Abstract

This paper discusses using preprocessing of multivariate data into projections converted to images and movies to provide graphical displays for large quantities of data. The views are augmented with interaction so that the user can paint areas and have small subsets of the full data set overlaid as real-time graphics. Looking up the full data set is achieved by an indexing of the projections that is created in the data pre-processing. An application of the methods on multivariate spatio-temporal data, seasonal metric data from the United States Geological Survey, is discussed.

1 Introduction

Large spatio-temporal data sets are being compiled to study global climate and environmental conditions across the globe. A lot of data is being collected with the support of the United States government, and consequently, it is being made available to the general public: See, for example, http://www.pmel.noaa.gov/tao/, http://www.usgs.gov/pubprod/ or http://www.ucar.edu/fac_data/data-resources.html.

One of the hurdles to using existing graphics tools on this data is the size. For real-time visualization, large data presents some challenges:

1. the views need to be created rapidly enough to be displayed in fractions of a second, especially for dynamic graphics;
2. a user needs to be able to interact with the views and have the response occur in fractions of a second;
3. the screen real estate is limited and with large data many cases could map to the same pixel;
4. some common case reduction methods for multivariate real-valued data while sufficient for mean and variance analysis are inadequate for visualization where the task is to find anomalies, rare events, uncommon relationships.

Thus the objective is to produce visual tools which provide displays swiftly, allow the user to interact with the view, and to maximize the amount of data represented in the limits of screen resolution. For this paper the goal is to demonstrate this is achievable for a million cases of multivariate data.

As far back as 1990 Dr John McDonald, at the University of Washington, demonstrated a software system for interactively visualizing with a million points. He could rotate 3 variables by generating images of the projected data: the rotation started slowly and discontinuously, one view, next view, next view, stepwise as the images for the 3D projection were created, and then scream along at break neck pace once the full set of images for 360° were created. He could brush in a scatterplot linked to other scatterplots: as the brush moved no updates were made, but once the mouse button was released the points would be highlighted. The scatterplots were displayed as pixel resolution, grey scale images, and color was overlaid in layers for each brush action. He used his software to examine data generated from remote sensing instruments, that is, images representing a spectral band. This type of data typically comes on the order of a million points: $1024 \times 1024 = 1048576$. He never published this work.

The research reported in this paper re-visits McDonald’s work, adding methods and software for visualizing large, multivariate space-time dependent data. It is organized as follows. The rest of this section introduces notation for the data, describes visual methods commonly used for small data sets, and provides more information on related work. Section 2 describes our approach to scaling methods to a million cases. Section 3 overviews the software that we have developed to test our approach, and section 4 applies the methodology on seasonal metric data from the United States Geological Survey (USGS).

1.1 Type of data

The data is real-valued multivariate data having at least a million cases (instances, examples) but probably less than 15 variables (features, attributes). In some examples there may be associated class information, such as an estimate of land use, and there may be associated time and spatial information.

In mathematical notation we expect to have data of the form

$$X = \begin{bmatrix}
  x_{11} & x_{12} & \ldots & x_{1p} \\
  x_{21} & x_{22} & \ldots & x_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \ldots & x_{np}
\end{bmatrix}_{n \times p}
\quad =
\begin{bmatrix}
  \text{tuple}_1 \\
  \text{tuple}_2 \\
  \vdots \\
  \text{tuple}_n
\end{bmatrix}
$$

where $n$ is the number of cases, and $p$ is the number of variables. The associated class vector may be denoted as $C = \begin{bmatrix} c_1 & c_2 & \ldots & c_n \end{bmatrix}^\prime$, the time context as
where \( A \) is an orthonormal matrix, that is the columns are vectors of length 1 and each is orthogonal to all other columns.

### 1.2 Visual methods for small datasets

There is a lot of choice in graphical methods for displaying aspects of high-dimensions: tours (Asimov 1985, Cook, Buja, Cabrera & Hurley 1995, Wegman 1991), scatterplot matrices (Carr 1985, Becker & Cleveland 1987), parallel coordinate plots (Inselberg 1985, Wegman 1990), and conditional plots (Becker, Cleveland & Shyu 1996). Scatterplots of two real-valued variables underly many of the above methods, and they can be denoted as projections, projecting the data from \( p - D \) to 2-D.

