Visualizing and Modeling Categorical Time Series Data

Randy Ribler Anup Mathur
Marc Abrams
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106
{ribler,mathur,abrams}@cs.vt.edu

June 15, 1995

Abstract

Categorical time series data can not be effectively visualized and modeled using methods developed for ordinal data. The arbitrary mapping of categorical data to ordinal values can have a number of undesirable consequences. New techniques for visualizing and modeling categorical time series data are described, and examples are presented using computer and communications network traces.

1 Introduction

Visualization tools allow scientists to comprehend very large data sets, and to discover relationships which otherwise are difficult to detect. Unfortunately, not all types of data can easily be visualized using existing tools. In particular, most of the large arsenal of tools that are available for the visualization and modeling of numeric time series data are inappropriate for categorical time series data. This paper identifies the particular problems that are encountered when visualizing and modeling categorical time series, and proposes new methods that are useful in many domains. Examples from computer and communications network performance analysis are examined.

Categorical data is identified by category rather than by ordinal value. The data set can be partitioned into a number of categories. Each element in the data set belongs to exactly one category. For example, gender is categorical data. Each item in a data set of people can be placed in one of the categories from the set {Male, Female}.

Categorical data arises naturally in many types of analysis. The analysis of computer trace files was the original motivation for this work. These files contain time series information describing the *state* of an executing computer program. Some examples of categorical data contained in these traces are:

- the name of the currently executing procedure in a program
- the name of the disk drive in use
- the names of programs loaded in memory
- the name of a remote computer sending information

Trace files provide low-level descriptions of the time dependent state transitions which occur during program execution, but trace files are often very large, and important information may be lost when they are summarized or modeled with statistics such as averages and variances.

A trace file is an n length sequence of ordered pairs $\{T_1, S_1\}, \{T_2, S_2\}, \ldots, \{T_n, S_n\}$, where $T_i (1 \leq i \leq n)$, is a time and S_i is a system state vector which records pertinent system information. To simplify the

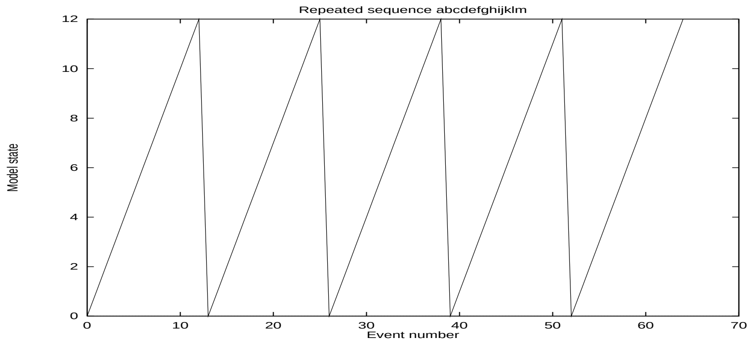

Figure 1: Sequence "abcdefghijklm" with one mapping of letters to numbers

analysis, and to create visualizations that are more intuitive, the system state vectors is usually broken down into its component members, yielding a separate scalar time series for each member of the state vector. In situations where it is important to consider the state vector as a unit, each unique vector is identified by a unique *model state*, which is a single categorical value [ADM92]. In either case, the reduction allows us to analyze a time series of scalar values. We use the term *event* to refer to the individual entries in the time series. We use the term *state* to refer to the category to which an individual event belongs.

2 Problem Statement

2.1 Problem 1: Visualization

Visualization is generally accomplished through mapping data values to a coordinate system, and displaying a 2-dimensional projection of a 2, 3, or more dimensional object. Often the same coordinate system that is used to model a physical phenomenon is used to produce its visualization. Each data point has a spatial relationship with every other data point in the set. This relationship can be expressed as a vector.

Generally, spatial relationships between *categories* of data do not exist or are incomplete. We can not compute a distance and a direction between *names* of things. This limitation renders the majority of conventional visualization techniques inappropriate for use with categorical data.

Imposing spatial relationships through the arbitrary assignment of ordinal values to each category can produce results that are either misleading or not particularly revealing. Figure 1 shows one mapping of a repeated sequence of the states "abcdefghijklm." This mapping is obtained by assigning sequential numbers to each of the letters in the sequence. The repeated ramps seem to match our intuition about how a repeated sequence should look when the letters are mapped to numbers. Figure 2 shows a mapping of different numbers to the same sequence of letters. Although it is clear from both graphs that a pattern exists, the graph in Figure 2 contains a high frequency component that is simply misleading. When there are a large number of unique states in the series, it becomes impossible to distinguish this phenomenon from real oscillations in the data.

The fact that two different mapping can produce such different visualizations is not surprising when we consider that n categories can be mapped to n ordinal values in $n!$ different ways. It seems unlikely that all $n!$ different views will convey the same information equally well.

