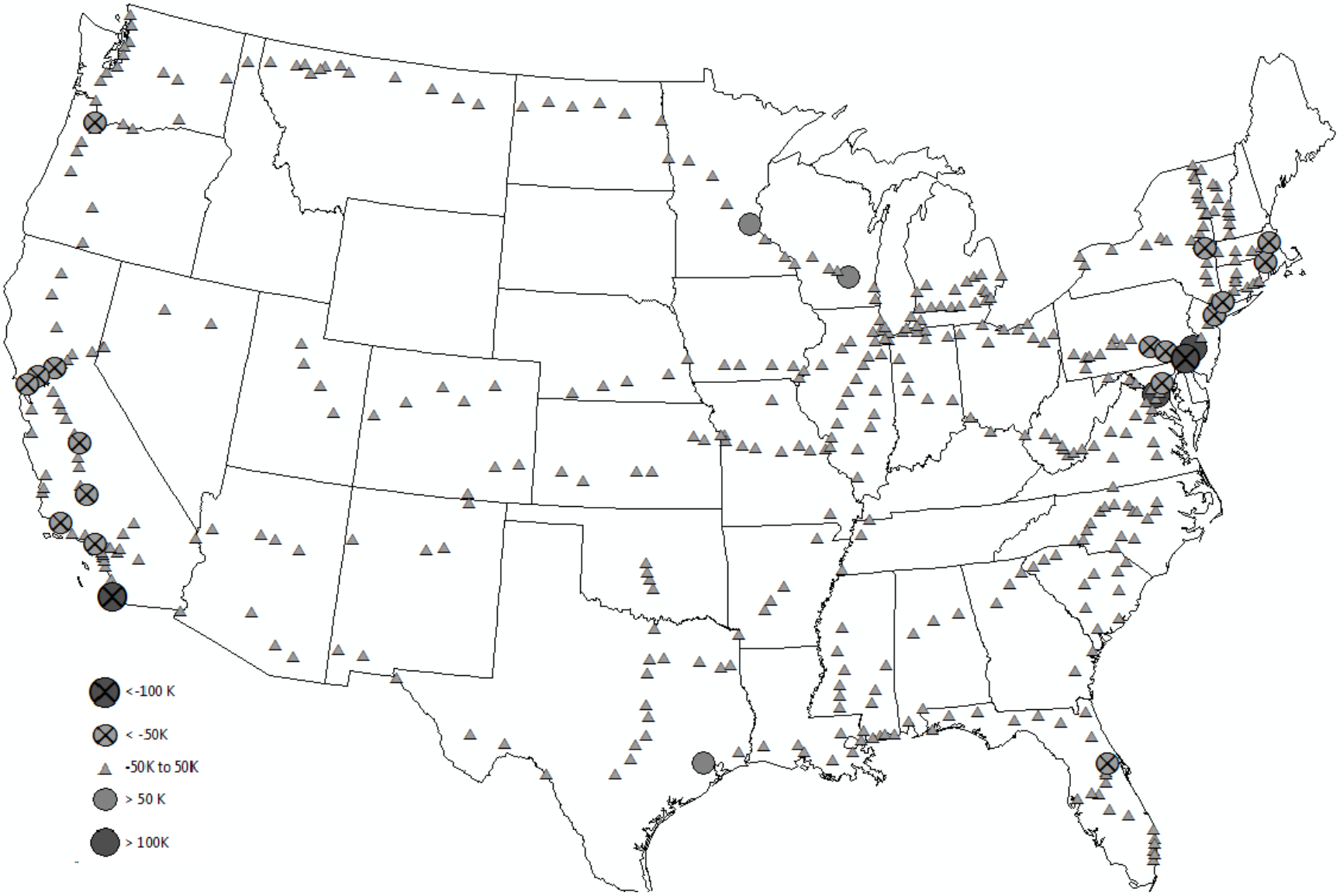


FIGURE 5 ESTIMATED TSAM RAIL RIDERSHIP DIFFERENCE FROM AMTRAK IN 2009 ROUND TRIPS.

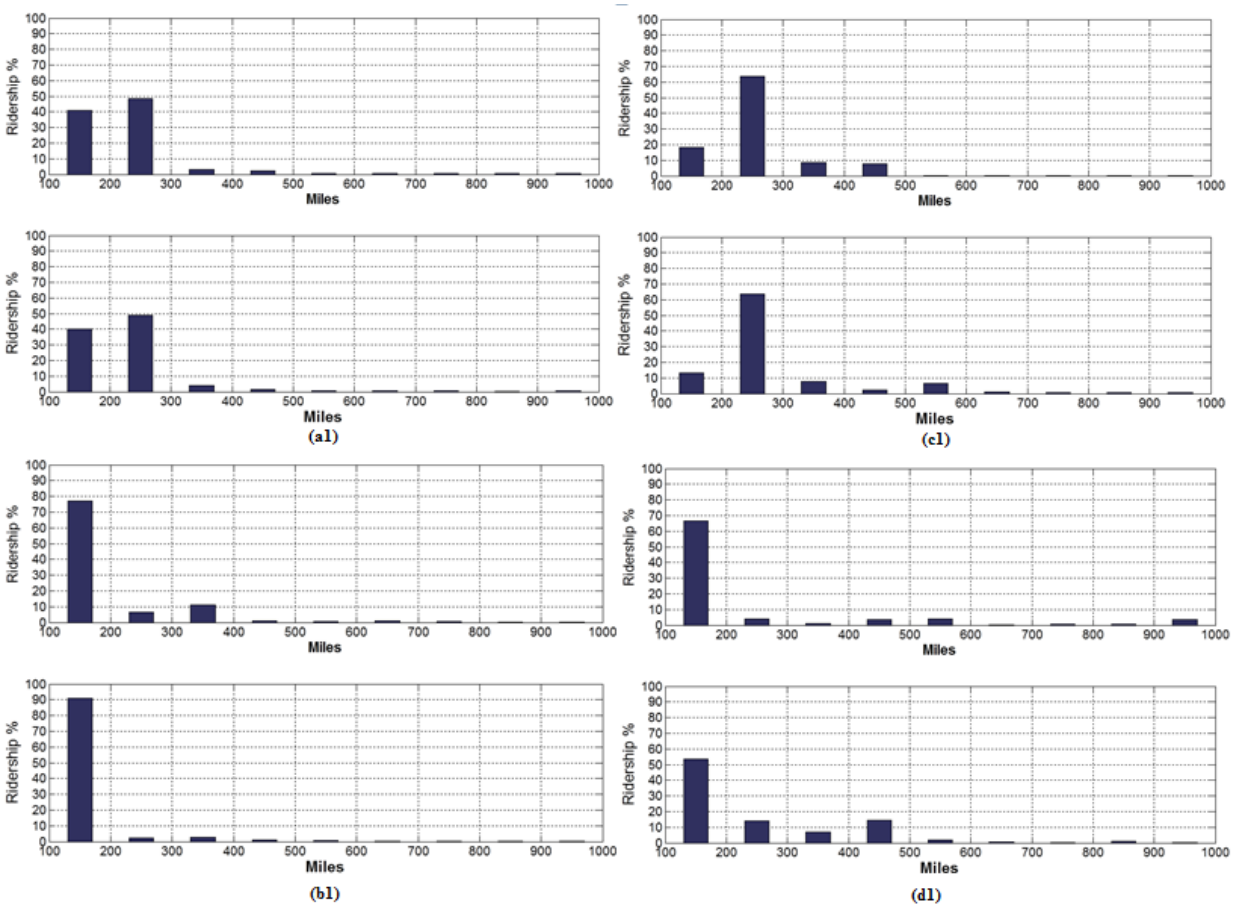


FIGURE 6 AMTRAK STATION TO STATION RIDERSHIP DISTRIBUTION WITH DISTANCE

Comparison for Selected Stations (Year 2009)

a. Penn Station, NY ($\Sigma\chi^2$= 0.79)

b. Philadelphia, PA ($\Sigma\chi^2$= 12.89)

c. Boston, MA ($\Sigma\chi^2$= 114.75)

d. Los Angeles, CA ($\Sigma\chi^2$= 96.93)

2. Amtrak Estimated Trend 2. Model Estimated Trend

3.10 HIGH-SPEED RAIL PREDICTION

One of the fundamental principles for developing this model is to estimate future demand ridership. In the future, the level of service parameters of the rail system and socio-economics are expected to change, thus inducing changes in demand. TSAM employs socio-economics forecast developed by Woods and Poole Economics *(22)*. The demand response for HSR is shown in Figure 7. Figure 7 (a) shows the change in annual nationwide rail ridership in the next three decades after the introduction of HSR. An increase in annual ridership of about 23 million round trips is predicted by the year 2040 nationwide. As seen from Figure 7(b), most of this ridership comes from the automobile mode - a surprising result. The speed performance parameters of the HSR rail system modeled assume a line speed of 130 mph (209 kph) and a cost 0.30$/mile in 2000$ (approximately 0.40 $/mile in 2012$). The HSR has been modeled currently for the following corridors: North-East Acela, California, Pacific Northwest, Florida, Chicago, Southeast, Empire, Northern New England, Keystone, South Central, Gulf coast and Vermont Routes. For this analysis, the HSR performance functions were activated in all these planned corridors.

An analysis of the market share for automobile, commercial air and rail in the New York area in 2040 predicts an increase of 7.3% share with the introduction of HSR. For Washington DC it is about 7.2%, while in the Philadelphia County, PA the increase is by 2.6%.

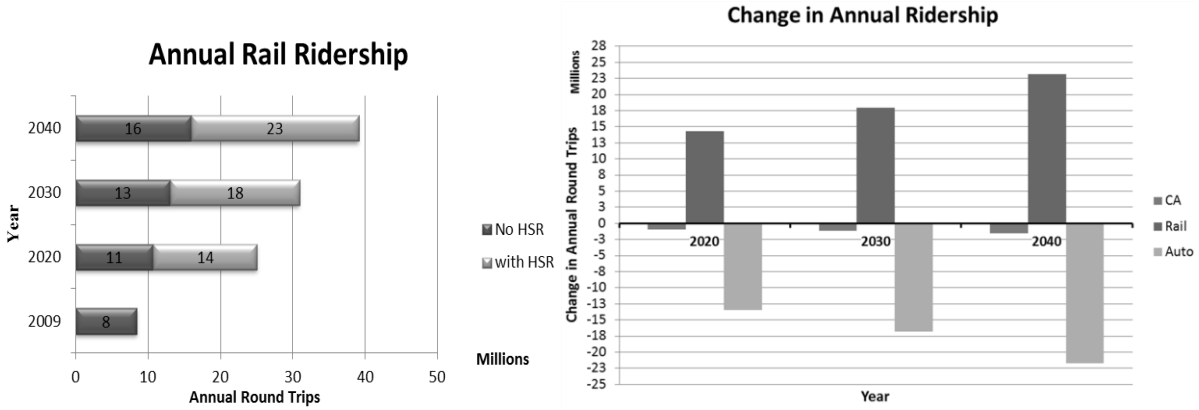


FIGURE 7 HSR EFFECT ON ANNUAL ROUND TRIP RIDERSHIP.

(a) Effect on Rail Ridership
(b) Change in Ridership for all modes

3.11 CONCLUSION

A tool has been developed to model and forecast nationwide rail demand. The model developed has been integrated into TSAM - the Transportation Systems Analysis Model. The market share for commercial air, automobile and rail can be estimated at three levels of fidelity: nationwide, county and station levels. The model developed employs a modified Box-Cox model that is able to better capture travel behavior over distance, as compared to traditional logit models.

The model is a flexible tool to predict rail ridership demand and serve as a decision making tool to estimate rail demand impacts on other modes of transportation. With simplified assumptions, the model in its current state can be used to study the effect of HSR in relieving the congestion in National Airspace Systems (NAS) and along the highways. The model can also be useful to the Federal Aviation Administration (FAA) and to airlines to study policy changes required with the introduction of HSR to maintain their market. Finally, the model could prove useful to airport authorities, AMTRAK, various Federal Agencies like the U.S. Department of Transportation, Federal Highway Administration (FHWA), and Federal Transit Administration (FTA) to understand the interplay between various modes of transportation under various policies.

3.12 RECOMMENDATIONS

To improve the proposed model, more details for AMTRAK data are needed. For example, transfer disutility must be improved. Modeling improvements to the rail station assignment model are the next aimed developments. The rail model is still in its nascent stage. Detailed data will help improve the calibration and gauge the credibility of assumed parameters. The current process of data collection employed by ATS must be revised in order to make the data more useful to the researchers. Detailed information about the zip code level origin-destination locations, costs and users actual airport/station selection data would also enhance usability. The induced demand factor, and possible policy changes in aviation due to HSR introduction must be studied and incorporated.

ACKNOWLEDGEMENTS

The authors thank NASA Langley Research Center for its support in developing this model. The authors are grateful to Jeff Viken, Samuel Dollyhigh and Teck-Seng Kwa from NASA, for their valuable inputs, constructive criticism and comments.

3.13 REFERENCES

1. Trani, A. A., H. Baik, H. Swingle, and S. Ashiabor. Integrated Model for Studying Small Aircraft Transportation System. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1850*, Transportation Research Board of the National Academies, Washington, D.C., 2003, pp. 1–10.
2. Ashiabor, S., H. Baik, and A. A. Trani. Logit Models for Forecasting Nationwide Intercity Travel Demand in the United States. In *Transportation Research Record: Journal of the Transportation Research Board, No. 2007*, Transportation Research Board of the National Academies, Washington, D.C., 2007, pp. 1–12.
3. Baik, H., A. A. Trani, N. Hinze, H. Swingle , S. Ashiabor and A. Seshadri. Forecasting Model for Air Taxi, Commercial Airline, and Automobile Demand in the United States. In *Transportation Research Record: Journal of the Transportation Research Board, No. 2052*, Transportation Research Board of the National Academies, Washington, D.C., 2008, pp. 9-20.
4. American Travel Survey: Technical Documentation. Bureau of Transportation Statistics, U.S. Department of Transportation, 1995.
5. Clifford.B., The Acela Express. Japan Railway & Transport Review, Issue 40, 2005, p. 18-21: ill.
6. Stopher, P., and J. Prashker. Intercity Passenger Forecasting: The Use of Current Travel Forecasting Procedures. *Proc., Annual Meeting of the Transportation Research Forum*, 1976, pp. 67–75.
7. Grayson, A. Disaggregate Model of Mode Choice in Intercity Travel. In *Transportation Research Record 385,* Transportation Research Board, Washington, D.C., 1981, pp. 36–42.
8. Morrison, S., and C. Winston. An Econometric Analysis of the Demand for Intercity Passenger Transportation. *Research in Transportation Economics*, Vol. 2, 1985, pp. 213–237.
9. Koppelman, F. S. Multidimensional Model System for Intercity Travel Choice Behavior. In *Transportation Research Record 1241*, Transportation Research Board, Washington, D.C., 1989, pp. 1–8.
10. Koppelman, F. S., G. Kuah, and M. Hirsh. *Review of Intercity Passenger Travel Demand Modeling: Mid 60's to the Mid 80's*. Transportation Center, Department of Civil Engineering, Northwestern University, Evanston, Ill. 1984
11. Albalate, D and Bel, G . High-speed rail: lessons for policy-makers from abroad. Research Institute of Applied Economics. Barcelona: University of Barcelona. http://www.ub.edu/irea/working_papers/2010/201003.pdf. Accessed Mar. 5, 2011.
12. Pagliara, F., J. M. Vassallo and C. Roman. High-Speed vs. air transportation: the Madrid Barcelona case study. In *Transportation Research Record: Journal of the Transportation Research Board,* No. 0553, Transportation Research Board 91st Annual Meeting, Washington, D.C., 2012.
13. López-Pita, A. and F. Robusté. Impact of High-Speed Lines in Relation to Very High Frequency Air Services. *Journal of Public Transportation, Vol. 8 (2), 2005,* pp. 17-35.
14. H. F. Gunn, et al. High-Speed Rail Market Projection - Survey Design and Analysis. *Transportation, vol. 19*, pp. 117-139, 1992.
15. . A disaggregate Box-Cox Logit mode choice model of intercity passenger travel in Germany and its implications for high-speed rail demand forecasts. The Annals of Regional Science, 1997, 31(2), 99-120.
16. National Household Travel Survey. FHWA, U.S. Department of Transportation, 2001. nhts.ornl.gov. Accessed June. 12, 2011.
17. *MapPoint: Business Mapping and Data Visualization Software.* Microsoft Corporation, Redmond, Wash., 2004.
18. Monthly Performance Report for January 2011. AMTRAK. http://www.AMTRAK.com/servlet/ContentServer?c=Page&pagename=am%2FLayout&cid=1241245669222. Accessed June. 12, 2011
19. Teodorovic, D. Airline Operations Research. Gordon and Breach Science. Publishers, New York, 1998.
20. Fact Sheets for all AMTRAK stations. National Association of Railroad Passengers. http://www.narprail.org/resources/fact-sheets/321-analysis-and-fact-sheets-of-AMTRAK-ridership. Accessed June. 12, 2011.
21. Transportation Statistics Annual Report 1995. Bureau of Transportation Statistics, U.S. Department of Transportation, 1995.
22. Complete Economic and Demographic Data Source 2005. Woods and Poole Economics, Washington, D.C., 2005.

4 MODEL DESCRIPTION

4.1 EXISTING MODEL

The previous version of TSAM had a train station network developed for 464 stations (Joshi 2005). The latitude-longitude coordinates of the train stations were used to compute the distances between the population centroids and stations, to evaluate the decision variables. Further work on TSAM allowed it to model the HSR network in eleven proposed HSR corridors (Vandyke 2011). Table 10 shows the list of major cities that are expected to be served by the corridors, included in TSAM. The city names that are in italics and underlined are not currently modeled, and the station list will need to be expanded in the future.

A nationwide travel time and cost function was developed to compute the decision variables. Processing time was estimated at 0.33 hours, and included in the computation of total travel time. Schedule delay was considered as described earlier in Section 3.6.3. The effect of transfer times was partly included, as the travel times were computed from AMTRAK data for a nationwide level, selecting random OD pairs. The following section discusses the TSAM results, as of June, 2011.

TABLE 10 MAJOR CITIES ALONG PROPOSED HIGH-SPEED RAIL CORRIDORS

Proposed Corridors	Major Cities Served by Each Corridor								
California	San Francisco, CA	San Jose, CA	Sacramento, CA	Merced, CA	Fresno, CA	Bakersfield, CA	Los Angeles, CA	Riverside, CA	San Diego, CA
Northeast Corridor	Boston, MA	New Haven, CT	New York, NY	Philadelphia, PA	Baltimore, MD	Washington, DC			
Pacific Northwest	*Vancouver, BC*	Seattle, WA	Tacoma, WA	Portland, OR	Eugene, OR				
Florida	Tampa, FL	Orlando, FL	Miami, FL						
Chicago Hub	Pontiac, MI	Detroit, MI	Kalamazoo, MI	Chicago, IL	St. Louis, MO	Kansas City, MO	*Iowa City, IA*	Omaha, NE	
Southeast	Charlotte, NC	Greensboro, NC	Raleigh, NC	Richmond, VA	Washington, DC				
Empire	New York, NY	Albany, NY	Rochester, NY	Buffalo, NY	Niagara Falls, NY				
Northern New England	*Montreal, QC*	Albany, NY	Springfield, MA	New Haven, CT	Boston, MA	*Portland, ME*			
Keystone	Philadelphia, PA	Harrisburg, PA	Pittsburgh, PA						
South Central	*Tulsa, OK*	Oklahoma City, OK	Dallas, TX	Fort Worth, TX	Austin, TX	San Antonio, TX	Texarkana, AR	Little Rock, AR	
Gulf Coast	Atlanta, GA	Birmingham, AL	Meridian, AL	New Orleans, LA	Houston, TX	Biloxi, MS	Mobile, AL		
Vermont	St. Albans, VT	Montpelier, VT	Northfield, MA	Springfield, MA	New Haven, CT				

4.1.1 *NORTH-EAST CORRIDOR*

Table 11 and Table 12 show the effect of introducing HSR in the NEC. The base ridership prediction for The Penn Station is about 77% lesser than the estimated AMTRAK ridership (eliminating short trips and transfers). Further, for the year 2030, without HSR the demand is estimated at 436,201 round trips. Also, it is observed from Table 12, that the change in commercial air ridership due to HSR is negligible. It is of the order of 0.6% for JFK, in the year 2030. The increase in rail demand in NYP is estimated at more than 1 million round trips in 2030, while the decrease in commercial air demand for JFK and LGA is around 100,000 round trips.

TABLE 11 Effect of HSR in NEC on Rail Market (TSAM 6.4, 175 mph HSR Speed, $0.20/mile 2010 HSR Cost)

Station Name		Without HSR			With HSR	
		2011	2020	2030	2020	2030
NYP	New York (Penn Sta.)	348,050	386,274	439,201	1,277,341	1,491,954
BBY	Back Bay - Boston	49,015	53,545	60,401	242,776	279,977
NWK	Newark	152,058	175,416	209,443	592,622	714,582
WAS	Washington	199,076	225,749	265,813	953,378	1,128,669
PHL	Philadelphia	181,170	204,832	240,726	646,764	772,475
BAL	Baltimore	93,380	104,171	119,324	405,503	478,421
MET	Metro park	114,001	135,657	164,029	469,379	574,155

TABLE 12 Effect of HSR in NEC on Commercial Air Market (TSAM 6.4, 175 mph HSR Speed, $0.20/mile 2010 HSR Cost)

Airport Name		Without HSR			With HSR	
		2011	2020	2030	2020	2030
JFK	John F Kennedy Intl	9,264,383	10,605,724	12,447,682	10,545,484	12,373,908
LGA	La Guardia	9,196,054	10,561,112	12,463,218	10,456,279	12,334,666
EWR	Newark Liberty Intl	6,101,638	7,066,690	8,405,922	7,036,406	8,368,208
HPN	Westchester County	2,091,199	2,398,383	2,810,154	2,379,639	2,787,433
ISP	Long Island Mac Arthur	992,879	1,103,451	1,242,150	1,093,356	1,230,034
SWF	Stewart Intl	243,978	275,693	315,580	275,599	315,489

The plots in Figure 8 show a sensitivity analysis for demand in the year 2030, as a factor of cost of HSR in TSAM 6.4 at 175 mph. The trends indicate a high relation of demand to cost. A reduction in cost from 40 cents/ mile to 20 cents/mile boosts the train demand by 1.5 million round trips. Again, the change in commercial air demand is only about 0.5 million round trips.

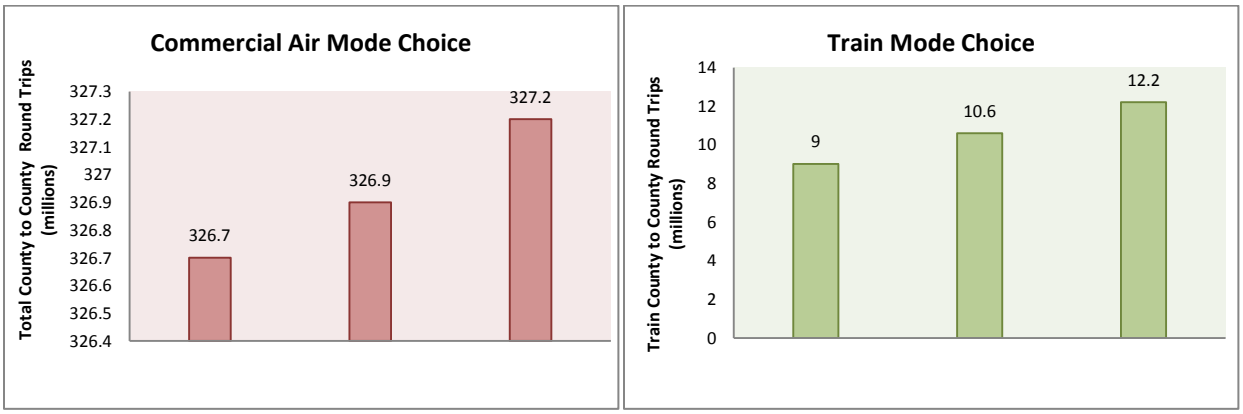


FIGURE 8 Sensitivity Analyses for Commercial Air and Rail Demand for 2030 (TSAM 6.4).

A market share analysis using TSAM 6.5 (175 mph speed and 20cents/mile cost) is presented in the Table 13 and Table 14. Amongst counties with high AMTRAK demand, two counties were selected for analysis. The changes in market share for New York County and Suffolk County, MA are tabulated in the Table 13 and Table 14. It also shows the sensitivity of the demand with respect to the trip purpose. Business rail ridership market is seen to be more sensitive to the introduction of HSR. Also, the automobile ridership is seen to vary more than the commercial air. For all purpose trips in the New York County, automobile demand varies by around 5%, compared to the 0.6% change in commercial air demand.

TABLE 13 Market Share Analyses for New York County, NY (TSAM 6.5, 175 mph, 20cents/mile, 2030)

		Auto %	Commercial Air %	Rail %
Without HSR	All	67.1	31.1	1.8
	Business	41.0	56.5	2.5
	Non-business	75.0	23.4	1.6
With HSR	All	62.4	30.5	7.1
	Business	34.2	54.7	11.0
	Non-business	70.9	23.2	5.9

TABLE 14 Market Share Analyses for Suffolk County, MA (TSAM 6.5, 175 mph, 20cents/mile, 2030)

		Auto %	Commercial Air %	Rail %
Without HSR	All	78.6	20.0	1.3
	Business	58.5	38.8	2.7
	Non-business	83.8	15.2	1.0
With HSR	All	75.0	19.5	5.8
	Business	48.8	35.5	15.7
	Non-business	81.7	15.0	3.3

4.1.2 *CALIFORNIA CORRIDOR*

A similar analysis for the California corridor is shown in the Table 15 and Table 16. The increase in train ridership in Glendale is about 1.7 million. This is about 60% more increase, that that observed in the previous analysis for Penn station in the North-East. Similar to the previous trend, the train demand seems to be extracted mostly from automobiles, with a small percentage of riders shifting from the commercial air mode. Figure 9 shows sensitivity plots for train and commercial air demand, to the cost of the HSR. Interestingly, the train demand is also not seen to vary significantly depending on the corridor in which the HSR is operative. The total county to county round train trips, generated by an HSR at 40cents/mile in the NEC attract 0.1 million more round trips than a similar HSR in California Corridor.

TABLE 15 Effect of HSR in California Corridor on the Rail Market (TSAM 6.4, 175 mph HSR Speed, $0.20/mile 2010 HSR Cost)

Station Name		Without HSR			With HSR	
		2011	2020	2030	2020	2030
GDL	Glendale	349,062	366,364	390,136	1,779,815	2,094,705
SAN	San Diego	112,684	130,542	152,455	849,658	1,067,333
SJC	San Jose	118,475	131,021	146,193	845,545	1,018,949
SAC	Sacramento	68,028	76,952	88,108	362,980	453,119
FNO	Fresno	56,424	61,070	66,750	359,815	435,038

TABLE 16 Effect of HSR in California Corridor on the Commercial Air Market (TSAM 6.4, 175 mph HSR Speed, $0.20/mile 2010 HSR Cost)

Airport Name		Without HSR			With HSR	
		2011	2020	2030	2020	2030
LAX	Los Angeles Intl	14,488,891	16,632,515	19,533,790	16,323,322	19,155,909
SFO	San Francisco Intl	9,614,201	11,040,772	12,954,950	10,773,856	12,627,398
SNA	Santa Ana	5,165,690	6,083,672	7,281,177	5,982,551	7,155,647
ONT	Ontario	3,585,086	4,263,806	5,160,648	4,204,559	5,086,217
BUR	Burbank	3,291,698	3,816,311	4,505,631	3,732,952	4,403,729
SAN	San Diego	2,812,757	3,467,400	4,387,081	3,413,124	4,314,863
LGB	Long Beach	2,474,878	2,885,901	3,450,514	2,844,545	3,398,924

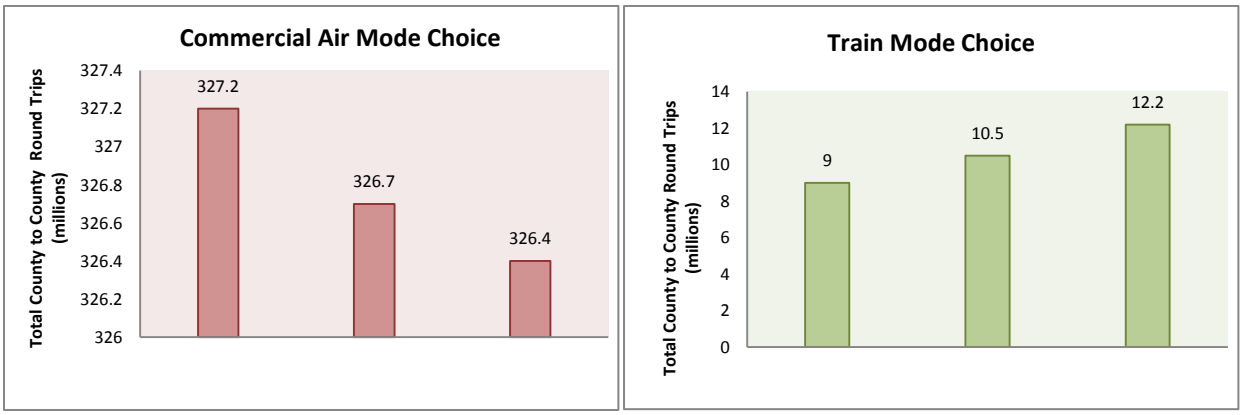


FIGURE 9 SENSITIVITY ANALYSES FOR COMMERCIAL AIR AND RAIL DEMAND FOR 2030 (TSAM 6.4, YEAR 2030, 175 MPH).

A market share analysis for business and non-business trip purposes is presented in the Table 17. The market share for the year 2030 is estimated using TSAM 6.5, with train speed at 175 mph and cost 20 cents/mile. Similar to the NEC, significant demand shift is seen in the business class with the introduction of HSR. The increase in business riders is about 4 -5 times the non-business train demand.

TABLE 17 Market Share Analyses for Suffolk County, MA (TSAM 6.5, 175 mph, 20cents/mile, 2030)

Los Angeles County		Auto %	Commercial Air %	Train %
Without HSR	All	72.1	27.4	0.5
	Business	54.3	45.3	0.4
	Non-business	77.1	23.4	0.5
With HSR	All	68.9	26.4	4.7
	Business	47.8	40.7	11.4
	Non-business	77	20.7	2.3
Santa Clara County		Auto %	Commercial Air %	Train %
Without HSR	All	62.4	36.8	0.7
	Business	41.6	57.9	0.5
	Non-business	69.6	29.5	0.8
With HSR	All	60.1	30.6	9.3
	Business	33.2	44.8	22
	Non-business	68.7	26	5.3
San Francisco County		Auto %	Commercial Air %	Train %
Without HSR	All	62.5	36.8	0.7
	Business	37.8	61.7	0.5
	Non-business	70.5	28.7	0.8
With HSR	All	63.7	31.7	4.5
	Business	35.8	52.9	11.3
	Non-business	71.9	25.5	2.5

4.1.3 *COMPARISON TO AMTRAK DATA*

The filtered AMTRAK data was compared to the output of TSAM 6.5, to validate the demand prediction for trains. As seen in Table 18, the predictions by TSAM are significantly off, and the model over predicts some areas and under predicts the others. The model seems to over predict the ridership in Milwaukee by 42% while it under predicts the demand in Washington DC by around 17%.

TABLE 18 Comparison of TSAM 6.5 and AMTRAK – Year 2009

City, State	TSAM 6.5 (2009)	AMTRAK (2009)
Glendale, CA	233,792	7,477
San Diego, CA	69,357	221,259
San Jose, CA	80,729	62,721
Washington, DC	260,880	1,870,251
New York – Penn, NY	383,630	2,906,515
Baltimore, MD	82,772	260,262
New Haven , CT	51,920	146,453
Boston-South Sta., MA	269,844	530,626
Philadelphia-30th St., PA	322,207	718,919
Milwaukee, WI	117,330	12,189

The distribution of demand over the distance travelled revealed that TSAM also predicts demand for long-range trips of about 2000 miles, which is not reflected in the data that AMTRAK provided. The difference in the TSAM and AMTRAK demand needs to be reduced. Different calibrations were attempted for it, and are described in the next chapter.

4.2 ATTEMPTED MODELS

The attempted models are summarized in Table 19. The basic Box-Cox logit model is represented by the 1st equation in Table 19. It could not adequately catch the characteristics of the NEC resulting in low ridership estimates. It has been observed that AMTRAK is preferred over commercial air and automobiles by business travelers. Available Wi-Fi, comfortable leg space and lesser time wastages due to security, taxiway, and driving make AMTRAK a more productive means of transport. The calibration attempted to introduce a factor other than time and cost to give the mode choice of rail an added degree of freedom, making the choice more attractive to business travelers. On account of limited surveys done regarding this, parameters were assumed to formulate the model. The productive times were attempted to be incorporated as a decision variable. The 2nd and 3rd models had a high rate of correlation between productive time and travel time.

Table 19 Summary of Attempted Models

1.	Box-Cox Logit Model	$Utility = \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
2.	Productive Time	$Utility = \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}}) + \alpha_{PT}(PT^{\lambda_{PT}})$
3.	Productive Time only	$Utility = \alpha_{TC}(TC^{\lambda_{TC}}) + \alpha_{PT}(PT^{\lambda_{PT}})$
4.	Unproductive Time only	$Utility = \alpha_{TC}(TC^{\lambda_{TC}}) + \alpha_{UPT}(UPT^{\lambda_{UPT}})$
5.	Productive Time Ratio	$Utility = \alpha_{TC}(TC^{\lambda_{TC}}) + \alpha_{PTR}(PTR^{\lambda_{PTR}})$
6.	Productive and Unproductive Time	$Utility = \alpha_{UPT}(UPT^{\lambda_{UPT}}) + \alpha_{TC}(TC^{\lambda_{TC}}) + \alpha_{PT}(PT^{\lambda_{PT}})$
7.	Filtered data To/From NEC	$Utility = \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
8.	Filtered data trips within NEC	$Utility = \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
9.	Filtered data for counties with 3 modes	$Utility = \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
10.	Mode Specific Constants	$Utility = \gamma_{(constant)} + \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
11.	Rail Mode Constant	$Utility = \gamma_{rail} + \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
12.	Constant For Rails in NEC	$Utility = \gamma_{rail\text{-}NEC} + \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$
13.	Constant For Rails in NY and DC	$Utility = \gamma_{rail\text{-}NY} + \gamma_{rail\text{-}DC} + \alpha_{TT}(TT^{\lambda_{TT}}) + \alpha_{TC}(TC^{\lambda_{TC}})$

4.2.1 *COMPUTATION OF PRODUCTIVE TIME*

The productive time for commercial airlines and rail, was computed as a percentage of total time based on the concept shown in Figure 10. The time available for productivity would be the slack time available at the origin airport/station and the in-vehicle portion. The in-vehicle time for commercial air is further broken down, and the cruise portion is treated as the time with potential productivity. The productive time in the cruise is limited to 3 hours, on account of the limited battery life for laptops, and absence of chargers for most of the seats. The cruise times was worked out on a regression analysis as a function of distance from the T-100 data.

For rail, a little time would be spent on set-up and getting comfortable. Other than that, the entire in-vehicle time could potentially be used productively, but the productivity would decrease after a certain time. Hence, only a percentage of the time is considered productive to even out the productivity, and for ease of computation. The flight productivity was assumed at a lower rate than the rail. The work cap imposed on train is 10 hours assuming Wi-Fi and battery life are not limiting factors, and the total working hours of work in a day is the governing factor. The productive time for automobiles was assumed to be zero.

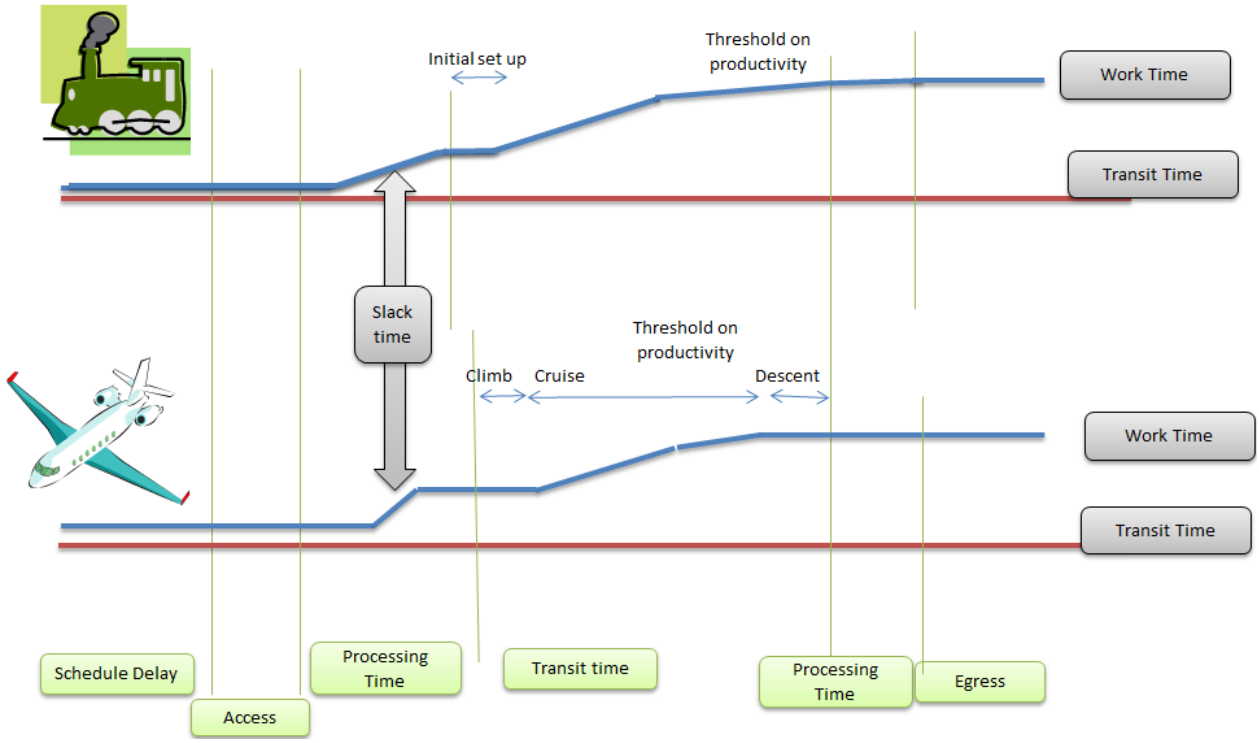


FIGURE 10 Concept of Productive Time.