In addition, direct manipulation between multiple plots provides ways of linking information, providing more insight into high-dimensional structure. The most common linking method is linked brushing, where highlighting features in one rendering similarly highlights the corresponding features in the other renderings. Linking several marginal views allows the viewer to explore conditional dependencies amongst variables, for example, if \( X \) is less than 10, then what is \( Y \). Figure 1 demonstrates linking between several views.


### 1.3 Why scale visual methods?

Visual methods provide different information about data structure than provided by numerical methods. We can think of visual displays as complex summaries of data that may be digested easily by the eye. Figure 6 gives an elemen-
Figure 1: Linking between several views: (left) brushing on class value, (middle) scatterplot of two variables, (right) tour projection.

Other example: four very different bivariate distributions of points that have the same correlation value. All four data sets can be described as having a positively linear association, but the deviations from this trend differ dramatically between data sets: (left) a random spread of values around the linear trend, (left middle) the trend is due to one single extreme case, (right middle) the data composed of six subgroups all of which have negative linear dependence, and (right) the horizontal variable takes discrete values. For data analysis it is good to have numerical summaries to succinctly describe the major trends and patterns, and visual methods to explain the nuances and anomalies in the data.

1.4 Previous work on scaling up multiple views for larger datasets

Dan Carr has published several articles on large data visualization, mostly providing static binned views such as a scatterplot matrix (Carr 1991). Wegman, Poston & Solka (1998) developed a “pixel tour” for displaying (multivariate spatial-context) spectral data, in which sequences of 1-dimensional projections of the multiple spectral bands are displayed as grey scale images. This is very useful for detecting major events in the spatial domain such as mines in a minefield.

2 Beginning to work with a million cases

2.1 What happens in a real-time system?

It is possible to get a million cases of multivariate data into the software system GGobi (at least on a linux system). The startup time is long: it took about 7 minutes. But once the data is imported into the system many operations are fairly snappy. There are a few modifications to the usual user behavior that accommodate viewing this many cases:

- Use a pixel sized glyph.
• Keep the brush off while positioning it in a plot, and turn it on when ready to paint a subset of the data.

• Keep the tour paused because the re-plotting is too slow to give smooth continuous motion. Instead use the scramble button to view random projections of the data.

• Avoid methods that use texture and ASH plots which involve binning the data. Binning is too slow in real-time.

• Keep the number of open displays to be small.

2.2 Reducing the number of cases

There are two possible directions for reducing the number of cases: working in the multivariate space \((p-d)\), or the imposed binning in the screen real estate, on the projected data. Figure 2 illustrates two types of binning in the multivariate space and binning projected data.

Figure 2: Illustration of three different types of binning for 3D data: (Left) Rectangular binning in the multivariate space, (middle) binning using a clustering algorithm, (right) binning of the projected data.

There has been considerable research into case reduction in the multivariate space. Techniques that are commonly in use are:

• **binning**: cases are gridded into cells (Figure 2 left), then the points in the cell are represented by the bin center or the average of the points in the cell, and a count of how many cases are in the cell. Using the average value better reflects the data distribution within the cell when the cases are non-uniformly distributed. Binning is inefficient when there are more than several variables, the number of bins needed is exponential in the number of variables, if each variable is binned into 10 cells for 2 variables there are \(10^2\) cells, for 10 variables there are \(10^{10}\) cells. It also creates visual artifacts, grid patterns, which can distract the user from seeing the structure of the data. There is also serious potential for loss of resolution with binning.
• **clustering:** a cluster algorithm is run, most usually $k$-means with $k$ set to be very large (Figure 2 middle). Then cases are binned into their closest cluster mean, and each case has an associated weight representing the count of the number of cases in the bin. (See for example Posse (2001).)

• **data squashing:** data is binned as a first step, then statistics are computed on each bin, e.g., mean, variance/covariance. These statistics are then used to generate a new sample in each bin to construct a reduced case data set (DuMouchel, Volinsky, Johnson, Cortes & Pregibon 1999).