It is tempting to try to use the view itself as a basis for coordinate assignment. A mapping that minimizes the distance between the state transitions, where distance is defined as the difference between the ordinal values assigned to the successive states, might at first seem desirable. But because the same

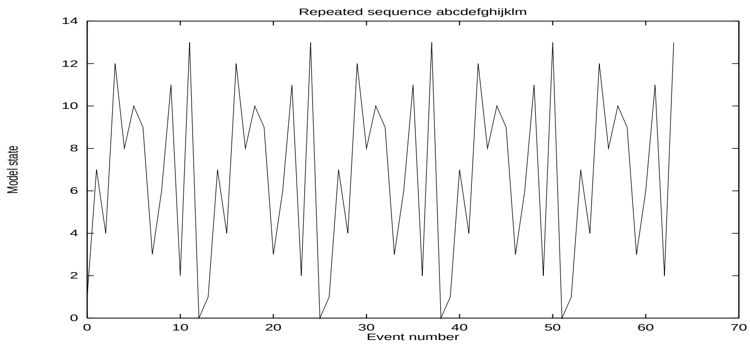

Figure 2: The same sequence as Figure 1, with another mapping

mapping must be used throughout the trace, a mapping that minimizes distances globally may still display very poor performance locally. In addition, it can be shown that determining this mapping, known as Optimal Linear Sequencing, is NP-hard [GJ79].

2.2 Problem 2: Modeling

We can not directly model categorical time series for the same reasons that we can not directly visualize them. That is, conventional models assume that ordinal relationships can be exploited. Another significant problem is that many models of time series data rely on time invariant probability distributions, and yet, it is often important to model and understand the transient behavior present in traces. When we model a trace, we view it as a realization of a stochastic process. In our models, the probability mass functions used to determine the stochastic process are time-dependent.

3 Traces Used for Illustration

Table 1 summarizes the traces used to demonstrate our visualizations and models.

Table 1: Example Traces

Traces		
Trace Name	Number of States	Unique States
Gaussian Elimination	409	31
World Wide Web (WWW) Traffic	229,256	2973
Dining Philosophers	5793	588
Cache Hit Trace of WWW Traffic	37	2

3.1 Gaussian Elimination

The Gaussian Elimination trace was obtained from an instrumented implementation which solves a 64×64 system of equations using 32 processors on an NCUBE multicomputer [RW93]. The system state recorded in the trace is the number of the processor that last sent a message. Process numbers are categorical because their exists no total ordering on their values – one processor is not “less than” another.

3.2 World Wide Web Traffic

The World Wide Web (WWW) traffic trace records the names of files which were accessed on a server used to store educational materials for computer science classes at Virginia Tech. This large trace includes the names of all the files accessed over a period of 75 days [AWA+95].

3.3 Five Dining Philosophers

The five dining philosophers trace was created by a parallel program running on a Sequent Symmetry at Argonne National Labs. The program simulates a problem representing resource contention from operating system design: five philosophers sit in a circle, eat spaghetti, and contend for a limited number of forks. At any point, each philosopher is in one of eight possible states (e.g., eating, or waiting for a fork). Upon a state change, the trace records a categorical data vector of length five, where each element in the vector records the state of one of the five philosophers. Each unique vector is mapped to a unique *model state*, which is a single categorical value [ADM92].

3.4 Cache Hits

Cache (or proxy) servers are used within organizations connected to the World Wide Web (WWW) to keep local copies of frequently used files from remote WWW sites. The cache hit for WWW traffic trace records a "hit" when a requested file is available locally, and records a "miss" when a file must be accessed remotely from the WWW. We use a small section of this trace, displayed in Table 2, to exemplify visualizations and modeling concepts.

Table 2: WWW Cache Hit trace

Event Number	Time	State	Event Number	Time	State
0	451420	miss	20	456250	miss
1	452660	miss	21	456340	miss
2	453270	hit	22	456380	hit
3	453300	miss	23	456850	miss
4	453460	hit	24	456869	miss
5	453690	miss	25	456870	miss
6	454090	miss	26	456871	miss
7	454140	miss	27	456872	miss
8	454170	miss	28	456880	miss
9	454290	miss	29	456900	miss
10	454410	miss	30	457190	miss
11	454510	miss	31	458050	miss
12	454640	miss	32	458490	miss
13	454730	miss	33	459210	miss
14	454750	miss	34	459300	miss
15	454890	miss	35	459320	miss
16	455460	miss	36	459440	miss
17	455480	miss	37	459450	miss
18	455510	miss	38	459490	hit
19	455860	miss			

4 Four Visualizations of Categorical Time Series

Because it is not productive to map categorical data directly to ordinal values, we propose a strategy which identifies numerical relationships between occurrences of categorical values in a time-series, and uses these relationships as a means to visualization. This section describes a number of visualizations which exploit the repetitions of states in the series to produce appropriate visualizations. We first illustrate each visualization with the cahe hit subtrace in Table 2 for pedagogical purposes, then with another trace from Table 1 to illustrate a practical application.

4.1 Cumulative Categorical Periodogram

The periodogram is a popular view for ordinal data. The discrete Fourier transform (DFT) transforms *numeric* time-series data into its component sine waves, which are used to produce the periodogram. But the concept of component sine waves is inapplicable to categorical time series data. We need a different definition of periodicity for categorical time series.

Intuitively, a categorical time series achieves a cycle when it returns to a previously encountered state. The *period* of this cycle can be defined as either the number of intervening events plus one, or as the time between the two occurrences of the state. We refer to the first definition as the *untimed* period, and to the second as the *timed* period.

4.1.1 Untimed Cumulative Categorical Periodogram

Recall that a trace is a sequence, $\{T_1, S_1\}, \{T_2, S_2\}, \ldots, \{T_n, S_n\}$. We compute an *Untimed Cumulative Categorical Periodogram (CCP)* by determining the number of times cycles with each of the possible untimed periods (1 to $n-1$) occurs. The number of occurrences of cycles with period i in a sequence S of length n is given by

$$\mathrm{Occurrences_i} = \sum_{\mathrm{j=i}}^{\mathrm{n}} \begin{cases} 1 & \text{if } S_j = S_{j-i}, \\ 0 & \text{otherwise.} \end{cases}$$

$\mathrm{Occurrences_i}$ can be computed for $1 \leq i \leq n$ in $\mathrm{O}(n)$ time and O(number of categories) space.