4.2.1.1 Assumptions

- Station Slack time 75% (of processing time)
- Work Time Cap on Flights 3 hours
- % time worked on rail 90% (of in-rail time)
- Flight Productivity 75% (as compared to office productivity)
- Rail Productivity 98% (as compared to office productivity)
- Airport slack time 30% (of processing time)
- Work Time Cap on Rail 10 hours

The processing times currently input in TSAM are given by Table 20. Greater processing times are used for larger airports to signify higher time spent in queues and walking in the terminal.

TABLE 20 Processing Times at Airport and Stations in Tsam

Commercial Airports:

Hub Type	Processing time at origin (hrs)	Processing time at destination (hrs)
Large Hub	1.5	0.5
Medium Hub	1	0.5
Small Hub	0.75	0.33
Non Hub	0.75	0.33

Air Taxi Airports:

	Processing time at origin (hrs)	Processing time at destination (hrs)
Airport	0.5	0.25

Train Stations:

	Processing time at origin (hrs)	Processing time at destination (hrs)
Station	0.33	0.17

To compute the cruise time of flights, linear regression functions were developed from the BADA data, and the plot is shown in Figure 11. The computational procedure for productive time is described in the sections 4.2.1.2 and 4.2.1.3, for train and commercial air respectively.

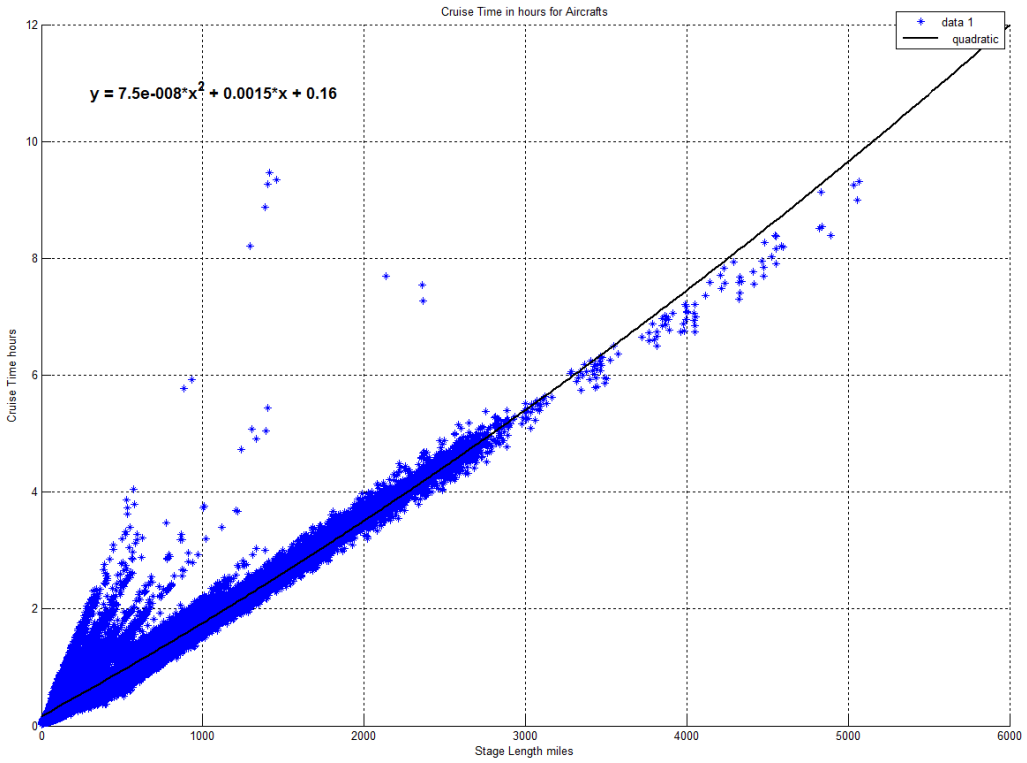
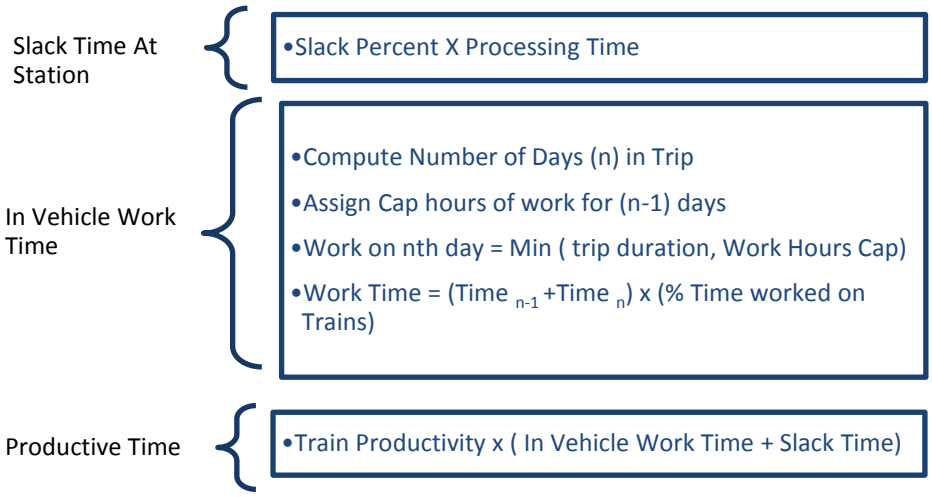
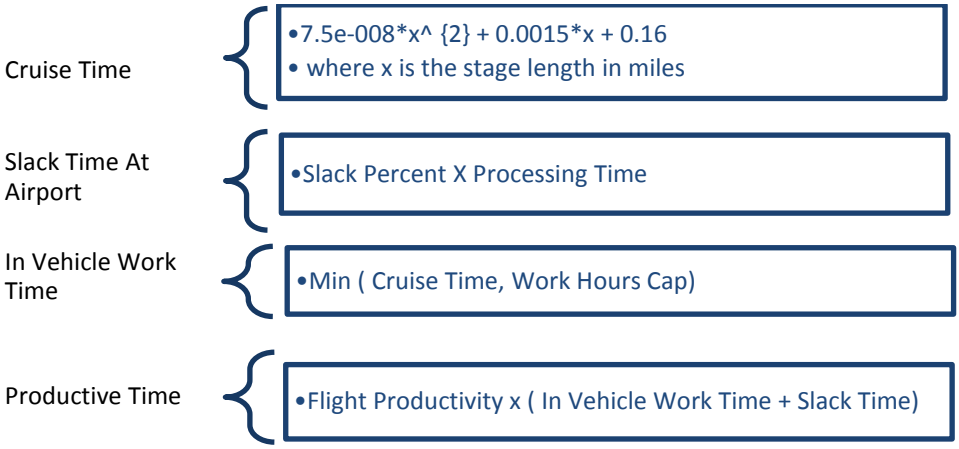


FIGURE 11 Cruise Time Plot for Aircraft.

4.2.1.2 Train productivity



4.2.1.3 Flight productivity



To reduce the correlation, the travel time was replaced by productive time and calibration was done using two variables. This approach failed to penalize the mode for the wasted time. For example, a commercial air trip of 4 hours was given the same utility as a 2 hours trip by train, due to similar productive times. The approach using only unproductive time has similar effects. Unproductive time was computed as the difference between total travel time and the productive time. The next approach was to use the productive time ratio, or productive time and unproductive time. Productive time ratio was computed as a ratio of the productive time and total travel time, and was replaced with the parameter of travel time, to reduce the correlation.

The NEC was attempted to be calibrated separately by filtering data. The models with filtering data resulted in few records that were not enough for a good calibration. Also, as many counties did not have rail data, the automobile and COMMERCIAL AIR modes were seen to have an advantage in the calibration. An attempt to filter the records for counties which show existence of all the three modes was attempted, to give equal utility weights to the three modes. Fewer records were the issue again with this method. The constant for NY and DC, seemed to have the best results among the tried models for the Non-Business trips. Business rail trips have fewer records that are not well

sampled over distance and hence the two constants could not be separately calibrated. A general mode constant for trains was calibrated for the NEC.

4.3 VALIDATION DATA

To compare the TSAM ridership to AMTRAK, AMTRAK data was converted to long-distance roundtrips. To eliminate the short distance trips, the AMTRAK station data distribution over distance was used. The average access distances to every station were extracted from TSAM. The egress distances were then assumed to be equal to the access distances. A linear relation of ridership over distance was assumed for the first 100 miles of data. A rough estimate of short distance trips for every station was estimated and filtered out. A summation of these trips at the national level results in about 13.3% of short trips eliminated in the year 2009.

The transfer rate estimates were incorporated in the station ridership for stations in Table 5. Once these factors were applied at the station level, the AMTRAK data shown is used for validation.

4.4 STATION CHOICE MODEL

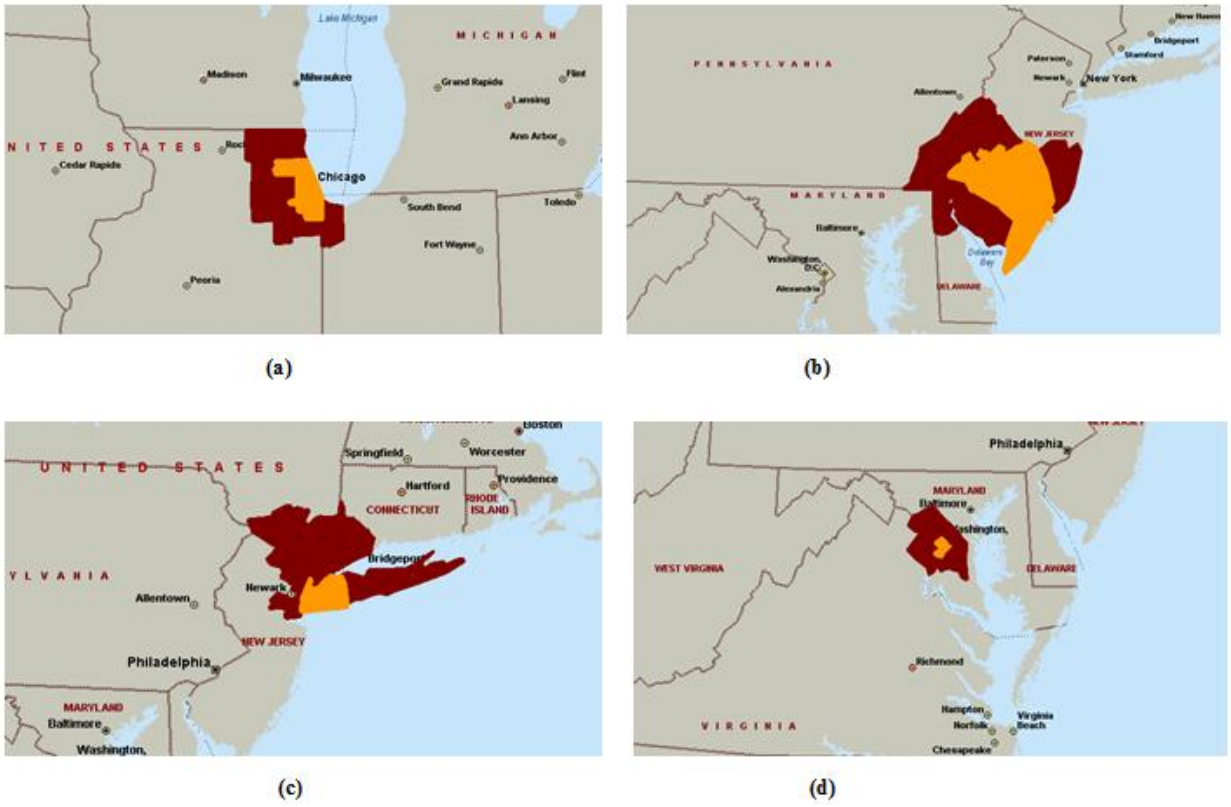


FIGURE 12 STATION CATCHMENT AREAS.
(A) Chicago, IL (B) Philadelphia, PA
(C) Penn Station, NY (D) Washington, DC

To develop the station choice model, a one county to one station assignment was employed. The stations with the least travel distances from the county population centroid were used to develop the initial iteration. Some of these selections were then modified for important rail stations. The stations with higher demand are expected to attract ridership from relatively farther counties as well. The assignment is then modified for the regions with a high rail commuter demand (Commuter Rail in North America n.d.). This would represent the "mode inertia", in areas like New York and DC. Also, areas with higher rail commuter demand are expected to have a lower car ownership percentage (List of U.S. Cities with Most Households without a Car 2012), this being a major flaw in using minimum driving distance as a part of the utility function. Figure 12 represents the modified station catchment area examples. The area in yellow is the catchment area when the assignment was based on least travel distances. The red area denotes the catchment area, considering mass transit access to the stations. The areas added to Penn station, NY (Figure 12 *(c)*) and Washington DC (Figure 12 *(d)*) are significant compared to those of Chicago, IL (Figure 12 *(a)*) and Philadelphia, PA Figure 12*(b)*).

Once the initial station assignment was performed, the assignment was modified further by trial and error. The rail demand from counties near selected major stations was directed towards it. This left the neighboring smaller stations with zero ridership. Further iterations were thus required. Validations were done with the filtered AMTRAK data at the station level, and approximate adjustments to some of the station assignments were done.

4.4.1 *CHICAGO STATION CATCHMENT AREA*

The counties assigned to Chicago AMTRAK station are shown in Table 21. Most of the counties have mass transit access, and the cost and times listed are assuming that all the people use the transit option to access the station. This assumption, while reduces the access cost, adds a time penalty. In future, an expected value of time and cost should be assigned. The costs used are for 2012 and are converted to the desired year using the inflation index, in the code that generates the input files.

TABLE 21 Chicago Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
Cook, IL	590	2.25	0.500	Chicago Transit Authority
Du Page, IL	596	5.25	1.000	Metra
Kane, IL	619	6.75	1.330	Metra
Kendall, IL	621	0.00	1.330	CAR
Mchenry, IL	630	7.25	1.600	Metra
Will, IL	673	6.75	1.083	Metra

The Kendall County and the Chicago station are not connected through mass transit and so, driving access is assigned to it (Google Maps n.d.). The access time used is driving time, while the access cost is set to a default zero. The input code filters out such stations with a default cost, while accounting for total travel cost. The access cost for such counties is then assigned based on the driving distance to the station.

4.4.2 *LOS ANGELES STATION CATCHMENT AREA*

The population centroid for Los Angeles is a 7 minute walk from the AMTRAK station, as computed from the Google maps (Google Maps n.d.). The county size for Los Angeles is quite huge, and it would be unfair to assume that walking would be a good mode of access. But due to limitations of the way the model is built, a minimum access time of 10 minutes and $2 access cost is assumed, shown in Table 22.

TABLE 22 Los Angeles Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
Los Angeles	186	2	0.1667	Foothill Transit
Ventura	223	22	1.5000	Car / Metro link

To reach LAX from the Ventura County, transit takes about 2 hours, while driving takes about 1 to 1.5 hours in congestion. As the difference is not significant in the congested conditions, it can be assumed that the split is 50%,

and an average value is assumed. The cost used is an average for the two costs again, $12 for Transit and $32 for driving (Google Maps n.d.).

4.4.3 *PENN STATION, NY CATCHMENT AREA*

The catchment area for NYP is shown in Table 23. All the counties have been assigned mass transit or bus access to the station. While there are multiple ways of getting to the station, the fastest route shown in Google maps transit is used. For counties like the Orange County, NY, the access times might make in unrealistic to travel all the way. Once the AMTRAK network has been developed, and transfer trips included, these counties could perhaps be reassigned.

TABLE 23 Penn Station Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
Fairfield, CT	289	12.00	2.0000	MTA, Metro-North Railroad
Bergen , NJ	1755	11.70	1.5000	MTA, NJ Transit Bus
Hudson, NJ	1762	4.25	0.7000	PATH
Passaic, NJ	1769	6.50	1.5000	NJ Transit Bus, NJ Transit Rail
Union, NJ	1773	11.50	1.000	NJ Transit Rail
Bronx, NY	1809	2.25	0.8330	MTA
Kings, NY	1830	2.25	0.5833	MTA
Nassau, NY	1836	15.00	1.3330	Long Island Rail Road, Nassau Inter-County Express
New York, NY	1837	2.25	0.5000	MTA
Orange, NY	1842	17.25	2.1667	NJ Transit Rail
Putnam, NY	1846	23.00	1.8333	MTA, Metro-North Railroad
Queens, NY	1847	2.25	0.4167	Long Island Rail Road
Richmond, NY	1849	2.25	1.5000	MTA
Suffolk, NY	1858	12.00	1.9500	Long Island Rail Road

4.4.4 *PROVIDENCE STATION CATCHMENT AREA*

TABLE 24 Providence Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
Bristol, RI	2290	2	0.830	RI Public Transit Authority
Kent, RI	2291	0	0.333	Driving
Newport, RI	2292	2	1.333	RI Public Transit Authority
Providence, RI	2293	2	0.350	RI Public Transit Authority

4.4.5 *PHILADELPHIA STATION CATCHMENT AREA*

For the Philadelphia AMTRAK station catchment counties shown in Table 25, it can be seen that most of the counties drive down to the station, according to the assumption. To understand this, let's take the example of the New Castle County population centroid in Delaware. The mass transit access takes 2 hours, while driving takes 45 minutes. Also, it takes 1 hour from the population centroid of Camden County via mass transit to reach the station, compared to 23 minutes of driving time. Philadelphia has a high commuter rail demand and a lower car ownership percentage (Commuter Rail in North America n.d.). Due to the limitations of the model and data available, the assignment has been based on logic and judgment, and should be checked and corrected in the future.

TABLE 25 Philadelphia Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
New Castle, DE	298	0	0.75	Driving
Atlantic, NJ	1754	0	1	Driving
Burlington, NJ	1756	0	0.4167	Driving
Camden, NJ	1757	4	1	Port Authority Transit Corporation SEPTA
Cape May, NJ	1758	0	1.33	Driving
Cumberland, NJ	1759	0	1	Driving
Ocean, NJ	1768	0	1.33	Driving
Salem, NJ	1770	0	0.9	Driving
Bucks, PA	2231	0	0.78	Driving
Chester, PA	2237	0	0.9	Driving
Delaware, PA	2245	3	0.85	SEPTA
Montgomery, PA	2268	5.5	1.66	SEPTA
Philadelphia, PA	2273	3	0.65	SEPTA

4.4.6 *WASHINGTON DC STATION CATCHMENT AREA*

TABLE 26 Washington Dc Rail Station Catchment Counties

County Name	County Number	Access Cost ($ 2012)	Access Time (hours)	Access Mode
District of Columbia	300	3	0.2	Metro
Montgomery, MD	1187	5	0.9	Metro, Montgomery County MD Ride On
Arlington, VA	2803	5	0.7	Metro
Loudoun, VA	2839	0	1	Driving
Fairfax, Fairfax City, Falls Church, VA	2887	5	1.5	Metro
Prince William, Manassas, Manassas Park	2895	0	0.9	Driving

4.5 TIME COST FUNCTIONS FOR AMTRAK

The time cost functions over distance were developed for a few AMTRAK lines, and are shown in Appendix A. A random future date was assumed for the computations. These functions were incorporated in TSAM

to compute the station to station travel time and costs. There are about 31 AMTRAK lines, and developing time and cost curves for all the lines is a tedious job. These functions were only developed for a few important routes. Developing these functions for all the routes could be a possible future amendment in TSAM.

4.6 MODEL RESULTS

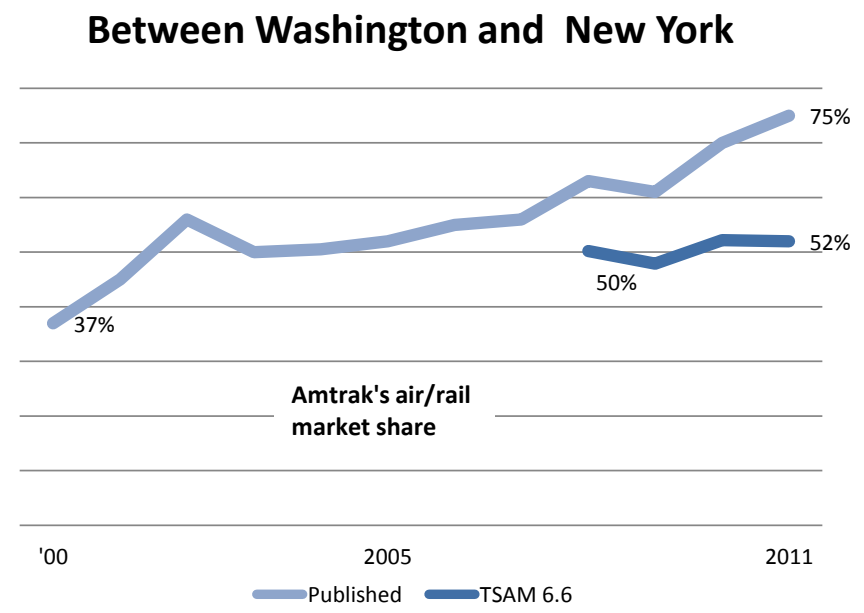
The model with the calibration coefficients presented in Table 8 and Table 9 was employed in the form of the Equations 6-9. Table 27 shows the change in ridership at the national level for the three modes and through different income groups. The results shown are assuming an HSR at 130 mph (209 kph) and a cost 0.30$/mile in 2000$ (approximately 0.40 $/mile in 2012$). It is evident that most of the train ridership comes from the automobile mode.

TABLE 27 Total County To County Household Round Trips (Millions)

	Year 2040						
No HSR	Mode	<$29k	$29k to $57k	$57k to $86k	$86k to $144k	>$144k	Total
	Auto	47.3	195.4	442.5	291.9	76.7	1,053.8
	Commercial Air	6.7	32.1	119.7	119.4	63.8	341.7
	Rail	0.6	1.2	3.6	4.3	0.9	10.6
HSR	Mode	<$29k	$29k to $57k	$57k to $86k	$86k to $144k	>$144k	Total
	Auto	46.7	194.1	440.2	285.2	74.1	1,040.3
	Commercial Air	6.7	32.0	119.4	118.9	63.5	340.5
	Rail	1.2	2.6	6.2	11.5	3.7	25.2

The model is currently unable to correctly predict the impact of train service on commercial air. This can be further illustrated by the comparison shown in Figure 13 (a) and (b). The base data for the comparison is reproduced from an article in the New York Times (Frustration of Air Travel Push Passengers to Amtrak 2012). The actual numbers on the graphs were not published in the article, and hence the graphs are reproduced based on judgment. Also, the area considered for the comparisons was not clear in the article. Hence, the comparison is based on the major stations of NYP, WAS and BOS, and the stations close to them. Figure 13 (a) depicts the AMTRAK share in the air rail market between Washington and New York. The stations included in the analysis are Newark, Penn station, Yonkers and Metropark for NY market. The stations included in the analysis of market in Washington are Washington, Alexandria and New Carrollton. Figure 13 (b) represents the share between New York and Boston, and stations included for Boston market are Boston South Station, Route 128, and Framingham. The trends show a 20% market share difference between commercial air and train. The reason for such a difference could be attributed to better productivity, reduced delays and inconvenience due to security and processing times offered by the train system. As the later factors have been incorporated to some extent in the model, productivity is probably the lacking factor. The article also points out that AMTRAK expects the ridership of the NEC to increase to 43.5 million passengers by 2040 with the improvements, based on predictions by an unknown method. The current model in TSAM predicts a nationwide ridership of 39 million round rail round trips in 2040 with HSR. Based on the current 1/3 ratio of NEC ridership in total AMTRAK ridership, the NEC ridership in 2040 by TSAM prediction is around 26 million one way trips.

It is also reasonable to assume that NEC would account for more than 1/3rd of the share once HSR sets in. It is difficult to further comprehend the difference in prediction with the current data.



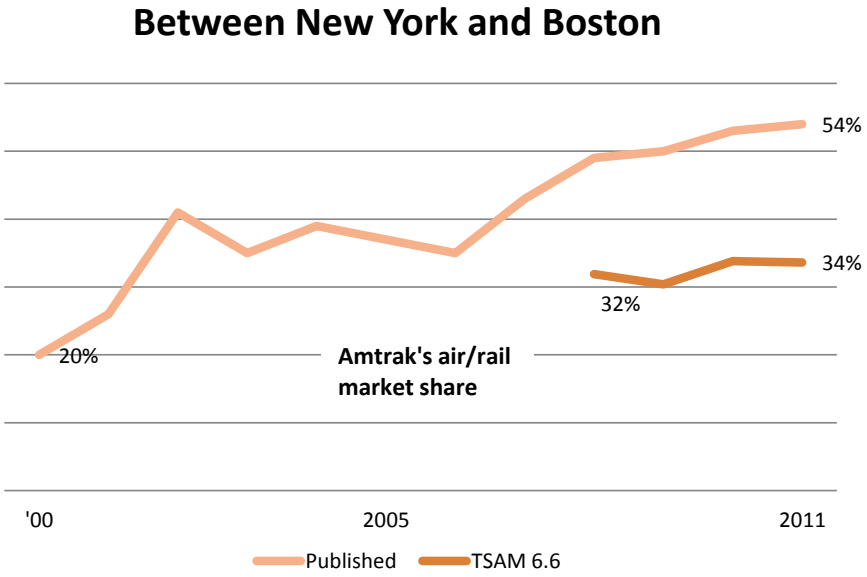
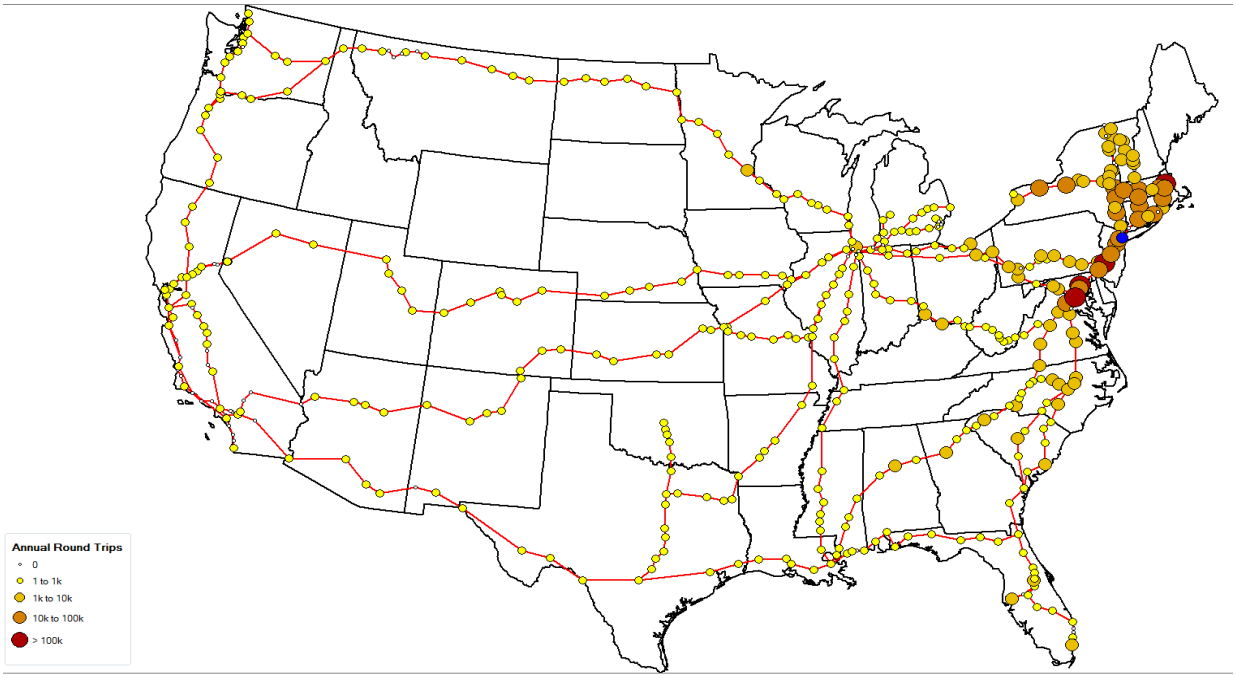
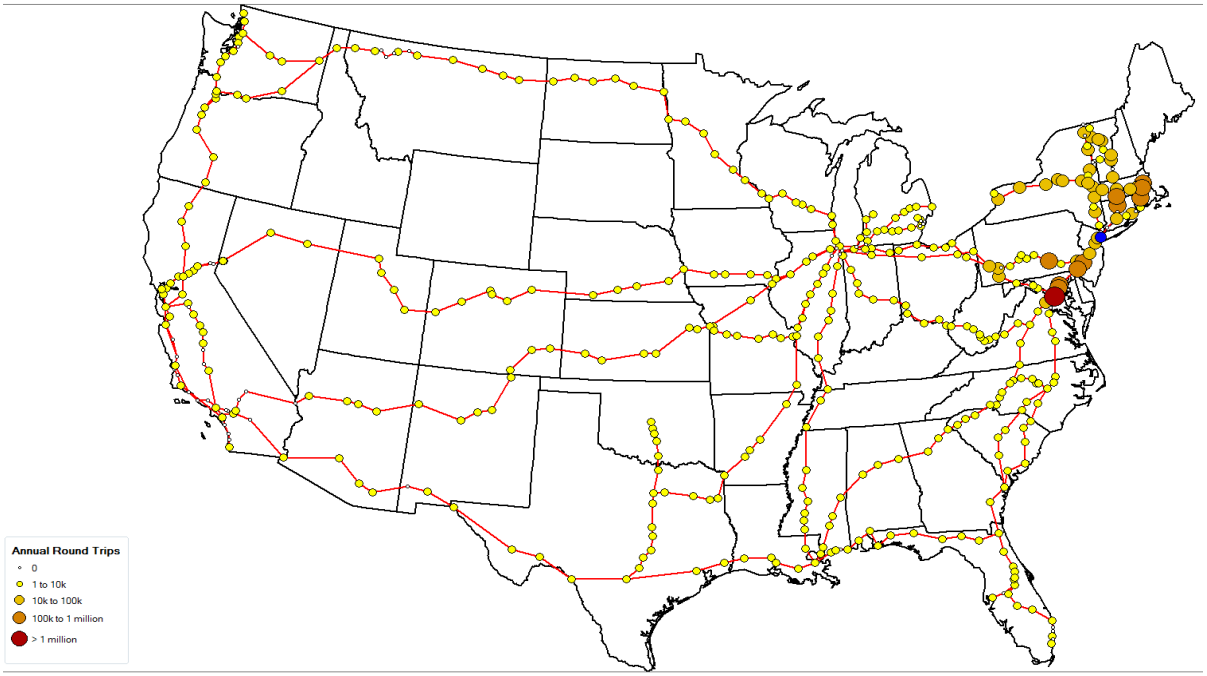


FIGURE 13 AMTRAK'S SHARE OF THE PASSENGER AIR AND RAIL MARKET IN THE NORTH EAST CORRIDOR.

Figure 14(a) and (b) show the difference in demand generated in 2040 at New York's Penn station with the introduction of HSR. There is a 111% increase in the ridership between Penn station and Washington. For an O-D pair like Penn station and Philadelphia, the increase is about 15%.



(A)

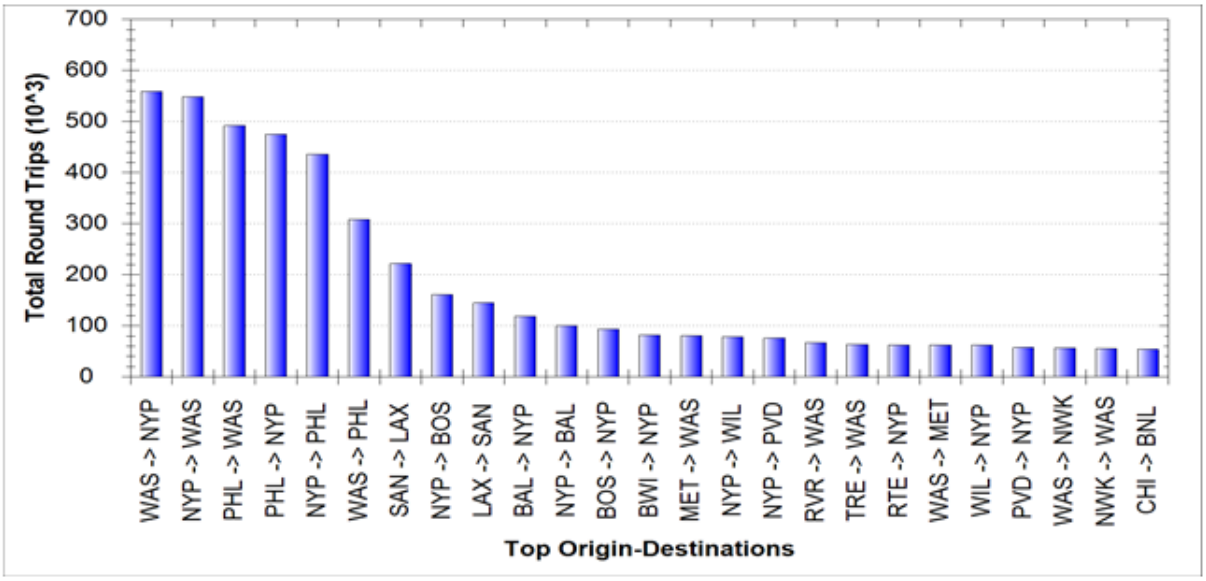


(B)

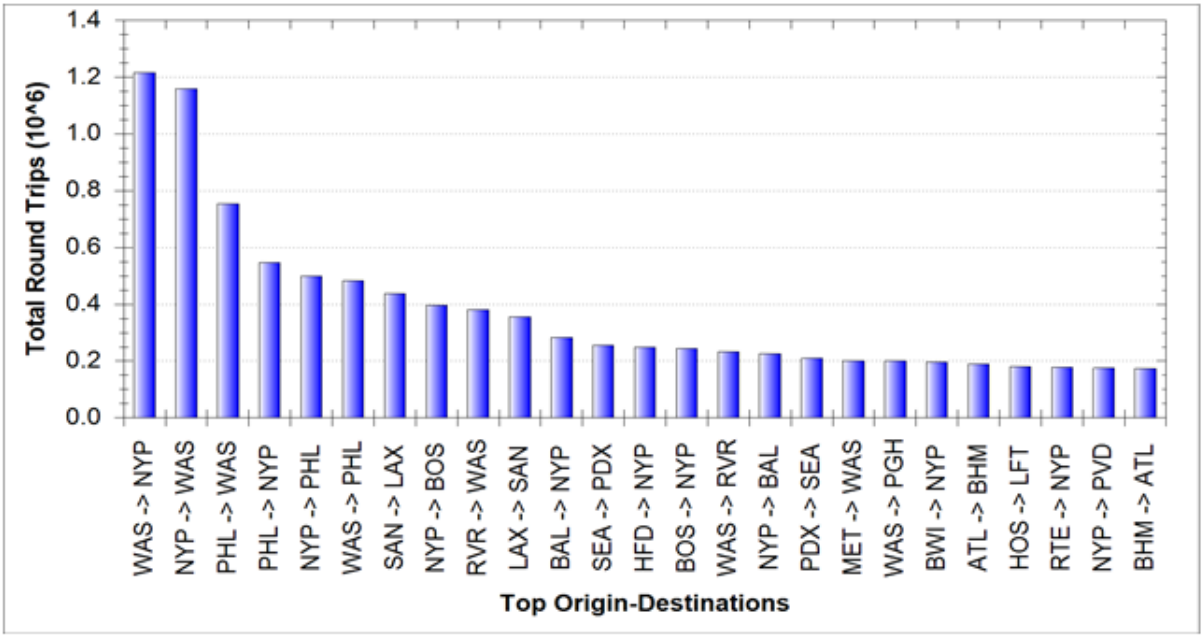
FIGURE 14 Rail Station Demand for Total Round Trips in 2040 from NYP - Penn Station, NEW YORK.

(A) Without High Speed Rail (B) With High Speed Rail

The plots in Figure 15 denote the top rail ridership station pairs, in 2040 with and without HSR. It is observed that the station pairs and the trend remain almost similar, the magnitude of the graph changes significantly. Penn station, Washington and Philadelphia again being the top ridership station pairs. The values of the ridership are presented in Appendix B.



(A)



(B)

FIGURE 15 TOTAL ROUND TRIP TRAIN RIDERSHIP AT TOP 25 ORIGINS - DESTINATIONS IN 2040.

A) WITHOUT HSR (B) WITH HSR

4.7 TSAM INTEGRATION

All computer codes for the developed tools are written in MATLAB. They can be easily integrated into TSAM. The Graphical User Interface (GUI) proposed will be an augmentation to the current GUI in TSAM. The proposed GUI shown in Figure 16 allows the user to modify various parameters in the rail model enabling sensitivity analysis. The proposed GUI includes rail as an optional mode, and the HSR option will be conditional to the rail mode selection. The HSR can be modeled for all of the corridors or can be modeled in one or any of the listed corridors. The year or operation, speed and cost will have a default value, but can be modified by the user. Once the model is incorporated, it will be easier for the user to access and analyze the output data and plots.

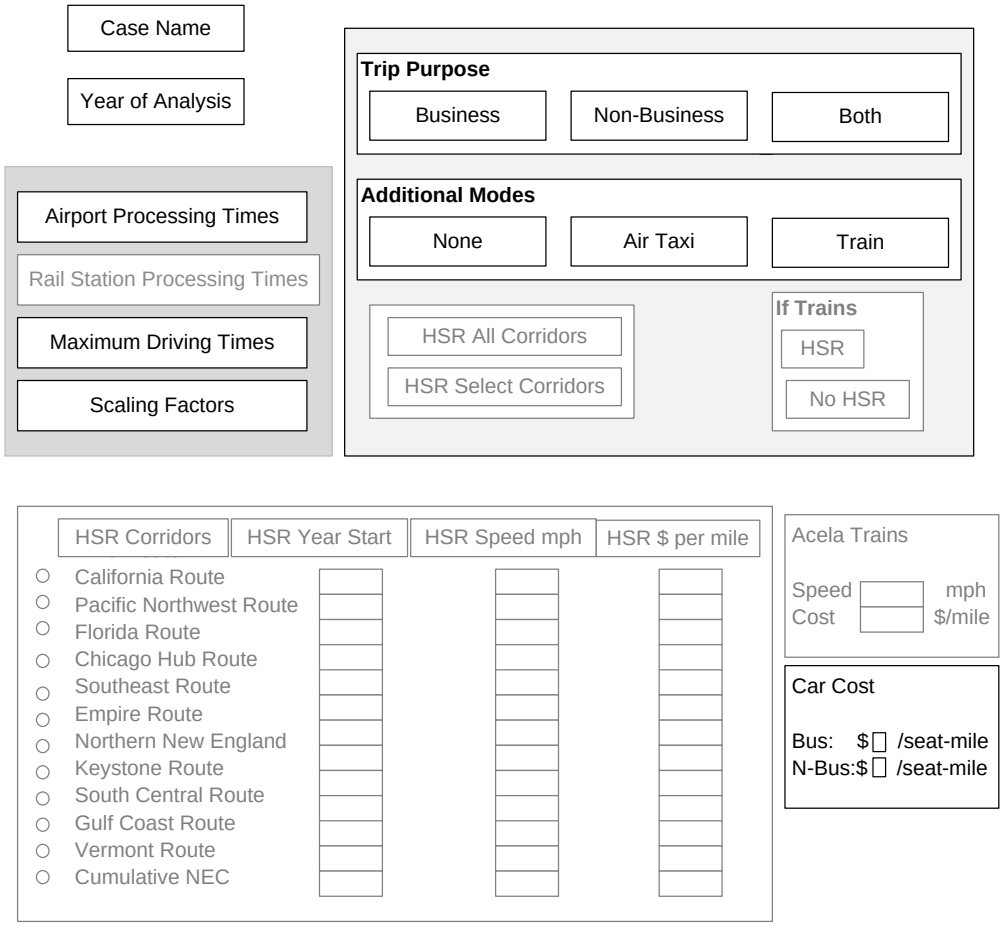


FIGURE 16 Proposed TSAM Graphical User Interface to Model Train Travel Demand.