With any of these methods the resulting data set is of the form:

$$X^* = \begin{bmatrix}
  x_{11} & x_{12} & \ldots & x_{1p} \\
  x_{21} & x_{22} & \ldots & x_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n^*1} & x_{n^*2} & \ldots & x_{n^*p}
\end{bmatrix}_{n^* \times p}, \quad W = \begin{bmatrix}
  w_1 \\
  w_2 \\
  \vdots \\
  w_{n^*}
\end{bmatrix}.$$

where $n^* < n$ and each case has an associated count or weight, $w_i, i = 1, \ldots, n^*$, corresponding to the relative representation of this combination of values in the full data set. These techniques were explored in (Han 2000), and we were not happy with any of the results. Binning may be appropriate for numerical analysis, when a mean or variance analysis is the focus but when finding anomalies and deviations from a trend the artificial cluster structure induced by the binning confounds the task (Figure 3).

![Figure 3: Two views of 5-dimensional data which has been binned to 30(!) cells per variable. There is artificial clustering simply due to the binning.](image)

To deal with the shortcomings of the techniques to reduce the size of the multivariate space we have concentrated on the binning induced by the screen real estate for each projection (Figure 2 right). The binning will be different for each projection, but this form of binning also induces a reduced size structure on the data. The binning induced by the screen real estate essentially results
in case reduction for each projection: Screen horizontal position, $S_x$, and screen vertical position, $S_y$, and a count or weight corresponding to the number of points plotted at the pixel, $W_{S_x,S_y}$. (We also denote an ID list, $P_{x,y,j}$, $j = 1,\ldots,\#\text{number of points at pixel } S_x, S_y$, to refer to the row value of the original data matrix.) For example, if $n = 1\,000\,000$ and the display window is $200 \times 200$ then there at most $40\,000$ plotting positions, so that the data is reduced from $n = 1\,000\,000$ to $n^* \leq 40\,000$ in each projection. And multiple projections of the data are created. We want to utilize this structure to build a fast lookup system for linking points between plots. Figure 4 illustrates the data projection into a window on screen: the raw data is standardized and projected and then again scaled to get it represented in a screen position.

![Sample Data Tuples](image)

Figure 4: Data is projected into pixel locations in a screen on the window. Black represents a pixel that contains at least one data point, grey represents no data at the pixel. The index is built only on pixels containing data.

The indexing has a second purpose, that is, to render the projected data as a density plot. Its possible to create some undesirable visual artifacts in doing this. It may be easier to explain by considering an analogy with histograms of 1-dimensional data. There are two parameters used to build these plots: shift and bin size. Shift is the numerical value that represents the start of the binning, usually the minimum data value. Bin size carves up the range of data values. Changing these two parameters can change (sometimes dramatically) the appearance of the histogram. These parameter values are fixed by the screen real estate rather than freely varied, but they should be considered as input parameters to the display. The next section illustrates some problems with using density displays.
2.3 Density estimation

The binned data projection is represented as a density plot where the bin count for each pixel is coded as a grey scale value. The transfer function we use is \( \text{grey scale} = 255 \times \left( \frac{\text{count}}{\text{maxcount}} \right)^p \). Figure 5 shows the function for several different power, \( p \), values. The reason to use a power transfer function is that the human eye cannot readily discern differences between adjacent grey values on a linear scale. We arrived at this transfer function after examining the techniques discussed in (Huang, McDonald & Stuetzle 1997) and the lecture notes for Purdue University course EE637 Digital Image Processing I http://dynamo.ecn.purdue.edu/~bouman/ee637/. A value of \( p = 0.2 \) is suggested and this is what we use for the images and movies in Limn.

The power transfer function effectively reveals more perceptible resolution of the data density in regions of high density, while also rendering the extreme points in dark ink. This can be seen in Figure 6: the highest row of density plots uses a linear transfer function, and the effect of “lightening” up the high density region as the power value decreases can be seen going down the page. The function renders low density as black and high density as white – similar to what we might physically see by flying over snow-capped mountains. Zero count is also rendered as white. In general this is not a problem in tour views especially as the high-density regions (white) are separated from the zero count (white) by low density pixels (black). And the dark ink for the low density regions draws the eye to the extreme anomalies – what we want to detect using graphics.

For multiple projections it may be useful to use the maximum bin count over all the projections, rather than the maximum bin count in a single image, to scale the bin counts. This will be particularly important for maintaining smoothness of grey scale from image to image in a tour movie sequence.