Figure 3 shows the CCP generated from the cache hit trace in Table 2. The x-axis represents periods and has domain $[0, n)$. The y-axis represents $\mathrm{occurrences_i}$, for $0 \leq\ i \leq\ n$. The events numbered 3, 4, 5, and 23 contribute to the bar at $x = 2$ in the periodigram, giving it the value 4. Events 22 and 38 are responsible for the bars at 18 and 16 respectively. All other events, with the exceptions of events 0 and 2, contribute to the first bin. Events 0 and 2 are the first occurrences of states and therefore do not represent the completion of a cycle.

Figure 4 shows the CCP generated from the five dining philosophers trace. The x-axis is logarithmic to help distinguish the lower frequencies. The y-axis is logarithmic so that we can clearly see a wide range of amplitudes. This particular graph identifies strong periodic behavior at periods near 6, 30, and 60.

4.1.2 Geometric Distribution of CCP

In uniformly distributed random data, the probability that a given event completes a cycle of length l is pq^{l-1}, where p is the reciprocal of the number of uniformly distributed states, and q is $1 - p$. This means that the periods of the cycles present in random data will have a geometric distribution, where the expected value for the number of occurrences of cycles of length l in a sequence of length n is

$$\mathrm{E(Occurrences_l) = (pq^{l-1})(n-l).}$$

where $\mathrm{Occurrences_l}$ is the number of occurrences of states which recur with period l.

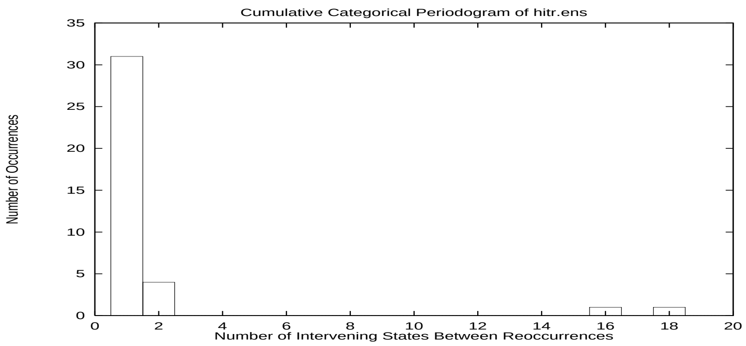

Figure 3: Untimed Cumulative Categorical Periodogram of Cache Hits

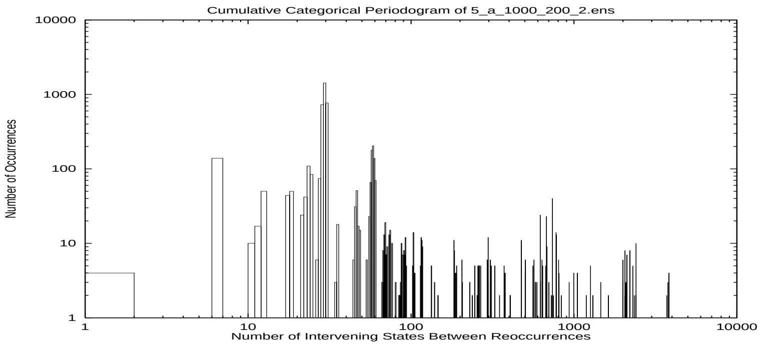

Figure 4: Untimed Cumulative Categorical Periodogram of Dining Philosophers

However, we expect to find strong periodic components in the computer and communications network traffic we analyze, and these periods will tend to be present for contiguous stretches. When periodic components of a trace have period l every lth event will also have period l. The probability of seeing k repetitions of a given period l in random data is:

$$(pq^{l-1})^k$$

This quickly becomes very small as k gets large, so when we see localized periodic behavior in a trace, we can be very confident that it is not from random chance. Still it may be desirable to normalize to correct for this bias toward geometric distributions. This would allow random data to produce a relatively flat curve. Normalization may be accomplished by subtracting the expected value from each bin and dividing by the standard error.

$$\mathrm{NormOccurrences_l} = \frac{(\mathrm{Occurrences_l} - \mathrm{E}(\mathrm{Occurrences_l}))}{\sqrt{\mathrm{Var}(\mathrm{Occurrences_l})}}$$

4.1.3 Timed CCP

The Timed CCP differs from the untimed CCP in that it defines the period of a cycle as the *time* between the reoccurrences of a state. Figure 5 shows the timed CCP generated from the same trace of the five dining philosophers simulation. This view shows that most of the periodic behavior occurs in between 3500 and 5500 time units.