4.8 SUMMARY

The earlier calibrations to the model grossly under predicted rail demand. The under prediction was been significantly corrected at a nationwide level, although it still requires work at the station level. The projected train

demand for the year 2030, having introduced HSR in all planned corridors was predicted earlier at 5 million round trips. With the improvement in calibrations, the rail ridership is now predicted at 31 million annual long-distance, round trips (Figure 7above). The changes to the model, presented below have significantly improved the ridership, by applying corrective measures to the input parameters.

The travel cost and time inputs were improved by developing regression functions for a few corridors (Appendix B). The travel times and costs initially were based on an average nationwide values estimated using regression curves. The regression data was collected for randomly selected OD pairs across the country. This method however, increases the disutility of the train mode significantly for some stations. The AMTRAK webpage was also used to compute the current frequencies between some lines, especially in the NEC (Appendix C). Assuming the frequencies have not changed much over the years, the schedule delays were then corrected for those lines. The schedule delay changed especially for the NEC which was set at a very high value. For some other routes like the California Zephyr, the frequency assumed was significantly higher than the actual frequency, leading to lower disutility and higher ridership.

The station choice model was improved by reassigning the stations for a few counties (Appendix A). The stations were initially assigned based on the closest driving distance from a county population centroid. The assignment method was based on a one-county-to-one-station assignment. This approach did not account for the counties which had a good network of commuter trains like the New York County. It is reasonable to assume that most of the people in such areas use commuter train as access mode due to parking limitations, car ownership percentages and mode inertia. To incorporate the effect of these factors, stations were reassigned for a few counties based on the mass transit network and ridership.

A Box-Cox model, shown in Table 8 and Table 9 was calibrated to best capture the rail demand with the available data. The limitation in improving the model further is the level of information in the available datasets. The ATS records do not contain information like the exact OD, the stations/airports that were actually used, the transfers made and others vital information. The data used is from 1995, and is limited in its capability to predict the decision behavior of people in 2040. Other surveys done after 1995 have few long-distance train records and do not serve our purpose.

Historic station level AMTRAK data and predicted ridership data published by AMTRAK were used for validation. From the validation plots it is evident that the model cannot capture adequate ridership shift from commercial airlines to rail. Most of the rail ridership is observed to come from automobiles.

The publicly available AMTRAK data is limited and terse in its level of information to serve as a good validation platform. The data was thus adapted for the project, based on several approximations and assumptions. Transfer rates (Table 7 above) were estimated, and short distance trips were approximated based on a set of assumptions. The parsed data was then used as a validation platform, and inconsistencies were identified and used as basis for the next set of corrective factors.

A GUI is proposed as an addition to the existing TSAM GUI. This would enable the user to select parameters for the rail mode to perform sensitivity analysis. This thesis presents the current state of the model TSAM 6.6 as of September 2012.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCULSIONS

A credible tool to model and predict train demand has been developed. The model is one of the first multimodal nationwide travel demand modeling software. The calibration model compiled uses previously developed tools for TSAM to incorporate rail demand as a mode choice. The proposed GUI allows the model to be incorporated into TSAM, to make the model user friendly. TSAM can now additionally model trains along with the existing capabilities of commercial air and automobiles. The capability can be extended to predict the HSR demand and its impact on the commercial airline and automobile demand. The final model presented shows significant improvement over the previous calibration, and can be satisfactorily used in its current state to make HSR predictions. The model is capable of performing sensitivity analysis and serving as an important decision making tool.

As seen from the calibration results, the non-business calibration plots are fairly good. The performance of the business calibrations are restricted by the number and sampling of the available rail records. Inconsistencies in the base data for rail business trips are evident from the spikes in the plots (Figure 3 above). The analysis of the ATS data also shows the poor geographical sampling of the data. With most of the train records concentrated in the NEC, modeling the other corridors is a difficult task. The data used in developing the model is from 1995, and hence cannot entirely capture the changes brought about by the Acela rail, and increased security inconvenience at the airports.

The research shows that a significant rail ridership is from the automobiles. Most of the relevant studies usually compare train and commercial air. The automobile ridership is seldom analyzed and this work shows that it is an important factor. This work is also the first to estimate transfer rates for the AMTRAK stations. Data on transfer rates though vital is not publicly available. Further, the work sheds light upon the importance of productivity factor to model commercial air and rail. We now know, that even with the introduction of HSR, the ease of work will be the most attractive factor over commercial air for business travelers. The increase in ridership in NEC with the HSR can be now used to justify the infrastructure improvements required by AMTRAK.

Validation results at the station level show a variation of the $\Sigma\chi^2$ value from 0.79 to 114.75. Limited data and information available about several assumed parameters are major hindrances in improving the validation further. The requirement of detailed data and more surveys has been realized to be of critical importance to develop the model further.

5.2 RECOMMENDATIONS

The work can be further improved by the following recommendations. There are improvements, which would be beneficial to the model and some which would be necessary once more information about the HSR is available.

5.2.1 *ADDITIONAL DATA*

Detailed surveys with elaborate information are highly recommended. The current publicly available best dataset is the ATS 1995 data. This data does not give us sufficient information about the options available, the zip code from which the trips were made, the station selected to make the trip, transfers made and other useful information. Moreover, the survey dates back to before the Acela trains were introduced. The current process of data collection needs to be modified to make the data more useful to the researchers. The survey should be designed to get the information to the zip code level without compromising the privacy of the participants.

Further, acquiring the O-D level AMTRAK demand data and transfer rates for validation purposes is essential. It will also serve as a platform to make sanity checks on the current assumptions and improve the model further. Once, the assumptions have been set right, the corridors could be modeled separately using the additional available data. Acela rail impact can also be studied and incorporated.

5.2.2 *STATION CHOICE MODEL*

An improvement suggested in the station choice model is development of a logit model. The one-county-to-one-station is not a very realistic way of distributing demand to stations. In the California Corridor, the county demarcations enclose a large area, resulting in assigning all the county trips to a particular station. As a result, several stations attract zero ridership leading to major discrepancies in the validation results. This problem although present, is not as prevalent in the NEC due to smaller county sizes. A logit model developed with the help of additional data will help an important improvement to the model. Similarly, the travel time and cost are presently assigned assuming a single mode of travel. It would be reasonable to assume there would be a mix usage of transit and car in some of these areas. The logit model will also need to account for this, when assigning travel times and costs.

5.2.3 *RAIL LINK-NODE NETWORK SCHEDULES*

The Train line schedules should be incorporated in TSAM, to estimate travel time realistically. Computing travel times and costs based on travel distances, results in large deviations when transfers come into the picture. The rail network can be modeled as a link-node type network, assigning the shortest path between stations. The model would greatly benefit from such an improvement because the number of stops, time at stations, transfers can be calculated. The travel speed could then be computed for each segment instead of using an average function for the entire corridor.

5.2.4 *TRANSFERS*

Another possible improvement to the model can be done by incorporating transfers in the estimation of total travel time, cost and disutility using the train mode. It is rational to believe that the number of transfers could have a significant disutility effect. This can be done in two ways: (a) include the number of transfers as a decision variable, assigning a suitable coefficient to it; (b) include the transfer time in computation of the total time. A penalty time could be added to project the inconvenience of transfers. The link-node type rail network along with incorporated AMTRAK schedules could be modeled to compute the transfer times.

5.2.5 *ACELA RAIL MODELING*

An enhancement to the model can be in the way Acela rail are modeled. The time and cost functions in the NEC are given weights to model the rail behavior. Modeling Acela rail separately should increase the accuracy of the model and will also make the model capable of predicting the Acela ridership separately. One way to do this would be by creating virtual stations for the Acela routes. It is interesting to note that while the travel time in Acela is improved over regional service, the increase in travel cost is relatively very high. Separate logit functions could then be developed to incorporate the differences in decision variable values, with the help of additional data. This would also enable computation of ridership separately for the NEC regional service and the Acela service.

5.2.6 *STATION LIST*

The sixth suggested improvement is updating the train station list, and using the correct station list for the corresponding years. A number of train stations which were functional earlier are no longer used, like the Akron station in Ohio. The currently available station list contains 464 stations. Currently, AMTRAK has 526 stations, and the list needs to be updated for additional stations in the corresponding years. Further, for the predicted HSR corridors, the major cities to be served have been determined. The location and number of stations that will sue the routes have still not been decided. AMTRAK stations in those major cities are all used for modeling the predicted corridors presently. Once the location and number of stations are determined, the stations list should be updated accordingly. This will be very important for ridership projections because the location of a station within a city will have an effect on the amount of potential riders.

5.2.7 *PRODUCTIVE TIME*

Additional calibrations are strongly recommended to incorporate the effect of productive time. The model can currently not attract significant ridership from the commercial airlines. The productive time factor should be incorporated such that the disutility of the available work time affects the commercial airline travel. The attempted calibrations in this model with productive time were done before the model was corrected for the other parameters. It would be worth an effort to try out a few of those calibrations again, to see if the model can now perform better. The model should see a significant increase for current train ridership and the HSR predictions.

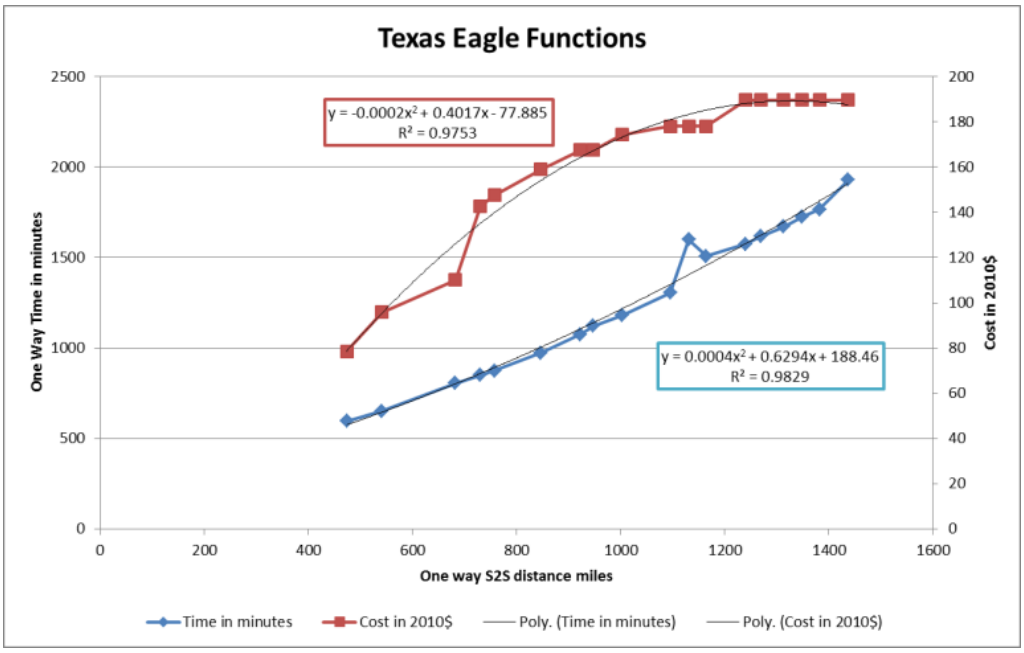
5.2.8 *LIFE CYCLE COST MODEL*

Lastly, the development of a life cycle cost model for trains could be an important tool. Energy consumption functions developed by Vandyke, 2010 could be used for this section. The functions could be expanded to include emissions. Such tools would enable the model to contribute significantly for budgetary decision making and environmental impact analysis. These monetary and practical additions to the rail mode choice will serve to boost the practical utility and commercial value of TSAM.

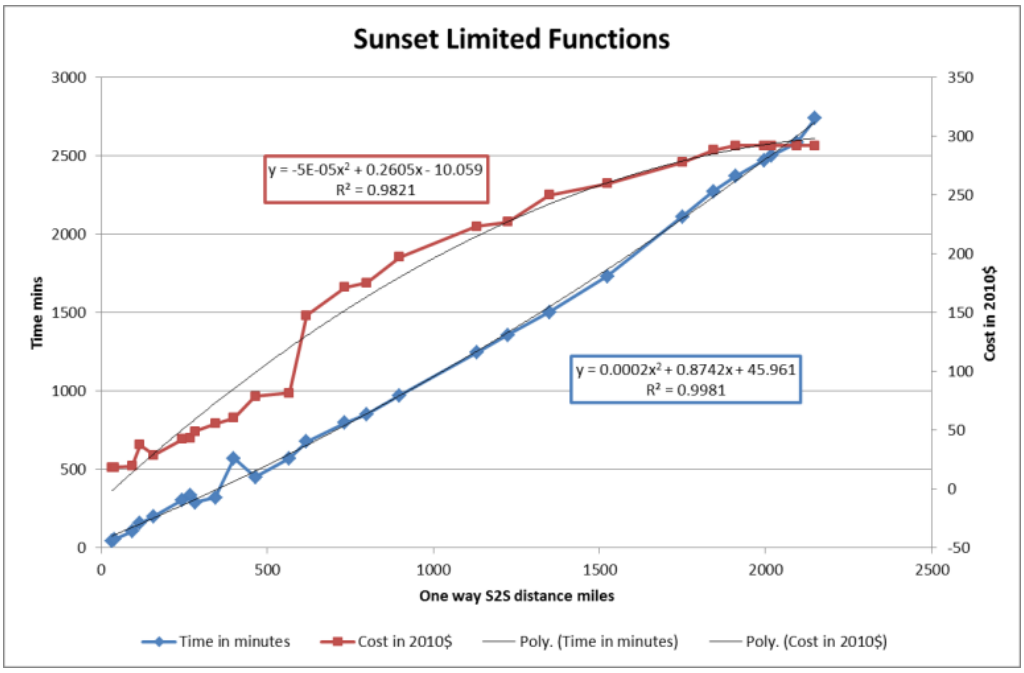
Appendix. A: RAIL TIME AND COST FUNCTIONS

The following plots have been developed selecting random O-D pairs on a given AMTRAK line. The station-to-station travel costs and travel times were then computed from the AMTRAK website. Regression curves for different lines, developed from the collected data are shown in the plots below.

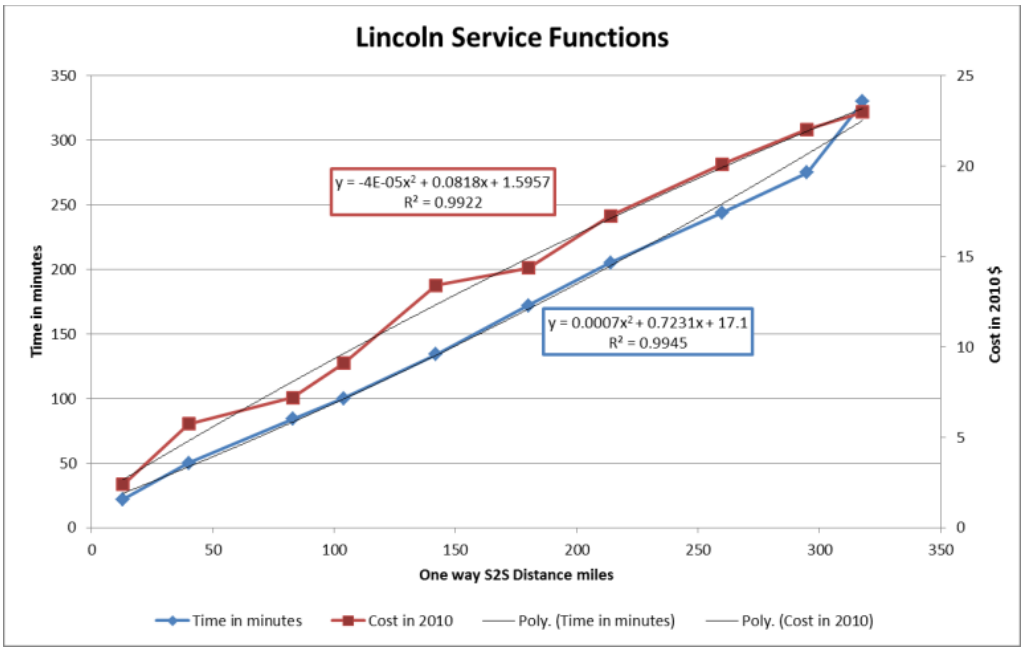
A.1 TEXAS EAGLE



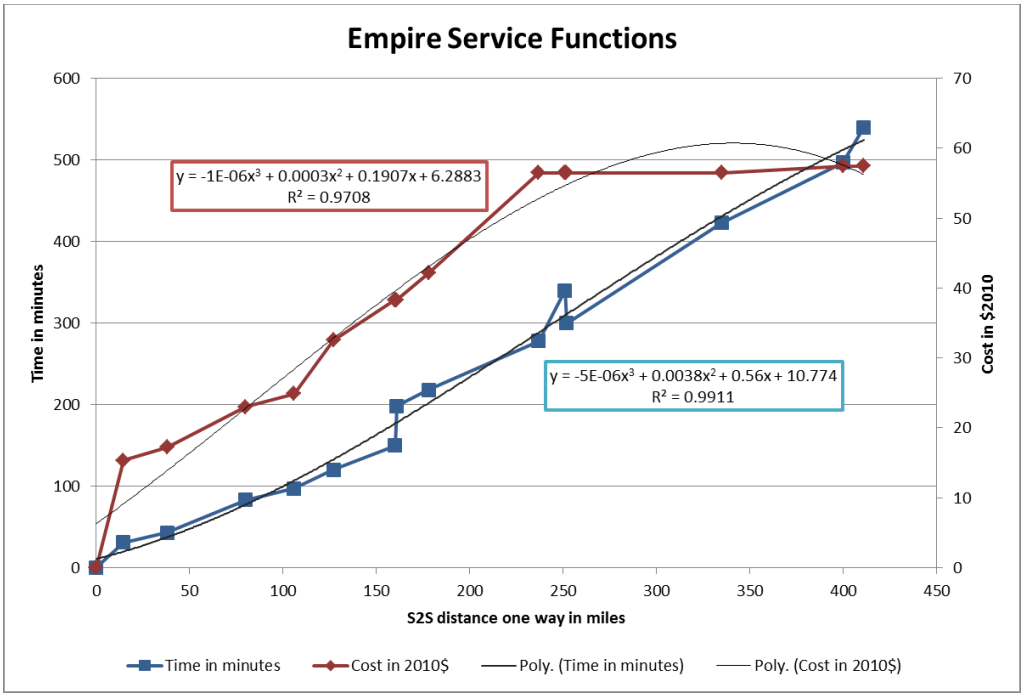
A.2 SUNSET LIMITED



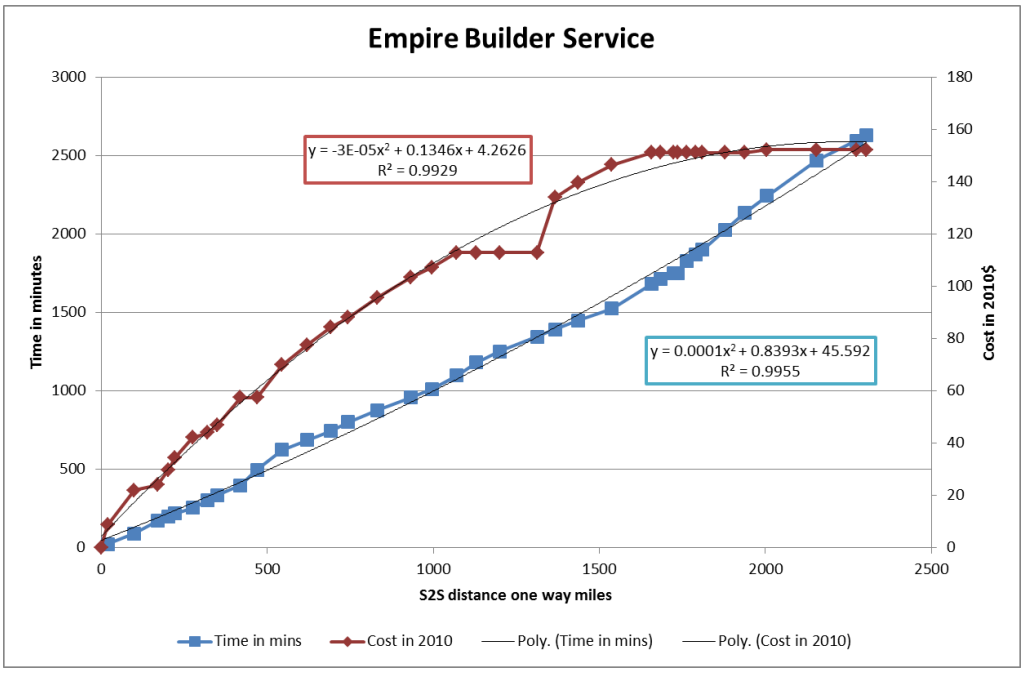
A.3 LINCOLN SERVICE



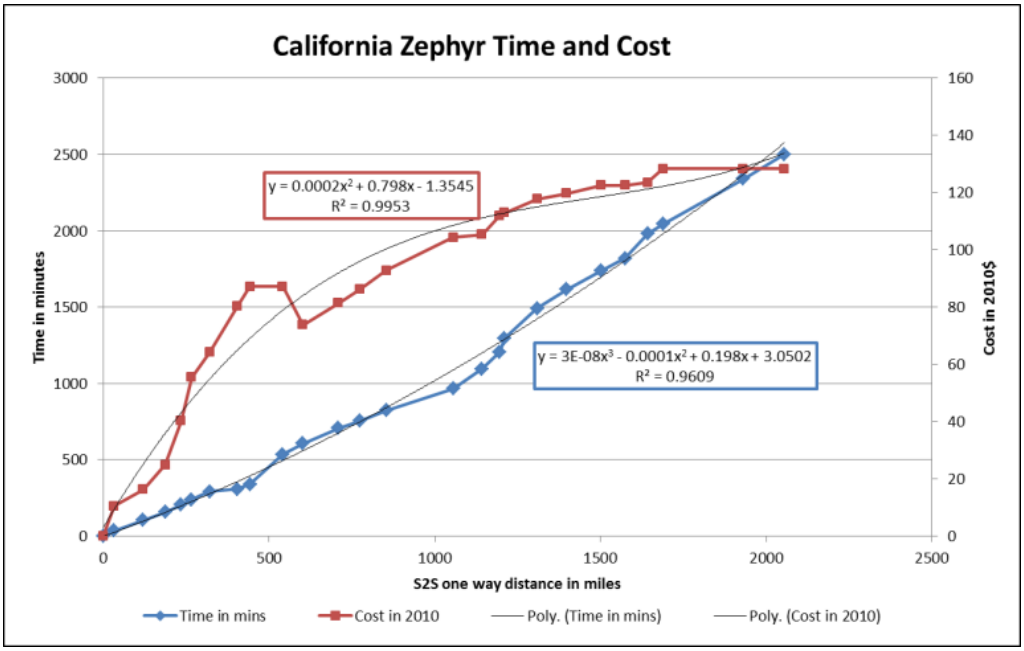
A.4 EMPIRE SERVICE



A.5 EMPIRE BUILDER

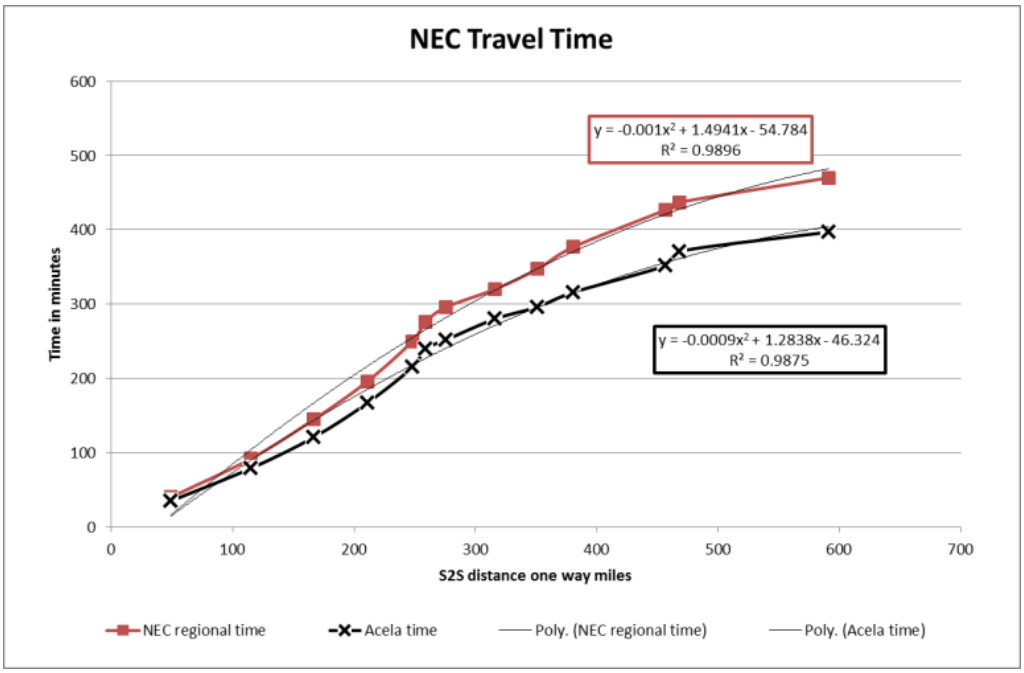


A.6 CALIFORNIA ZEPHYR

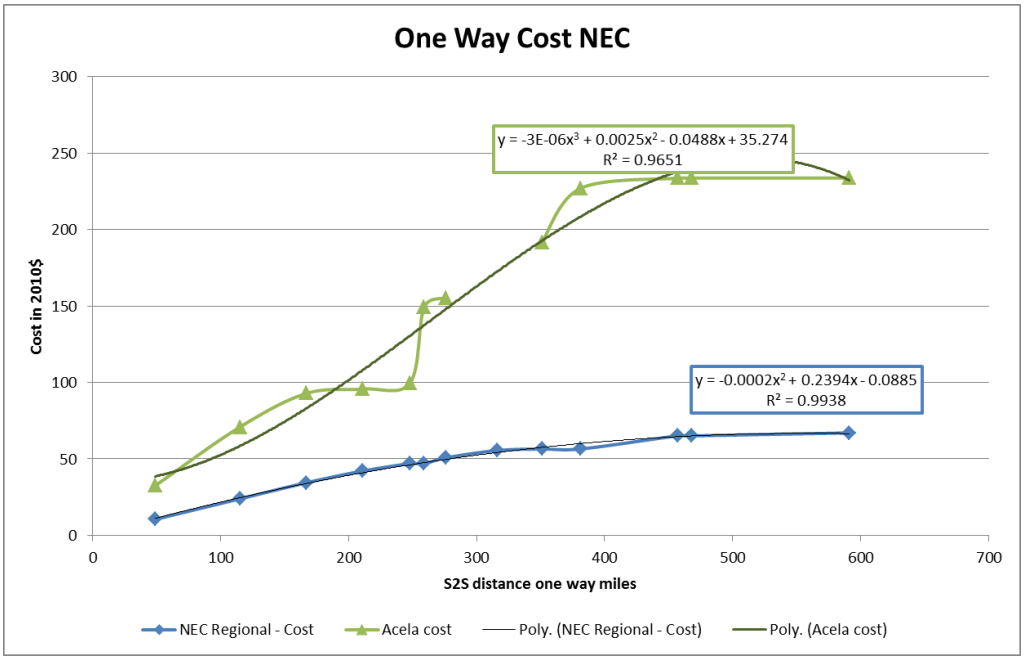


A.7 NEC TRAVEL TIME

For the NEC, travel time and cost curves were developed separately for the Acela and Regional Service. The cost difference is seen to be much higher than the time savings.

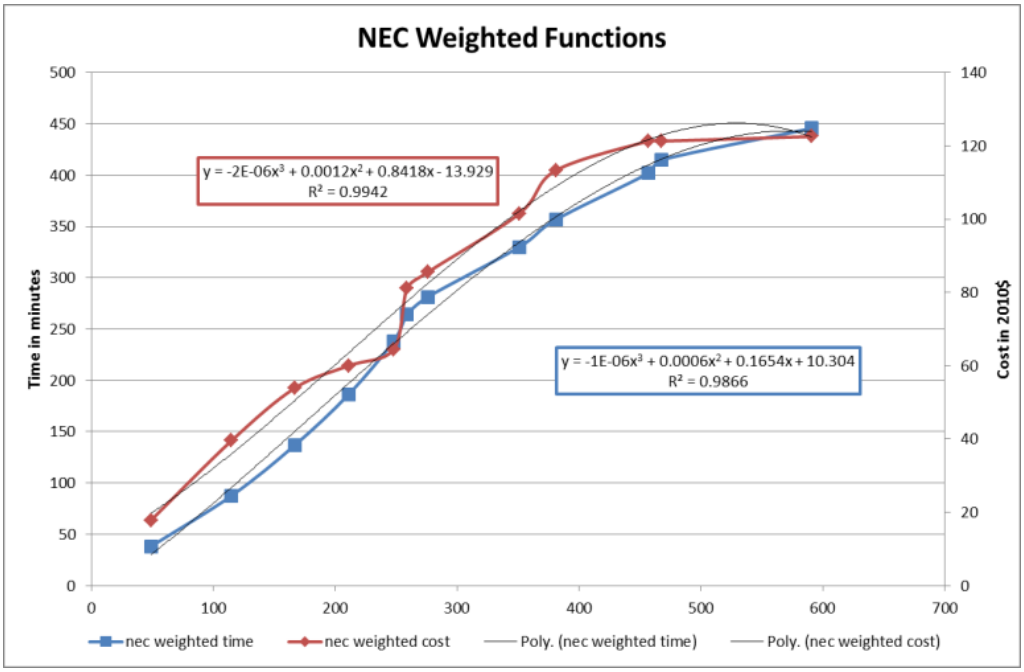


A.8 NEC TRAVEL COST



A.9 NEC WEIGHTED FUNCTIONS

A weighted regression curve is developed to model the NEC. The Acela demand is observed to be 1/3rd of the total NEC train demand. Hence, weighted travel time and costs are computed using a weight of 1/3 for Acela service and 2/3 for Regional service.



Appendix. B: RAIL TOP 50 ORIGIN DESTINATIONS FOR TOTAL RIDERSHIP IN 2040 WITH HSR.

The following table has the top train ridership pairs, as computed in TSAM, in the year 2040. The HSR rail system modeled assumes a line speed of 130 mph (209 kph) and a cost 0.30$/mile in 2000$ (approximately 0.40 $/mile in 2012$).

Origin State	Origin Airport ID	Origin Airport Name	Origin City Name	Destination State	Destination Airport ID	Destination Airport Name	Destination City Name	Total Round Trips
AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	NV	LAS	Mc Carran Intl	Las Vegas	1,366,658
CA	LAX	Los Angeles Intl	Los Angeles	NV	LAS	Mc Carran Intl	Las Vegas	1,270,749
CA	LAX	Los Angeles Intl	Los Angeles	CA	SFO	San Francisco Intl	San Francisco	1,098,620
CA	SFO	San Francisco Intl	San Francisco	CA	LAX	Los Angeles Intl	Los Angeles	1,055,037
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	900,879
NY	JFK	John F Kennedy Intl	New York	CA	LAX	Los Angeles Intl	Los Angeles	893,223
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	NV	LAS	Mc Carran Intl	Las Vegas	881,682
CA	LAX	Los Angeles Intl	Los Angeles	NY	JFK	John F Kennedy Intl	New York	786,479
IL	ORD	Chicago O'hare Intl	Chicago	NY	LGA	La Guardia	New York	759,808
CO	DEN	Denver Intl	Denver	NV	LAS	Mc Carran Intl	Las Vegas	747,725
UT	SLC	Salt Lake City Intl	Salt Lake City	NV	LAS	Mc Carran Intl	Las Vegas	706,203
CO	DEN	Denver Intl	Denver	CA	LAX	Los Angeles Intl	Los Angeles	697,840
CO	DEN	Denver Intl	Denver	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	687,176
AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	CO	DEN	Denver Intl	Denver	658,217
NY	LGA	La Guardia	New York	IL	ORD	Chicago O'Hare Intl	Chicago	645,299
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	642,756
TX	DAL	Dallas Love Field	Dallas	TX	HOU	William P Hobby	Houston	642,529
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	CA	LAX	Los Angeles Intl	Los Angeles	639,615
NY	LGA	La Guardia	New York	GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	625,970
TX	HOU	William P Hobby	Houston	TX	DAL	Dallas Love Field	Dallas	606,844
CA	BUR	Bob Hope	Burbank	NV	LAS	Mc Carran Intl	Las Vegas	604,730
WA	SEA	Seattle-Tacoma Intl	Seattle	NV	LAS	Mc Carran Intl	Las Vegas	601,170
AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	CA	SNA	John Wayne Airport-Orange County	Santa Ana	600,556
AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	CA	LAX	Los Angeles Intl	Los Angeles	593,964
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	CO	DEN	Denver Intl	Denver	593,326
NM	ABQ	Albuquerque Intl Sunport	Albuquerque	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	590,183
CA	LAX	Los Angeles Intl	Los Angeles	CO	DEN	Denver Intl	Denver	574,399
GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	564,861
AZ	PHX	Phoenix Sky	Phoenix	CA	ONT	Ontario Intl	Ontario	561,162

		Harbor Intl						
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	IL	ORD	Chicago O'hare Intl	Chicago	553,243
GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	NY	LGA	La Guardia	New York	537,963
UT	SLC	Salt Lake City Intl	Salt Lake City	CO	DEN	Denver Intl	Denver	529,317
WA	SEA	Seattle-Tacoma Intl	Seattle	CA	LAX	Los Angeles Intl	Los Angeles	501,892
CA	SNA	John Wayne Airport-Orange County	Santa Ana	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	500,934
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	FL	MCO	Orlando Intl	Orlando	495,609
IL	ORD	Chicago O'Hare Intl	Chicago	CA	LAX	Los Angeles Intl	Los Angeles	492,407
GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	FL	FLL	Fort Lauderdale/Hollywood Intl	Fort Lauderdale	491,659
CO	DEN	Denver Intl	Denver	UT	SLC	Salt Lake City Intl	Salt Lake City	483,047
CA	ONT	Ontario Intl	Ontario	NV	LAS	Mc Carran Intl	Las Vegas	478,684
IL	ORD	Chicago O'Hare Intl	Chicago	TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	476,970
AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	NM	ABQ	Albuquerque Intl Sunport	Albuquerque	475,929
CA	LAX	Los Angeles Intl	Los Angeles	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	474,697
CA	ONT	Ontario Intl	Ontario	AZ	PHX	Phoenix Sky Harbor Intl	Phoenix	472,915
GA	ATL	Hartsfield - Jackson Atlanta Intl	Atlanta	FL	MCO	Orlando Intl	Orlando	467,880
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	MN	MSP	Minneapolis-St Paul Intl/Wold-Chamberlain	Minneapolis	467,229
MN	MSP	Minneapolis-St Paul Intl/Wold-Chamberlain	Minneapolis	CO	DEN	Denver Intl	Denver	465,468
NY	JFK	John F Kennedy Intl	New York	CA	SFO	San Francisco Intl	San Francisco	461,607
CO	DEN	Denver Intl	Denver	TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	449,812
TX	DFW	Dallas/Fort Worth Intl	Dallas-Fort Worth	CA	SNA	John Wayne Airport-Orange County	Santa Ana	447,692
WA	SEA	Seattle-Tacoma Intl	Seattle	CO	DEN	Denver Intl	Denver	446,972

Appendix. C: TRAIN FREQUENCY TABLES

These frequency tables were evaluated for different lines, using data collected from the AMTRAK Website. These were then used to compute the schedule delays for the O-D station pairs.