We encountered two problems with the density plots: both are evident in the examples in Figure 6. When the bin count is small and the max count is large the bin may get rendered as a zero count cell. This happens with the linear transfer function for the example which has a single extreme value (second left column). The other transfer functions correctly display the cell as having a non-zero count. The second problem occurs when a variable takes integer values with the total number being less than the screen real estate size (right column). The result is that strips of white space are visible periodically, a very undesirable effect. These problems are similar to those encountered when using histogram displays of single variables.

2.4 Screen real estate indexing

A three level index is created for each stored data projection. In its simplest format, the index consists of three files.

- The Sx index file stores \( S_x \) coordinates of those non-empty pixels by increasing order, and pointers to the starting position of the corresponding Sy-blocks in the Pixel index file.
Figure 5: Transfer function used to convert count to a grey scale: three different power function, and dotted line represents linear transfer function.

Figure 6: Four very different bivariate distributions that have the approximately the same correlation, and their binned views using different power transfer functions: (second top) linear, (second bottom) power = 0.6, (bottom) power = 0.3.
• The Pixel index file stores $S_y$ coordinates of those non-empty bins, and pointers to the starting position of the corresponding ID list in the IDList index file. Each block of $S_y$ coordinates are ordered by their increasing order.

• The IDList index file stores ID numbers that point to the raw data tuples that projected to a point in this plot image.

For a small data set, we can store a full ID list for each non-empty bin of a binned projection. But for large data sets, with millions of tuples, space considerations make it impractical to maintain such a full index. To make use of the index for large data sets, it is necessary to reduce the size of the ID lists while maintaining as much information as possible. Two approaches have been used in Linn to this end. First, the complete data set is used to calculate the count, mean, minimum, and maximum of the set of tuples that plot to a given point. The statistics are stored in a fourth file that is linked back to the index files allowing Linn users access to the statistics during Linn execution. Second, sampling of the complete set of tuples that plot to a bin is used to provide access to a reasonable set of tuples that represent the point. In the current prototype random sampling is being used. Future development plans call for other methods of sampling to be evaluated as Linn testing continues. A block diagram of the current index structure is shown in Figure 7.

3 Software system

The Linn systems approaches the performance issues associated with real time viewing of large data sets by preprocessing the data when possible. This preprocessing can take the form of both indexing the data for faster searches and by generating sets of images of the data that provide an animated backdrop where dynamic graphics can be generated in front of the images. The system is designed as a pluggable framework that allows new modules to be easily incorporated into the data preprocessing and viewing stages. Current modules provide support for reading in data from multiple sources, creating images and their indexes to the original data for a scatterplot matrix, and viewing programs to display the images. A tour animation sequence can be built as a quicktime movie or assembled from a collection of image files. The View classes can be used to view the images and interact with them.

Functionally it reduces to two main activities: preprocessing the data and viewing the results. Figure 8 shows roughly the structure of the two systems.

• The Linn program controls processing the data into projections, indexing the projections, and if an animation sequence is to be the result then a quicktime movie is created. A properties file is used to specify animation creation and appearance attributes. The program creates the tour engine, rendering and recording type objects that are specified in the properties file and starts animation cycle that the modules comprise. If a scatterplot
Figure 7: Diagram illustrating the three layer index files structure.

Figure 8: Diagram illustrating software structure (left) Limn) (right (View).
matrix is to be the result the Linn program stops at simply building the index, and leaves the image rendering to the View program. For large data sets, the indexing routine will keep only a sample of indexes for the points for each pixel.

- The View program will display a scatterplot matrix or a tour animation that allows the user to control speed and direction of the playback. Additionally users can interact with the display while an animation is running. By brushing on areas plot, data points will be selected and the viewer will draw graphics around each point to highlight it. Because the highlight graphics are drawn on top of the scatterplot image performance will begin to lag as the subset size gets larger.

Users interact with a scatterplot image by brushing areas on the plot and have the corresponding subset overlaid on other plots as real-time graphics. When a bounding box is brushed on a scatterplot by a user, the bounding box determines the range of $S_x$ and $S_y$. The $S_x$ index file makes use of the $S_x$ range to determine the relevant $S_y$ lists in the Pixel index file. The Pixel index file then match the $S_y$ range only to those $S_y$ values that satisfied the $S_x$ range criteria to determine the ID lists in the sample IDList index file. The tuples from the sample list for each relevant scatterplot point are retrieved. The resulting data can then be brushed into the other scatterplots of interest to the user. The result is that the brushing technique can be applied to much larger data sets. In addition to using the sample lists, users can also make use of the statistics to brush other interesting scatterplots.