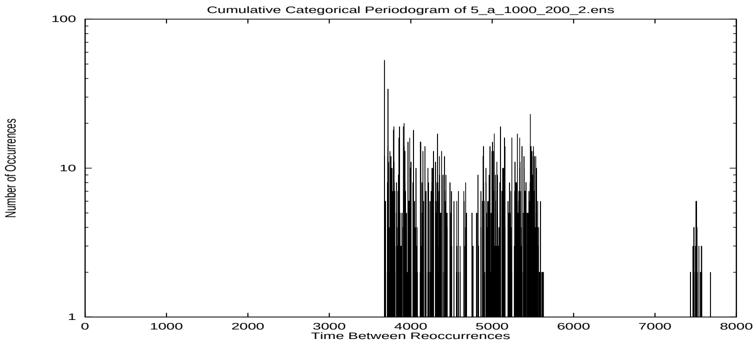

Figure 5: Timed Cumulative Categorical Periodogram of Dining Philosophers

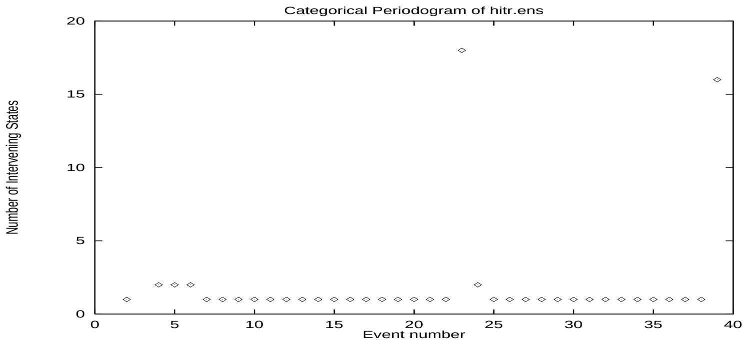

Figure 6: Categorical Periodogram of Cache Hits

4.2 Categorical Periodogram (CP)

The CCP provides a visualization of the total number of occurrences of each period. However, it is often useful to see exactly where the cycles occur in the time series. Figure 6 shows the *categorical periodogram* of our cache hit example. In this view we see not the events themselves, but the period since the previous occurrence of each state. A dot is plotted on the graph for every event that completes a cycle. The x-coordinate of the entry is the event sequence number, and the y-coordinate is the period of the cycle which has been completed. For example, the point $(1, 1)$ is plotted because in Table 2, $S_1 = S_0 =$"miss," thus the period of "miss" at event one is one. The point $(4, 2)$ is plotted because $S_4 = S_2 =$"hit," yielding a period of two. The categorical periodogram is designed to generate horizontal lines across sections of the graph that contain strong periodic components.

An example of a categorical periodogram from trace data is shown in Figure 7, which shows the CP for the dining philosophers trace. Several strong periodic components can be seen. Two central components with periods of approximately 30 and 60 are particularly prominent, while some shorter components at the left and right, which result from processes starting and stopping, also stand out clearly.

4.2.1 Timed Categorical Periodogram

The timed categorical periodogram, like the timed CCP, uses time between repetitions of states rather than number of intervening events to determine period. Figure 8 shows the timed categorical periodogram for the five dining philosophers problem. Several sequences with strong periodic behavior can also be

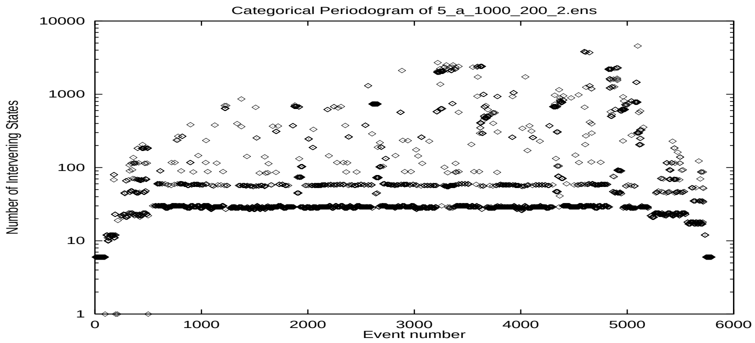

Figure 7: Untimed Categorical Periodogram

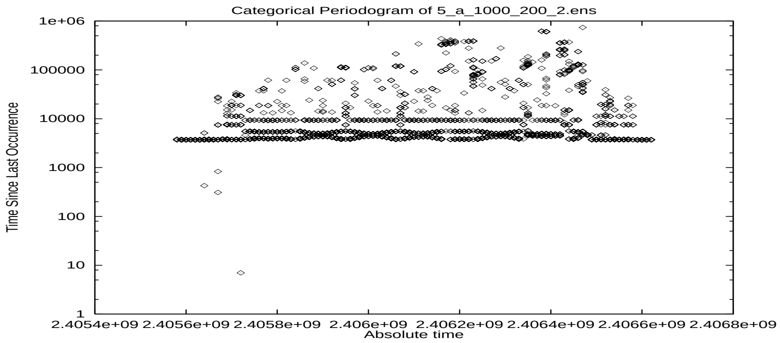

Figure 8: Timed Categorical Periodogram

seen here. In addition, some oscillating features can be seen. When a frequency is present in a trace for a contiguous time interval, *lines* will form from the concentration of plotted points. When these lines appear to move upward it is an indication of periods becoming longer, and when lines move downward, it is an indication of periods becoming shorter. There appears to be a strong relationship between two components in this trace. As the period of one component increases, the other decreases.

Figure 9 shows the oscillating section from the dining philosophers trace in greater detail and with a linear rather than a log y-axis.

4.3 Mass Evolution Graph

It is difficult to identify which states comprise periods of interest using the categorical periodogram. A single horizontal line may be made up of a series of different states. For example, the subsequence "abcabcabc" will produce a solid horizontal line at $y = 3$. If, later in the sequence, these same states are included in a subsequence of period six, there will be no way to identify the fact that these individual states have a new period of reoccurrence. The Mass Evolution Graph (MEG) shows how the periodicity of each state changes in time. The x-axis is the event number, and the y-axis indicates the number of times the state has previously appeared in the trace.