C.1 NEC Regional Service

	BOS	BBY	RTE	PVD	NLC	NHV	STM	NYP	NWK	MET	TRE	PHL	WIL	BAL	BWI	WAS
BOS	0	9	9	9	9	9	9	9	8	8	8	8	8	8	8	8
BBY		0	9	9	9	9	9	9	8	8	8	8	8	8	8	8
RTE			0	9	9	9	9	9	8	8	8	8	8	8	8	8
PVD				0	9	9	9	9	8	8	8	8	8	8	8	8
NLC					0	9	9	9	8	8	8	8	8	8	8	8
NHV						0	10	10	9	9	9	9	9	9	9	9
STM							0	10	9	9	9	9	9	9	9	9
NYP								0	20	20	20	20	20	20	20	20
NWK									0	20	20	20	20	20	20	20
MET										0	20	20	20	20	20	20
TRE											0	20	20	20	20	20
PHL												0	20	20	20	20
WIL													0	20	20	20
BAL														0	20	20
BWI															0	20
WAS																0

C.2 Acela Service

	BOS	BBY	RTE	PVD	NLC	NHV	STM	NYP	NWK	MET	TRE	PHL	WIL	BAL	BWI	WAS
BOS	0	10	10	10	1	9	9	10	9	1	0	9	9	9	9	9
BBY		0	10	10	1	9	9	10	9	1	0	9	9	9	9	9
RTE			0	10	1	9	9	10	9	1	0	9	9	9	9	9
PVD				0	1	9	9	10	9	1	0	9	9	9	9	9
NLC					0	1	1	1	1	1	0	1	1	1	1	1
NHV						0	9	9	9	1	0	9	9	9	9	9
STM							0	9	9	1	0	8	8	8	7	8
NYP								0	15	5	1	15	15	15	10	15
NWK									0	5	1	15	15	15	10	15
MET										0	1	5	5	5	5	5
TRE											0	1	1	1	1	1
PHL												0	15	15	10	15
WIL													0	15	10	15
BAL														0	10	15
BWI															0	10
WAS																0

C.3 Vermonter

	NHV	STM	NYP	NWK	MET	TRE	PHL	WIL	BAL	BWI	WAS
NHV	0	1	2	2	1	2	2	2	2	1	2
STM		0	1	1	1	1	1	1	1	1	1
NYP			0	2	1	2	2	2	2	1	2
NWK				0	1	2	2	2	2	1	2
MET					0	1	1	1	1	1	1
TRE						0	2	2	2	1	2
PHL							0	2	2	1	2
WIL								0	2	1	2
BAL									0	1	2
BWI										0	1
WAS											0

C.4 Keystone Express

	NYP	NWK	MET	TRE	PHL
NYP	0	9	0	9	9
NWK		0	0	9	9
MET			0	0	0
TRE				0	9
PHL					0

C.5 Crescent

	NYP	NWK	MET	TRE	PHL	WIL	BAL	BWI	WAS
NYP	0	1	0	1	1	1	1	0	1
NWK		0	0	1	1	1	1	0	1
MET			0	0	0	0	0	0	0
TRE				0	1	1	1	0	1
PHL					0	1	1	0	1
WIL						0	1	0	1
BAL							0	0	1
BWI								0	0
WAS									0

C.6 Pennsylvanian

	NYP	NWK	MET	TRE	PHL
NYP	0	1	0	1	1
NWK		0	0	1	1
MET			0	0	0
TRE				0	1
PHL					0

Appendix. D: MATLAB CODES FOR RAIL MODEL

D.1 Main file creator

```
function S2S_File_Creator(Install_Dir,CaseYear,Train,...

NEC_Route,NEC_Route_Average_Cost,NEC_Route_Initial_Year,NEC_Route_Average_Spe
ed,...

California_Route,California_Route_Average_Cost,California_Route_Initial_Year,
California_Route_Average_Speed,...

Pacific_Northwest_Route,Pacific_Northwest_Route_Average_Cost,Pacific_Northwes
t_Route_Initial_Year,Pacific_Northwest_Route_Average_Speed,...

Florida_Route,Florida_Route_Average_Cost,Florida_Route_Initial_Year,Florida_R
oute_Average_Speed,...

Chicago_Hub_Route,Chicago_Hub_Route_Average_Cost,Chicago_Hub_Route_Initial_Ye
ar,Chicago_Hub_Route_Average_Speed,...

Southeast_Route,Southeast_Route_Average_Cost,Southeast_Route_Initial_Year,Sou
theast_Route_Average_Speed,...

Empire_Service_Route,Empire_Service_Route_Average_Cost,Empire_Service_Route_I
nitial_Year,Empire_Service_Route_Average_Speed,...

Northern_New_England_Route,Northern_New_England_Route_Average_Cost,Northern_N
ew_England_Route_Initial_Year,Northern_New_England_Route_Average_Speed,...

Keystone_Route,Keystone_Route_Average_Cost,Keystone_Route_Initial_Year,Keysto
ne_Route_Average_Speed,...

South_Central_Route,South_Central_Route_Average_Cost,South_Central_Route_Init
ial_Year,South_Central_Route_Average_Speed,...

Gulf_Coast_Route,Gulf_Coast_Route_Average_Cost,Gulf_Coast_Route_Initial_Year,
Gulf_Coast_Route_Average_Speed,...

Vermont_Route,Vermont_Route_Average_Cost,Vermont_Route_Initial_Year,Vermont_R
oute_Average_Speed,...

Cumulative_NEC_Route,Cumulative_NEC_Route_Average_Cost,Cumulative_NEC_Route_I
nitial_Year,Cumulative_NEC_Route_Average_Speed)


%This file will create the S2S travel time and S2S travel cost
%matrices.
%%-----------------------------------------------------------------------
%This file needs the functions: Train_Station_Index Finder,
%Schedule_Delay_Calculator

%Travel Cost Equation Values.  A cost analysis was conducted on Mar. 23,
%2011 for travel on March 9, 2011 on the Amtrak Network Nationwide. Coach
%class with no room.
```

```
% Last Edited : Saloni Chirania
% Date last edited : 08/08/12

%%-------------------------------------------------------------------------
--



% clear all;
% clc;
% %uncomment install dir line
% Install_Dir = 'C:\Program Files (x86)\TSAM\6.6';  % location with input
files
% CaseYear    = 2020; % for which the model is run. the model will convert it
to 2011 later, if it is greater than 2011
% Train       = 'YES';
% Year_Start_High_Speed_Service = 2011;
% %

global TrainStation_List
global Inflation_Factor

load(strcat(Install_Dir,'\data\mode_choice\input\TrainStation_List.mat'));
%This file containts the Amtrak "TrainStation_List"
%load('E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data\mode_choice\input\PCEP
_Index.mat')    %%This is the inflation index using PCEP index
load(strcat(Install_Dir,'\data\mode_choice\input\BLS_CPI_Index.mat')); %This
is the inflation index using BLS_CPI_U_RS_Index changed in 2011 from PCEP
index
load(strcat(Install_Dir,'\data\mode_choice\input\S2S_Train_ScheduleDelay.mat'
)); %This loads the original schedule delay values from TSAM. IT is assumed
this is the scheule delay unless changes are made in the code below.
load(strcat(Install_Dir,'\data\mode_choice\input\S2S_Train_Distance')); %This
is the original matrix of distance between stations. This is the original
data and is assumed to be correct unless specific changes have been noted
below.
%
load(strcat(Install_Dir,'\data\mode_choice\input\S2S_Train_ScheduleDelay_Acel
a')); %This is the matrix which has the frequencies of the NEC as computed on
13th April 2012.

Year_analysis = min(CaseYear,2011);  % the maximum year is 2011, as that is
the last year for train input files.
Cost_Year                     = 2000; %This should always be 2000 to be
consistent with TSAM.  TSAM needs all input cost in 2000 $'s. So the created
table of S2S costs for 1995, or 1996, or 1997, etc. should all be in 2000
$'s.
Regression_Cost_Year_Access  = 2012; % this is the year when the costs were
calculated. Should not be changed unless new values are determined.
Regression_Cost_Year         = 2010; % THIS HAS TO BE =2010 UNLESS NEW
REGRESSION CURVES ARE DEVELOPED. This is the year of the cost data that was
used to develop regression curves. All curves were developed in 2010 $'s
unless noted otherwise.
Inflation_Factor             =BLS_CPI_Index((Cost_Year-
1994),2)/BLS_CPI_Index((Regression_Cost_Year-1994),2);
Inflation_Factor_Access      =BLS_CPI_Index((Cost_Year-
1994),2)/BLS_CPI_Index((Regression_Cost_Year_Access-1994),2);
```

```
Output_Save_Directory
=(strcat(Install_Dir,'\data\mode_choice\input\Y',num2str(Year_analysis),'\'))
;  % file location to save the S2S files.

%% load the station selection matrices for selected corridors
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\chicago_metro.mat')); %This is the matrix containing the enforced
station selction for countris in the chicago corridor. The columns are :
County number - Cost in 2012$ - Time in minutes
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\LA_metro.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\newyork_subway.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\phl_metro.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\pvd_transit.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\washington_metro.mat'));
Subway_station_access = [chicago_metro ;newyork_subway; phl_metro;
pvd_transit; washington_metro]; % with all the counties.
b = Subway_station_access(:,2) == 0; % the subway station access also
contains stations which do not comply to the closest driving station rule.
These are farther stations, but drive down to these important stations. These
stations have been assigned a value of 0 in travel cost. hence, eliminate
these here.
Subway_station_access(b,:) = [];
clear b;

%%
if strcmp(Train,'YES')== 0
    % i.e. if train mode is not selcted, it assigns a very high time
    % and cost to the modes, ensuring that no one takes the train

    S2S_Train_TravelTime=S2S_Train_Distance .* 10; %This means the S2S travel
speed is  1/10 mph.
    S2S_Train_TravelCost=S2S_Train_Distance .* 5000; %This means the S2S
travel cost is  5000 $/mi.

else


    %% ------------------------------------------------
    % the follwing lines are commented, while running in TSAM. To give
    % inputs manually in the code, uncomment them

    %     disp(['Year:',num2str(Year_analysis)])

    %This section is used to determine what projected corridor lines are
    %included in the model.  Variables should be 'YES' if the corridor is to
be
    %modeled.  Descriptions of the corridors can be found below. The initial
    %year variables are the first year that the given corridor is to be
    %modeled. These variable should not be less than 2011.
```

```
    %-----------------------------------------------------------------
%     NEC_Route                   = 'NO'; % YES NO
%     California_Route            = 'NO';
%     Pacific_Northwest_Route     = 'NO';
%     Florida_Route               = 'NO';
%     Chicago_Hub_Route           = 'NO';
%     Southeast_Route             = 'NO';
%     Empire_Service_Route        = 'NO';
%     Northern_New_England_Route = 'NO';
%     Keystone_Route              = 'NO';
%     South_Central_Route         = 'NO';
%     Gulf_Coast_Route            = 'NO';
%     Vermont_Route               = 'NO';
%     Cumulative_NEC_Route        = 'NO'; %IF Cumulative_NEC_Route= YES, this
will combine and overwrite the NEC_Route, Empire_Route,
Northern_New_England_Route, Keystone_Route, and Northern_NEC_Route. Turn off
these corridors to speed the code.
    %
    if ~exist('NEC_Route') || isempty(NEC_Route) % Default value of NO if
nothing is assigned
        NEC_Route = 'NO';
    end
    if ~exist('California_Route') || isempty(California_Route)
        California_Route  = 'NO';
    end
    if ~exist('Pacific_Northwest_Route') || isempty(Pacific_Northwest_Route)
        Pacific_Northwest_Route = 'NO';
    end
    if ~exist('Florida_Route') || isempty(Florida_Route)
        Florida_Route = 'NO';
    end
    if ~exist('Chicago_Hub_Route') || isempty(Chicago_Hub_Route)
        Chicago_Hub_Route = 'NO';
    end
    if ~exist('Southeast_Route') || isempty(Southeast_Route)
        Southeast_Route = 'NO';
    end
    if ~exist('Empire_Service_Route') || isempty(Empire_Service_Route)
        Empire_Service_Route = 'NO';
    end
    if ~exist('Northern_New_England_Route') ||
isempty(Northern_New_England_Route)
        Northern_New_England_Route = 'NO';
    end
    if ~exist('Keystone_Route') || isempty(Keystone_Route)
        Keystone_Route  = 'NO';
    end
    if ~exist('South_Central_Route') || isempty(South_Central_Route)
        South_Central_Route = 'NO';
    end
    if ~exist('Gulf_Coast_Route') || isempty(Gulf_Coast_Route)
        Gulf_Coast_Route = 'NO';
    end
    if ~exist('Vermont_Route') || isempty(Vermont_Route)
        Vermont_Route  = 'NO';
    end
    if ~exist('Cumulative_NEC_Route') || isempty(Cumulative_NEC_Route)
```

```
        Cumulative_NEC_Route  = 'NO';
    end

    %CHANGE INITIAL YEAR BACK TO 2011 AFTER TESTING IS COMPLETE

%             NEC_Route_Initial_Year                  =
Year_Start_High_Speed_Service; %This implies the year that the NEC corridor
gets updated (higher speeds).
%             California_Route_Initial_Year           =
Year_Start_High_Speed_Service;  % enter a different year if required
%             Pacific_Northwest_Route_Initial_Year    =
Year_Start_High_Speed_Service;
%             Florida_Route_Initial_Year              =
Year_Start_High_Speed_Service;
%             Chicago_Hub_Route_Initial_Year          =
Year_Start_High_Speed_Service;
%             Southeast_Route_Initial_Year            =
Year_Start_High_Speed_Service;
%             Empire_Service_Route_Initial_Year       =
Year_Start_High_Speed_Service;
%             Northern_New_England_Route_Initial_Year =
Year_Start_High_Speed_Service;
%             Keystone_Route_Initial_Year             =
Year_Start_High_Speed_Service;
%             South_Central_Route_Initial_Year        =
Year_Start_High_Speed_Service;
%             Gulf_Coast_Route_Initial_Year           =
Year_Start_High_Speed_Service;
%             Vermont_Route_Initial_Year              =
Year_Start_High_Speed_Service;
%             Cumulative_NEC_Route_Initial_Year       =
Year_Start_High_Speed_Service;
%     %
%
    % If the weighted curves option is yes, the cost and time curves will be
    % applied a factor of 2/3 for NEC regional service. and 1/3 for Acela.

    %If Acela_Cost_XXXXXX_Route = 'YES' then that corridor is modeled with
    %the Amtrak Acela Cost curve. Otherwise, Cost is currently based on
$/mile.
    %When new cost function is developed,the code in the double for loop
below
    %will have to be changed.

    %This needs to be in 2010 dollars because Regression_Cost_Year = 2010
    % if weights are used instead of the new functions, uncomment the
following
    % three lines
    %         Acela_NEC_Weighted_Curves               = 'YES';  % YES or NO
    %         Regional_Service_Weight                 = 2/3;
    %         Acela_weight                            = 1 -
Regional_Service_Weight ;
    %

    %             Acela_Cost_NEC_Route         = 'NO';
    %             Acela_Cost_California_Route           = 'NO';
```

```
    %             Acela_Cost_Pacific_Northwest_Route    = 'NO';
    %             Acela_Cost_Florida_Route              = 'NO';
    %             Acela_Cost_Chicago_Hub_Route          = 'NO';
    %             Acela_Cost_Southeast_Route            = 'NO';
    %             Acela_Cost_Empire_Route               = 'NO';
    %             Acela_Cost_Northern_New_England_Route = 'NO';
    %             Acela_Cost_Keystone_Route             = 'NO';
    %             Acela_Cost_South_Central_Route        = 'NO';
    %             Acela_Cost_Gulf_Coast_Route           = 'NO';
    %             Acela_Cost_Vermont_Route              = 'NO';
    %             Acela_Cost_Cumulative_NEC_Route       = 'NO';
    %
    %
%                     NEC_Route_Average_Cost                 = 0.50; %This
applies only after year 2010.
%                     California_Route_Average_Cost          = 0.50; % Costs
are in 2010$.
%                     Pacific_Northwest_Route_Average_Cost   = 0.50;
%                     Florida_Route_Average_Cost             = 0.50;
%                     Chicago_Hub_Route_Average_Cost         = 0.50;
%                     Southeast_Route_Average_Cost           = 0.50;
%                     Empire_Service_Route_Average_Cost      = 0.50;
%                     Northern_New_England_Route_Average_Cost= 0.50;
%                     Keystone_Route_Average_Cost            = 0.50;
%                     South_Central_Route_Average_Cost       = 0.50;
%                     Gulf_Coast_Route_Average_Cost          = 0.50;
%                     Vermont_Route_Average_Cost             = 0.50;
%                     Cumulative_NEC_Route_Average_Cost      = 0.50;
%     %
    %     %Travel Time is currently based on an average speed across the
corridor.
    %     %This speed is the average speed including stops.  When new time
functions
    %     %are developed, the code in the double for loop below will have to
be
    %     %changed. The speed is presented in miles/hr.
%                     NEC_Route_Average_Speed                = 130;  %175;
%Planned Value = 119.   This applies only after year 2010.
%                     California_Route_Average_Speed         = 130; %Planned
Value = 153
%                     Pacific_Northwest_Route_Average_Speed  = 130; %Planned
Value = 105
%                     Florida_Route_Average_Speed            = 130; %Planned
Value = 118
%                     Chicago_Hub_Route_Average_Speed        = 130;
%Planned Value = 77
%                     Southeast_Route_Average_Speed          = 130;
%Planned Value = 77
%                     Empire_Service_Route_Average_Speed     = 130;
%Planned Value = 77
%                     Northern_New_England_Route_Average_Speed= 130;
%Planned Value = 77
%                     Keystone_Route_Average_Speed           = 130;
%Planned Value = 77
%                     South_Central_Route_Average_Speed      = 130; %Planned
Value = 105
```

```
%                       Gulf_Coast_Route_Average_Speed          = 130; %Planned
Value = 105
%                       Vermont_Route_Average_Speed             = 130;
%Planned Value = 77
%                       Cumulative_NEC_Route_Average_Speed      = 130; %Planned
Value = 100


    %% ----------------------------------------------------
    %Nationwide Regression Coefficients  Currently valid for 1995-2040. This
curve is used as baseline.  Other routes have been modified below.
    %--------------------------------------------------------------------
    % A power function was fit for the Travel Cost regression.

    %y=A*x^B.   R^2 = 0.9057 for this analysis.
    A=0.6454;  %A=0.6454 for original nationwide regression analysis
    B=0.7622;  %B=0.7622 for original nationwide regression analysis
    Cost_Factor=1; % Factor=1 for original regression. If ~=1 then a trial
value is used. This factor will increase each cost by the factor value.

    %A linear function was fit for the Travel Time regression.(Intercept was
    %set to 0)
    %y=C*x
    C=0.022; %C=0.022 for original regression of nationwide times.  If
C~=0.022, then it is a trial of a different time.

    [~,Number_Stations_Total_Network] = size(TrainStation_List);
    [subway_stations_number,~]=size(Subway_station_access);

    Station_Index       = 1:Number_Stations_Total_Network;

    S2S_Train_TravelTime =
zeros(Number_Stations_Total_Network,Number_Stations_Total_Network);
    S2S_Train_TravelCost =
zeros(Number_Stations_Total_Network,Number_Stations_Total_Network);
    C2S_Train_Access_Cost = zeros(subway_stations_number,2);
    S2S_Train_TravelTime = C*S2S_Train_Distance; %hrs
    S2S_Train_TravelCost =
max(25,(A*(S2S_Train_Distance.^B)*Cost_Factor))*Inflation_Factor; % A minimum
of 25 dollars is assigned to cost because regression curve produces negative
values for short distances. The $30 is valid for 2011 and it is adjusted for
inflation for the desired year.
    C2S_Train_Access_Cost(:,1)=Subway_station_access(:,1);
    C2S_Train_Access_Cost(:,2) = max(2,
(Subway_station_access(:,2)*Inflation_Factor_Access));  % a minimum of 2 $ is
assigned
    %% ------------------------------------------------
    % Time, cost and schedule delay for different modeled lines
    % only a few lines were modeled.

    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Pacific_Surfliner_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
```

```
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Lincoln_Service_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
California_Zephyr_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Sunset_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Texas_Eagle_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Empire_Builder_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Acela_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost);
    [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Empire_Service_Route_Input(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Di
stance,S2S_Train_TravelTime,S2S_Train_TravelCost);


    %% Time cost inputs for HSR
    % ------------------------------------------------------------
    % Assign Year HSR Starts, HSR Cost, HSR SPEED by DEfault if Empty
    %-------------------------------------------------------------
    if strcmp(NEC_Route, 'NO') ~= 1
        if ~exist('NEC_Route_Initial_Year') ||
isempty(NEC_Route_Initial_Year) == 1
            NEC_Route_Initial_Year = 2011;
        end
        if ~exist('NEC_Route_Average_Cost') || isempty(NEC_Route_Average_Cost
) == 1
            NEC_Route_Average_Cost  = 0.30;
        end
        if ~exist('NEC_Route_Average_Speed') ||
isempty(NEC_Route_Average_Speed) == 1
            NEC_Route_Average_Speed = 119;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
NEC_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,NEC_Route,NEC_Route_Average_Cost,NEC_Route_Initial_Yea
r,NEC_Route_Average_Speed); %This applies only after year 2010.
    end

    if strcmp(California_Route, 'NO') ~= 1
        if ~exist('California_Route_Initial_Year') || isempty(
California_Route_Initial_Year) == 1
```

```
            California_Route_Initial_Year = 2011;
        end
        if ~exist('California_Route_Average_Cost') ||
isempty(California_Route_Average_Cost) == 1
            California_Route_Average_Cost = 0.30;
        end
        if ~exist('California_Route_Average_Speed') ||
isempty(California_Route_Average_Speed) == 1
            California_Route_Average_Speed = 153;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
California_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,California_Route,California_Route_Average_Cost,Califor
nia_Route_Initial_Year,California_Route_Average_Speed);

    end

    if strcmp(Pacific_Northwest_Route, 'NO') ~= 1
        if ~exist('Pacific_Northwest_Route_Initial_Year ') ||
isempty(Pacific_Northwest_Route_Initial_Year) == 1
            Pacific_Northwest_Route_Initial_Year = 2011;
        end
        if ~exist('Pacific_Northwest_Route_Average_Cost') || isempty(
Pacific_Northwest_Route_Average_Cost ) == 1
            Pacific_Northwest_Route_Average_Cost   = 0.30;
        end
        if ~exist('Pacific_Northwest_Route_Average_Speed') ||
isempty(Pacific_Northwest_Route_Average_Speed) == 1
            Pacific_Northwest_Route_Average_Speed = 105;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Pacific_Northwest_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Pacific_Northwest_Route,Pacific_Northwest_Route_Averag
e_Cost,Pacific_Northwest_Route_Initial_Year,Pacific_Northwest_Route_Average_S
peed);

    end

    if strcmp(Florida_Route, 'NO') ~= 1
        if ~exist(' Florida_Route_Initial_Year') ||
isempty(Florida_Route_Initial_Year) == 1
            Florida_Route_Initial_Year = 2011;
        end
        if ~exist('Florida_Route_Average_Cost') ||
isempty(Florida_Route_Average_Cost) == 1
            Florida_Route_Average_Cost  = 0.30;
        end
        if ~exist('Florida_Route_Average_Speed') ||
isempty(Florida_Route_Average_Speed) == 1
            Florida_Route_Average_Speed = 118;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Florida_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
```

```
e,S2S_Train_TravelCost,Florida_Route,Florida_Route_Average_Cost,Florida_Route
_Initial_Year,Florida_Route_Average_Speed);

    end

    if strcmp(Chicago_Hub_Route, 'NO') ~= 1
        if ~exist('Chicago_Hub_Route_Initial_Year') ||
isempty(Chicago_Hub_Route_Initial_Year) == 1
            Chicago_Hub_Route_Initial_Year = 2011;
        end
        if ~exist('Chicago_Hub_Route_Average_Cost') ||
isempty(Chicago_Hub_Route_Average_Cost) == 1
            Chicago_Hub_Route_Average_Cost  = 0.30;
        end
        if ~exist('Chicago_Hub_Route_Average_Speed') ||
isempty(Chicago_Hub_Route_Average_Speed ) == 1
            Chicago_Hub_Route_Average_Speed  = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Chicago_Hub_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Chicago_Hub_Route,Chicago_Hub_Route_Average_Cost,Chica
go_Hub_Route_Initial_Year,Chicago_Hub_Route_Average_Speed);

    end

    if strcmp(Southeast_Route, 'NO') ~= 1
        if ~exist('Southeast_Route_Initial_Year') ||
isempty(Southeast_Route_Initial_Year) == 1
            Southeast_Route_Initial_Year = 2011;
        end
        if ~exist('Southeast_Route_Average_Cost') ||
isempty(Southeast_Route_Average_Cost) == 1
            Southeast_Route_Average_Cost  = 0.30;
        end
        if ~exist('Southeast_Route_Average_Speed') ||
isempty(Southeast_Route_Average_Speed) == 1
            Southeast_Route_Average_Speed = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Southeast_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Southeast_Route,Southeast_Route_Average_Cost,Southeast
_Route_Initial_Year,Southeast_Route_Average_Speed);

    end

    if strcmp(Empire_Service_Route, 'NO') ~= 1
        if ~exist('Empire_Service_Route_Initial_Year') ||
isempty(Empire_Service_Route_Initial_Year) == 1
            Empire_Service_Route_Initial_Year = 2011;
        end
        if ~exist('Empire_Service_Route_Average_Cost') ||
isempty(Empire_Service_Route_Average_Cost) == 1
            Empire_Service_Route_Average_Cost  = 0.30;
        end
```

```
        if ~exist('Empire_Service_Route_Average_Speed') ||
isempty(Empire_Service_Route_Average_Speed) == 1
            Empire_Service_Route_Average_Speed = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Empire_Service_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Empire_Service_Route,Empire_Service_Route_Average_Cost
,Empire_Service_Route_Initial_Year,Empire_Service_Route_Average_Speed);

    end

    if strcmp(Northern_New_England_Route, 'NO') ~= 1
        if ~exist('Northern_New_England_Route_Initial_Year') ||
isempty(Northern_New_England_Route_Initial_Year) == 1
            Northern_New_England_Route_Initial_Year= 2011;
        end
        if ~exist('Northern_New_England_Route_Average_Cost') ||
isempty(Northern_New_England_Route_Average_Cost) == 1
            Northern_New_England_Route_Average_Cost  = 0.30;
        end
        if ~exist('Northern_New_England_Route_Average_Speed') ||
isempty(Northern_New_England_Route_Average_Speed) == 1
            Northern_New_England_Route_Average_Speed = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Northern_New_England_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Northern_New_England_Route,Northern_New_England_Route_
Average_Cost,Northern_New_England_Route_Initial_Year,Northern_New_England_Rou
te_Average_Speed);

    end

    if strcmp(Keystone_Route, 'NO') ~= 1
        if ~exist('Keystone_Route_Initial_Year ') ||
isempty(Keystone_Route_Initial_Year ) == 1
            Keystone_Route_Initial_Year  = 2011;
        end
        if ~exist('Keystone_Route_Average_Cost') ||
isempty(Keystone_Route_Average_Cost) == 1
            Keystone_Route_Average_Cost  = 0.30;
        end
        if ~exist('Keystone_Route_Average_Speed') ||
isempty(Keystone_Route_Average_Speed) == 1
            Keystone_Route_Average_Speed = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Keystone_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Keystone_Route,Keystone_Route_Average_Cost,Keystone_Ro
ute_Initial_Year,Keystone_Route_Average_Speed);

    end

    if strcmp(South_Central_Route, 'NO') ~= 1
```

```
        if ~exist('South_Central_Route_Initial_Year') ||
isempty(South_Central_Route_Initial_Year) == 1
            South_Central_Route_Initial_Year = 2011;
        end
        if ~exist('South_Central_Route_Average_Cost') ||
isempty(South_Central_Route_Average_Cost) == 1
            South_Central_Route_Average_Cost  = 0.30;
        end
        if ~exist('South_Central_Route_Average_Speed') ||
isempty(South_Central_Route_Average_Speed) == 1
            South_Central_Route_Average_Speed = 105;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
South_Central_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,South_Central_Route,South_Central_Route_Average_Cost,S
outh_Central_Route_Initial_Year,South_Central_Route_Average_Speed);

    end

    if strcmp(Gulf_Coast_Route, 'NO') ~= 1
        if ~exist('Gulf_Coast_Route_Initial_Year') ||
isempty(Gulf_Coast_Route_Initial_Year) == 1
            Gulf_Coast_Route_Initial_Year = 2011;
        end
        if ~exist('Gulf_Coast_Route_Average_Cost') ||
isempty(Gulf_Coast_Route_Average_Cost ) == 1
            Gulf_Coast_Route_Average_Cost   = 0.30;
        end
        if ~exist('Gulf_Coast_Route_Average_Speed') ||
isempty(Gulf_Coast_Route_Average_Speed ) == 1
            Gulf_Coast_Route_Average_Speed  = 105;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Gulf_Coast_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Gulf_Coast_Route,Gulf_Coast_Route_Average_Cost,Gulf_Co
ast_Route_Initial_Year,Gulf_Coast_Route_Average_Speed);
    end

    if strcmp(Vermont_Route, 'NO') ~= 1
        if ~exist('Vermont_Route_Initial_Year') ||
isempty(Vermont_Route_Initial_Year) == 1
            Vermont_Route_Initial_Year = 2011;
        end
        if ~exist('Vermont_Route_Average_Cost') ||
isempty(Vermont_Route_Average_Cost) == 1
            Vermont_Route_Average_Cost  = 0.30;
        end
        if ~exist('Vermont_Route_Average_Speed') ||
isempty(Vermont_Route_Average_Speed) == 1
            Vermont_Route_Average_Speed = 77;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Vermont_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
```

```
e,S2S_Train_TravelCost,Vermont_Route,Vermont_Route_Average_Cost,Vermont_Route
_Initial_Year,Vermont_Route_Average_Speed);

    end

    if strcmp(Cumulative_NEC_Route, 'NO') ~= 1
        if ~exist('Cumulative_NEC_Route_Initial_Year') ||
isempty(Cumulative_NEC_Route_Initial_Year) == 1
            Cumulative_NEC_Route_Initial_Year = 2011;
        end
        if ~exist('Cumulative_NEC_Route_Average_Cost') ||
isempty(Cumulative_NEC_Route_Average_Cost) == 1
            Cumulative_NEC_Route_Average_Cost  = 0.30;
        end
        if ~exist('Cumulative_NEC_Route_Average_Speed') ||
isempty(Cumulative_NEC_Route_Average_Speed) == 1
            Cumulative_NEC_Route_Average_Speed = 100;
        end
        [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Cumulative_NEC_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Cumulative_NEC_Route,Cumulative_NEC_Route_Average_Cost
,Cumulative_NEC_Route_Initial_Year,Cumulative_NEC_Route_Average_Speed);

    end



    % %
    %     %          [~,Number_Stations_Total_Network] =
size(TrainStation_List);
    %         Station_Index       = 1:Number_Stations_Total_Network;

    %         S2S_Train_ScheduleDelay=S2S_Train_ScheduleDelay_HSR;
    %         clear S2S_Train_ScheduleDelay_HSR

    %         S2S_Train_Distance=S2S_Train_Distance_HSR(Station_Index,
Station_Index); %this renames the distance matrix.
    %         clear S2S_Train_Distance_HSR
end % if strcmp(Train,'YES')==0
% ----------------------------
% set diagonal elements to zero
% ----------------------------
S2S_Train_TravelCost(logical(eye(size(S2S_Train_TravelCost)))) = 0;
S2S_Train_TravelTime(logical(eye(size(S2S_Train_TravelTime)))) = 0;
% ----------------------------
% save matrices
% ----------------------------
save([Output_Save_Directory,'S2S_Train_Distance_',num2str(Year_analysis),'.ma
t'],'S2S_Train_Distance')
save([Output_Save_Directory,'S2S_Train_TravelTime_',num2str(Year_analysis),'.
mat'],'S2S_Train_TravelTime')
save([Output_Save_Directory,'S2S_Train_TravelCost_',num2str(Year_analysis),'.
mat'],'S2S_Train_TravelCost')
save([Output_Save_Directory,'S2S_Train_ScheduleDelay_',num2str(Year_analysis)
,'.mat'],'S2S_Train_ScheduleDelay')
```

```
save([Output_Save_Directory,'C2S_Train_Access_Cost_',num2str(Year_analysis),'
.mat'],'C2S_Train_Access_Cost')
```

D.2 Assisting route functions

D.2.1. Acela Rail

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Acela_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

Station_Abbreviation_List =
{'STM','NHV','NLC','WIL','WAS','BWI','BAL','RTE','BOS','NWK','TRE','MET','NYP
','PHL','PVD','BBY'}; %Stations in the NEC. The file Train_Station_ID_Finder.
m can be used to find these codes based on the desired city.
%The Station Abbreviation List is representative of Stamford CT, New Haven
%CT, New London CT, Wilmington DE, Washington DC, Baltimore MD (airport),
%Baltimore MD, Westwood MA, Boston MA, Newark NJ, Trenton NJ, Metropark NJ,
%New York NY, Phidelphia PA, and Providence RI.

[~,Number_Stations]        =size(Station_Abbreviation_List);
Station_Route_Index =zeros(1,Number_Stations);

for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});  % finds the respective station ID's
end

Train_Frequency      = ones(Number_Stations,Number_Stations)*21; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
Train_Service_Period = 16; %The service period of trains. This is used for
the calculation of schedule delay.
Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.

S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.


if Year_analysis <= 2000
    %----------------------------------------
    % YEARS 1995-2000

    %A 3rd order polynomial function was fit for the Travel Cost in the NEC
    %regions.  The intercept was fixed at 0.
    %y=A*X^3 + B*X^2 + C *X

    A = 2E-7;
    B = -0.0006;
```

```
    C = 0.4439;

    %A linear function was fit for Travel Time in the NEC. Intercept was fit
to 0.
    %y=M*x
    M=0.9629; %This regression was done using minutes therefore in the
equation below, it is divided by 60
       %---------------------------------------

    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(M*S2S_Train_Distance(Station_Route_Index,Station_Route_Index))/60;  % in
hours
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,((A*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3))+
(B*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+(C*(S2S_
Train_Distance(Station_Route_Index,Station_Route_Index)))))*Inflation_Factor;
% A minimum of 25 dollars is assigned to cost because regression curve
produces negative values for short distances. The $30 is valid for 2011 and
it is adjusted for inflation for the desired year.

elseif Year_analysis > 2000
    %YEARS 2001-2010  Acela Travel Time and Cost Curves
    %A 3rd order polynomial function was fit for the Travel Cost in the NEC
    %(years 2001-2010)
    %y=A*X^3 + B*X^2 + C *X + D

    A = -1E-06;
    B = 0.0006;
    C = 0.1654;
    D = 10.304;

    M = -2e-06;
    N = 0.0012;
    O = 0.8418;
    P = -13.929;

    % Cost and time curve by Alex Vadnyke in 2010
    %     M = 7E-7;
    %     N = -0.0015;
    %     P = 1.0279;
    %     Q = 20.018;
        %---------------------------------------
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
max(0.15,((M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3)
)+(N*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+(O*(S2
S_Train_Distance(Station_Route_Index,Station_Route_Index))+P))/60);
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,((A*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3))+
(B*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+(C*(S2S_
Train_Distance(Station_Route_Index,Station_Route_Index))+D)))*Inflation_Facto
r; % A minimum of 25 dollars is assigned to cost because regression curve
produces negative values for short distances. The $30 is valid for 2011 and
it is adjusted for inflation for the desired year.
end
```

D.2.2. California HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= California_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,California_Route,California_Route_Average_Cost,Califor
nia_Route_Initial_Year,California_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if California_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'SJC','SAC','MCD','FNO','BFD','LAX','RIV','SAN','ANA','DAV','EMY','FUL','GDL
','IRV','OAC','ONA','POS','PRB','RIC','RLN','RSV','SNA'}; %Major stations end
at 'SAN'
    %The Station Abbreviation List is representative of San Jose, Sacramento,
    %Merced, Fresno, Bakersfield, Los Angeles, Riverside, and San Diego. All
    %stations are in CA.  Other Stations listed are non-major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are
    %located.
    [~,Number_Stations]        = size(Station_Abbreviation_List);
    Station_Route_Index   = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*5; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/California_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Sta
tion_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,California_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,S
tation_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.3. California Zephyr Input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
California_Zephyr_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

Station_Abbreviation_List =
{'CHI','BRL','MTP','OTM','OSC','CRN','OMA','LNK','HAS','HLD','MCK','FMG','WIP
','GRA','GSC','GJT','GRI','HER','PRO','SLC','ELK','WNN'};
[~,Number_Stations]        =size(Station_Abbreviation_List);
Califronia_zephyr_index =zeros(1,Number_Stations);
for a=1:1:Number_Stations

Califronia_zephyr_index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_
List{1,a});
end


% The California Zephyr route has a quadratic function for time in
% minutes and a 3rd degree polynomial function for cost in 2010 $
%y=A*X^2 + B*X + C
A = 0.0002;
B = 0.798;
C = - 1.3545;

%y=M*X^3 + N*X^2 + O *X + P
M = 3E-08;
N = - 0.0001 ;
O = 0.198;
P = 3.0502;

S2S_Train_TravelTime(Califronia_zephyr_index,Califronia_zephyr_index) =
(A*S2S_Train_Distance(Califronia_zephyr_index,Califronia_zephyr_index).^2+B*(
S2S_Train_Distance(Califronia_zephyr_index,Califronia_zephyr_index)+ C))./60;
%hrs

S2S_Train_TravelCost(Califronia_zephyr_index,Califronia_zephyr_index) =
max(12,((M*(S2S_Train_Distance(Califronia_zephyr_index,Califronia_zephyr_inde
x).^3))+(N*(S2S_Train_Distance(Califronia_zephyr_index,Califronia_zephyr_inde
x).^2))+(O*(S2S_Train_Distance(Califronia_zephyr_index,Califronia_zephyr_inde
x)))+P))*Inflation_Factor; % A minimum of 12 dollars is assigned to cost
because regression curve produces negative values for short distances. The
$30 is valid for 2011 and it is adjusted for inflation for the desired year.
```

D.2.4. Chicago Hub Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Chicago_Hub_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Chicago_Hub_Route,Chicago_Hub_Route_Average_Cost,Chica
go_Hub_Route_Initial_Year,Chicago_Hub_Route_Average_Speed);


global TrainStation_List
global Inflation_Factor


if Chicago_Hub_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'CHI','KCY','STL','OMA','DET','PNT','ALN','BNL','CRV','DWT','JOL','LCN','NPV
','PON','SMT','SPI','IDP','KWD','DYE','HMI','MCI','ALI','ARB','BAM','BMM','BT
L','JXN','KAL','NBM','ROY','SJM'};
    %The Station Abbreviation List is representative of Chicago, Kansas City,
St. Louis, Omaha NE, Detroit, Pontiac MI. Other Stations listed are non-major
stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.
    %located.

    [~,Number_Stations]        = size(Station_Abbreviation_List);
    Station_Route_Index  = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.

S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Chicago_Hub_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,St
ation_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Chicago_Hub_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,
Station_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.5. Cumulative NEC Route HSR

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Cumulative_NEC_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Cumulative_NEC_Route,Cumulative_NEC_Route_Average_Cost
,Cumulative_NEC_Route_Initial_Year,Cumulative_NEC_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Cumulative_NEC_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'STM','NHV','NLC','WIL','WAS','BWI','BAL',...
        'RTE','BOS','NWK','TRE','MET','NYP','PHL','PVD','ROC','BFX',...
        'NFL','ALB','AMS','HUD','POU','RHI','ROM','SDY','SYR','UCA','YNY',...
        'SPG','FED','FTC','GZZ','PLB','POH','PRK','RSP',...
        'WHL','WSP','FRA','WOR','BER','HFD','MDN','HAR','PGH','COV',...
        'GNB','LAB','PAO','MPR','SAB','SPG','HFD','BLF','BRA','ESX',...
        'RPH','WAB','WNM','WRJ','AMM','BER','MDN'};
    %The Station Abbreviation List is representative of all stations that
    %are in the various NEC Corridors.

    [~,Number_Stations]    = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.