4 Application

4.1 Data description

The U.S. Geological Survey’s (USGS) EROS Data Center produces biweekly composite images of Normalized Difference Vegetation Index (NDVI). NDVI is a spectral index based on the surface reflectivity estimated by the first two spectral bands of the Advanced Very High Resolution Radiometer (AVHRR) which operates on several National Oceanographic and Atmospheric Administration (NOAA) satellite platforms. This combination is used as an index to measure the health and density of plant biomass.

The data sets include estimates for the start, end, and peak of the growing season for each square kilometer of the U.S., as well as measures for biomass at those times and net productivity over the growing season. The value of this data set is that it can used to classify land cover by how it functions, not just how it appears.

The seasonal metrics used in this study are: SoST (Time of the start of the growing season), SoSN (NDVI at the start of the growing season), EoST (Time at the end of the growing season), EoSN (NDVI at the end of the growing season), TotalNDVI (Integrated NDVI over the season - an indicator for net
productivity). In addition each spatial region is classified into one of 159 EPA ecoregion. We are interested in agricultural use and use 15 classes (11/17 is corn/soybeans, 56 is grasslands, 94 is deciduous forest, 139 is coniferous forest, 155 is barren ...) at each spatial location, and measurements for several time periods. We use data from 1989 for this example, which comes to a little over a million points.

The primary objective of the data collection is to monitor the health and density of the nation’s biomass over the growing season, and also to examine land cover class in relation to these seasonal metrics.

The data arrives as multi-band, 16-bit, binary flat files with each band corresponding to a different year and each file containing a single variable. Each band in this data set can be thought of as an image map of a single variable for each year. Figure 9 demonstrates a three-year binary file for Start of Season Time. The data input capabilities of Limn are limited to reading ASCII columns of text where each column represents a variable, and each row represents a single latitude-longitude pair for a single year. We preprocessed the data using other software to get it into a format that Limn could use.

Figure 9: Representation of three year binary file for the Start of Season Time.

4.2 Viewing a tour of the data

Figure 10 displays some snapshots from the tour animation. The top left display shows the full window with the user beginning to brush, and the right top shows
the results of the brushing (green). The plots at bottom shows a sequence of three tour projections, where the subset is displayed as real-time graphics on the movie images. What can be seen? In the high-density region there is substantial multimodality which most likely corresponds to different land use types. Around this region there are numerous singleton or low density pockets. The brushed subset is one such pocket of points. As the tour progresses this group spreads out, so it contains spatial locations that have similar extreme values in two variables but quite different values on other variables: that is, it is not clustered tightly in the 5-D space.

In the same way, many of these anomalies on the fringe of the scattercloud can be investigated. Small subsets brushed and watched through the movie, then turned off and another group brushed and watched. Could these be plots of land which are growing some unorthodox crop, or is it a problem with the data cleaning procedures?

4.3 Scatterplot matrix

Figure 11 displays the scatterplot matrix of the class variable and the seasonal metric data. What can we see from this?

In the figure the classes corresponding to corn/soybean (blue) and grasslands (pink) are brushed. There are substantial differences between these groups. One occurs in the SoSN and SoST plot (second row, right plot): grassland has relatively high SoSN (start of season NDVI) whereas corn/soybeans have a very diverse distribution in these two variables. There is also a difference in the EoSN, again grasslands have a fairly concentrated moderate value for EoSN. The distributions of the MaxT and MaxN take primarily discrete values.

Overlaid on the matrix is a snapshot of the user controls. When the user brushes in an image a subset control tab is added to the control panel, and also a subset information frame is added to the information panel. The user can then control whether a sample from each pixel location, or the full data, or statistics such as the mean is plotted over the images. The user can also choose a different color to represent the subset.