Figure 10 shows the untimed MEG for the sub-trace of Table 2. The two different states (cache hit and cache miss) are each represented by a different line. The ith occurrence of a state is plotted at y-coordinate i. The x-coordinate is the event sequence number. MEG lines that are perfectly straight identify periodic behavior. This can be seen in the segment of the trace that contains only cache misses.

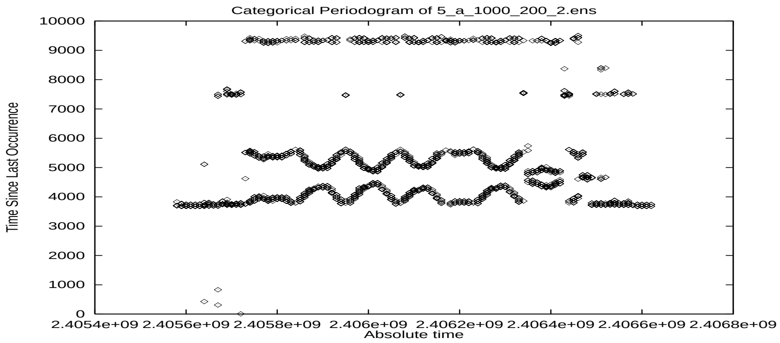

Figure 9: Oscillating Section of Timed Categorical Periodogram

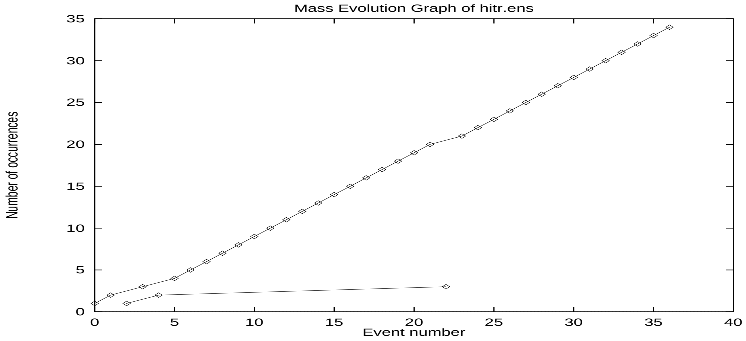

Figure 10: Mass Evolution Graph of Cache Hits

In a system where a state s occurs with a probability $p(s)$, we expect that a linear regression of the s path in the MEG will have a slope equal to $p(s)$. We can construct a MEG using either untimed or timed definitions of period.

The MEG frequently extracts characteristics of a trace that are not easily discernible in any other view. For example, Figure 11 shows that the Gaussian Elimination trace contains a series of distinct *phases*, that is, it has distinct sections of the trace which have different probability density functions. Places in the graph where lines touch the x-axis indicate times when model states appear for the first time.

The slope of the MEG line between points (x_i, y_i) and (x_{i+1}, y_{i+1}) is:

$$\frac{(y_{i+1} - yi)}{(x_{i+1} - x_i)} = \frac{1}{(x_{i+1} - x_i)},$$

which is equal to the frequency, or the reciprocal of the period in the categorical periodogram. So the derivative of the mass evolution graph is equal to the reciprocal of the categorical periodogram. When the i^{th} model state in the sequence appears at period p in the categorical periodogram, the i^{th} model state in the MEG forms a line of slope $1/p$ with the preceding occurrence of that model state.

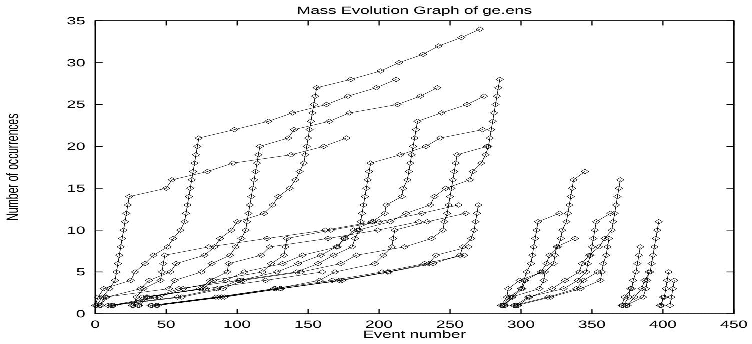

Figure 11: Mass Evolution Graph of Gaussian Elimination Trace

4.4 Pattern Visualization for Categorical Time Series

4.4.1 Trie Visualization

A *pattern* is a sequence of events that consists of a specified sequence of states. The trie visualization is designed to find patterns in categorical time series. Because we do not know what patterns to look for, we need a method that will find *interesting* patterns automatically. Depending on the application, interesting patterns are patterns that occur frequently, infrequently, or within a certain range of repetitions.

If we define k as the maximum length of patterns that we want to detect, we can solve the problem by partitioning the trace into k-space, where a section of the trace: T_i to T_{i+k} is mapped to k-space coordinate $(T_i, T_{i+1}, .., T_{i+k})$. If we use a k-dimensional array s that is initialized to zero, and increment the value $s[T_i, T_{i+1}, ..T_{i+k}]$ each time that the corresponding subsequence is encountered, we will generate a sparse k-dimensional matrix which will contain values other than 0 only in locations corresponding to sequences that occur in the trace. The nonzero entries are the number of times the corresponding patterns occur. We can compute s in O(n) time and space.