S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Cumulative_NEC_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index
,Station_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Cumulative_NEC_Route_Average_Cost*S2S_Train_Distance(Station_Route_Ind
ex,Station_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.6. Empire Builder Input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Empire_Builder_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

Station_Abbreviation_List =
{'CHI','CBS','POG','WDL','TOH','LSE','WIN','RDW','MSP','SCD','SPL','DLK','FAR
','GFK','DVL','RUG','MOT','STN','WTN','WPT','GGW','MAL','HAV','SBY','CUT','BR
O','GPK','ESM','WGL','WFH','LIB','SPT','SPK','PSC','WIH','BNG'};

[~,Number_Stations]        =size(Station_Abbreviation_List);
Station_Route_Index =zeros(1,Number_Stations);
for a=1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
end


% Empire Builder Regression Coefficients
% Linear relaton for Time travel in minutes : y = A *x2 + B*x +C

A = 0.0001;
B = 0.8393;
C = 45.592;

% Polynomial relation of order 2 for Cost in 2010 $
% y=M*X^2 + N*X + O

M = -3E-05;
N = 0.1346;
O = 4.2626;

S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A.*S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2+B.*S2S_Tra
in_Distance(Station_Route_Index,Station_Route_Index)+C)./60; %hrs

S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(12,((M.*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))
+(N.*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index)))+O))*Infla
tion_Factor;

% A minimum of 25 dollars is assigned to cost because regression curve
produces negative values for short distances. The $30 is valid for 2011 and
it is adjusted for inflation for the desired year.
```

D.2.7. Empire Service Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Empire_Service_Route_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor


Station_Abbreviation_List =
{'NYP','ROC','BFX','NFL','ALB','AMS','HUD','POU','RHI','ROM','SDY','SYR','UCA
','YNY'};
%The Station Abbreviation List is representative of New York NY, Rochester
%NY, Buffalo NY, ,Niagara Falls NY and Rensselaer NY(Albany). Other Stations
listed are non-major stations, see
%Projected Rail Corridor Data sheet to see where other stations are.

[~,Number_Stations]    = size(Station_Abbreviation_List);
Station_Route_Index   = zeros(1,Number_Stations);

for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
end

% Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
% Train_Service_Period = 16; %The service period of trains. This is used for
the calculation of schedule delay.
% Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.

    %-------------------------------------------------------------------
    %     Empire Service Time and Cost Fundtions
    %A 3rd order polynomial function was fit for the Travel Cost on half of
the the
    %Empire Service Route from NYP - ALB
    %     A linear equation is set for the time
    %  time = A*distance^3 + B*distance^2 +C*distance + D
    A = -5E-06;
    B = 0.0038;
    C = 0.56;
    D = 10.774;
    % cost = M*X^3 + N*X^2 + O *X + P
    %

    M = -1E-06;
    N = 0.0003;
    O = 0.1907;
```

```
    P = 6.2883;
    %This regression was done using minutes
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3)+B*(S2S_Tr
ain_Distance(Station_Route_Index,Station_Route_Index).^2)+C*(S2S_Train_Distan
ce(Station_Route_Index,Station_Route_Index)+D))./60; %minutes
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(20,(M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3)+N*
(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2)+O*S2S_Train_
Distance(Station_Route_Index,Station_Route_Index)+P))*Inflation_Factor; % $.
A minimum of 25 dollars is assigned to cost. The cost is in 2010 $ and then
adjusted back to inflation.

    %-----------------------------------------------------------------------
end
```

D.2.8. Empire Service Route

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Empire_Service_Route_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor


Station_Abbreviation_List =
{'NYP','ROC','BFX','NFL','ALB','AMS','HUD','POU','RHI','ROM','SDY','SYR','UCA
','YNY'};
%The Station Abbreviation List is representative of New York NY, Rochester
%NY, Buffalo NY, ,Niagara Falls NY and Rensselaer NY(Albany). Other Stations
listed are non-major stations, see
%Projected Rail Corridor Data sheet to see where other stations are.

[~,Number_Stations]    = size(Station_Abbreviation_List);
Station_Route_Index   = zeros(1,Number_Stations);

for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
end

% Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
% Train_Service_Period = 16; %The service period of trains. This is used for
the calculation of schedule delay.
```

```
% Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.

    %-----------------------------------------------------------------------
    %     Empire Service Time and Cost Fundtions
    %A 3rd order polynomial function was fit for the Travel Cost on half of
the the
    %Empire Service Route from NYP - ALB
    %     A linear equation is set for the time
    %  time = A*distance^3 + B*distance^2 +C*distance + D
    A = -5E-06;
    B = 0.0038;
    C = 0.56;
    D = 10.774;
    % cost = M*X^3 + N*X^2 + O *X + P
    %

    M = -1E-06;
    N = 0.0003;
    O = 0.1907;
    P = 6.2883;
    %This regression was done using minutes
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3)+B*(S2S_Tr
ain_Distance(Station_Route_Index,Station_Route_Index).^2)+C*(S2S_Train_Distan
ce(Station_Route_Index,Station_Route_Index)+D))./60; %minutes
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(20,(M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^3)+N*
(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2)+O*S2S_Train_
Distance(Station_Route_Index,Station_Route_Index)+P))*Inflation_Factor; % $.
A minimum of 25 dollars is assigned to cost. The cost is in 2010 $ and then
adjusted back to inflation.

    %-----------------------------------------------------------------------
end
```

D.2.9. Florida Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Florida_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Florida_Route,Florida_Route_Average_Cost,Florida_Route
_Initial_Year,Florida_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Florida_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'TPA','ORL','MIA','DFB','DLB','FTL','HOL','KIS','LAK','SFD','WPB','WPK','WTH
'};
```

```
    %The Station Abbreviation List is representative of Tampa, Orlando,
Miami. Other Stations listed are non-major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are
    %located.

    [~,Number_Stations]   = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Florida_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Statio
n_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Florida_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,Stat
ion_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is assigned
to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.10. Gulf Coast Route HSR

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Gulf_Coast_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Gulf_Coast_Route,Gulf_Coast_Route_Average_Cost,Gulf_Co
ast_Route_Initial_Year,Gulf_Coast_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Gulf_Coast_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'ATL','BHM','MOE','BIX','MEI','NOL','HOS','ATN','TCL','HBG','LAU','PIC','LCH
','LFT','BMT'};
```

```
    %The Station Abbreviation List is representative of Atlanta, Birmingham
AL, Mobile AL, Biloxi MS, Meridian MS, New Orleans, Houston TX . Other
Stations listed are non-major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.

    [~,Number_Stations]    = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Gulf_Coast_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Sta
tion_Route_Index);
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Gulf_Coast_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,S
tation_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.11. Keystone Route HSR

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Keystone_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Keystone_Route,Keystone_Route_Average_Cost,Keystone_Ro
ute_Initial_Year,Keystone_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Keystone_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List = {'PHL','HAR','PGH','COV','GNB','LAB','PAO'};
    %The Station Abbreviation List is representative of Philadelphia PA,
    %Harrisburg PA, Pittsburgh PA. Other Stations listed are non-major
stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.
```

```
    [~,Number_Stations]    = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Keystone_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Stati
on_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Keystone_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,Sta
tion_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.12. Lincoln Service Input

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Lincoln_Service_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

if Year_analysis >= 2006
    % The lincoln service runs from Chicago to St. Louis, Missouri
    % This part of the code corrects the frequency and time and cost
    % functions for the lincoln service
    % the first run was in 2006

    Station_Abbreviation_List =
{'CHI','SMT','JOL','DWT','PON','BNL','LCN','SPI','CRV','ALN','STL'};
    [~,Number_Stations]        =size(Station_Abbreviation_List);
    Station_Route_Index =zeros(1,Number_Stations);
```

```
    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*5; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 16; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.

    %-----------------------------------------
    %Lincoln Service Coefficients
    %-------------------------------------
    % For Travel Time , the regression is done in minutes, so needs to be
    % divided by 60 to conveert it into hours
    % A*x^2+B*x+C
    A = 0.0007;
    B = 0.7213;
    C = 17.1;

    % For travel cost, the regression is done for 2010$
    %M*x^2 + N*x+O
    M = -4E-05;
    N = 0.0818;
    O = 1.5957;
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A*S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2+B*S2S_Train
_Distance(Station_Route_Index,Station_Route_Index)+B)./60; %hrs

    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(2,((M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+(
N*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index)))+O))*Inflatio
n_Factor;

 % A minimum of 2 dollars is assigned to cost because regression curve
produces negative values for short distances. The $30 is valid for 2011 and
it is adjusted for inflation for the desired year.

end

end
```

D.2.13. NEC Route HSR Input

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= NEC_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,NEC_Route,NEC_Route_Average_Cost,NEC_Route_Initial_Yea
r,NEC_Route_Average_Speed); %This applies only after year 2010.

global TrainStation_List
global Inflation_Factor

if NEC_Route_Initial_Year <= Year_analysis
    %NEC
    %-------------------------------------------------------------------------
-----------------
    Station_Abbreviation_List =
{'STM','NHV','NLC','WIL','WAS','BWI','BAL','RTE','BOS','NWK','TRE','MET','NYP
','PHL','PVD','BBY'}; %Stations in the NEC. The file Train_Station_ID_Finder.
m can be used to find these codes based on the desired city.
    %The Station Abbreviation List is representative of Stamford CT, New
Haven
    %CT, New London CT, Wilmington DE, Washington DC, Baltimore MD (airport),
    %Baltimore MD, Westwood MA, Boston MA, Newark NJ, Trenton NJ, Metropark
NJ,
    %New York NY, Phidelphia PA, and Providence RI.

    [~,Number_Stations]        =size(Station_Abbreviation_List);
    Station_Route_Index =zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*21; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 16; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/NEC_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Station_Ro
ute_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,NEC_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,Station_
```

```
Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is assigned to
cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.14. Northern NEW England Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Northern_New_England_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Northern_New_England_Route,Northern_New_England_Route_
Average_Cost,Northern_New_England_Route_Initial_Year,Northern_New_England_Rou
te_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Northern_New_England_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'ALB','SPG','NHV','BOS','FED','FTC','GZZ','PLB','POH','PRK','RSP','WHL','WSP
','FRA','WOR','BER','HFD','MDN'};
    %The Station Abbreviation List is representative of Rensselaer NY
(Albany),
    %Springfield MA, New Haven CT, Boston MA. Other Stations listed are non-
major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.

    [~,Number_Stations]                  = size(Station_Abbreviation_List);
    Station_Route_Index   = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Northern_New_England_Route_Average_Speed)*S2S_Train_Distance(Station_Route
_Index,Station_Route_Index); %hrs
```

```
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Northern_New_England_Route_Average_Cost*S2S_Train_Distance(Station_Rou
te_Index,Station_Route_Index))*Inflation_Factor; % $.  A minimum of 25
dollars is assigned to cost. The cost is in 2010 $ and then adjusted back to
inflation.
end
```

D.2.15. Pacific Northwest Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Pacific_Northwest_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Pacific_Northwest_Route,Pacific_Northwest_Route_Averag
e_Cost,Pacific_Northwest_Route_Initial_Year,Pacific_Northwest_Route_Average_S
peed);


global TrainStation_List
global Inflation_Factor

if Pacific_Northwest_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'SEA','TAC','PDX','EUG','ALY','ORC','SLM','BEL','CTL','EDM','EVR','KEL','MVW
','OLW','TUK'};
    %The Station Abbreviation List is representative of Seattle WA, Tacoma
WA,
    %Portland OR, and Eugene OR. Other Stations listed are non-major
stations, see
    %Projected Rail Corridor Data sheet to see where other stations are
    %located.

    [~,Number_Stations]                = size(Station_Abbreviation_List);
    Station_Route_Index    = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay;
```

```
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Pacific_Northwest_Route_Average_Speed)*S2S_Train_Distance(Station_Route_In
dex,Station_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Pacific_Northwest_Route_Average_Cost*S2S_Train_Distance(Station_Route_
Index,Station_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars
is assigned to cost. The cost is in 2010 $ and then adjusted back to
inflation.

end
```

D.2.16. Pacific Surfliner Input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Pacific_Surfliner_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

if Year_analysis > 1999   % Pacific Surfliner started in 2000

    Pacific_Surfliner_CA_Abbreviation_List =
{'SAN','SOL','OSD','SNC','IRV','SNA','ANA','FUL','LAX','GDL','SIM','OXN','SBA
','SAT','GVB','SLO'}; % these have a differenct cost and time function than
other corridors.
    Pacific_Surfliner_CA_half_Abbreviation_List =
{'SAN','SOL','OSD','SNC','IRV','SNA','ANA','FUL','LAX'}; % this portion of
the Paicific surfliner has greater frequency

    %-------------------------------------------------------
    % this portion assigns schedule delay to the entire list
    [~,Number_Stations]        =
size(Pacific_Surfliner_CA_Abbreviation_List);
    Station_Route_Index  = zeros(1,Number_Stations);

    for a=1:1:Number_Stations
        Station_Route_Index
(1,a)=Train_Station_Index_Finder(Pacific_Surfliner_CA_Abbreviation_List{1,a})
;
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*6; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 16; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.
```

```
S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.

    %-----------------------------------------------
    % for the shorter part where the frequency is higher
    [~,Number_Stations]        =
size(Pacific_Surfliner_CA_half_Abbreviation_List);
    Station_Route_Index1  = zeros(1,Number_Stations);

    for b=1:1:Number_Stations
        Station_Route_Index1
(1,b)=Train_Station_Index_Finder(Pacific_Surfliner_CA_half_Abbreviation_List{
1,b});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*11; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index1,Station_Route_Index1)=Schedule_D
elay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.

    %-----------------------------------------------------


End
```

D.2.17. South Central Route HSR input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
South_Central_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,South_Central_Route,South_Central_Route_Average_Cost,S
outh_Central_Route_Initial_Year,South_Central_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if South_Central_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'OKC','AUS','DAL','FTW','SAS','LRK','TXA','ADM','NOR','PUR','PVL','CBR','GLE
','MCG','SMC','TAY','TPL','ARK','MVN'};
    %The Station Abbreviation List is representative of Oklahoma City, Austin
TX, Dallas, Ft. Worth TX, San Antonio, Little Rock AR, and Texarkana AR .
Other Stations listed are non-major stations, see
```

```
    %Projected Rail Corridor Data sheet to see where other stations are.

    [~,Number_Stations]         = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/South_Central_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,
Station_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,South_Central_Route_Average_Cost*S2S_Train_Distance(Station_Route_Inde
x,Station_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.18. Southeast Route HSR

```
function [S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost]
= Southeast_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Southeast_Route,Southeast_Route_Average_Cost,Southeast
_Route_Initial_Year,Southeast_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if Southeast_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'CLT','GRO','RGH','RVR','WAS','BNC','CYN','DNC','HPT','KAN','SAL','ALX','FBG
','MSS','PTB','QAN'};
    %The Station Abbreviation List is representative of Charlotte NC,
    %Greensboro NC, Raleigh NC, Richmond VA, and Washington DC. Other
Stations listed are non-major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.
```

```
    [~,Number_Stations]       = size(Station_Abbreviation_List);
    Station_Route_Index   = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end

    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Southeast_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Stat
ion_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Southeast_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,St
ation_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is
assigned to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.19. Sunset Input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Sunset_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

Station_Abbreviation_List =
{'LAX','POS','ONA','PSN','YUM','MRC','TUS','BEN','LDB','DEM','ELP','ALP','SND
','DRT','SAS','HOS','BMT','LCH','LFT','NIB','SCH','NOL'};
[~,Number_Stations]        =size(Station_Abbreviation_List);
Station_Route_Index              = zeros(1,Number_Stations);

for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
end
%-----------------------------------------
```

```
%Sunet Limited Coefficients
%-------------------------------------
% For Travel Time , the regression is done in minutes, so needs to be
% divided by 60 to conveert it into hours
% A*x^2+B*x+C
A = 0.0002;
B = 0.8742;
C = 45.961;

% For travel cost, the regression is done for 2010$
%M*x^2 + N*x+O
M = -5E-05;
N = 0.2605;
O = -10.059;


S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A*S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2+B*S2S_Train
_Distance(Station_Route_Index,Station_Route_Index)+C)./60; %hrs

S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(12,((M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+
(N*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index)))+O))*Inflati
on_Factor; % A minimum of 12 dollars is assigned to cost because regression
curve produces negative values for short distances. The $30 is valid for 2011
and it is adjusted for inflation for the desired year.
```

D.2.20. Texas Eagle input

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Texas_Eagle_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost)

global TrainStation_List
global Inflation_Factor

Station_Abbreviation_List =
{'CHI','KKI','CHM','MAT','EFG','CEN','CDL','PBF','WNR','LRK','MVN','ARK','TXA
','MHL','LVW','MIN','DAL','FTW','CBR','MCG','TPL','TAY','AUS','SMC','SAS'}';

[~,Number_Stations]        = size(Station_Abbreviation_List);
Station_Route_Index  = zeros(1,Number_Stations);

for a=1:1:Number_Stations
    Station_Route_Index
(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List{1,a});
end

%-------------------------------------------------------------------------
%     Texas Eagle Service
%-------------------------------------------------------------------------
%    % For Travel Time , the regression is done in minutes, so needs to be
```

```
% divided by 60 to conveert it into hours
% A*x^2+B*x+C
A = 0.0004;
B = 0.629;
C = 188.46;

% For travel cost, the regression is done for 2010$
%M*x^2 + N*x+O
M = -0.0002;
N = 0.4087;
O = -80.612;

S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(A*S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2+B*S2S_Train
_Distance(Station_Route_Index,Station_Route_Index)+C)./60; %hrs

S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(12,((M*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index).^2))+
(N*(S2S_Train_Distance(Station_Route_Index,Station_Route_Index)))+O))*Inflati
on_Factor; % A minimum of 25 dollars is assigned to cost because regression
curve produces negative values for short distances. The $30 is valid for 2011
and it is adjusted for inflation for the desired year.
```

D.2.21. Vermont Route HSR

```
function[S2S_Train_ScheduleDelay,S2S_Train_TravelTime,S2S_Train_TravelCost] =
Vermont_Route_HSR_Input
(Year_analysis,S2S_Train_ScheduleDelay,S2S_Train_Distance,S2S_Train_TravelTim
e,S2S_Train_TravelCost,Vermont_Route,Vermont_Route_Average_Cost,Vermont_Route
_Initial_Year,Vermont_Route_Average_Speed);

global TrainStation_List
global Inflation_Factor

if  Vermont_Route_Initial_Year <= Year_analysis
    Station_Abbreviation_List =
{'MPR','SAB','SPG','HFD','NHV','BLF','BRA','ESX','RPH','WAB','WNM','WRJ','AMM
','BER','MDN'};
    %The Station Abbreviation List is representative of Montpelier VT, St.
Albans VT, Springfield MA, Hartford CT, New Haven CT. Other Stations listed
are non-major stations, see
    %Projected Rail Corridor Data sheet to see where other stations are.

    [~,Number_Stations]    = size(Station_Abbreviation_List);
    Station_Route_Index = zeros(1,Number_Stations);

    for a=1:1:Number_Stations

Station_Route_Index(1,a)=Train_Station_Index_Finder(Station_Abbreviation_List
{1,a});
    end
```

```
    Train_Frequency      = ones(Number_Stations,Number_Stations)*8; %This is
assigning the frequency of train between all possible combinations of
stations above to be 6 trains. Once an actual schedule is developed, then a
new matrix can be created.
    Train_Service_Period = 18; %The service period of trains. This is used
for the calculation of schedule delay.
    Schedule_Delay       =
Schedule_Delay_Calculator(Train_Frequency,Train_Service_Period); %Schedule
delay for stations listed above.


S2S_Train_ScheduleDelay(Station_Route_Index,Station_Route_Index)=Schedule_Del
ay; %Updates the main S2S schedule delay with the delay calculated for
stations listed above.
    S2S_Train_TravelTime(Station_Route_Index,Station_Route_Index) =
(1/Vermont_Route_Average_Speed)*S2S_Train_Distance(Station_Route_Index,Statio
n_Route_Index); %hrs
    S2S_Train_TravelCost(Station_Route_Index,Station_Route_Index) =
max(25,Vermont_Route_Average_Cost*S2S_Train_Distance(Station_Route_Index,Stat
ion_Route_Index))*Inflation_Factor; % $.  A minimum of 25 dollars is assigned
to cost. The cost is in 2010 $ and then adjusted back to inflation.
end
```

D.2.22. Train station index finder

```
%This file finds the index number of a rail station.  The city name is used
%for the station, name must be entered in all caps.

function [Station_Index]=Train_Station_Index_Finder(Train_Station_Code)
global TrainStation_List
load('C:\Program Files
(x86)\TSAM\6.6\data\mode_choice\input\TrainStation_List.mat')  % Train
station list


Station_Index=0;

for i=1:1:464
    if strcmp(TrainStation_List(1,i).Station_ID,Train_Station_Code)==1
        Station_Index=i;
    end
end
```

D.2.23. Schedule Delay Calculator

```
%This file calculates the schedule delay for service between stations.
function
[Schedule_Delay]=Schedule_Delay_Calculator(Train_Frequency,Train_Service_Peri
od)

Schedule_Delay=Train_Service_Period./(Train_Frequency .* 4);
```

D.3 Rail station selection file

```
function C2S_update(Analysis_Year)
%This file creates the C2SData_Train variable that is used by TSAM.  The 4
%columns of this file are the County Index (0:3091), Closest Station Index,
%Driving Distance, Driving Time.
global TrainStation_List
 Install_Dir = 'C:\Program Files (x86)\TSAM\6.6';
 Analysis_Year = 1995;

Number_of_Counties=3091;  %number of counties in the analysis.
Number_of_Stations=464; %number of stations in analysis.
% Analysis_Year=2020; %Desired Year for Creating the C2SData_Train.mat file.
This file changes because of the congestion factors that are applied to the
driving time. %This currently runs between 1995 and 2040.

load(strcat(Install_Dir,'\data\mode_choice\input\C2S\C2SDriveDist_Population_
Centroids.mat'));  %load the County to Station driving distance that were
computed with mappoint.   The driving distance and time were computed for
only the 5 closest rail stations to each county. That is why there is -999
vlaues in most of the C2SDriveDist_Population_Centroids file.
load(strcat(Install_Dir,'\data\mode_choice\input\C2S\C2SDriveTime_Population_
Centroids_Congestion_',num2str(Analysis_Year),'.mat')) % Driving Times from
County to Station
% Load the subway metro files
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\chicago_metro.mat'));  % has counties list assigned to the stations
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\LA_metro.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\newyork_subway.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\phl_metro.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\washington_metro.mat'));
load(strcat(Install_Dir,'\data\mode_choice\input\Train Station Subway
Access\pvd_transit.mat'));


% ------------------------------------------------
% create train network
%-------------------------------------------------
if Analysis_Year >= 1995 && Analysis_Year <= 2002
    Train_Network_Year = 2002;
elseif Analysis_Year >= 2003 || Analysis_Year <= 2006
    Train_Network_Year = str2num(CaseYear);
elseif Analysis_Year < 2006 || Analysis_Year <= 2008
    Train_Network_Year = 2006;
elseif Analysis_Year >= 2009
    Train_Network_Year = 2009;
end
load(strcat(Install_Dir,'\data\mode_choice\input\Y',num2str(Train_Network_Yea
r),'\TrainStation_List.mat'));
```

```
% ------------------------------------------------
% Find stations
%-------------------------------------------------
PHL_code = Train_Station_Index_Finder('PHL');  % finds the station code
NYP_code = Train_Station_Index_Finder('NYP');
WAS_code = Train_Station_Index_Finder('WAS');
PVD_code = Train_Station_Index_Finder('PVD');
CHI_code = Train_Station_Index_Finder('CHI');
LAX_code = Train_Station_Index_Finder('LAX');
C2SData_Train=zeros(3091,4);

for a=1:Number_of_Counties
% a=8;
    Drive_Distance=1000000; %preallocation
    if isempty(intersect(phl_metro(:,1),a))== 0 % check if the station
belongs to the phiadelphia metro area
        a1= find(phl_metro == a); % find the index
        Drive_Time=phl_metro(a1,3); % subway time
        b=PHL_code; % station index for PHL
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

    elseif isempty(intersect(washington_metro(:,1),a))== 0 % check if the
station belongs to the washington metro area
        b=WAS_code; % station index for WAS
        a1= find(washington_metro == a); % find the index
        Drive_Time=washington_metro(a1,3); % subway time
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

    elseif isempty(intersect(pvd_transit(:,1),a))== 0 % check if the station
belongs to the washington metro area
        b=PVD_code; % station index for PVD
        a1= find(pvd_transit == a); % find the index
        Drive_Time=pvd_transit(a1,3); % subway time
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

    elseif isempty(intersect(newyork_subway(:,1),a))== 0 % check if the
station belongs to the NY metro area
        b=NYP_code; % station index for NYP
        a1= find(newyork_subway == a); % find the index
        Drive_Time=newyork_subway(a1,3); % subway time
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

    elseif isempty(intersect(chicago_metro(:,1),a))== 0 % check if the
station belongs to the chicago metro area
        b=CHI_code; % station index for CHI
        a1= find(chicago_metro == a); % find the index
        Drive_Time=chicago_metro(a1,3); % subway time
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);
```

```
    elseif isempty(intersect(LA_metro(:,1),a))== 0 % check if the station
belongs to the LAX metro area
        b=LAX_code; % station index for LAX
        a1= find(LA_metro == a); % find the index
        Drive_Time=LA_metro(a1,3); % subway time
          Assigned_Station_Index=b;
    Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

    else
        for b=1:1:Number_of_Stations %This determines the origin station
number and the county index number where the station is located.
            if C2SDriveDist_Population_Centroids(a,b)>0 &&
C2SDriveDist_Population_Centroids(a,b)<Drive_Distance %&&
C2SDriveDist_Population_Centroids(orgCountyReal,b)<Drive_Distance_Buffer
%this ensures that the distance exists(since not all C2S pairs were
calculated), that the distance is within the buffer range, and that the
station with shortest drive distance is selected for that county.

                Assigned_Station_Index=b;
                Drive_Time=C2SDriveTime_Population_Centroids_Congestion(a,b);
                Drive_Distance=C2SDriveDist_Population_Centroids(a,b);

                %orgStation_County_Index=Station_County_Index_Number(1,b);
            end
        end
    end



        if Drive_Distance==1000000 %Drive_Distance =1000000 means there were
no driving calculations from that county to any given station.  Mappoint
wasn't able to find directions for all coutnies.  If this is true, -999 is
assigned as a placeholder.
        C2SData_Train(a,1)=a;
        C2SData_Train(a,2)=-999;
        C2SData_Train(a,3)=-999;
        C2SData_Train(a,4)=-999;
        else
        C2SData_Train(a,1)=a;
        C2SData_Train(a,2)=Assigned_Station_Index;
        C2SData_Train(a,3)=Drive_Distance;
        C2SData_Train(a,4)=Drive_Time;
        end

end


%save(['C:\Users\avandyke\Documents\TSAM_GUI_VB_2005\bin\data\mode_choice\inp
ut\C2SData_Train',num2str(Analysis_Year),'.mat'],'C2SData_Train');
save((strcat(Install_Dir,'\data\mode_choice\input\TrainCounty2Airport\C2SData
_Train_',num2str(Analysis_Year),'.mat')),'C2SData_Train');
```

D.4 Analyze Train Access Distances

```
% This m-file analyzes the commercial airport access distance
clc;
clear all;
close all;
format long g;

% ------------------
% Options
% ------------------
MSA_Combination = 'All'; % All, MSA to MSA, MSA to Non-MSA, Non-MSA to MSA,
Non-MSA to Non-MSA
Year            = '1995';
TripPurpose     = 'Both'; %Both Business, Non-Business

% ------------------
% File Laoding
% ------------------
Input_Dir  = 'E:\Analyze_Train_Access_Distance';
Output_Dir = 'E:\Analyze_Train_Access_Distance';
TSAM_Dir   = 'E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data';

load ([TSAM_Dir, '\trip_distribution\input\stateNames.mat']);
Number_Of_States = size(stateNames,1);

% ------------------
% Read ATS Data File
% ------------------
% ATS_File = fopen([Input_Dir, '\', TripPurpose, ' - ', MSA_Combination, ' -
ATS Data - 02_25_2009.txt'], 'r');
if strcmp (TripPurpose,'Business') == 1 || strcmp (TripPurpose,'Non-
Business') == 1
ATS_File = fopen([Input_Dir, '\', TripPurpose, ' - ', MSA_Combination, ' -
ATS Data - HH - 04_29_2009 - Distance Filtered.txt'], 'r');
ATS_Field_Names = fgets(ATS_File);
ATS_Data = textscan(ATS_File,
'%s%s%f%s%s%s%s%f%s%s%s%f%s%f%s%s%s%f%s%s%f%f%s%f%f%f%f', 'delimiter', '\t');
elseif strcmp (TripPurpose,'Both') == 1
ATS_File1 = fopen([Input_Dir, '\', 'Business', ' - ', MSA_Combination, ' -
ATS Data - HH - 04_29_2009 - Distance Filtered.txt'], 'r');
ATS_File2 = fopen([Input_Dir, '\', 'Non-Business', ' - ', MSA_Combination, '
- ATS Data - HH - 04_29_2009 - Distance Filtered.txt'], 'r');
ATS_Field_Names1 = fgets(ATS_File1);
ATS_Field_Names = fgets(ATS_File2);
ATS_Data1= textscan(ATS_File1,
'%s%s%f%s%s%s%s%f%s%s%s%f%s%f%s%s%s%f%s%s%f%f%s%f%f%f%f', 'delimiter', '\t');
ATS_Data2 = textscan(ATS_File2,
'%s%s%f%s%s%s%s%f%s%s%s%f%s%f%s%s%s%f%s%s%f%f%s%f%f%f%f', 'delimiter', '\t');
[a1,b1] = size (ATS_Data1);
[a2,b2] = size (ATS_Data2);
ATS_Data = zeros(a1+a2,b1);
ATS_Data(a1,b1) = ATS_Data1;
ATS_Data(a1:a2,b1) = ATS_Data2;  % combining the two data sets
end
% Field List (PERSON TRIP TABLE):
```

```
% Field 1: OMETNAME
% Field 2: OMETCODE
% Field 3: ORIGIN CEEDS CODE
% Field 4: OSTNAME
% Field 5: OSTPOST
% Field 6: DMETNAME
% Field 7: DMETCODE
% Field 8: DESTINATION CEEDS CODE
% Field 9: DSTNAME
% Field 10: DSTPOST
% Field 11: REASON
% Field 12: TRANSOD
% Field 13: TRANSDO
% Field 14: GCDOD
% Field 15: GCDDO
% Field 16: GCDRT
% Field 17: GCDTOSTA
% Field 18: RTEDUSOD
% Field 19: RTEDUSDO
% Field 20: RTEDUSRT
% Field 21: RTDTOSTA
% Field 22: HHINC
% Field 23: PERTRIPS
% Field 24: TRPARTY
% Field 25: PERTRWGT
% Field 26: HHID
% Field 27: TRPRTYHH

% Field List (HOUSEHOLD TRIP TABLE):
% Field 1: OMETNAME
% Field 2: OMETCODE
% Field 3: ORIGIN CEEDS CODE
% Field 4: OSTNAME
% Field 5: OSTPOST
% Field 6: DMETNAME
% Field 7: DMETCODE
% Field 8: DESTINATION CEEDS CODE
% Field 9: DSTNAME
% Field 10: DSTPOST
% Field 11: REASON
% Field 12: TRANSOD
% Field 13: TRANSDO
% Field 14: GCDOD
% Field 15: GCDDO
% Field 16: GCDRT
% Field 17: GCDTOSTA
% Field 18: RTEDUSOD
% Field 19: RTEDUSDO
% Field 20: RTEDUSRT
% Field 21: RTDTOSTA
% Field 22: HHINC
% Field 23: TRIPS
% Field 24: TRPARTY
% Field 25: PERTRWGT
% Field 26: HHID
% Field 27: TRPRTYHH
```

```
OMETNAME                = ATS_Data{1};
ORIGIN_CEEDS_CODE       = ATS_Data{3};
OSTPOST                 = ATS_Data{5};
DMETNAME                = ATS_Data{6};
DESTINATION_CEEDS_CODE  = ATS_Data{8};
DSTPOST                 = ATS_Data{10};
TRANSOD                 = ATS_Data{12};
RTEDUSOD                = ATS_Data{18};
RTDTOSTA                = ATS_Data{21};
HHINC                   = ATS_Data{22};
TRPARTY                 = ATS_Data{24};
PERTRWGT                = ATS_Data{25};

% ------------------
% Simplify Mode Of Transportation
% ------------------
Index_Of_Auto  = (TRANSOD == 1 | TRANSOD == 2 | TRANSOD == 3 | TRANSOD == 15
| TRANSOD == 17);
Index_Of_CA    = TRANSOD == 4;
Index_Of_Train  = TRANSOD == 9;
Index_Of_Other = TRANSOD ~= 1 & TRANSOD ~= 2 & TRANSOD ~= 3 & TRANSOD ~= 9 &
TRANSOD ~= 4 & TRANSOD ~= 15 & TRANSOD ~= 17;

TRANSOD(Index_Of_Auto)  = 1;
TRANSOD(Index_Of_CA)    = 2;
TRANSOD(Index_Of_Train) = 3;
TRANSOD(Index_Of_Other) = 4;

clear Index_Of_Auto;
clear Index_Of_CA;
clear Index_Of_Train;
clear Index_Of_Other;

% ------------------
% Simplify HouseHold Income Group
% The simplyfication takes into account the conversion from 1995 $ to 2000
% $ using the BLS CPI index of 1.1299.
% ------------------
Income_Group_List = {'<$25K', '$25K - $50K', '$50K - $75K', '$75K - $125K',
'>$125K', 'All'};
Index_Of_Inc1 = find(HHINC == 1 | HHINC == 2 | HHINC == 3); % <$250K
Index_Of_Inc2 = find(HHINC == 4 | HHINC == 5 | HHINC == 6); % $25K - $50K
Index_Of_Inc3 = find(HHINC == 7 | HHINC == 8); % $50K - $75K
Index_Of_Inc4 = find(HHINC == 9 | HHINC == 10); % $75K - $125K
Index_Of_Inc5 = find(HHINC == 11 | HHINC == 12); % >$125K

HHINC(Index_Of_Inc1) = 1;
HHINC(Index_Of_Inc2) = 2;
HHINC(Index_Of_Inc3) = 3;
HHINC(Index_Of_Inc4) = 4;
HHINC(Index_Of_Inc5) = 5;

clear Index_Of_Inc1;
clear Index_Of_Inc2;
```

```
clear Index_Of_Inc3;
clear Index_Of_Inc4;
clear Index_Of_Inc5;

Number_Of_IncomeGroups = 5;
% for inc_grp = 3%1: Number_Of_IncomeGroups
% ------------------
% Calculate averages and bining
% ------------------
Min_Distance = 0;
Max_Distance = 250;
Bin_Step     = 1; % mile

Bin_Edges      = Min_Distance : Bin_Step : Max_Distance;
Number_Of_Bins = length(Bin_Edges) - 1;

AccessDistance_PDF = zeros(Number_Of_Bins,4);
AccessDistance_CDF = zeros(Number_Of_Bins,4);

for Region = 1:4

    if Region == 1 % MSA2MSA
        Index_Of_Train = TRANSOD == 3 & isnan(ORIGIN_CEEDS_CODE) == 0 &
isnan(DESTINATION_CEEDS_CODE) == 0 ;
    elseif Region == 2 % MSA2NonMSA
        Index_Of_Train = TRANSOD == 3 & isnan(ORIGIN_CEEDS_CODE) == 0 &
isnan(DESTINATION_CEEDS_CODE) == 1 ;
    elseif Region == 3 % NonMSA2MSA
        Index_Of_Train = TRANSOD == 3 & isnan(ORIGIN_CEEDS_CODE) == 1 &
isnan(DESTINATION_CEEDS_CODE) == 0 ;
    elseif Region == 4 % NonMSA2NonMSA
        Index_Of_Train = TRANSOD == 3 & isnan(ORIGIN_CEEDS_CODE) == 1 &
isnan(DESTINATION_CEEDS_CODE) == 1;
    end

    Access_Distances     = RTDTOSTA(Index_Of_Train);
    Trip_Weights         = PERTRWGT(Index_Of_Train);

    % Averages
    Unweighted_Mean_Access_Distance(Region,1) = mean(Access_Distances);
    Weighted_Mean_Access_Distance(Region,1)   = round(sum(Access_Distances .*
Trip_Weights) / sum(Trip_Weights));

    % Binning
    for i = 1 : Number_Of_Bins

        Start_Dist = Bin_Edges(i);
        End_Dist   = Bin_Edges(i+1);

        Dist_Index = find(Access_Distances >= Start_Dist & Access_Distances <
End_Dist);

        AccessDistance_PDF(i,Region) = sum(Trip_Weights(Dist_Index));
        if i == 1
            AccessDistance_CDF(i,Region) = sum(Trip_Weights(Dist_Index));
```

```
        else
            AccessDistance_CDF(i,Region) = AccessDistance_CDF(i-1,Region) +
sum(Trip_Weights(Dist_Index));
        end

    end % for i = 1 : Number_Of_Bins + 1

    Total_Person_Trips = sum(Trip_Weights);

    AccessDistance_PDF(:,Region) = AccessDistance_PDF(:,Region) /
Total_Person_Trips * 100;
    AccessDistance_CDF(:,Region) = AccessDistance_CDF(:,Region) /
Total_Person_Trips * 100;

end % for Region = 1:4

% % ------------------
% % Plot PDF
% % ------------------
% plot(Bin_Edges(1:end-1), AccessDistance_PDF(:,2))
% title(['ATS 1995 - ', TripPurpose, ' - ' , MSA_Combination, ' - Airport
Access Distance PDF - Mean: ', num2str(Weighted_Mean_Access_Distance), '
miles.'], 'FontSize', 20);
% xlabel('Access Distance (miles)', 'FontSize', 16);
% ylabel('%', 'FontSize', 16);
% % xlim([0 500]);
% grid on;

% ------------------
% Plot CDF
% ------------------
figure;
hold on;
plot(Bin_Edges(1:end-1), AccessDistance_CDF(:,1), 'b')
plot(Bin_Edges(1:end-1), AccessDistance_CDF(:,2), 'r')
plot(Bin_Edges(1:end-1), AccessDistance_CDF(:,3), 'g')
plot(Bin_Edges(1:end-1), AccessDistance_CDF(:,4), 'k')
title(['ATS 1995 - ', TripPurpose, ' - Station Access Distance CDF'],
'FontSize', 20);
legend('MSA2MSA', 'MSA2NonMSA', 'NonMSA2MSA', 'NonMSA2NonMSA', 'Location',
'SouthEast');
xlabel('Access Distance (miles)', 'FontSize', 12);
ylabel('Round Trips (%)', 'FontSize', 12);
grid on;

% figure;
% hold on;
% plot(Bin_Edges(1:end-1), AccessDistance_PDF(:,1), 'b')
% title(['ATS 1995 - ', TripPurpose, ' - Station Access Distance PDF MSA-
MSA'], 'FontSize', 20);
% xlabel('Access Distance (miles)', 'FontSize', 12);
% ylabel('Round Trips (%)', 'FontSize', 12);
% grid on;
% figure
% plot(Bin_Edges(1:end-1), AccessDistance_PDF(:,2), 'r')
```

```
% title(['ATS 1995 - ', TripPurpose, ' - Station Access Distance PDF MSA-
NonMSA'], 'FontSize', 20);
% xlabel('Access Distance (miles)', 'FontSize', 12);
% ylabel('Round Trips (%)', 'FontSize', 12);
% grid on;
% figure
% plot(Bin_Edges(1:end-1), AccessDistance_PDF(:,3), 'g')
% title(['ATS 1995 - ', TripPurpose, ' - Station Access Distance PDF NonMSA-
MSA'], 'FontSize', 20);
% xlabel('Access Distance (miles)', 'FontSize', 12);
% ylabel('Round Trips (%)', 'FontSize', 12);
% grid on;
% figure
% plot(Bin_Edges(1:end-1), AccessDistance_PDF(:,4), 'k')
% title(['ATS 1995 - ', TripPurpose, ' - Station Access Distance PDF NonMSA-
NonMSA'], 'FontSize', 20);
% % legend('MSA2MSA', 'MSA2NonMSA', 'NonMSA2MSA', 'NonMSA2NonMSA',
'Location', 'SouthEast');
% xlabel('Access Distance (miles)', 'FontSize', 12);
% ylabel('Round Trips (%)', 'FontSize', 12);
% grid on;

% end
% ------------------
% Saving
% ------------------
if strcmp(TripPurpose, 'Business') == 1
   ATS_AccessDistance_CDF_Business(:,1) = Bin_Edges(1:end-1);
   ATS_AccessDistance_CDF_Business(:,2) = AccessDistance_CDF(:,1);
   ATS_AccessDistance_CDF_Business(:,3) = AccessDistance_CDF(:,2);
   ATS_AccessDistance_CDF_Business(:,4) = AccessDistance_CDF(:,3);
   ATS_AccessDistance_CDF_Business(:,5) = AccessDistance_CDF(:,4);
   save ATS_AccessDistance_CDF_Business.mat ATS_AccessDistance_CDF_Business;
elseif strcmp(TripPurpose, 'Non-Business') == 1
   ATS_AccessDistance_CDF_NonBusiness(:,1) = Bin_Edges(1:end-1);
   ATS_AccessDistance_CDF_NonBusiness(:,2) = AccessDistance_CDF(:,1);
   ATS_AccessDistance_CDF_NonBusiness(:,3) = AccessDistance_CDF(:,2);
   ATS_AccessDistance_CDF_NonBusiness(:,4) = AccessDistance_CDF(:,3);
   ATS_AccessDistance_CDF_NonBusiness(:,5) = AccessDistance_CDF(:,4);
   save ATS_AccessDistance_CDF_NonBusiness.mat
ATS_AccessDistance_CDF_NonBusiness;
end

fclose('all');
```