Given s we can produce a visualization which shows, for each event i in the sequence, the number of times the subsequence beginning at event i and continuing to event $i + k$ occurs in the entire sequence.

4.4.2 Examples Using Tries

The trie visualization method can be used in a variety of applications. We have used it to analyze World Wide Web traffic, and to detect sections of code that are common to multiple computer programs

We can use the Trie technique to view the repetitions present in the WWW Traffic trace. Figure 12 shows, for each event, the number of times the 14 state subsequence beginning at that event occurs within the entire trace. There are a large number of sequences that are repeated 10-30 times. Surprisingly, there is at least one sequence that occurs over 50,000 times in the trace. This sequence consists of repeated accesses to the same file.

5 Modeling Approach

Having proposed a number of visualizations methods, we next consider the use of one of them (MEG) to generate a model. This section describes how the MEG is used as a basis for modeling. The goal of MEG based modeling is to produce a time-dependent probability mass function (pmf) $f_i(t)$ for each state i recorded in the trace. The first step towards recovering $f_i(t)$ is to plot a curve with the *Number of Occurrences* of state i as the abscissa and *Time* or *Event Number* as the ordinate. The slope of this curve at any instant t is directly proportional to $f_i(t)$. We denote the set of such curves for all unique

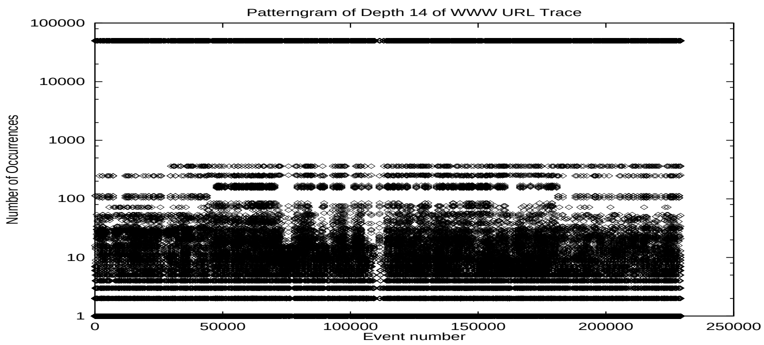

Figure 12: Trie Visualization of WWW Trace

states in the trace as the Mass Evolution Graph (MEG). The next step in our approach is to fit a curve to each path in the MEG using linear regression techniques and compute $s_i(t)$, the time-dependent slope function for each fitted curve. Since $s_i(t)$ represents the proportion of time state i occurs, instead of $s_i(t)$ we use the normalized $s_i(t)$, $s_i^{'}(t)$, defined as follows:

$$s_i^{'}(t) = \frac{s_i(t)}{\sum_i s_i(t)}$$

Thus $\sum_i s_i^{'}(t) = 1$. $s_i^{'}(t)$ is the estimated probability mass function for state i.

Table 3 below shows the normalized estimated pmf for the WWW Cache Hit trace. The columns *From Event* and *To Event* in this table define an event interval for the pmf. The interval is closed at *From Event* and open at *To Event*. The column *P(0)* is the probability of a cache miss during the corresponding event interval in the table. Similarly the column *P(1)* is the probability of a cache hit during the corresponding event interval in the table. The first line of Table 3 says that between event numbers 0 and 2, $p(0) = 1$ and $p(1) = 0$.

Table 3: Model for the Cache Hit trace

Estimated pmf			
From Event	To Event	$P(0)$	$P(1)$
0	2	1.00	0.00
2	5	0.50	0.50
5	6	0.90	0.10
6	22	0.95	0.05
22	26	1.00	0.00

The steps in the process of building a MEG model from a trace are discussed in detail below.

5.1 Partitioning Paths in the MEG

A path i in the MEG is a realization of the time-dependent pmf, $f_i(t)$, for the corresponding state. Our goal is to obtain an estimate of this pmf. As discussed in section 5 above, the pmf for a state i at event t is directly proportional to the slope of the corresponding MEG path $s_i(t)$ at that event. This holds true for all events except where the time-dependent pmf $f_i(t)$ is not differentiable. Denote such events as *partitioning events*. Partitioning events define the boundaries of *event-intervals* such that:

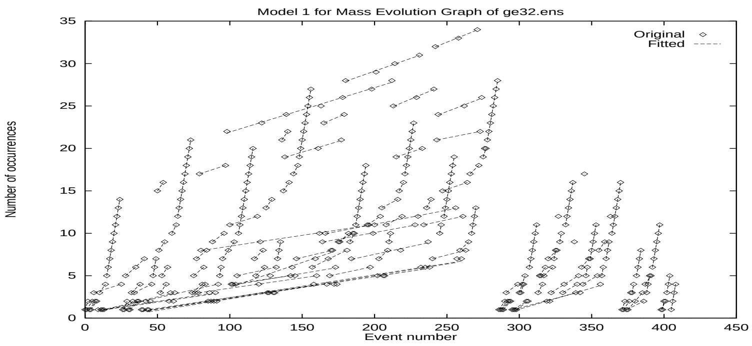

Figure 13: MEG of the Gaussian Elimination Trace After Fitting Straight Lines to Partitioned Segments

- during each event-interval the function $f_i(t)$ has a constant value, and
- for each consecutive event-interval the function $f_i(t)$ has a distinct value.

To estimate the function $f_i(t)$ we must determine all the partitioning events for it. In a MEG path, events where the slope of the path changes significantly are strong indicators of a partitioning event for the corresponding state. In this step we determine the partitioning events for the pmf of a state by partitioning the corresponding MEG path into segments of nearly-constant slope. In [MA95] we describe two algorithms that form a UNIX-style pipeline to partition a path in the MEG into segments having a constant slope. The first algorithm computes the slope at every event for each path in the MEG. The input to this algorithm is the MEG. The second algorithm partitions each path in the MEG into segments of constant slope. The input to this algorithm are the MEG and the output from the first algorithm.

The partitioning algorithm inspects the slope at each consecutive pair of events in a MEG path. If the slope at event E_i differs from the slope at the immediately previous event E_{i-1} by more than the user defined parameter $threshold$ then the event E_i is a partitioning event. Another user defined parameter $runlength$ enforces a lower bound on the number of events in a segment (e.g., if $runlength = 2$ then a segment must have at least two points).

5.2 Fitting straight line segments to MEG paths

This step can be described by the following simple algorithm:

```
for every path i in the MEG {
   for every partitioned segment j in path i {
      fit a straight line to segment j using linear regression;
   } /* endfor */
} /* end for */
```

Figure 13 shows the MEG for the Gaussian Elimination trace after each path in it has been partitioned (using the approach described in section 5.1) and each partitioned segment has been fitted with a straight line.

Note that a single straight line can be fitted to the entire MEG path. This saves us from the partitioning step described in the previous section (section 5.1). However, this usually leads to a less accurate model.

5.3 Generating the Model

Generating the model after fitting straight lines to segments involves inferring an empirical probability mass function using the fitted lines. To illustrate the mechanism for generating the model we start with a simpler scenario, namely a single straight line is fitted to every path in the MEG output. Recall that every path in the MEG output corresponds to a state in the trace. Also the slope of a MEG path is proportional to the pmf for corresponding state. Using the above we can derive the probabilities for the simple scenario by using the slopes of the straight lines fitting each path in the MEG output. Recall that we already have these $FittedSlopes$ from the previous step described in section 5.2.

The same idea is extended to MEG paths that are partitioned into segments. Consider the scenario where a MEG path i (corresponding to state i) is partitioned into k segments and a straight line is fitted to each of these k segments. From the $FittedSlopes$ of each of the k segments we can build a time-dependent estimated pmf for state i. Applied to all paths (and hence all states) in the MEG output this generates an estimated time-dependent probability mass function for all states.

6 Model Validation

This section describes how we validate the MEG model. We also demonstrate how well our model compares to a *time-homogeneous* semi-Markov model that does not take into account the time-varying behavior of one trace: Gaussian Elimination.

6.1 Tests for Model Validation

We use two tests for model validation – autocorrelation and homogeneity.

6.1.1 Autocorrelation

An important guide to the properties of time-dependent data is provided by a series of quantities called sample autocorrelation coefficients, which measure the correlation between observations of a system at different instances of time. Denote the state of a system S at time t to be X_t, and the state of the same system at time $t+k$ to be X_{t+k}. Then the autocorrelation of S at lag k, denoted $r(k)$, is defined in terms of all recorded $X_i s$ which are k time units distant from each other. By varying the lag k we can plot the autocorrelation function $r(k)$ versus k to generate a *correlogram*. The correlogram provides an insight into the probability model which generated the data. We use the correlogram as a validation test by generating synthesized traces from our model, and then comparing correlograms obtained from our traces to the correlogram obtained from the original trace.

6.1.2 Test of Homogeneity

Another validation test we use is the test of homogeneity. It is a statistical test which tests the following null hypothesis:

- H_0 : A trace synthesized from our model is generated from the same distribution that generated the original trace.

We use the Kruskal-Wallis rank sum test with correction for ties [Ott88]. The Kruskal-Wallis test is non-parametric test, so we do not need to make any distributional assumptions about the trace data.

To validate our model using the Kruskal-Wallis test we first synthesize traces from our model. A synthesized trace is then compared with the original trace using the Kruskal-Wallis test. To make the test more rigorous we partition the synthesized and the original traces into ten partitions and then apply the Kruskal-Wallis test to each such partition in the synthesized trace and the corresponding partition in the original trace. Denote by n the number of partition pairs for which H_0 is not rejected. If $n = 10$ then all ten partitions of the synthesized trace match their corresponding partitions in the original trace.

6.2 Validation Results

The results from the validation tests for the model generated for the Gaussian Elimination trace are described below.

6.2.1 Plot of the sample autocorrelation

The sample autocorrelation plot for the original Gaussian Elimination trace (*ge32.original*), a trace synthesized from the MEG model (*ge32.MEG*), and a trace synthesized from the semi-Markov model (*ge32.semi-markov*) is shown in figure 14. Figure 14 clearly shows that the trace ge32.MEG follows ge32.original very closely (especially when compared to the trace ge32.semi-markov).