D.5 Transfer rate computing

D.5.1 Compute Transfer Ridership

```
function [Transfer_Ridership] =
Compute_Transfer_Ridership(orstn,dsstn,input_dir,transfr_no,S2Sridership,Tran
sfer_Ridership)

% Code to compute Transfer Ridership
```

```
% %%
% Last Edited : 18th July 2012
% Saloni
%% Load Variables
clc
% clear all

global Major_hub_count_Ridership
global Major_hub_count_Ridership_2
 global TrainStation_List
global number_major_hubs
% load ([input_dir,'\s2sTrain_pTripTable_Total.mat']); % the total trip
purpose, roudned to positive infinity, train ridership from MC output. change
the file location according to the latest file
% input_dir = 'E:\train_network_stn';
load([input_dir,'\Station_hubs.mat']); % This variable contains the selected
hubs nationwide in the rank order of their annual ridership
load ([input_dir,'\Route_train.mat']); % the file has the Stations for the
different Amtrak Routes in the US. The list of the Route trains can be found
in the Excel Sheet : Train Route list . xls
% load([input_dir,'\Transfer_Ridership.mat']); % this variable would compute
the transfer ridership along with the station ID
load('C:\Program Files
(x86)\TSAM\6.6\data\mode_choice\input\Y2009\S2S_Train_Distance_2009.mat'); %
The S2S distances are sued to select the best transfer station
% load('C:\Program Files
(x86)\TSAM\6.6\data\mode_choice\input\TrainStation_List.mat')  % Train
station list

%%-------------------------------------------------------

[Routes_total,stn_no] = size(Route_train);  % The number of Routes and the
number of stations
[a,hub_count] = size(Station_hubs); % The number of hubs considered
% [total_stations] = length(TrainStation_List);

% number_major_hubs=2; % is used if the transfer between two small stations
is to be restrictited from the major hubs
%%---------------------------------------

% for orstn = 295:total_stations
%     for dsstn = 141:total_stations  % For every OD pair combinations
try
combo_no = 0;
combo_hub=[];
same_rt_combo=[];
ds_combo_stn=[];
or_combo_stn=[];
[or_route,x] = find(Route_train == orstn); % find routes that have the origin
station
[ds_route,y] = find(Route_train == dsstn); % find routes that have the
destination station
same_rt = intersect(or_route,ds_route); % to check if there is a direct train
between the rotues

if isempty(same_rt)== 1  % if they are in the same route , quit
```

```
    or_route_1 = Route_train(or_route,:); % finds all the stations with the
Routes that have the Origin station
    ds_route_1 = Route_train(ds_route,:);% finds all the stations with the
Routes that have the Destination station
    [b1 b2 ] = size (or_route_1); % to find the number of Routes
    trnsfr1 = double(ismember(or_route_1,ds_route_1)); % Checks for common
stations in between the route stations
    for c= 1:b1  % exclude the station IDs with zero from further
consideration
        d = find( or_route_1(c,:) == 0);
        trnsfr1(c,d) = 0;  %
    end %c= 1:b1
    or_hub_chk = double(ismember(Station_hubs,or_route_1)); % checks if there
are hubs in the origin routes
    ds_hub_chk = double(ismember(Station_hubs,ds_route_1)); % assigns a value
of 1 or 0 for matching records
    or_ds_chk = or_hub_chk+ ds_hub_chk;
    index = find (or_ds_chk == 2); % checks for common hubs i.e. those with a
value of 1 in both matrix
    trnsfr_hub =Station_hubs (index ); % the Trains_Station_ID of the
transfer hub, if any.

    %%-----------------------------------------------
    % if there are more than one common hubs, distance and
    % ranks will be used to give priority to a specific
    % transferring hub

    % Case 1 :  Only one common station and that is  hub
    if numel(trnsfr_hub) == 1  % if there is only one common station and that
is a hub
        index = find (Station_hubs ==  trnsfr_hub); % find the ID of the
station in the hubs
        Transfer_Ridership(index ,2) = Transfer_Ridership(index ,2) +
S2Sridership; % add the ridership to the transfer ridership of the hub

        % Case 2 : Many common potentail transfer hubs
    elseif numel(trnsfr_hub) > 1  % if there are many common hubs
        dist_hub=
(S2S_Train_Distance(orstn,trnsfr_hub)+S2S_Train_Distance(dsstn,trnsfr_hub));
% computes the total travel distance for every potential transfer hub
        dist_min = min(dist_hub);% find the minimum distance
        longer_dist = (dist_hub>1.05*dist_min); % 05% greater distance to be
eliminated
        dist_hub_1 = dist_hub(longer_dist==0); % seelcts only the hubs within
the least distance ranges
        hub_index = trnsfr_hub(longer_dist==0);  % choose the one with the
minimum total travle distance
        for i = 1: numel(hub_index)
            ranks_hub(i) = min(find(Station_hubs==hub_index(i), 1 )); % finds
the most prefered lowest dstance route
        end
        % Case 1 very low ridership
        if (numel(hub_index) > 1&& ... % if there are more than 1 minimum
distance stations, use rank preference.
                S2Sridership < Major_hub_count_Ridership)% if several chocies
are still available
```

```
            n = find (ranks_hub<=number_major_hubs); % finds if the first two
stations viz. NYP and WAS need to be included

            if  numel(n) < numel(hub_index)
                ranks_hub (:,n) = [];  % Delete the ones with major hubs
            else n=[];
                n = find (ranks_hub<=number_major_hubs-1); % else check only
for the first major hub
                if  numel(n) < numel(hub_index)
                    ranks_hub (:,n) = [];  % Delete the ones with major hubs
                end %umel(n) < numel(combo_hub(1,:))
            end %numel(n) < numel(combo_hub(1,:))
            % Case 2 : Low Ridership
        elseif (numel(hub_index) > 1&& ... % if there are more than 1 minimum
distance stations, use rank preference.
                S2Sridership < Major_hub_count_Ridership_2)% if several
chocies are still available
            n = find (ranks_hub<=number_major_hubs-1); % else check only for
the first major hub
            if  numel(n) < numel(hub_index)
                ranks_hub (:,n) = [];  % Delete the ones with major hubs

            end %

        end  %(numel(hub_index) > 1&& ... % if there are more than 1 minimum
distance stations, use rank preference.

        index = min(ranks_hub); % the one with the highest preference

        %                           index = min(find(Station_hubs==hub_index));

        Transfer_Ridership(index ,2) = Transfer_Ridership(index ,2) +
S2Sridership; % add the ridership to the transfer ridership of the hub

        %% Case 3 no common hubs
    elseif numel(trnsfr_hub) == 0 % there is no common hub
        % Subcase 1 : Some other common station - not among the
        % selcted hubs
        if max(max(trnsfr1,[],1),[],2) == 1 % if there is some other common
station, do nothing. trnsfr1 is a logical array of 0 or 1,
            % subcase 2: A Transfer posibility
        else
            % two or more transfers
            or_hub = Station_hubs(or_hub_chk==1); % checks hubs in the origin
routes
            ds_hub = Station_hubs(ds_hub_chk==1);
            % compute minimum distance for all the
            % combinations

            for or_combo = 1: length (or_hub)
                for ds_combo= 1: length (ds_hub)  % for every combination of
OD hubs
                    combo_no = combo_no+1;
```

```
                    combo_hub(1,combo_no) = S2S_Train_Distance
(orstn,or_hub(or_combo))+ S2S_Train_Distance
(or_hub(or_combo),ds_hub(ds_combo))+ S2S_Train_Distance
(ds_hub(ds_combo),dsstn);
                    combo_hub(2,combo_no) = or_hub(or_combo);   % notes the
first hub of transfer
                    combo_hub(3,combo_no) = ds_hub(ds_combo);  % notes the
2nd hub
                    combo_hub(4,combo_no) =
find(Station_hubs==or_hub(or_combo));   % notes the first hub of transfer
Station_hub Rank
                    combo_hub(5,combo_no) =
find(Station_hubs==ds_hub(ds_combo));  % notes the 2nd hub of transfer
Station_hub Rank
                    [or_combo_stn,x] = find(Route_train == or_hub(or_combo));
% find routes that have the origin station
                    [ds_combo_stn,y] = find(Route_train == ds_hub(ds_combo));
% find routes that have the destination station
                    same_rt_combo(1,combo_no) =1-
isempty(intersect(or_combo_stn,ds_combo_stn)); % to check if there is a
direct train between the rotues
                    combo_hub(6,combo_no) = same_rt_combo(1,combo_no);
%Stores 1 or 0 depedning on whether there is a connection between the selcted
hubs
                    ds_combo_stn=[];
                    or_combo_stn=[];
                end %ds_combo= 1: length (ds_hub)
            end %or_combo = 1: length (or_hub)

            combo_hub=sortrows(combo_hub',1)'; % Sorts according to ascending
order.

            if (max (combo_hub(6,:))==0 && S2Sridership ~= 0 &&
transfr_no==1) % i.e. no common routes, and has more than 2 transfers.
                distance_max = 1.15*combo_hub(1,1);
                longer_combo = find(combo_hub(1,:) > distance_max); %Delete
the station combos with a distance of more than 15% of the minimum distance
                combo_hub (:,longer_combo) = [];

                if (numel(combo_hub(1,:)) > 1 && ... % if several chocies are
still available
                        S2Sridership < Major_hub_count_Ridership)
                    n = find (combo_hub(4,:)<=number_major_hubs); % finds if
the first two stations viz. NYP and WAS need to be included

                    if  numel(n) < numel(combo_hub(1,:))
                        combo_hub (:,n) = [];  % Delete the ones with major
hubs
                    else n=[];
                        n = find (combo_hub(4,:)<=(number_major_hubs-1)); %
else check only for the first major hub
                        if  numel(n) < numel(combo_hub(1,:))
                            combo_hub (:,n) = [];  % Delete the ones with
major hubs
                        end %umel(n) < numel(combo_hub(1,:))
                    end %numel(n) < numel(combo_hub(1,:))
```

```
                    nm =  find (combo_hub(5,:)<=number_major_hubs);   %
destination hub
                    if (numel(combo_hub(1,:)) > 1 && numel(nm) <
numel(combo_hub(1,:)))
                        combo_hub (:,nm) = [];  % Delete the ones with major
hubs
                    else nm=[];
                        nm = find (combo_hub(5,:)<=(number_major_hubs-1)); %
else check only for the first major hub
                        if     numel(nm) < numel(combo_hub(1,:))
                            combo_hub (:,nm) = [];  % Delete the ones with
major hubs
                        end
                    end
                elseif (numel(combo_hub(1,:)) > 1&& ... % if there are more
than 1 minimum distance stations, use rank preference.
                        S2Sridership < Major_hub_count_Ridership_2)% if
several chocies are still available
                    n = find (combo_hub(4,:)<=number_major_hubs-1); % else
check only for the first major hub
                    if  numel(n) < numel(combo_hub(1,:))
                        combo_hub (:,n) = [];  % Delete the ones with major
hubs
                    end
                end
                combo_hub(7,:) = combo_hub(4,:) + combo_hub(5,:) ;% sum up
the total ranks
            combo_hub=sortrows(combo_hub',7)'; % Sorts according to ascending
order.
                or_hub_1 =   combo_hub(2,1);
                ds_hub_1 =  combo_hub(3,1); % assigns the selected O and D
hubs
                transfr_no = 2;
                [Transfer_Ridership] =
Compute_Transfer_Ridership(or_hub_1,ds_hub_1,input_dir,transfr_no,S2Sridershi
p,Transfer_Ridership);

                index1 =combo_hub(4,1);
                index2 =  combo_hub(5,1); % assigns the selected O and D hubs
index numbers

                Transfer_Ridership(index1 ,2) = Transfer_Ridership(index1 ,2)
+ S2Sridership; % add the ridership to the transfer ridership of the hub
                Transfer_Ridership(index2 ,2) = Transfer_Ridership(index2 ,2)
+ S2Sridership; % add the ridership to the transfer ridership of the hub

            elseif  (max (combo_hub(6,:))==0 &&S2Sridership ~= 0 &&
transfr_no >1) % i.e. no common routes, and has more than 4 transfers.

                disp ('OD pair');
                disp (orstn);
                disp (dsstn)
                disp('Ridership');
                disp(S2Sridership);
```

```
            elseif (max (combo_hub(6,:))==1)
                % There are only two transfers

                no_connection = find(combo_hub(6,:)==0); % find combinations
with no connections
                combo_hub (:,no_connection) = []; % delete the ones with no
conenctions. Assuming that extra transfers is a disutility.
                distance_max = 1.15*combo_hub(1,1);
                longer_combo = find(combo_hub(1,:) > distance_max); %Delete
the station combos with a distance of more than 15% of the minimum distance
                combo_hub (:,longer_combo) = [];
                % if the ridership between the stations is really
                % low, it is assumed that major transfer hubs wud
                % be avoided if other options are available.
                if (numel(combo_hub(1,:)) > 1 && ... % if several chocies are
still available
                        S2Sridership < Major_hub_count_Ridership)
                    n = find (combo_hub(4,:)<=number_major_hubs); % finds if
the first two stations viz. NYP and WAS need to be included

                    if  numel(n) < numel(combo_hub(1,:))
                        combo_hub (:,n) = [];  % Delete the ones with major
hubs
                    else n=[];
                        n = find (combo_hub(4,:)<=(number_major_hubs-1)); %
else check only for the first major hub
                        if  numel(n) < numel(combo_hub(1,:))
                            combo_hub (:,n) = [];  % Delete the ones with
major hubs
                        end %umel(n) < numel(combo_hub(1,:))
                    end %numel(n) < numel(combo_hub(1,:))
                    nm =  find (combo_hub(5,:)<=number_major_hubs);   %
destination hub
                    if (numel(combo_hub(1,:)) > 1 && numel(nm) <
numel(combo_hub(1,:)))
                        combo_hub (:,nm) = [];  % Delete the ones with major
hubs
                    else nm=[];
                        nm = find (combo_hub(5,:)<=(number_major_hubs-1)); %
else check only for the first major hub
                        if     numel(nm) < numel(combo_hub(1,:))
                            combo_hub (:,nm) = [];  % Delete the ones with
major hubs
                        end
                    end

                elseif (numel(combo_hub(1,:)) > 1&& ... % if there are more
than 1 minimum distance stations, use rank preference.
                        S2Sridership < Major_hub_count_Ridership_2)% if
several chocies are still available
                    n = find (combo_hub(1,:)<=number_major_hubs-1); % else
check only for the first major hub
                    if  numel(n) < numel(combo_hub(1,:))
                        combo_hub (:,n) = [];  % Delete the ones with major
hubs
                    end
```

```
                end %(numel(combo_hub(1,:)) > 1 &&
s2sTrain_pTripTable_Total(orstn,dsstn) < 1
                index1 =    combo_hub(4,1);
                index2 = combo_hub(5,1); % assigns the selected O and D hubs
                Transfer_Ridership(index1 ,2) = Transfer_Ridership(index1 ,2)
+ S2Sridership; % add the ridership to the transfer ridership of the hub
                Transfer_Ridership(index2 ,2) = Transfer_Ridership(index2 ,2)
+ S2Sridership; % add the ridership to the transfer ridership of the hub


            end %(max (combo_hub(6,:))==0 &&
s2sTrain_pTripTable_Total(orstn,dsstn) ~= 0)
        end %  max(max(trnsfr1,[],1),[],2) == 1
    end %numel(trnsfr_hub) == 1
end  %  if isempty(same_rt)== 1 % no transfer and quit
catch ME
%             disp(orstn,'Origin Station',dsstn,'Destination Station')
clear or_*
clear b*
clear d*
clear x*
clear y*
clear trnsfr*
clear r*
clear hub*
clear c*
clear sam*
% end
% save ([input_dir, '\Transfer_Ridership.mat'], 'Transfer_Ridership');
%     end % for dsstn = 462:total_stations
end  % for orstn = 462:total_stations
```

D.5.2 To run Compute Transfer

```
clc
clear all
%
% global S2S_Train_Distance
% global TrainStation_List

input_dir = 'E:\train_network_stn\';
% load('C:\Program Files
(x86)\TSAM\6.6\data\mode_choice\input\Y2009\S2S_Train_Distance_2009.mat'); %
The S2S distances are sued to select the best transfer station
load('C:\Program Files
(x86)\TSAM\6.6\data\mode_choice\input\TrainStation_List.mat')  % Train
station list
```

```
load ([input_dir,'\s2sTrain_pTripTable_Total.mat']); % the total trip
purpose, roudned to positive infinity, train ridership from MC output. change
the file location according to the latest file
load([input_dir,'\Transfer_Ridership.mat']); % this variable would compute
the transfer ridership along with the station ID

global Major_hub_count_Ridership
global Major_hub_count_Ridership_2
global number_major_hubs
[total_stations] = length(TrainStation_List);
Major_hub_count_Ridership = 1;
Major_hub_count_Ridership_2 = 4;
number_major_hubs=2; % is used if the transfer between two small stations is
to be restrictited from the major hubs



for orstn = 1:total_stations
    if mod(orstn,100)==0
        disp([num2str(orstn),'/',num2str(total_stations),' Origins
Processed'])
    end
    for dsstn = 1: total_stations  % For every OD pair combinations
        if mod(dsstn,total_stations)==0
            disp([num2str(dsstn),'/',num2str(total_stations),' Destinations
Processed'])
        end
        transfr_no = 1;
        S2Sridership = s2sTrain_pTripTable_Total(orstn,dsstn);
        %         disp(['Origin ',num2str(orstn)])
        [Transfer_Ridership] =
Compute_Transfer_Ridership(orstn,dsstn,input_dir,transfr_no,S2Sridership,Tran
sfer_Ridership);

    end
end

save ([input_dir, '\Transfer_Ridership_1.mat'], 'Transfer_Ridership');
```

D.6 Comparing TSAM distribution with AMTRAK

```
% This code will find the ridership along distances from a particular
station, and put them into different bins.
% It will then calculate the percentage of the ridership for distances in
groups of 100 miles
% THIS FILE needs mode choice output files pasted in a folder called
% s2s_trips on the desktop
% it uses the function Train_Station_Index_Finder.m
% you need to enter the train station ID from the AMTRAK id list
clc
clear all

load
C:\Users\saloni\Documents\MATLAB\AMTRAK\AMTRAK_ridership_percentage_wout_100m
iles.mat % this file contains the ridership percentage along different miles,
as computed using the AMTRAK data
load('C:\Users\saloni\Desktop\s2s_trips\business\s2sTrain_pTripTable_Total.ma
t') %This file contains the business trips output - mode choice
business_trips = s2sTrain_pTripTable_Total;
clear s2sTrain_pTripTable_Total
load('C:\Users\saloni\Desktop\s2s_trips\non-
business\s2sTrain_pTripTable_Total.mat')  % This file contains the non
business trips output for mode choice
non_business_trips = s2sTrain_pTripTable_Total;
clear s2sTrain_pTripTable_Total

load('E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data\mode_choice\input\S2S_T
rain_Distance_HSR'); %This is the original matrix of distance between
stations. This is the original data and is assumed to be correct unless
specific changes have been noted below.

global TrainStation_List
load('C:\Users\saloni\Desktop\s2s_trips\TrainStation_List.mat')  %This  file
containts the AMTRAK "TrainStation_List"

total_stations = length (business_trips);
total_ridership = zeros (1,20);

reply = input ( ' Please enter the abbreviations three letter in CAPS ',
's');
AMTRAK_id = input('The AMTRAK station ID, Find it in the excel file called
AMTRAK ID xls ');
station_abbreviation  = {reply}; % Converts to a cell array
Ridership_Station_Index =  Train_Station_Index_Finder(station_abbreviation);

for i = 1: total_stations
    dist = S2S_Train_Distance_HSR (i,Ridership_Station_Index);

           if (dist > 99 && dist < 200) % distances between 100 and 200
          total_ridership  (1,1)  =  total_ridership  (1,1)+  business_trips
(i,Ridership_Station_Index)                  +                 business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 199 && dist < 300) % distances between 200 - 300
          total_ridership  (1,2)  =  total_ridership  (1,2)+  business_trips
(i,Ridership_Station_Index)                  +                 business_trips
```

```
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 299 && dist < 400)
           total_ridership  (1,3)  =  total_ridership  (1,3)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 399 && dist < 500)
            total_ridership  (1,4)  =  total_ridership  (1,4)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 499 && dist < 600)
           total_ridership  (1,5)  =  total_ridership  (1,5)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 599 && dist < 700)
           total_ridership  (1,6)  =  total_ridership  (1,6)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 699 && dist < 800)
           total_ridership  (1,7)  =  total_ridership  (1,7)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 799 && dist < 900)
           total_ridership  (1,8)  =  total_ridership  (1,8)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 899 && dist < 1000)
           total_ridership  (1,9)  =  total_ridership  (1,9)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 999 && dist < 1100)
           total_ridership  (1,10)  =  total_ridership  (1,10)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 1099 && dist < 1200)
           total_ridership  (1,11)  =  total_ridership  (1,11)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 1199 && dist < 1300)
           total_ridership  (1,12)  =  total_ridership  (1,12)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
         elseif (dist > 1299 && dist < 1400)
           total_ridership  (1,13)  =  total_ridership  (1,13)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
```

```
          elseif (dist > 1399 && dist < 1500)
           total_ridership  (1,14)  =  total_ridership  (1,14)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1499 && dist < 1600)
           total_ridership  (1,15)  =  total_ridership  (1,15)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1599 && dist < 1700)
           total_ridership  (1,16)  =  total_ridership  (1,16)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1699 && dist < 1800)
           total_ridership  (1,17)  =  total_ridership  (1,17)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1799 && dist < 1900)
           total_ridership  (1,18)  =  total_ridership  (1,18)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1899 && dist < 2000)
           total_ridership  (1,19)  =  total_ridership  (1,19)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1999)
           total_ridership  (1,20)  =  total_ridership  (1,20)+  business_trips
(i,Ridership_Station_Index)                  +                  business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);

           end
end

total = sum(total_ridership);  % total ridership for tsam
percent_ridership = total_ridership*100/total; % converts to percentage tsam
ridership
AMTRAK_percent_ridership   =   new_percent(AMTRAK_id,:);   %   extracts   the
corresponding AMTRAK data
groups = linspace (100,1000,10);

subplot(2,1,1)
bar (groups,AMTRAK_percent_ridership(1:10),0.4,'histc')
title          ('Ridership          Distribution          NYP-          AMTRAK
data(2009)','fontweight','b','color','b','fontsize',16)
xlabel ('Miles','fontweight','b','fontsize',12)
ylabel ('Ridership %','fontsize',12)
axis ([100 1000 0 100])
grid on
% set ( gca, 'xtick', [100 200 300 400 500 600 700 800 900 1000 1100 1200
1300 1400 1500 1600 1700 1800 1900 2000],'ytick' ,[0 10 20 30 40 50 60 70 80
90 100],'Fontsize',14)
```

```
set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 ],'ytick' ,[0 10
20 30 40 50 60 70 80 90 100],'Fontsize',12)

subplot(2,1,2)
bar ( groups, percent_ridership(1:10),0.4,'histc')
% title ('Model Estimate','fontweight','b','color','b','fontsize',24)
title      ('Ridership      Distribution      by      TSAM      6.6      for
2009','fontweight','b','color','b','fontsize',16)
xlabel ('Miles','fontweight','b','fontsize',12)
ylabel ('Ridership %','fontsize',12)
axis ([100 1000 0 100])
% set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 1100 1200 1300
1400 1500 1600 1700 1800 1900 2000],'ytick' ,[0 10 20 30 40 50 60 70 80 90
100],'Fontsize',14)
set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 ],'ytick' ,[0 10
20 30 40 50 60 70 80 90 100],'Fontsize',12)
grid on

clear s*
% clear t*
clear d*
clear b*
clear R*
clear r*
clear T*
clear n*
clear i
clear S*
clear g*
clear AMTRAK_id
```

D.7 Box Cox Function with productive time

```
function [Fval,CAshare,AutoTrips,CATrips,TrainTrips] =
objective_fcn_BoxCox_prod_v3(X,AUTO_reg,CA_reg,Train_reg,auto,CA,train,Trips,
weight,hastrain,purpose)

% objective_fcn_BoxCox__v_2 : USes a constant for train for Business, and two
ocnstant for Non Business.
% the constant is effective only if its in the NEC....
%

%Last Edited : Saloni : 1/30/2012
%   Calculates the Utility, Probability, and Objective Functionvalue
%   Inputs:
%       X - model Utility equation coefficients
%       AUTO_reg - regressors for the utility equation
%       CA_reg - regressors for the utility equation
%       Train_reg - regressors for the utility equation
%       auto - logic value indicating if auto was chosen for a specific
record
%       CA - logic value indicating if CA was the chosen model for each
record
%       train - logic value indicating if train was the chosen model for each
record
```

```
%       Trips - trip weight for each record
%       weight - use ATS weighting factors if = 1, do not if = 0
%       hastrain - flag indicating whether train is included (1=yes,0=no)
%   Outputs:
%       Fval - objective function value
%       CAshare - nationwide-level CA share
%       AutoTrips - auto trips for each record
%       CATrips - CA trips for each record
%       TrainTrips - Train Trips for each record
%
%   Nelson Guerreiro, ATK, 09/20/2010, revised 04/21/2011
% Saloni Chirania , 01/30/2012

% Mode Utility
% X: alp_time alp_cost lamTauto lamCauto lamTCA lamCCA lamTtrain lamCtrain
% note: with lambdas close to zero, use the log version of the BoxCox, per
the Box-Cox paper
% auto time utility

if abs(X(3))<1e-4
    U_auto_T = X(1) * log(AUTO_reg(:,1));
else
    U_auto_T = X(1) * (AUTO_reg(:,1).^X(3)-1)/X(3);
end

% auto cost utility
if abs(X(4))<1e-4
    U_auto_C = X(2) * log(AUTO_reg(:,2));
else
    U_auto_C = X(2) * (AUTO_reg(:,2).^X(4)-1)/X(4);
end
% CA time utility

U_auto = U_auto_T + U_auto_C;

if strcmp(purpose,'Business'==1)
    if abs(X(6))<1e-4
        U_CA_T   = X(1) * log(CA_reg(:,1));  % business purpose has one extra
term for producitvity in CA mode.
    else
        U_CA_T   = X(1) * (CA_reg(:,1).^X(6)-1)/X(6);
    end
    % CA cost utility
    if abs(X(7))<1e-4
        U_CA_C = X(2) * log(CA_reg(:,2));
    else
        U_CA_C = X(2) * (CA_reg(:,2).^X(7)-1)/X(7);
    end

    U_CA = CA_reg(:,3).*X(5)+ U_CA_T + U_CA_C;  % the productive term
coeffiecint is considered to be constant

    if hastrain==1
        % train time utility
        if abs(X(10))<1e-4
```

```
            U_train_T = X(1) * log(Train_reg(:,1));
        else
            U_train_T = X(1) * (Train_reg(:,1).^X(10)-1)/X(10);
        end
        % train cost utility
        if abs(X(11))<1e-4
            U_train_C = X(2) * log(Train_reg(:,2));
        else
            U_train_C = X(2) * (Train_reg(:,2).^X(11)-1)/X(11);
        end

        U_train = Train_reg(:,3)*X(9)+Train_reg(:,6)*X(8)  + U_train_T +
U_train_C;  % one constant for NEC , logical multiplier ensures that the
constant term is added only for those areas that belong to NEC, producitivity
is used for business class only
    end
else % Non-Business
    if abs(X(5))<1e-4
        U_CA_T   = X(1) * log(CA_reg(:,1));  % business purpose has one extra
term for producitvity in CA mode.
    else
        U_CA_T   = X(1) * (CA_reg(:,1).^X(5)-1)/X(5);
    end
    % CA cost utility
    if abs(X(6))<1e-4
        U_CA_C = X(2) * log(CA_reg(:,2));
    else
        U_CA_C = X(2) * (CA_reg(:,2).^X(6)-1)/X(6);
    end
    U_CA =  U_CA_T + U_CA_C;  % the productive term coeffiecint is considered
to be constant

    if hastrain==1
        % train time utility
        if abs(X(9))<1e-4
            U_train_T = X(1) * log(Train_reg(:,1));
        else
            U_train_T = X(1) * (Train_reg(:,1).^X(9)-1)/X(9);
        end
        % train cost utility
        if abs(X(10))<1e-4
            U_train_C = X(2) * log(Train_reg(:,2));
        else
            U_train_C = X(2) * (Train_reg(:,2).^X(10)-1)/X(10);
        end

        U_train = Train_reg(:,4).*X(7)+Train_reg(:,5).*X(8) + U_train_T +
U_train_C; % one constant for NYP, and one for DC
    end
end


%% correct for the records that cause NaNs
% exponential cannot handle values less than -700 and greater than 700 due to
precision (shift up)
if hastrain==1
```

```
    Uall = [U_auto U_CA U_train];
else
    Uall = [U_auto U_CA];
end

minU = min(Uall,[],2);
ind2shift = find(minU<-700);
U_auto(ind2shift) = U_auto(ind2shift) - minU(ind2shift) - 700;
U_CA(ind2shift) = U_CA(ind2shift) - minU(ind2shift) - 700;
if hastrain==1
    U_train(ind2shift) = U_train(ind2shift) - minU(ind2shift) - 700;
end


%% Mode Probability
expauto = exp(U_auto);
expCA   = exp(U_CA);
if hastrain==1
    exptrain = exp(U_train);
    P_auto = expauto ./ (expauto + expCA + exptrain);
    P_CA   = expCA ./ (expauto + expCA + exptrain);
    P_train = exptrain ./ (expauto + expCA +exptrain);
else
    P_auto = expauto ./ (expauto + expCA);
    P_CA   = expCA ./ (expauto + expCA);
end

%% correct for the records that cause NaNs
% remove any zero probabilities and replace with very low values to avoid Inf
log
ind = find(P_auto<1e-100);
P_auto(ind) = 1e-100;
ind = find(P_CA<1e-100);
P_CA(ind) = 1e-100;
if hastrain==1
    ind = find(P_train<1e-100);
    P_train(ind) = 1e-100;
end


%% calculate the resulting CA share
AutoTrips = round(Trips.*P_auto);
CATrips = round(Trips.*P_CA);
if hastrain==1
    TrainTrips = round(Trips.*P_train);
else
    TrainTrips = 0*AutoTrips;
end
calc_autotrips = sum(AutoTrips);
calc_CAtrips = sum(CATrips);
calc_traintrips = sum(TrainTrips);
TotalTrips = calc_autotrips + calc_CAtrips + calc_traintrips;
CAshare = calc_CAtrips/TotalTrips;
```

```
%% Log Likelihood (is calculated with natural logarithm)
if hastrain==1
    LL = auto.*log(P_auto).*Trips.^weight + CA.*log(P_CA).*Trips.^weight +
train.*log(P_train).*Trips.^weight;
else
    LL = auto.*log(P_auto).*Trips.^weight + CA.*log(P_CA).*Trips.^weight;
end
Fval = -sum(LL);

% make NaNs and Infs very large values
if isnan(Fval) || isinf(Fval)
    Fval = sum(Trips);  % default to worst LL (not a valid coefficient
selection)
end
```