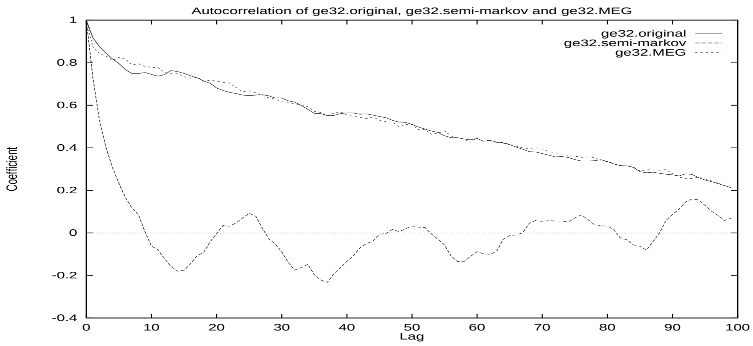

Figure 14: AutoCorrelation Plot for the Gaussian Elimination Trace

6.2.2 Results of the Kruskal-Wallis test

Table 4 shows the results of the Kruskal-Wallis test on original Gaussian Elimination trace (*ge32.original*), a trace synthesized from the MEG model (*ge32.MEG*), and a trace synthesized from the semi-Markov model (*ge32.semi-markov*). As shown in Table 4 only 5 (out of 10) of the trace ge32.semi-markov pass the Kruskal-Wallis test when compared to ge32.original. In a similar test all 10 segments of the trace ge32.MEG passed.

Table 4: Results of the Kruskal-Wallis test for the Gaussian Elimination Trace

Kruskal-Wallis test results ($\alpha = 0.05, \chi^2 = 3.84146$)			
Model	No. segments passing the test	Mean of H'	Standard Deviation of H'
Semi-Markov	5 of 10	25.12	27.67
MEG	10 of 10	0.34	0.38

7 Conclusions and Future Work

In this paper we have demonstrated a number of visualization techniques for categorical time series data, and a modeling approach which can build accurate models of time-dependent data. These techniques can be used to help draw insights from very large data sets.

We are currently developing methods for categorical correlation. Correlation for numeric data is based on the *strength of the relationship* [Ott88] between an independent and dependent variable. So for example, if the dependent variable gets larger when the independent variable gets larger, there is said to be a strong correlation between the two values. Categorical data does not share this concept of larger and smaller values (which is larger "a" or "b?") so another definition of correlation is required for categorical data. The definition should reflect our strong intuitive feelings about correlated categorical data. For example, it seems obvious that two sequences, "abcdefg" and "abcdefg," should posses the highest degree (i.e. perfect) correlation. Similarly "abcabc" *seems* to be correlated to "xyzxyz," but *seems* to be more highly correlated to "cbacba" and also correlated to "abcabd." These concepts of correlation depend not on the one-to-one relationship between independent and dependent variables, but on the similarities in *sequences* of independent/dependent variables.

Our intuitions, though strong in some instances, seem weak when we ask questions like, "Is 'abcabc' more highly correlated to 'abcefg' or 'cbacba?'" These questions are difficult because they compare different aspects of our intuitive idea of correlation. For example, "abcefg" matches "abcabc" for the first three elements in the sequence, but "cbacba" has a one-to-one matching for each element in the sequence and it also uses the same alphabet.

These different aspects suggest that the correlation should be a *vector* rather than a single value, where each element of the vector corresponds to particular aspects of categorical correlation. We are implementing and testing a number of categorical correlation measures as candidate members for a correlation vector.

Because it is risky to draw conclusions about a population of traces from a single trace, all of these techniques are currently being expanded to support *ensembles* of data. Many of the visualizations are also being expanded to provide 3-dimensional views of the categorical attributes that we have described.

All of the views and models described in this paper have been incorporated into *Chitra95*, a third generation performance analysis and visualization tool developed at Virginia Tech [ALG94, ABRV94], available via WWW from http://info.cs.vt.edu/chitra.

References

[ABRV94] M. Abrams, A. Batongbacal, R. Ribler, and D. Vazirani. CHITRA94: A tool to dynamically characterize ensembles of traces for input data modeling and output analysis. Technical Report TR 94-21, Computer Sci. Dept., Virginia Tech, Blacksburg, VA 24061-0106, June 1994. Available from World Wide Web location http://info.cs.vt.edu/chitra.

[ADM92] M. Abrams, N. Doraswamy, and A. Mathur. Chitra: Visual analysis of parallel and distributed programs in the time, event, and frequency domain. *IEEE Trans. on Parallel and Distributed Systems*, 3(6):672–685, November 1992.

[ALG94] M. Abrams, T. Lee, and K. Ganugapati. Constructing software performance models from trace data with Chitra92. *submitted to Software – Practice and Experience*, February 1994. Computer Sci. Dept., Virginia Tech, TR 94-07.

[AWA+95] M. Abrams, S. Williams, G. Abdulla, S. Patel, R. Ribler, and E. Fox. Multimedia traffic analysis using chitra95. Technical Report TR 95-05, Computer Sci. Dept., Virginia Tech, Blacksburg, VA 24061-0106, May 1995. Available from World Wide Web location http:// succeed/95multimediaAWAFPR/95multimediaAWAFPR.html.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[MA95] A. Mathur and M. Abrams. Time dependant rate-based modeling of trace data. Technical Report TR-95-xx, Computer Sci. Dept., Virginia Tech, Blacksburg, VA 24061-0106, May 1995. Available from World Wide Web location http://info.cs.vt.edu/chitra.

[Ott88] L. Ott. *An Introduction to Statistical Methods and Data Analysis.* PWS-Kent, Boston, 3rd edition, 1988.

[RW93] Diane T. Rover and Abdul Waheed. Multiple-domain analysis methods. In *ACM/ONR Workshop on Parallel Debugging and Performance*, proceedings appeared in *ACM SIGPLAN Notices,* 28(12), Dec. 1993, pages 53–63, San Diego, May 1993. ACM.