D.8 input Plots For Calibration File

```
clear all
close all
clc
purpose = 'Business'; % Non-Business
Year = '1995';
% file = [pwd,'\Input1\NEC-HH-ToFr\',purpose,' - All - With Travel Time and
Cost - NY-BOS.txt'];  % input file
% file = ['C:\Users\saloni\Documents\MC_Calib_v_1_2- edited-saloni-
09_09_2011\Input\',purpose,' - All - With Travel Time and Cost -
03_21_2012_MedianFares_AllTTTCSDDS.txt'];  % input file
file = ['E:\tsam 6.6 input files\MC_Calib_v_2\input\',purpose,' - All - With
Travel Time and Cost - 09_11_2012-10.txt'];  % input file within NEC
% file = ['E:\ATS Analysis 1995\',purpose,' - All - With Travel Time and Cost
- 09_04_2012-3.txt'];%['C:\Users\saloni\Documents\MC_Calib_v_1_2- edited-
saloni-09_09_2011\Input\',purpose,' - All - With Travel Time and Cost -
09_03_2012.txt'];
% file = ['E:\',purpose,' - All - With Travel Time and Cost -
01_11_2012\',purpose,' - All - With Travel Time and Cost - 01_11_2012.txt'];
% file = ['E:\MC_Calib_v_1_2\Input\',purpose,' - All - With Travel Time and
Cost - 03_09_2011.txt'];
% file = ['C:\Users\saloni\Documents\MC_Calib_v_1_2- edited-saloni-
09_09_2011\Input1\',purpose,' - All - With Travel Time and Cost -
10_12_2011.txt'];
TSAM_Dir = 'C:\Program Files (x86)\TSAM\6.6\data';
CEEDS_MSA_County_Matching_File =
fopen([TSAM_Dir,'\trip_generation\input\CEEDS 2001 - County To MSA - Matching
List.txt'], 'r');
CEEDS_MSA_County_Matching_File_Field_Names =
fgets(CEEDS_MSA_County_Matching_File);
CEEDS_MSA_County_Matching_Data = textscan(CEEDS_MSA_County_Matching_File,
'%s%s%s%s%s%s%f%f%f%f%s%s', 'delimiter', '\t');
MSA   = CEEDS_MSA_County_Matching_Data{7};
STATE = CEEDS_MSA_County_Matching_Data{12};
 OnlyAutoCA = '';
VOTDate         = '09_04_2012-10';
TT_TC_Dir      = ['E:\Travel Time and Cost Calculations - 09_01_2011-
Modified_Saloni\VOT_',
VOTDate,'\',purpose];%['C:\Users\saloni\Documents\Travel Time and Cost
Calculations - 09_01_2011-Modified_Saloni\VOT_', VOTDate];
```

```
TD_Project_Dir = 'E:\tsam 6.6\TSAM Project';%'C:\Users\saloni\Documents\TSAM
Project - TSAM 6_5';

Income_Group= 1;
load([TD_Project_Dir, '\Trip_Distribution\Output\Y', Year,
'\businessTripTable_county_Inc', num2str(Income_Group), '_', OnlyAutoCA,
Year, '.mat']);

        eval(['Demand_Inc', num2str(Income_Group), ' =
businessTripTable_county_Inc', num2str(Income_Group), ';']);
        eval(['clear businessTripTable_county_Inc', num2str(Income_Group),
';']);

load([TT_TC_Dir, '\Average_Travel_Cost_CA_Weighted.mat']);
load([TT_TC_Dir, '\Average_Travel_Time_CA_Weighted.mat']);
load([TT_TC_Dir, '\Average_Travel_Cost_Train_Weighted.mat']);
load([TT_TC_Dir, '\Average_Travel_Time_Train_Weighted.mat']);

load([TT_TC_Dir, '\Average_Ost_wait_Train_Weighted.mat']);
load([TT_TC_Dir, '\Average_ProcessingTime_origin_CA_Weighted.mat']);
load([TT_TC_Dir, '\Average_Time_onTrain_Train_Weighted.mat']);
load([TT_TC_Dir, '\Average_StageLength_CA_2way_CA_Weighted.mat']);
load([TT_TC_Dir, '\Average_TravelTime_noSD_CA_Weighted.mat']);
load([TT_TC_Dir, '\Average_TravelTime_noSD_Train_Weighted.mat']);

load([TT_TC_Dir, '\c2cCA_TravelTime_2way_All.mat']);
load([TT_TC_Dir, '\c2cCA_TravelCost_2way_All.mat']);
load([TT_TC_Dir, '\c2cCA_AccessEgressCost_CA_2way_All.mat']);
load([TT_TC_Dir, '\c2cCA_LodgingCost_CA_2way_All.mat']);

load([TT_TC_Dir, '\c2cCA_StageLength_CA_2way_All.mat']);
load([TT_TC_Dir, '\c2cCA_TravelTime_noSD_CA_2way_All.mat']);
load([TT_TC_Dir, '\c2cCA_ProcessingTime_origin_2way_All.mat']);


%Train
load([TT_TC_Dir, '\c2cTrain_TravelTime_2way_All.mat']);
load([TT_TC_Dir, '\c2cTrain_TravelCost_2way_All.mat']);

load([TT_TC_Dir, '\c2cTrain_Ost_wait_2way_All.mat']);
load([TT_TC_Dir, '\c2cTrain_Time_onTrain_2way_All.mat']);
load([TT_TC_Dir, '\c2cTrain_TravelTime_noSD_2way_ALL.mat']);

load([TSAM_Dir,
'\mode_choice\input\DrivingDistanceTime\C2CDriveDist_Population_Centroids.mat
']);
C2CDriveDist_Population_Centroids_2Way = C2CDriveDist_Population_Centroids +
C2CDriveDist_Population_Centroids';
load([TSAM_Dir,'\mode_choice\input\DrivingDistanceTime\C2CDriveTime_Populatio
n_Centroids_Congestion_',Year,'.mat']);
C2CDriveTime_Population_Centroids_Congestion_2Way =
C2CDriveTime_Population_Centroids_Congestion +
C2CDriveTime_Population_Centroids_Congestion';
```

```
% %% Read the input file for calibration and extract the required data
[Fields,numfields,Data,numrecords,fmat] =
read_single_header_csv_file(file,[],0);
%
% Comment lines 22 - 28 if you do not want analyssi by income groups.
% incgroup = 5;
% hhinccol  = strmatch('TSAM_HHINC',Fields,'exact');  % immediately filter by
income group
% ind = find(Data{hhinccol}==incgroup);
% numrecords = length(ind);
% for i = 1:length(Data)
%     Data{i} = Data{i}(ind);
% end

ORIGIN_CEEDS_CODE=Data{strmatch ('ORIGIN_CEEDS_CODE',Fields,'exact')};
ORIGIN_CEEDS_CODE=str2double(ORIGIN_CEEDS_CODE);
% origin = Data {origin_temp};
% length_origin = length (orig_temp);
% origin = zeros (length_origin , 1);
% for i = 1:length (orig_temp)
%     origin(i,1) = str2double (orig_temp{i,1});
% end
des_temp=strmatch ('DESTINATION_CEEDS_CODE',Fields,'exact');
des_temp = Data {des_temp};
% identify some of the data columns
autotimecol = strmatch('AUTO_TT',Fields,'exact');
autocostcol = strmatch('AUTO_TC',Fields,'exact');
CAtimecol = strmatch('COMMAIR_TT',Fields,'exact');
CAcostcol = strmatch('COMMAIR_TC',Fields,'exact');
CA_prodtimecol = strmatch('COMMAIR_PT',Fields,'exact');
CA_unprodtimecol = strmatch('COMMAIR_UPT',Fields,'exact');
transod   = strmatch('TSAM_TRANSOD',Fields,'exact');
RTEDUSRT   = [strmatch('TSAM_ROUTEDIST',Fields,'exact')
strmatch('TSAM_RTEDUSRT',Fields,'exact')];
OSTPOST =Data{strmatch('OSTPOST',Fields,'exact')};
DSTPOST =Data{strmatch('DSTPOST',Fields,'exact')};
traintimecol =  strmatch('TRAIN_TT',Fields,'exact');
traincostcol  =  strmatch('TRAIN_TC',Fields,'exact');
train_prodtimecol =  strmatch('TRAIN_PT',Fields,'exact');
train_unprodtimecol  =  strmatch('TRAIN_UPT',Fields,'exact');
PERTRWGT= Data{strmatch('HHTRPWGT',Fields,'exact')};
% traintimecol =  strmatch('Train_TT',Fields,'exact');
% traincostcol  =  strmatch('Train_TC',Fields,'exact');
trpwgt    = [strmatch('HHTRPWGT',Fields,'exact')
strmatch('PERTRWGT',Fields,'exact')];
%
% % to convert the origin cell array into matrix
% for i = 1:length (orig_temp)
%     origin(i,1) = str2double (orig_temp{i,1});
% end
DESTINATION_CEEDS_CODE = Data{strmatch
('DESTINATION_CEEDS_CODE',Fields,'exact')};

auto_total_time = Data{autotimecol};
CA_total_time = Data{CAtimecol};
```

```
train_total_time = Data{traintimecol};
Trips = Data{trpwgt};
RTEDUSRT = Data{RTEDUSRT};

autocost = Data{autocostcol};
CAcost = Data{CAcostcol};
traincost = Data{traincostcol};
numrows = length(autocost);
Origin_Counties_Index = zeros(numrows ,1);
Origin_MSA_Dummy = zeros(numrows ,1);
Destination_Counties_Index = zeros(numrows ,1);
Destination_MSA_Dummy = zeros(numrows ,1);

for i = 1:numrows
    if isnan(ORIGIN_CEEDS_CODE(i)) == 0 % MSA
        Origin_Counties_Index = find(MSA == ORIGIN_CEEDS_CODE(i));
        Origin_MSA_Dummy(i) = 1;
    else % Not MSA (Take all counties in State that are not MSA)
        % if strcmp(OSTPOST(Current_HHINC_Index), 'NJ') == 1 % Some States
are all MSA counties
        %     Origin_Counties_Index = find(strcmp(STATE,
OSTPOST(Current_HHINC_Index)) == 1);
        % else
        Origin_Counties_Index = find(MSA == 0 & strcmp(STATE, OSTPOST(i)) ==
1);
        % end
        Origin_MSA_Dummy(i) = 0;
    end

    % Get destination counties for destination region
    if isnan(DESTINATION_CEEDS_CODE(i)) == 0 % MSA
        Destination_Counties_Index = find(MSA == DESTINATION_CEEDS_CODE(i));
        Destination_MSA_Dummy(i) = 1;
    else % Not MSA (Take all counties in State that are not MSA)
        % if strcmp(DSTPOST(Current_HHINC_Index), 'NJ') == 1 % Some States
are all MSA counties
        %     Destination_Counties_Index = find(strcmp(STATE,
DSTPOST(Current_HHINC_Index)) == 1);
        % else
        Destination_Counties_Index = find(MSA == 0 & strcmp(STATE,
DSTPOST(i)) == 1);
        % end
        Destination_MSA_Dummy(i) = 0;
    end

    Min_Distance = max(RTEDUSRT(i) * 0.8, 200);
        Max_Distance = RTEDUSRT(i) * 1.2;
        [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination
_Counties_Index) >= Min_Distance &
C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination_Coun
ties_Index) <= Max_Distance);
        if isempty(Selected_OD_Pairs_Index_i) == 1 % If no counties selected
use less stringent disance limits
```

```
            [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination
_Counties_Index) >= 200);
        end

        if isempty(Selected_OD_Pairs_Index_i) == 1 % Skip records if no
county OD pair found
            Skipped_Records(1,1)  = Skipped_Records(1,1) + 1;
            Skipped_PERTRWGT(1,1) = Skipped_PERTRWGT(1,1) + PERTRWGT(i);
            continue;
        end

        Dist_Diff =
abs(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination_
Counties_Index) - RTEDUSRT(i));
        Min_Diff = min(min(Dist_Diff));
        [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(Dist_Diff == Min_Diff);
        if isempty(Selected_OD_Pairs_Index_i) == 1 % If no counties selected
=> ERROR
            disp('ERROR!: No county found using distances!');
            return;
        end

        % Remove county OD pairs < 100 miles route distance, Remove OD Pairs
Not In Range Of ATS Data
        Min_Distance = max(RTEDUSRT(i) * 0.8, 200);
        Max_Distance = RTEDUSRT(i) * 1.2;
        [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination
_Counties_Index) >= Min_Distance &
C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination_Coun
ties_Index) <= Max_Distance);
        if isempty(Selected_OD_Pairs_Index_i) == 1 % If no counties selected
use less stringent disance limits
            [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination
_Counties_Index) >= 200);
        end

        if isempty(Selected_OD_Pairs_Index_i) == 1 % Skip records if no
county OD pair found
            Skipped_Records(1,1)  = Skipped_Records(1,1) + 1;
            Skipped_PERTRWGT(1,1) = Skipped_PERTRWGT(1,1) + PERTRWGT(i);
            continue;
        end

        % The new Distance Filtering introduced assigns already a valid
        % TSAM driving distance for each record. We just need to find the
        % closest County2County to the TSAM Distance
        Dist_Diff =
abs(C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination_
Counties_Index) - RTEDUSRT(i));
        Min_Diff = min(min(Dist_Diff));
        [Selected_OD_Pairs_Index_i, Selected_OD_Pairs_Index_j] =
find(Dist_Diff == Min_Diff);
```

```
        if isempty(Selected_OD_Pairs_Index_i) == 1 % If no counties selected
=> ERROR
            disp('ERROR!: No county found using distances!');
            return;
        end


%         % Get trip distribution demand to weight travel time and cost
        eval(['Demand_Current = Demand_Inc', num2str(Income_Group),
'(Origin_Counties_Index,Destination_Counties_Index);']);
%
%         % Retrieve valid OD Pairs
        Number_Of_OD_Pairs = length(Selected_OD_Pairs_Index_i);
%
%         c2cAuto_TravelTime_2way_All_Temp       =
c2cAuto_TravelTime_2way_All(Origin_Counties_Index,Destination_Counties_Index)
;
%         c2cAuto_TravelCost_2way_All_Temp       =
c2cAuto_TravelCost_2way_All(Origin_Counties_Index,Destination_Counties_Index,
Income_Group);
%         c2cAuto_LodgingCost_2way_All_Temp      =
c2cAuto_LodgingCost_2way_All(Origin_Counties_Index,Destination_Counties_Index
,Income_Group);
%
        c2cCA_TravelTime_2way_All_Temp       =
c2cCA_TravelTime_2way_All(Origin_Counties_Index,Destination_Counties_Index);
        c2cCA_TravelCost_2way_All_Temp       =
c2cCA_TravelCost_2way_All(Origin_Counties_Index,Destination_Counties_Index,In
come_Group);
        c2cCA_AccessEgressCost_CA_2way_Temp  =
c2cCA_AccessEgressCost_CA_2way_All(Origin_Counties_Index,Destination_Counties
_Index,Income_Group);
        c2cCA_LodgingCost_CA_2way_Temp       =
c2cCA_LodgingCost_CA_2way_All(Origin_Counties_Index,Destination_Counties_Inde
x,Income_Group);

        c2cCA_StageLength_CA_2way_All_Temp       =
c2cCA_StageLength_CA_2way_ALL(Origin_Counties_Index,Destination_Counties_Inde
x);
        c2cCA_TravelTime_noSD_CA_2way_All_Temp  =
c2cCA_TravelTime_noSD_CA_2way_ALL(Origin_Counties_Index,Destination_Counties_
Index);
        c2cCA_ProcessingTime_origin_CA_2way_All_Temp       =
c2cCA_ProcessingTime_origin_2way_All(Origin_Counties_Index,Destination_Counti
es_Index);

        c2cTrain_TravelTime_2way_All_Temp       =
c2cTrain_TravelTime_2way_All(Origin_Counties_Index,Destination_Counties_Index
);
        c2cTrain_TravelCost_2way_All_Temp       =
c2cTrain_TravelCost_2way_All(Origin_Counties_Index,Destination_Counties_Index
,Income_Group);

        c2cTrain_Ost_wait_2way_All_Temp       =
c2cTrain_Ost_wait_2way_All(Origin_Counties_Index,Destination_Counties_Index);
```

```
        c2cTrain_Time_onTrain_2way_All_Temp  =
c2cTrain_Time_onTrain_2way_AlL(Origin_Counties_Index,Destination_Counties_Ind
ex);
        c2cTrain_TravelTime_no_SD_2way_ALL_Temp       =
c2cTrain_TravelTime_noSD_2way_ALL(Origin_Counties_Index,Destination_Counties_
Index);

        C2CDriveDist_Population_Centroids_Temp =
C2CDriveDist_Population_Centroids_2Way(Origin_Counties_Index,Destination_Coun
ties_Index);
        C2CDriveTime_Population_Centroids_Temp =
C2CDriveTime_Population_Centroids_Congestion_2Way(Origin_Counties_Index,Desti
nation_Counties_Index);
%
%         c2cAuto_TravelTime_2way_All_Temp2       =
zeros(Number_Of_OD_Pairs,1);
%         c2cAuto_TravelCost_2way_All_Temp2       =
zeros(Number_Of_OD_Pairs,1);
%         c2cAuto_LodgingCost_2way_All_Temp2      =
zeros(Number_Of_OD_Pairs,1);
%
        c2cCA_TravelTime_2way_All_Temp2       = zeros(Number_Of_OD_Pairs,1);
        c2cCA_TravelCost_2way_All_Temp2       = zeros(Number_Of_OD_Pairs,1);
        c2cCA_AccessEgressCost_CA_2way_Temp2  = zeros(Number_Of_OD_Pairs,1);
        c2cCA_LodgingCost_CA_2way_Temp2       = zeros(Number_Of_OD_Pairs,1);

        c2cCA_ProcessingTime_origin_CA_2way_Temp2 =
zeros(Number_Of_OD_Pairs,1);
        c2cCA_StageLength_CA_2way_Temp2 = zeros(Number_Of_OD_Pairs,1);
        c2cCA_TravelTime_noSD_Temp2 = zeros(Number_Of_OD_Pairs,1);

        c2cTrain_TravelTime_2way_All_Temp2       =
zeros(Number_Of_OD_Pairs,1);
        c2cTrain_TravelCost_2way_All_Temp2       =
zeros(Number_Of_OD_Pairs,1);

        c2cTrain_Ost_wait_2way_All_Temp2 = zeros(Number_Of_OD_Pairs,1);
        c2cTrain_Time_onTrain_2way_All_Temp2 = zeros(Number_Of_OD_Pairs,1);
        c2cTrain_TravelTime_no_SD_2way_ALL_Temp2 =
zeros(Number_Of_OD_Pairs,1);
%
%
        C2CDriveDist_Population_Centroids_Temp2 =
zeros(Number_Of_OD_Pairs,1);
        C2CDriveTime_Population_Centroids_Temp2 =
zeros(Number_Of_OD_Pairs,1);
        Demand_Current_Temp2                    =
zeros(Number_Of_OD_Pairs,1);
%
        for j = 1:Number_Of_OD_Pairs
%             c2cAuto_TravelTime_2way_All_Temp2(j,1)       =
c2cAuto_TravelTime_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
%             c2cAuto_TravelCost_2way_All_Temp2(j,1)       =
c2cAuto_TravelCost_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
```

```
%             c2cAuto_LodgingCost_2way_All_Temp2(j,1)      =
c2cAuto_LodgingCost_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));

            c2cCA_TravelTime_2way_All_Temp2(j,1)       =
c2cCA_TravelTime_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cCA_TravelCost_2way_All_Temp2(j,1)       =
c2cCA_TravelCost_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cCA_AccessEgressCost_CA_2way_Temp2(j,1)  =
c2cCA_AccessEgressCost_CA_2way_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cCA_LodgingCost_CA_2way_Temp2(j,1)       =
c2cCA_LodgingCost_CA_2way_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));

            c2cCA_ProcessingTime_origin_CA_2way_Temp2(j,1) =
c2cCA_ProcessingTime_origin_CA_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cCA_StageLength_CA_2way_Temp2(j,1) =
c2cCA_StageLength_CA_2way_All_Temp (Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cCA_TravelTime_noSD_Temp2(j,1)
=c2cCA_TravelTime_noSD_CA_2way_All_Temp (Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));

            c2cTrain_TravelTime_2way_All_Temp2(j,1)       =
c2cTrain_TravelTime_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cTrain_TravelCost_2way_All_Temp2(j,1)       =
c2cTrain_TravelCost_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));

            c2cTrain_Ost_wait_2way_All_Temp2(j,1) =
c2cTrain_Ost_wait_2way_All_Temp (Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cTrain_Time_onTrain_2way_All_Temp2(j,1) =
c2cTrain_Time_onTrain_2way_All_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            c2cTrain_TravelTime_no_SD_2way_ALL_Temp2(j,1)
=c2cTrain_TravelTime_no_SD_2way_ALL_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));

            C2CDriveDist_Population_Centroids_Temp2(j,1) =
C2CDriveDist_Population_Centroids_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            C2CDriveTime_Population_Centroids_Temp2(j,1) =
C2CDriveTime_Population_Centroids_Temp(Selected_OD_Pairs_Index_i(j),
Selected_OD_Pairs_Index_j(j));
            Demand_Current_Temp2(j,1)                     =
Demand_Current(Selected_OD_Pairs_Index_i(j), Selected_OD_Pairs_Index_j(j));
        end
%
%         % Remove OD pairs with no CA Travel Time Or Cost
```

```
        No_CA_Index = find(c2cCA_TravelTime_2way_All_Temp2 <= 0 |
c2cCA_TravelCost_2way_All_Temp2 <= 0);

        if isempty(No_CA_Index) == 0
            c2cCA_TravelTime_2way_All_Temp2(No_CA_Index)      = [];
            c2cCA_TravelCost_2way_All_Temp2(No_CA_Index)      = [];
            c2cCA_AccessEgressCost_CA_2way_Temp2(No_CA_Index) = [];
            c2cCA_LodgingCost_CA_2way_Temp2(No_CA_Index)      = [];
            c2cCA_ProcessingTime_origin_CA_2way_Temp2(No_CA_Index)= [];
            c2cCA_StageLength_CA_2way_Temp2(No_CA_Index)      = [];
            c2cCA_TravelTime_noSD_Temp2(No_CA_Index)      = [];

        end
%
%
% %        % Remove OD pairs with no Train Travel Time or Cost
% %         No_Train_Index = find(c2cTrain_TravelTime_2way_All_Temp2 <= 0 |
c2cTrain_TravelCost_2way_All_Temp2 <= 0);
% %
% %         if isempty(No_Train_Index) == 0
% %             c2cTrain_TravelTime_2way_All_Temp2(No_Train_Index)      = [];
% %             c2cTrain_TravelCost_2way_All_Temp2(No_Train_Index)      = [];
% %             c2cTrain_Ost_wait_2way_All_Temp2(No_Train_Index)      = [];
% %             c2cTrain_Time_onTrain_2way_All_Temp2(No_Train_Index)      =
[];
% %             c2cTrain_TravelTime_no_SD_2way_ALL_Temp2(No_Train_Index)
= [];
% %         end
% %
%
%         Demand_Current_Temp2_Auto = Demand_Current_Temp2;
        Demand_Current_Temp2_CA   = Demand_Current_Temp2;
        Demand_Current_Temp2_Train = Demand_Current_Temp2;
%         if isempty(No_CA_Index) == 0
%             Demand_Current_Temp2_CA(No_CA_Index) = [];
%         end
%
        Demand_Total      = sum(sum(Demand_Current_Temp2));
%         Demand_Total_Auto = sum(sum(Demand_Current_Temp2_Auto));
        Demand_Total_CA   = sum(sum(Demand_Current_Temp2_CA));
        Demand_Total_Train = sum(sum(Demand_Current_Temp2_Train));
%
%         % Calculte average Travel Time
        if Demand_Total > 0
%             Average_Travel_Time_Auto      =
sum(sum(c2cAuto_TravelTime_2way_All_Temp2 .* Demand_Current_Temp2_Auto)) /
Demand_Total_Auto;
%             Average_Travel_Cost_Auto      =
sum(sum(c2cAuto_TravelCost_2way_All_Temp2 .* Demand_Current_Temp2_Auto)) /
Demand_Total_Auto;
%             Average_Lodging_Cost_Auto     =
sum(sum(c2cAuto_LodgingCost_2way_All_Temp2 .* Demand_Current_Temp2_Auto)) /
Demand_Total_Auto;
```

```
            Average_Travel_Time_CA        =
sum(sum(c2cCA_TravelTime_2way_All_Temp2 .* Demand_Current_Temp2_CA)) /
Demand_Total_CA;
            Average_Travel_Cost_CA        =
sum(sum(c2cCA_TravelCost_2way_All_Temp2 .* Demand_Current_Temp2_CA)) /
Demand_Total_CA;
            Average_Access_Egress_Cost_CA =
sum(sum(c2cCA_AccessEgressCost_CA_2way_Temp2 .* Demand_Current_Temp2_CA)) /
Demand_Total_CA;
            Average_Lodging_Cost_CA       =
sum(sum(c2cCA_LodgingCost_CA_2way_Temp2 .* Demand_Current_Temp2_CA)) /
Demand_Total_CA;

            Average_ProcessingTime_origin_CA
=sum(sum(c2cCA_ProcessingTime_origin_CA_2way_Temp2 .*
Demand_Current_Temp2_CA)) / Demand_Total_CA;
            Average_TravelTime_noSD_CA  =sum(sum( c2cCA_TravelTime_noSD_Temp2
.* Demand_Current_Temp2_CA)) / Demand_Total_CA;
            Average_StageLength_CA_2way = sum(sum(
c2cCA_StageLength_CA_2way_Temp2 .* Demand_Current_Temp2_CA)) /
Demand_Total_CA;

            Average_Travel_Time_Train      =
sum(sum(c2cTrain_TravelTime_2way_All_Temp2 .* Demand_Current_Temp2_Train)) /
Demand_Total_Train;
            Average_Travel_Cost_Train      =
sum(sum(c2cTrain_TravelCost_2way_All_Temp2 .* Demand_Current_Temp2_Train)) /
Demand_Total_Train;

            Average_Ost_wait_Train=  sum(sum(c2cTrain_Ost_wait_2way_All_Temp2
.* Demand_Current_Temp2_Train)) / Demand_Total_Train;
            Average_TravelTime_noSD_Train =
sum(sum(c2cTrain_TravelTime_no_SD_2way_ALL_Temp2 .*
Demand_Current_Temp2_Train)) / Demand_Total_Train;
            Average_Time_onTrain_Train =
sum(sum(c2cTrain_Time_onTrain_2way_All_Temp2 .* Demand_Current_Temp2_Train))
/ Demand_Total_Train;

            Average_Route_Distance    =
sum(sum(C2CDriveDist_Population_Centroids_Temp2 .* Demand_Current_Temp2)) /
Demand_Total;
            Average_Route_DrivingTime =
sum(sum(C2CDriveTime_Population_Centroids_Temp2 .* Demand_Current_Temp2)) /
Demand_Total;
        else
%             Average_Travel_Time_Auto      =
mean(c2cAuto_TravelTime_2way_All_Temp2);
%             Average_Travel_Cost_Auto      =
mean(c2cAuto_TravelCost_2way_All_Temp2);
%             Average_Lodging_Cost_Auto     =
mean(c2cAuto_LodgingCost_2way_All_Temp2);

            Average_Travel_Time_CA        =
mean(c2cCA_TravelTime_2way_All_Temp2);
            Average_Travel_Cost_CA        =
mean(c2cCA_TravelCost_2way_All_Temp2);
```

```
            Average_Access_Egress_Cost_CA =
mean(c2cCA_AccessEgressCost_CA_2way_Temp2);
            Average_Lodging_Cost_CA       =
mean(c2cCA_LodgingCost_CA_2way_Temp2);

            Average_ProcessingTime_origin_CA  =
mean(c2cCA_ProcessingTime_origin_CA_2way_Temp2);
            Average_TravelTime_noSD_CA  = mean(c2cCA_TravelTime_noSD_Temp2);
            Average_StageLength_CA_2way =
mean(c2cCA_StageLength_CA_2way_Temp2);

            Average_Travel_Time_Train      =
mean(c2cTrain_TravelTime_2way_All_Temp2);
            Average_Travel_Cost_Train      =
mean(c2cTrain_TravelCost_2way_All_Temp2);

            Average_Ost_wait_Train= mean(c2cTrain_Ost_wait_2way_All_Temp2);
            Average_TravelTime_noSD_Train =
mean(c2cTrain_TravelTime_no_SD_2way_ALL_Temp2);
            Average_Time_onTrain_Train =
mean(c2cTrain_Time_onTrain_2way_All_Temp2);
%
%
            Average_Route_Distance    =
mean(C2CDriveDist_Population_Centroids_Temp2);
%             Average_Route_DrivingTime =
mean(C2CDriveTime_Population_Centroids_Temp2);
        end
%
% % %         Average_Travel_Time_Train      =
mean(mean(c2cTrain_TravelTime_2way_All(Origin_Counties_Index,Destination_Coun
ties_Index)));
% % %         Average_Travel_Cost_Train      =
mean(mean(c2cTrain_TravelCost_2way_All(Origin_Counties_Index,Destination_Coun
ties_Index,Income_Group)));
%
%
% %         if Origin_MSA_Dummy == 1 && Destination_MSA_Dummy == 1
% %             1
% %         end
%
%         % Check if less than 100 miles 1 way
        if Average_Route_Distance < 200 || isnan(Average_Route_Distance) == 1
% Skip record
            % Skipped_Records(2,1)  = Skipped_Records(2,1) + 1;
            % Skipped_PERTRWGT(2,1) = Skipped_PERTRWGT(2,1) + PERTRWGT(i);
%             % continue;
            Average_Route_Distance = 200;
            Average_Route_Distance_1Way = 100;
            Bin_Index = find(Average_Route_Distance_1Way >= Bin_Start &
Average_Route_Distance_1Way < Bin_End);
%
%             Average_Travel_Time_Auto      =
Average_Travel_Time_Auto_VOT(Bin_Index);
%             Average_Travel_Cost_Auto      =
mean(Average_Travel_Cost_Auto_VOT(Bin_Index,:));
```

```
%             Average_Lodging_Cost_Auto     = -999;
%
            Average_Travel_Time_CA        =
Average_Travel_Time_CA_VOT(Bin_Index);
            Average_Travel_Cost_CA        =
mean(Average_Travel_Cost_CA_VOT(Bin_Index,:));
            Average_Access_Egress_Cost_CA = -999;
            Average_Lodging_Cost_CA       = -999;

            Average_ProcessingTime_origin_CA =
Average_ProcessingTime_origin_CA_VOT(Bin_Index);
            Average_TravelTime_noSD_CA =
Average_TravelTime_noSD_CA_VOT(Bin_Index);
            Average_StageLength_CA_2way =
Average_StageLength_CA_2way_VOT(Bin_Index);

            Average_Travel_Time_Train =
Average_Travel_Time_Train_VOT(Bin_Index);
            Average_Travel_Cost_Train =
mean(Average_Travel_Cost_Train_VOT(Bin_Index));

            Average_Ost_wait_Train =  Average_Ost_wait_Train_VOT(Bin_Index);
            Average_TravelTime_noSD_Train =
Average_TravelTime_noSD_Train_VOT(Bin_Index);
            Average_Time_onTrain_Train =
Average_Time_onTrain_Train_VOT(Bin_Index);
        end
%
%         % Skip if ATS AUTO and No CA between counties
        if isnan(Average_Travel_Time_CA) == 1 && TRANSOD(Current_HHINC_Index)
~= 2 % Auto
%             % Skipped_Records(5,1)  = Skipped_Records(5,1) + 1;
%             % Skipped_PERTRWGT(5,1) = Skipped_PERTRWGT(5,1) +
PERTRWGT(Current_HHINC_Index);
%             % continue;
            Average_Route_Distance_1Way = Average_Route_Distance / 2;
            Bin_Index = find(Average_Route_Distance_1Way >= Bin_Start &
Average_Route_Distance_1Way < Bin_End);
            Average_Travel_Time_CA        =
Average_Travel_Time_CA_VOT(Bin_Index);
            Average_Travel_Cost_CA        =
mean(Average_Travel_Cost_CA_VOT(Bin_Index,:));
            Average_Access_Egress_Cost_CA = -999;
            Average_Lodging_Cost_CA       = -999;
%
            Average_ProcessingTime_origin_CA =
Average_ProcessingTime_origin_CA_VOT(Bin_Index);
            Average_TravelTime_noSD_CA =
Average_TravelTime_noSD_CA_VOT(Bin_Index);
            Average_StageLength_CA_2way =
Average_StageLength_CA_2way_VOT(Bin_Index);
%
%
        elseif isnan(Average_Travel_Time_CA) == 1 &&
TRANSOD(Current_HHINC_Index) == 2 % CA
            % Skipped_Records(6,1)  = Skipped_Records(6,1) + 1;
```

```
            % Skipped_PERTRWGT(6,1) = Skipped_PERTRWGT(6,1) +
PERTRWGT(Current_HHINC_Index);
            % continue;
            Average_Route_Distance_1Way = Average_Route_Distance / 2;
            Bin_Index = find(Average_Route_Distance_1Way >= Bin_Start &
Average_Route_Distance_1Way < Bin_End);
            Average_Travel_Time_CA        =
Average_Travel_Time_CA_VOT(Bin_Index);
            Average_Travel_Cost_CA        =
mean(Average_Travel_Cost_CA_VOT(Bin_Index,:));
            Average_Access_Egress_Cost_CA = -999;
            Average_Lodging_Cost_CA       = -999;

            Average_ProcessingTime_origin_CA =
Average_ProcessingTime_origin_CA_VOT(Bin_Index);
            Average_TravelTime_noSD_CA =
Average_TravelTime_noSD_CA_VOT(Bin_Index);
            Average_StageLength_CA_2way =
Average_StageLength_CA_2way_VOT(Bin_Index);
%
        end
%
        if Average_Travel_Time_Train ==-999 || Average_Travel_Time_Train <0
||Average_Travel_Cost_Train ==-999 || Average_Travel_Cost_Train <0
            Average_Route_Distance_1Way = Average_Route_Distance / 2;
            Bin_Index = find(Average_Route_Distance_1Way >= Bin_Start &
Average_Route_Distance_1Way < Bin_End);
            Average_Travel_Time_Train =
Average_Travel_Time_Train_VOT(Bin_Index);
            Average_Travel_Cost_Train =
mean(Average_Travel_Cost_Train_VOT(Bin_Index));

            Average_Ost_wait_Train =  Average_Ost_wait_Train_VOT(Bin_Index);
            Average_TravelTime_noSD_Train =
Average_TravelTime_noSD_Train_VOT(Bin_Index);
            Average_Time_onTrain_Train =
Average_Time_onTrain_Train_VOT(Bin_Index);
        end
%
%         % Check if zero TT or TC
%         if Average_Travel_Time_Auto == 0 || Average_Travel_Cost_Auto == 0
|| Average_Travel_Time_CA == 0 || Average_Travel_Cost_CA == 0
%             disp('Error! Zero TC or TT!');
%             disp(num2str(Current_HHINC_Index));
%             return;
%         end
%         Check for NaN
%         if isnan(Average_Route_Distance) == 1 ||
isnan(Average_Travel_Time_Auto) == 1 || isnan(Average_Travel_Cost_Auto) == 1
|| isnan(Average_Travel_Time_CA) == 1 || isnan(Average_Travel_Cost_CA) == 1
%             disp('Error! NaN Found!');
%             disp(num2str(Current_HHINC_Index));
%             return;


end
```

D.9 Creating input File with Productivity & NEC and NY,DC constants

```
% Filtering the NEC inputs for 150 miles from the total ATS records.

clc
clear all

%%
% DataDate        = '04_20_2011';
RunDate         = '09_24_2012';
VOTDate         = '09_24_2012';
purpose     = 'Non-Business'; % Business, Non-Business
MSA_Combination = 'All'; % All, MSA to MSA, MSA to Non-MSA, Non-MSA to MSA,
Non-MSA to Non-MSA
Weighted        = 'Yes';
Exclude_Other_Trips = 'No';
Year            = '1995';
Use_ATS_Average_Party_Size = 'Yes';

Input_Dir = 'E:\ATS Analysis 1995';
%'C:\Users\saloni\Documents\MC_Calib_v_1_2- edited-saloni-09_09_2011';
TSAM_Dir       = 'E:\Version_6_6_Release_Date_01_17_2012\TSAM Project - TSAM
6_6'; %'E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data';
ATS_Output_File = fopen([pwd,'\input\', purpose, ' - ', MSA_Combination, ' -
With Travel Time and Cost - ', RunDate, '.txt'], 'w');

%%
file = [Input_Dir,'\',purpose, ' - All - With Travel Time and Cost - ',
VOTDate, '.txt'];
[Fields,numfields,Data,numrecords,fmat] =
read_single_header_csv_file(file,[],0);
%ATS_File = fopen(
% ATS_Field_Names = fgets(ATS_File);
% ATS_Data = textscan(ATS_File,
'%f%s%f%s%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%s%s%f
','delimiter', ',');

CEEDS_MSA_County_Matching_File = fopen([TSAM_Dir,
'\Trip_Generation\input\CEEDS 2001 - County To MSA - Matching List.txt'],
'r');
CEEDS_MSA_County_Matching_File_Field_Names =
fgets(CEEDS_MSA_County_Matching_File);
CEEDS_MSA_County_Matching_Data = textscan(CEEDS_MSA_County_Matching_File,
'%s%s%s%s%s%f%f%f%f%s%s', 'delimiter', '\t');
%--------------------------------------------------------------------------
% Reading Data
%--------------------------------------------------------------------------
% ATS_Data1 = {};
% ATS_Data2 = {};
% ATS_Data_filtered1={};
% ATS_Data_filtered={};
[rows,fields] = size (Data);

or_ceeds_id = strmatch ('ORIGIN_CEEDS_CODE',Fields,'exact');
des_ceeds_id = strmatch ('DESTINATION_CEEDS_CODE',Fields,'exact');
```

```
ostatecol = strmatch('OSTPOST',Fields,'exact');
dstatecol = strmatch('DSTPOST',Fields,'exact');

ostpost = Data {ostatecol};
dstpost = Data {dstatecol};
or_ceeds = Data{or_ceeds_id};
des_ceeds = Data{des_ceeds_id};

commair_tt = strmatch('COMMAIR_TT',Fields,'exact');
Average_Travel_Time_CA= Data {commair_tt};

commair_tc = strmatch('COMMAIR_TC',Fields,'exact');
Average_Travel_Cost_CA= Data{commair_tc};

orproca_t = strmatch('ORPROCA_T',Fields,'exact');
Processing_Time_Origin_CA= Data{orproca_t};

nosdca_tt= strmatch('NOSDCA_TT',Fields,'exact');
No_Schedule_Delay_Time_CA= Data{nosdca_tt};

stlgth_ca = strmatch('STLGTH_CA',Fields,'exact');
Stage_Length_CA= Data{stlgth_ca};

train_tt = strmatch('TRAIN_TT',Fields,'exact');
Average_Travel_Time_Train= Data {train_tt};

train_tc = strmatch('TRAIN_TC',Fields,'exact');
Average_Travel_Cost_Train= Data {train_tc};

orproctrain_t = strmatch('ORPROTRAIN_T',Fields,'exact');
Processing_Time_Origin_TRAIN= Data{orproctrain_t};

nosdtrain_tt= strmatch('NOSDTRAIN_TT',Fields,'exact');
No_Schedule_Delay_Time_TRAIN= Data{nosdtrain_tt};

ontrain_tt = strmatch('ONTRAIN_TT',Fields,'exact');
On_veh_time_TRAIN= Data{ontrain_tt};

distance= strmatch('RTEDUSRT',Fields,'exact');
RTEDUSRT= Data{distance};

% wait_tt= strmatch('Wait_1_TT',Fields,'exact');
% Wait_1_TRAIN= Data{wait_tt};

% accegg_tt= strmatch('ACCEGG_1_TT',Fields,'exact');
% ACCEGG_1_TRAIN= Data{accegg_tt};

%-------------------------------------------------------------------------
% Addidng Logical value for NEC, and NY DC
%-------------------------------------------------------------------------

list_orig = length (ostpost);
% Define the states for which it needs to be checked
state1 = {'DC'};
```

```
state2 = {'NY'};
nec_ceeds = [160 240 560 720 875 1123 2281 3240 3283 3640 4000 ,...
    5015 5190 5380 5483 5523 5600 5640 5660 6160 6483 6680 6760 7560 8003
8480 8840 9160 9280]';
% or_ceeds = str2double (or_ceeds);
% des_ceeds = str2double (des_ceeds);

stateDC_chk = zeros(list_orig,1);
stateNY_chk = zeros(list_orig,1);
NEC_chk = zeros(list_orig,1);
or_chk = ismember (or_ceeds,nec_ceeds); % checks if origin ceeds fall in nec
ceeds set
des_chk = ismember (des_ceeds,nec_ceeds);% checks if destination ceeds fall
in nec ceeds set
Orig_Or_Dest_NEC= double(max(or_chk,des_chk));

for k = 1 : list_orig
    stateDC_chk(k,1) =
max(strcmp(ostpost{k},state1),strcmp(dstpost{k},state1));  % checks if they
match
    stateNY_chk(k,1) =
max(strcmp(ostpost{k},state2),strcmp(dstpost{k},state2));

end
%-------------------------------------------------------------------------
% Calculating Productive Time
%-------------------------------------------------------------------------
 % Define Constants
 slack_percent_CA = 30; % percent
 slack_percent_TRAIN = 75; % percent
 work_percent_time_TRAIN = 90; % percent
 cap_TRAIN_time = 10; %hrs
 prod_CA_percent = 75 ;% percent
 prod_TRAIN_percent = 98 ;% percent
 cap_CA_time = 3; % hours

 % Compute Cruise Time
 Cruise_time = 7.5e-008*(Stage_Length_CA).^2 + 0.0015*(Stage_Length_CA)+0.16;
% equation extracted from T-100 data 2003 for all the flights toegther

% Finding Work Times for CA
slack_time_CA = Processing_Time_Origin_CA * slack_percent_CA*0.01;   % time
available to work from procesing time
work_time_CA= 2*min(Cruise_time/2 , cap_CA_time);  % as we are limiting the
work done for laptop battery , fatigue.;
prod_time_CA = prod_CA_percent *0.01 * (work_time_CA+slack_time_CA);
% time_ratio_CA = prod_time_CA./CA_noSD_TTime;
unprod_time_CA = Average_Travel_Time_CA - prod_time_CA; % all times computed
are 2 way

% Finding Work Times for TRAIN
slack_time_TRAIN = Processing_Time_Origin_TRAIN *slack_percent_TRAIN*0.01;
On_veh_time_TRAIN_one_way = On_veh_time_TRAIN/2;
number_days = floor(On_veh_time_TRAIN_one_way/24);% they can work 10 hrs
everyday on train, wer they are not restricted by the battery
```

```
last_day_hrs = mod(On_veh_time_TRAIN_one_way,24); % to find hours on the last
day
% last_day_hrs = min (last_day_hrs_temp(i)*work_percent_time_Train*0.01,
cap_Train_time);
work_time_TRAIN =slack_time_TRAIN
+2*(number_days*cap_TRAIN_time+min(last_day_hrs,cap_TRAIN_time))*work_percent
_time_TRAIN*0.01;
% work_time_Train = slack_time_train + last_day_hrs+
number_days*cap_Train_time;
prod_time_TRAIN = prod_TRAIN_percent * work_time_TRAIN*0.01;
% time_ratio_Train = prod_time_Train./TRAIN_noSD_TTime;
unprod_time_TRAIN = Average_Travel_Time_Train - prod_time_TRAIN;
%--------------------------------------------------------------------------
%Printing Records
%--------------------------------------------------------------------------

%% prnting record without productive time


% ATS_Output_File = fopen([pwd,'\input\', purpose, ' - ', MSA_Combination, '
- With Travel Time and Cost - ', RunDate, '.txt'], 'w');
% fprintf(ATS_Output_File, '%s\n',
'ORIGIN_CEEDS_CODE,OSTPOST,DESTINATION_CEEDS_CODE,DSTPOST,WAS_CHECK,NY_CHECK,
NEC_CHECK,TSAM_TRANSOD,TSAM_HHINC,HHTRPWGT,TRPARTYHH,RTEDUSRT,TSAM_RTEDUSRT,A
UTO_TT,AUTO_TC,COMMAIR_TT,COMMAIR_TC,TRAIN_TT,TRAIN_TC,ORIGIN_MSA_DUMMY,DESTI
NATION_MSA_DUMMY,NITEDEST,AUTO_DT');
%  ORIGIN_CEEDS_CODE = Data {1};
% OSTPOST = Data {2};
% DESTINATION_CEEDS_CODE= Data{3};
% DSTPOST= Data {4};
% WAS_CHECK = stateDC_chk;  % new additions
% NY_CHECK = stateNY_chk;
% NEC_CHECK=Orig_Or_Dest_NEC;
% TRANSOD= Data {5};
% HHINC= Data {6};
% PERTRWGT=Data {7};
% TRPARTY= Data{8};
% RTEDUSRT= Data {9};
% Average_Route_Distance= Data{10};
% Average_Travel_Time_Auto= Data{11};
% Average_Travel_Cost_Auto= Data {12};
% Average_Travel_Time_CA= Data {13};
% Average_Travel_Cost_CA= Data{14};
% Average_Travel_Time_Train= Data {15};
% Average_Travel_Cost_Train= Data {16};
% Origin_MSA_Dummy= Data{17};
% Destination_MSA_Dummy= Data {18};
% NITEDEST =Data {19};
% Average_Route_DrivingTime= Data {20};
%
% Dest = Data {3};
% records = length (Dest);
% for i = 1 : records
%
```

```
%  fprintf(ATS_Output_File,'%s,%s,%s,%s,%.0f,%.0f, %.0f,
%.0f,%.0f,%.2f,%.0f,%.0f,%.0f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.0f,%.0f,%.0f,%.
1f\n',...
%             char(ORIGIN_CEEDS_CODE(i)) , char(OSTPOST(i)),
char(DESTINATION_CEEDS_CODE(i)) ,...
%             char(DSTPOST(i)), WAS_CHECK(i), NY_CHECK(i), NEC_CHECK(i),
TRANSOD(i) , HHINC(i) , PERTRWGT(i) , ...
%             TRPARTY(i) , RTEDUSRT(i) , Average_Route_Distance(i),
Average_Travel_Time_Auto(i), Average_Travel_Cost_Auto(i),...
%             Average_Travel_Time_CA(i),
Average_Travel_Cost_CA(i),Average_Travel_Time_Train(i),Average_Travel_Cost_Tr
ain(i), ...
%             Origin_MSA_Dummy(i),Destination_MSA_Dummy(i), NITEDEST(i) ,
Average_Route_DrivingTime(i));
% end

%% prnting record including productive time variables
fprintf(ATS_Output_File, '%s\n',
'ORIGIN_CEEDS_CODE,OSTPOST,DESTINATION_CEEDS_CODE,DSTPOST,WAS_CHECK,NY_CHECK,
NEC_CHECK,TSAM_TRANSOD,TSAM_HHINC,HHTRPWGT,TRPARTYHH,RTEDUSRT,TSAM_RTEDUSRT,A
UTO_TT,AUTO_TC,COMMAIR_TT,COMMAIR_TC,COMMAIR_PT,COMMAIR_UPT,TRAIN_TT,TRAIN_TC
,TRAIN_PT,TRAIN_UPT,ORIGIN_MSA_DUMMY,DESTINATION_MSA_DUMMY,NITEDEST,AUTO_DT')
;
 ORIGIN_CEEDS_CODE = or_ceeds;
OSTPOST = ostpost;
DESTINATION_CEEDS_CODE= des_ceeds;
DSTPOST= dstpost;
WAS_CHECK = stateDC_chk;  % new additions
NY_CHECK = stateNY_chk;
NEC_CHECK=Orig_Or_Dest_NEC;

transod   = strmatch('TSAM_TRANSOD',Fields,'exact');
TRANSOD= Data {transod};

hhinc   = strmatch('TSAM_HHINC',Fields,'exact');
HHINC= Data {hhinc};

trpwt   = strmatch('PERTRWGT',Fields,'exact');
PERTRWGT=Data {trpwt};

partysz   = strmatch('TRPARTY',Fields,'exact');
TRPARTY= Data{partysz};

rtedusrt = strmatch('RTEDUSRT',Fields,'exact');
RTEDUSRT= Data {rtedusrt};

tsam_rtedusrt = strmatch('TSAM_RTEDUSRT',Fields,'exact');
Average_Route_Distance= Data{tsam_rtedusrt};

auto_tt = strmatch('AUTO_TT',Fields,'exact');
Average_Travel_Time_Auto= Data{auto_tt};

auto_tc = strmatch('AUTO_TC',Fields,'exact');
Average_Travel_Cost_Auto= Data {auto_tc};
```

```
or_msa = strmatch('ORIGIN_MSA_DUMMY',Fields,'exact');
Origin_MSA_Dummy= Data{or_msa};

ds_msa = strmatch('DESTINATION_MSA_DUMMY',Fields,'exact');
Destination_MSA_Dummy= Data {ds_msa};

nitedest = strmatch('NITEDEST',Fields,'exact');
NITEDEST =Data {nitedest};

auto_dt = strmatch('AUTO_DT',Fields,'exact');
Average_Route_DrivingTime= Data {auto_dt};

Dest = Data {3};
records = length (Dest);
for i = 1 : records
%
 fprintf(ATS_Output_File,'%.0f,%s,%.0f,%s,%.0f,%.0f, %.0f,
%.0f,%.0f,%.2f,%.0f,%.0f,%.0f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.
1f,%.0f,%.0f,%.0f,%.1f\n',...
            ORIGIN_CEEDS_CODE(i) , char(OSTPOST(i)),
DESTINATION_CEEDS_CODE(i) ,...
            char(DSTPOST(i)), WAS_CHECK(i), NY_CHECK(i), NEC_CHECK(i),
TRANSOD(i) , HHINC(i) , PERTRWGT(i) , ...
            TRPARTY(i) , RTEDUSRT(i) , Average_Route_Distance(i),
Average_Travel_Time_Auto(i), Average_Travel_Cost_Auto(i),...
            Average_Travel_Time_CA(i),
Average_Travel_Cost_CA(i),prod_time_CA(i), unprod_time_CA(i),
Average_Travel_Time_Train(i),Average_Travel_Cost_Train(i),prod_time_TRAIN(i),
unprod_time_TRAIN(i), Origin_MSA_Dummy(i),Destination_MSA_Dummy(i),...
            NITEDEST(i) , Average_Route_DrivingTime(i));
end

% for i = 1 : records
%

%  fprintf(ATS_Output_File,'%.0f,%s,%.0f,%s,%.0f,%.0f, %.0f,
%.0f,%.0f,%.2f,%.0f,%.0f,%.0f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.
1f,%.0f,%.0f,%.0f,%.1f\n',...
%             ORIGIN_CEEDS_CODE(i) , char(OSTPOST(i)),
DESTINATION_CEEDS_CODE(i) ,...
%             char(DSTPOST(i)), WAS_CHECK(i), NY_CHECK(i), NEC_CHECK(i),
TRANSOD(i) , HHINC(i) , PERTRWGT(i) , ...
%             TRPARTY(i) , RTEDUSRT(i) , Average_Route_Distance(i),

Average_Travel_Time_Auto(i), Average_Travel_Cost_Auto(i),...
%             Average_Travel_Time_CA(i),

Average_Travel_Cost_CA(i),prod_time_CA(i), unprod_time_CA(i),

Average_Travel_Time_Train(i),Average_Travel_Cost_Train(i),prod_time_TRAIN(i),
unprod_time_TRAIN(i), Origin_MSA_Dummy(i),Destination_MSA_Dummy(i),...
%             NITEDEST(i) , Average_Route_DrivingTime(i));
% end
```

D.10 Creating input File with NEC and NY,DC constants

```
% Filtering the NEC inputs for 150 miles from the total ATS records.

clc
clear all

Output_Dir = 'E:\ATS Analysis 1995';
%'C:\Users\saloni\Documents\MC_Calib_v_1_2- edited-saloni-09_09_2011';
TSAM_Dir       = 'E:\Version_6_6_Release_Date_01_17_2012\TSAM Project - TSAM
6_6'; %'E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data';

%%
% DataDate        = '04_20_2011';
RunDate         = '09_24_12';
VOTDate         = '09_03_2012';
purpose     = 'Business'; % Business, Non-Business
MSA_Combination = 'All'; % All, MSA to MSA, MSA to Non-MSA, Non-MSA to MSA,
Non-MSA to Non-MSA
Weighted        = 'Yes';
Exclude_Other_Trips = 'No';
Year            = '1995';
Use_ATS_Average_Party_Size = 'Yes';
%%
file = [Output_Dir,'\',purpose, ' - All - With Travel Time and Cost - ',
VOTDate, '.txt'];
[Fields,numfields,Data,numrecords,fmat] =
read_single_header_csv_file(file,[],0);
%ATS_File = fopen(
% ATS_Field_Names = fgets(ATS_File);
% ATS_Data = textscan(ATS_File,
'%f%s%f%s%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%s%s%f
','delimiter', ',');

CEEDS_MSA_County_Matching_File = fopen([TSAM_Dir,
'\Trip_Generation\input\CEEDS 2001 - County To MSA - Matching List.txt'],
'r');
CEEDS_MSA_County_Matching_File_Field_Names =
fgets(CEEDS_MSA_County_Matching_File);
CEEDS_MSA_County_Matching_Data = textscan(CEEDS_MSA_County_Matching_File,
'%s%s%s%s%s%s%f%f%f%f%s%s', 'delimiter', '\t');
%%
% ATS_Data1 = {};
% ATS_Data2 = {};
% ATS_Data_filtered1={};
% ATS_Data_filtered={};
[rows,fields] = size (Data);

or_ceeds_id = strmatch ('ORIGIN_CEEDS_CODE',Fields,'exact');
des_ceeds_id = strmatch ('DESTINATION_CEEDS_CODE',Fields,'exact');
ostatecol = strmatch('OSTPOST',Fields,'exact');
dstatecol = strmatch('DSTPOST',Fields,'exact');

ostpost = Data {ostatecol};
dstpost = Data {dstatecol};
```

```
or_ceeds = Data{or_ceeds_id};
des_ceeds = Data{des_ceeds_id};

list_orig = length (ostpost);
% Define the states for which it needs to be checked
state1 = {'DC'};
state2 = {'NY'};
nec_ceeds = [160 240 560 720 875 1123 2281 3240 3283 3640 4000 ,...
    5015 5190 5380 5483 5523 5600 5640 5660 6160 6483 6680 6760 7560 8003
8480 8840 9160 9280]';
or_ceeds = str2double (or_ceeds);
% des_ceeds = str2double (des_ceeds);

stateDC_chk = zeros(list_orig,1);
stateNY_chk = zeros(list_orig,1);
NEC_chk = zeros(list_orig,1);
or_chk = ismember (or_ceeds,nec_ceeds); % checks if origin ceeds fall in nec
ceeds set
des_chk = ismember (des_ceeds,nec_ceeds);% checks if destination ceeds fall
in nec ceeds set
Orig_Or_Dest_NEC= double(max(or_chk,des_chk));

for k = 1 : list_orig
    stateDC_chk(k,1) =
max(strcmp(ostpost{k},state1),strcmp(dstpost{k},state1));  % checks if they
match
    stateNY_chk(k,1) =
max(strcmp(ostpost{k},state2),strcmp(dstpost{k},state2));

end

%% prnting record without productive time


% ATS_Output_File = fopen([pwd,'\input\', purpose, ' - ', MSA_Combination, '
- With Travel Time and Cost - ', RunDate, '.txt'], 'w');
% fprintf(ATS_Output_File, '%s\n',
'ORIGIN_CEEDS_CODE,OSTPOST,DESTINATION_CEEDS_CODE,DSTPOST,WAS_CHECK,NY_CHECK,
NEC_CHECK,TSAM_TRANSOD,TSAM_HHINC,HHTRPWGT,TRPARTYHH,RTEDUSRT,TSAM_RTEDUSRT,A
UTO_TT,AUTO_TC,COMMAIR_TT,COMMAIR_TC,TRAIN_TT,TRAIN_TC,ORIGIN_MSA_DUMMY,DESTI
NATION_MSA_DUMMY,NITEDEST,AUTO_DT');
%  ORIGIN_CEEDS_CODE = Data {1};
% OSTPOST = Data {2};
% DESTINATION_CEEDS_CODE= Data{3};
% DSTPOST= Data {4};
% WAS_CHECK = stateDC_chk;  % new additions
% NY_CHECK = stateNY_chk;
% NEC_CHECK=Orig_Or_Dest_NEC;
% TRANSOD= Data {5};
% HHINC= Data {6};
% PERTRWGT=Data {7};
% TRPARTY= Data{8};
% RTEDUSRT= Data {9};
% Average_Route_Distance= Data{10};
% Average_Travel_Time_Auto= Data{11};
```

```
% Average_Travel_Cost_Auto= Data {12};
% Average_Travel_Time_CA= Data {13};
% Average_Travel_Cost_CA= Data{14};
% Average_Travel_Time_Train= Data {15};
% Average_Travel_Cost_Train= Data {16};
% Origin_MSA_Dummy= Data{17};
% Destination_MSA_Dummy= Data {18};
% NITEDEST =Data {19};
% Average_Route_DrivingTime= Data {20};
%
% Dest = Data {3};
% records = length (Dest);
% for i = 1 : records
%
%  fprintf(ATS_Output_File,'%s,%s,%s,%s,%.0f,%.0f, %.0f,
%.0f,%.0f,%.2f,%.0f,%.0f,%.0f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.0f,%.0f,%.0f,%.
1f\n',...
%             char(ORIGIN_CEEDS_CODE(i)) , char(OSTPOST(i)),
char(DESTINATION_CEEDS_CODE(i)) ,...
%             char(DSTPOST(i)), WAS_CHECK(i), NY_CHECK(i), NEC_CHECK(i),
TRANSOD(i) , HHINC(i) , PERTRWGT(i) , ...
%             TRPARTY(i) , RTEDUSRT(i) , Average_Route_Distance(i),
Average_Travel_Time_Auto(i), Average_Travel_Cost_Auto(i),...
%             Average_Travel_Time_CA(i),
Average_Travel_Cost_CA(i),Average_Travel_Time_Train(i),Average_Travel_Cost_Tr
ain(i), ...
%             Origin_MSA_Dummy(i),Destination_MSA_Dummy(i), NITEDEST(i) ,
Average_Route_DrivingTime(i));
% end

%% prnting record including productive time variables
ATS_Output_File = fopen([pwd,'\input\', purpose, ' - ', MSA_Combination, ' -
With Travel Time and Cost - ', RunDate, '.txt'], 'w');
fprintf(ATS_Output_File, '%s\n',
'ORIGIN_CEEDS_CODE,OSTPOST,DESTINATION_CEEDS_CODE,DSTPOST,WAS_CHECK,NY_CHECK,
NEC_CHECK,TSAM_TRANSOD,TSAM_HHINC,HHTRPWGT,TRPARTYHH,RTEDUSRT,TSAM_RTEDUSRT,A
UTO_TT,AUTO_TC,COMMAIR_TT,COMMAIR_TC,ORPROCA_T,NOSDCA_TT,STLGTH_CA,TRAIN_TT,T
RAIN_TC,ORPROTRAIN_T,NOSDTRAIN_TT,ONTRAIN_TT,ORIGIN_MSA_DUMMY,DESTINATION_MSA
_DUMMY,NITEDEST,AUTO_DT');
 ORIGIN_CEEDS_CODE = or_ceeds;
OSTPOST = ostpost;
DESTINATION_CEEDS_CODE= des_ceeds;
DSTPOST= dstpost;
WAS_CHECK = stateDC_chk;  % new additions
NY_CHECK = stateNY_chk;
NEC_CHECK=Orig_Or_Dest_NEC;

transod   = strmatch('TSAM_TRANSOD',Fields,'exact');
TRANSOD= Data {transod};

hhinc   = strmatch('TSAM_HHINC',Fields,'exact');
HHINC= Data {hhinc};

trpwt   = strmatch('PERTRWGT',Fields,'exact');
PERTRWGT=Data {trpwt};
```

```
partysz   = strmatch('TRPARTY',Fields,'exact');
TRPARTY= Data{partysz};

rtedusrt = strmatch('RTEDUSRT',Fields,'exact');
RTEDUSRT= Data {rtedusrt};

tsam_rtedusrt = strmatch('TSAM_RTEDUSRT',Fields,'exact');
Average_Route_Distance= Data{tsam_rtedusrt};

auto_tt = strmatch('AUTO_TT',Fields,'exact');
Average_Travel_Time_Auto= Data{auto_tt};

auto_tc = strmatch('AUTO_TC',Fields,'exact');
Average_Travel_Cost_Auto= Data {auto_tc};

commair_tt = strmatch('COMMAIR_TT',Fields,'exact');
Average_Travel_Time_CA= Data {commair_tt};

commair_tc = strmatch('COMMAIR_TC',Fields,'exact');
Average_Travel_Cost_CA= Data{commair_tc};

orproca_t = strmatch('ORPROCA_T',Fields,'exact');
Processing_Time_Origin_CA= Data{orproca_t};

nosdca_tt= strmatch('NOSDCA_TT',Fields,'exact');
No_Schedule_Delay_Time_CA= Data{nosdca_tt};

stlgth_ca = strmatch('STLGTH_CA',Fields,'exact');
Stage_Length_CA= Data{stlgth_ca};

train_tt = strmatch('TRAIN_TT',Fields,'exact');
Average_Travel_Time_Train= Data {train_tt};

train_tc = strmatch('TRAIN_TC',Fields,'exact');
Average_Travel_Cost_Train= Data {train_tc};

orproctrain_t = strmatch('ORPROTRAIN_T',Fields,'exact');
Processing_Time_Origin_TRAIN= Data{orproctrain_t};

nosdtrain_tt= strmatch('NOSDTRAIN_TT',Fields,'exact');
No_Schedule_Delay_Time_TRAIN= Data{nosdtrain_tt};

ontrain_tt = strmatch('ONTRAIN_TT',Fields,'exact');
On_veh_time_TRAIN= Data{ontrain_tt};

or_msa = strmatch('ORIGIN_MSA_DUMMY',Fields,'exact');
Origin_MSA_Dummy= Data{or_msa};

ds_msa = strmatch('DESTINATION_MSA_DUMMY',Fields,'exact');
Destination_MSA_Dummy= Data {ds_msa};

nitedest = strmatch('NITEDEST',Fields,'exact');
NITEDEST =Data {nitedest};
```

```
auto_dt = strmatch('AUTO_DT',Fields,'exact');
Average_Route_DrivingTime= Data {auto_dt};

Dest = Data {3};
records = length (Dest);
for i = 1 : records

 fprintf(ATS_Output_File,'%.0f,%s,%.0f,%s,%.0f,%.0f, %.0f,
%.0f,%.0f,%.2f,%.0f,%.0f,%.0f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,%.
1f,%.1f,%.1f,%.0f,%.0f,%.0f,%.1f\n',...
            ORIGIN_CEEDS_CODE(i) , char(OSTPOST(i)),
DESTINATION_CEEDS_CODE(i) ,...
            char(DSTPOST(i)), WAS_CHECK(i), NY_CHECK(i), NEC_CHECK(i),
TRANSOD(i) , HHINC(i) , PERTRWGT(i) , ...
            TRPARTY(i) , RTEDUSRT(i) , Average_Route_Distance(i),
Average_Travel_Time_Auto(i), Average_Travel_Cost_Auto(i),...
            Average_Travel_Time_CA(i),
Average_Travel_Cost_CA(i),Processing_Time_Origin_CA(i),No_Schedule_Delay_Time
_CA(i),...
            Stage_Length_CA(i),
Average_Travel_Time_Train(i),Average_Travel_Cost_Train(i),Processing_Time_Ori
gin_TRAIN(i), ...
            No_Schedule_Delay_Time_TRAIN(i),On_veh_time_TRAIN(i),
Origin_MSA_Dummy(i),Destination_MSA_Dummy(i),...
            NITEDEST(i) , Average_Route_DrivingTime(i));
end
```

D.11 Comparing distribution along distance with AMTRAK

```
% This code will find the ridership along distances from a particular
station, and put them into different bins.
% It will then calculate the percentage of the ridership for distances in
groups of 100 miles
% THIS FILE needs mode choice output files pasted in a folder called
% s2s_trips on the desktop
% it uses the function Train_Station_Index_Finder.m
% you need to enter the train station ID from the amtrak id list
clc
clear all

load
C:\Users\saloni\Documents\MATLAB\amtrak\Amtrak_ridership_percentage_wout_100m
iles.mat % this file contains the ridership percentage along different miles,
as computed using the Amtrak data
load('C:\Users\saloni\Desktop\s2s_trips\business\s2sTrain_pTripTable_Total.ma
t') %This file contains the business trips output - mode choice
business_trips = s2sTrain_pTripTable_Total;
clear s2sTrain_pTripTable_Total
load('C:\Users\saloni\Desktop\s2s_trips\non-
business\s2sTrain_pTripTable_Total.mat') % This file contains the non
business trips output for mode choice
non_business_trips = s2sTrain_pTripTable_Total;
clear s2sTrain_pTripTable_Total
```

```
load('E:\avandyke\Documents\TSAM_GUI_VB_2005\bin\data\mode_choice\input\S2S_T
rain_Distance_HSR'); %This is the original matrix of distance between
stations. This is the original data and is assumed to be correct unless
specific changes have been noted below.

global TrainStation_List
load('C:\Users\saloni\Desktop\s2s_trips\TrainStation_List.mat') %This file
containts the Amtrak "TrainStation_List"

total_stations = length (business_trips);
total_ridership = zeros (1,20);

reply = input ( ' Please enter the abbreviations three letter in CAPS ',
's');
amtrak_id = input('The Amtrak station ID, Find it in the excel file called
Amtrak ID xls ');
station_abbreviation  = {reply}; % Converts to a cell array
Ridership_Station_Index =  Train_Station_Index_Finder(station_abbreviation);

for i = 1: total_stations
    dist = S2S_Train_Distance_HSR (i,Ridership_Station_Index);

            if (dist > 99 && dist < 200) % distances between 100 and 200
           total_ridership (1,1) = total_ridership (1,1)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 199 && dist < 300) % distances between 200 - 300
           total_ridership (1,2) = total_ridership (1,2)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 299 && dist < 400)
           total_ridership (1,3) = total_ridership (1,3)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 399 && dist < 500)
            total_ridership (1,4) = total_ridership (1,4)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 499 && dist < 600)
           total_ridership (1,5) = total_ridership (1,5)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 599 && dist < 700)
           total_ridership (1,6) = total_ridership (1,6)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
           elseif (dist > 699 && dist < 800)
           total_ridership (1,7) = total_ridership (1,7)+ business_trips
(i,Ridership_Station_Index) + business_trips
```

```
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 799 && dist < 900)
           total_ridership (1,8) = total_ridership (1,8)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 899 && dist < 1000)
           total_ridership (1,9) = total_ridership (1,9)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 999 && dist < 1100)
           total_ridership (1,10) = total_ridership (1,10)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1099 && dist < 1200)
           total_ridership (1,11) = total_ridership (1,11)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1199 && dist < 1300)
           total_ridership (1,12) = total_ridership (1,12)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1299 && dist < 1400)
           total_ridership (1,13) = total_ridership (1,13)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1399 && dist < 1500)
           total_ridership (1,14) = total_ridership (1,14)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1499 && dist < 1600)
           total_ridership (1,15) = total_ridership (1,15)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1599 && dist < 1700)
           total_ridership (1,16) = total_ridership (1,16)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1699 && dist < 1800)
           total_ridership (1,17) = total_ridership (1,17)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1799 && dist < 1900)
           total_ridership (1,18) = total_ridership (1,18)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
```

```
          elseif (dist > 1899 && dist < 2000)
           total_ridership (1,19) = total_ridership (1,19)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);
          elseif (dist > 1999)
           total_ridership (1,20) = total_ridership (1,20)+ business_trips
(i,Ridership_Station_Index) + business_trips
(Ridership_Station_Index,i)+non_business_trips
(i,Ridership_Station_Index)+non_business_trips (Ridership_Station_Index,i);

            end
end

total = sum(total_ridership);  % total ridership for tsam
percent_ridership = total_ridership*100/total; % converts to percentage tsam
ridership
amtrak_percent_ridership = new_percent(amtrak_id,:); % extracts the
corresponding amtrak data
groups = linspace (100,1000,10);

subplot(2,1,1)
bar (groups,amtrak_percent_ridership(1:10),0.4,'histc')
title ('Ridership Distribution NYP- AMTRAK
data(2009)','fontweight','b','color','b','fontsize',16)
xlabel ('Miles','fontweight','b','fontsize',12)
ylabel ('Ridership %','fontsize',12)
axis ([100 1000 0 100])
grid on
% set ( gca, 'xtick', [100 200 300 400 500 600 700 800 900 1000 1100 1200
1300 1400 1500 1600 1700 1800 1900 2000],'ytick' ,[0 10 20 30 40 50 60 70 80
90 100],'Fontsize',14)
set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 ],'ytick' ,[0 10
20 30 40 50 60 70 80 90 100],'Fontsize',12)

subplot(2,1,2)
bar ( groups, percent_ridership(1:10),0.4,'histc')
% title ('Model Estimate','fontweight','b','color','b','fontsize',24)
title ('Ridership Distribution by TSAM 6.6 for
2009','fontweight','b','color','b','fontsize',16)
xlabel ('Miles','fontweight','b','fontsize',12)
ylabel ('Ridership %','fontsize',12)
axis ([100 1000 0 100])
% set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 1100 1200 1300
1400 1500 1600 1700 1800 1900 2000],'ytick' ,[0 10 20 30 40 50 60 70 80 90
100],'Fontsize',14)
set ( gca,'xtick', [100 200 300 400 500 600 700 800 900 1000 ],'ytick' ,[0 10
20 30 40 50 60 70 80 90 100],'Fontsize',12)
grid on

clear s*
% clear t*
clear d*
clear b*
clear R*
clear r*
```

```
clear T*
clear n*
clear i
clear S*
clear g*
clear amtrak_id
```

D.12 Extracting Time Spent in Cruise from T100

```
% this will extract the time spent in the various flight modes and the
% stage lengths.
clc
clear all

load('C:\Users\saloni\Documents\MATLAB\flight_tracks.mat') %This file
containts the BADA data for aircrafts
load('C:\Users\saloni\Documents\MATLAB\turbo_planes.mat') %This file
containts the aircrfat names with turboprop engines
load('C:\Users\saloni\Documents\MATLAB\jet_planes.mat') %This file containts
the aircrfat names with jet engines
load('C:\Users\saloni\Documents\MATLAB\piston_planes.mat') %This file
containts the aircrfat names with piston engines

size_tracks = length(flight_tracks);

for i=1: size_tracks
    x(i,1) = flight_tracks(1,i).Stage_Length_miles;
    y(i,1) = flight_tracks(1,i).Cruise_Time_hrs;
end

scatter (x,y,'*','b');
ylabel('Cruise Time hours'), xlabel('Stage Length miles'), title('Cruise Time
in hours for Aircrafts'), grid on

% size_tracks = length(flight_tracks);
% % turboprop =
{'AT45__','AT72__','ATP___','BE20__','BE99__','BE9L__','C130__','D228__','D32
8__','DH8A__','DH8C__','DH8D__','E120__','F27___','H25B__','JS31__','JS32__',
'JS41__','PA27__','PAY2__','PAY3__','SF34__','SH36__','SW3___','SW4___'}';
% % jet =
{'B742__','A319__','A30B__','A306__','A310__','A320__','A332__','A343__','A31
9__','A321__','A343__','A346__','RJ85__','B712__','','','','','','','','','',
'','','','','','',''}
% m=1;
% n=1;
% s=1;
% for i=1: size_tracks
%
%     size_turbo = length (turbo);
%     for a = 1: size_turbo
%     if strcmp(flight_tracks(i).Aircraft_type,turbo(a))
%        turbo_climb_time(m,1) = flight_tracks(1,i).Climb_Time_hrs; % gives
the time to climb
%        turbo_cruise_time(m,1) = flight_tracks(1,i).Cruise_Time_hrs;
%        turbo_descent_time(m,1) = flight_tracks(1,i).Descent_Time_hrs;
```

```
%        turbo_stage_length_miles(m,1) =
flight_tracks(1,i).Stage_Length_miles;
%        m=m+1;
%     end
%     end
%
%       size_jet = length (jet);
%     for a = 1: size_jet
%     if strcmp(flight_tracks(1,i).Aircraft_type,jet(a))
%        jet_climb_time(n,1) = flight_tracks(1,i).Climb_Time_hrs; % gives the
time to climb
%        jet_cruise_time(n,1) = flight_tracks(1,i).Cruise_Time_hrs;
%        jet_descent_time(n,1) = flight_tracks(1,i).Descent_Time_hrs;
%        jet_stage_length_miles(n,1) = flight_tracks(1,i).Stage_Length_miles;
%        n=n+1;
%     end
%     end
%
%     size_piston = length (piston);
%     for a = 1: size_piston
%     if strcmp(flight_tracks(1,i).Aircraft_type,piston(a))
%        piston_climb_time(s,1) = flight_tracks(1,i).Climb_Time_hrs; % gives
the time to climb
%        piston_cruise_time(s,1) = flight_tracks(1,i).Cruise_Time_hrs;
%        piston_descent_time(s,1) = flight_tracks(1,i).Descent_Time_hrs;
%        piston_stage_length_miles(s,1) =
flight_tracks(1,i).Stage_Length_miles;
%        s=s+1;
%     end
%     end
% end
%
% clear m
% clear n
% clear s
% clear a
% clear i
% clear jet
% clear turbo
% clear piston
% clear flight_tracks
% clear size_jet
% clear size_piston
% clear size_turbo
% clear size_tracks
% clc
% figure
% scatter ( jet_climb_time,jet_stage_length_miles,'*','b')
% xlabel('Climb Time hours'), ylabel('Stage Length miles'), title('Climb Time
in hours for Aircrafts'), grid on
% hold on
% scatter ( piston_climb_time,piston_stage_length_miles,'*','r')
% scatter ( turbo_climb_time,turbo_stage_length_miles,'*','k')
% legend('Jet Climb time','Piston Climb time','Turboprop climb time')
% hold off
%
% figure
```

```
% scatter ( jet_cruise_time,jet_stage_length_miles,'*','b')
% xlabel('Cruise Time hours'), ylabel('Stage Length miles'), title('Cruise
Time in hours for Aircrafts'), grid on
% hold on
% scatter ( piston_cruise_time,piston_stage_length_miles,'*','r')
% scatter ( turbo_cruise_time,turbo_stage_length_miles,'*','k')
% legend('Jet Cruise time','Piston Cruise time','Turboprop Cruise time')
% hold off
%
% figure
% scatter ( jet_descent_time,jet_stage_length_miles,'*','b')
% xlabel('Descent Time hours'), ylabel('Stage Length miles'), title('Descent
Time in hours for Aircrafts'), grid on
% hold on
% scatter ( piston_descent_time,piston_stage_length_miles,'*','r')
% scatter ( turbo_descent_time,turbo_stage_length_miles,'*','k')
% legend('Jet Descent time','Piston Descent time','Turboprop Descent time')
% hold off
```

REFERENCES

"High-Speed Rail Strategic Plan: Press Release & Highlights". U.S. Department of Transportation, 2009, April 16.

Albalate, D.,Bel,G. "High-Speed rail: Lessons for Policy Makers from Abroad." Research Institute of Applied Economics, Barcelon: University of Barcelona, n.d.

American Travel Survey: Technical Documentation. Bureau of Transportation Stastics, U.S.Department of Transportation, 1995.

Ashiabor, S., H. Baik and A.A.Trani. "Logit Models for Forecasting Nationwide Intercity Travel Demand in the United States." *Transportation Research Record: Journal of the Transportation Research Board* (Transportation Reseatch Board of the National Academics, Washington, D.C.), no. 2007 (2007): 1-12.

Baik, H., A.A.Trani, N.Hinze, H.Swingle ,S.Ashiabor and A.Seshadri. "Forecasting Model for Air Taxi, Commercial Airline and Automobile Demand in the United States." *Transportation Research Record : Journal of the Transportation Research Board* (Transportation Research Board of the National Academics, Washington D.C.), no. 2052 (2008): 9-20.

Breure A, van Meel J. "Airport offices: facilitating nomadic workers." *Facilities* 21, no. 7 (2003): 175-179.

Central Japan Railway Company. *About the Shinkansen.* n.d. http://english.jr-central.co.jp/about/safety.html (accessed February 02, 2012).

Clifford, B.R. *The Acela Express.* Japan Railway & Transport Review, March 2005.

Commuter Rail in North America. Wikipedia. Wikimedia Foundation. n.d. http://en.wikipedia.org/wiki/Commuter_rail_in_North_America (accessed June 25, 2012).

Department of Transportation. *High-Speed Groun Transportation for America.* Federal Railroad Administration, 1997.

Fact Sheets for all Amtrak Stations. National Association of Railroad Passengers, n.d.

Feng, David. *A Complete Guide to China's High Speed Rail |CNNGo.com.* Travel Asia with CNNGo.com - The Insider's Guide to Travelling in Asia. n.d. http://www.cnngo.com/shanghai/play/complete-guide-chinas-high-speed-rail-038363 (accessed February 03, 2012).

Frittelli, J. "High Speed Rail (HSR) in the United States." *CRS Report for Congress*, 2009.

Frustration of Air Travel Push Passengers to Amtrak. The New York Times. August 15, 2012. http://www.nytimes.com/2012/08/16/business/hassles-of-air-travel-push-passengers-to-amtrak.html?_r=2&pagewanted=2&ref=business (accessed August 20, 2012).

Google Maps. Google Maps. n.d. https://www.google.com/maps?ie=UTF8 (accessed August 01, 2012).

Grayson, A. "Disaggregate Model of Mode Choice in Intercity Travel." *Transportation Research Record*, no. 385 (1981): 36-42.

Gunn, Bradley and Hensher. "High-Speed Rail Marker Projection to Very High Frequency Air Services." *Journal of Public Transportation*, no. 19 (1992): 117-139.

High Speed Rail in Germany: Inter-city Planes are Grounded by Faster Trains, World News. Latest News, Sport and Comment from The Guardian. n.d. http://www.guardian.co.uk/world/2009/aug/05/high-speed-rail-grounds-city-planes (accessed February 03, 2012).

High Speed Rails in the United States. Wikipedia, The Free Encyclopedia . n.d. http://en.wikipedia.org/wiki/High-speed_rail_in_the_United_States (accessed February 2012, 2012).

History of the French TGV. Squidoo: Welcome to Squidoo. n.d. http://www.railway-technology.com/projects/spain/>. (accessed February 03, 2012).

Joshi, C. *Development of a Decision Support Tool for Planning Rail Systems: An Implementation in TSAM.* Civil Engineering, Blacksburg: Virginia Polytechnic Institute and State University, 2005.

Koppelman, F.S. "Multidimensional Model System for Intercity Travel Choice Behavior." *Transportation Research Record*, no. 1241 (1989): 1-8.

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

Lew, Alexander. *Today in History: Train a Grande Vitesse (TGV) Delivers First Passengers to Lyon.* Wired.com. n.d. http://www.wired.com/autopia/2007/09/today-in-hist-3/>. (accessed February 02, 2012).

List of U.S. Cities with Most Households without a Car. Wikipedia. Wikimedia Foundation. April 08, 2012. http://en.wikipedia.org/wiki/List_of_U.S._cities_with_most_households_without_a_car (accessed June 25, 2012).

López-Pita, A. and F. Robusté. "Impact of High-Speed Lines in Relation to Very High Frequency Air Services." *Journal of Public Transportation* II, no. 8 (2005): 17-35.

Lyons, G., Jain, J. and Holley, D. "The use of travel time by rail passengers in Great Britain." *Transportation Research* A, no. 41 (2007): 107-120.

Mandel .B, M. Gaudry, and W. Rothengatter. *A disaggregate Box-Cox Logit mode choice model of Intercity Passenger Travel in Germany and its implications for high speed rail demand forecasts.* The Annals of Regional Science 31(2), 1997, 99-120.

"MapPoint: Business Mapping and Data Visualization Software." Redmond, Washington: Microsoft Corporation, 2004.

Monthly Performance Report. Amtrak, January 2011.

Morrison, S., and C. Winston. "An Econometric Analysis of the Demand for Intercity Passenger Transportation." *Research in Transportation Economics* 2 (1985): 213-237.

N700 Shinaknsen, Japan. n.d. http://www.railway-technology.com/projects/n700-shinkansen/specs.html. (accessed February 02, 2012).

National Household Travel Survey. U.S. Department of Transportation, F.H.W.A, 2001.

Pagliara, F., J. M. Vassallo and C. Roman. "High Speed vs. Air transportation: The Madrid Barcelona Case Study." *Transportation Research Record: Journal of the Transportation Research Board*, no. 0553 (2012).

Poorzahedy, Tabatabaee, Kermanshah, Aashtiani, and Toobaei. "World City Model Choice : Choice of Rail Public Transportation." *Scintia Iranica* II (2004): 320-331.

R Fickling, Mott MacDonald, UK, HGA and TRi Napier University, UK H Gunn, TRi Napier University, UK H Kirby, Mark Bradley Research and Consulting, USA M Bradley, and Accent, UK C Heywood. "The productive use of rail travel time and value of travel time saving for travellers in the course of work." *European Transport Conference 2008.* Netherlands, 2008.

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

Teodorovic, D. *Airline Operations Research* (Gordon and Breach Science), 1998.

Tiller, C, and Y Nottara. *TGV Web.* n.d. http://www.trainweb.org/tgvpages/tgvindex.html. (accessed February 02, 2012).

Trani, A.A., H.Baik, H.Swingle and S. Ashiabor. "Integrated Model for Studying Small Aircraft Transportation System." *Transportation Research Record : Journal of Transportation Research Board* (Transportation Research Board of the National Academics, Washington DC), no. No. 1850 (2003): 1 -10.

Transportation Stastics Annual Report. U.S. Department of Transportation, Bureau of Transportation Stastics, 1995.

Vandyke, A. *Development of a High-Speed Rail Model to Study Current and Future high-Speed Rail Corridors in the United States.* Civil Engineering, Blacksburg: Virginia Polytechnic Institute and state University, 2011.

Yao, Morikawa, Kurauchi, and Tokida. "A study on nested logit mode choice model for intercity high speed rail system with combined RP/SP data." *3rd International Conference on Traffic and Transportation Studies, ICTTS.* Guilin, China: American Society of Civil Engineers, 2002.