5 Current and future developments

This paper has described building graphics on a data set containing a million cases of 7-dimensional measurements. About a year ago we drafted a paper on the project and in the future work section we wrote that one pressing issue was to migrate from using the full data set to a screen real estate binning and indexing of the full data, and also, that it would be fun to experiment with overlaying multiple movies. Both ideas push the code further to function with larger data sets, and both are now implemented (although the movie overlay work is not reported on in this paper). So let’s take a step back and think of where the approach needs improving, where the software needs developing and what might the tools be useful for in the real world.
Figure 10: Sequence of tour views demonstrating brushing a subset and subsequent touring. The anomalous points on the fringe of the point cloud are the interesting ones to investigate. One such group has been brushed here. It spreads out and separates into smaller groups in other projections, so it is not a cluster in 5-D space but rather in perhaps 2 of the variables only. Other small anomalies can be similarly investigated, brushed, watched through the movie, turned off, new set brushed... Could these anomalies be plots of land which are growing some unorthodox crop, or is it a problem with the data cleaning procedures?
Figure 11: Scatterplot matrix of class and seasonal metric variables.
5.1 Improving the methods

With large data there are too many points represented at each pixel to store and overlay on other projections so we plan to explore sampling schemes for the pixel populations. We currently use a threshold number, and keep all pointers to the raw data up to a fixed sample size. Thus pixels where few points are represented maintain the full list of cases, but where there is high density a fixed number of randomly sampled points are kept.

5.2 Software

Large data sets are often stored as specially structured binary files, for example, netCDF (www.unidata.ucar.edu/packages/netcdf/), hdf (hdf.ncsa.uiuc.edu/), or some type of image format. We currently extract the data out of this format with other image manipulation tools and store it in ascii files. This is laborious and space-consuming. It makes more sense to develop data parsers to work with the binary formats.

The data sets still will be too large to operate the index pre-processing on the entire data set simultaneously. The index pre-processing will need to be run on pieces of the data and this may be ideally conducted as a parallel processing application. We'd like to explore this.

5.3 How might we use these tools?

As shocking as it might sound, we think one of the most likely uses of these tools is data cleaning! We've already experienced this in writing this paper. Through generating the example application in this paper we discovered several serious problems with the data. A new data set was generated. But it's clear there are still some interesting subsets that may be due to the cleaning process. Much of the data that this paper addresses will be made available to the public after processing and cleaning so that it is important that it is reliable and accurate. The graphics tools will allow the data analyst find problems, especially of a multivariate nature.

But there is a data analysis purpose, too. It's hoped that these tools can be used to examine large data in a multivariate manner. Especially we are interested in global weather patterns and anomalies. We need to take the time to explore the web for available data.

Software availability

The software is available through www.sourceforge.net/projects/limn.

Acknowledgments

This research is supported by funding from National Science Foundation #9982341.
Appendix: Software Details

The object-oriented framework of pluggable components that make up the Limn software system is written in the Java programming language. Java allows for the definition of abstract objects through specification of interfaces. Concrete objects can then be written that interact with one another through the interface specifications and in doing so allow for a flexible software system where objects that share the same interface can be used interchangeably. Limn has been designed to provide a flexible framework where any combination of concrete implementations can be assembled and executed by simply changing a few parameters in the properties file. For example a new module can be written that provides a novel method for calculating projection basis and as long as that module implements the LimnBasisFactory interface it can be easily integrated with the existing system.

For Limn six interfaces form the core framework and the main applications will build and bind a collection of concrete objects that are specified in the main properties file. These main components of the system are described as follows:

- **LimnData** provides some basic data access and selection features. A LimnData object may contain an entire data set or may only cache sections of data as needed.

- **LimnBasisFactory** is responsible for building appropriate projections. In some instances these will need to be generated in batches before the animation process begins to find the best combinations of projections for the tour. in others it might make more sense to build them as needed. The factory is only responsible for producing the main key frames of the projection, the LimnEngine is responsible for iterating between them.

- **LimnBasis** is a small wrapper class that contains the active projection. It is little more than a convenience as it provides methods for performing the projections across rows of data. This object is accessed by both the LimnEngine which writes new projections into it and LimnRender which reads the projection information out of it.

- **LimnEngine** manages the creation and interpolation between tour projection basis. It will notify the LimnRender component when a new projection has been created and is ready for rendering.

- **LimnRender** is the core rendering object. While other objects may augment the graphic with navigation tools or indices, the basic structure and type of the graphic will be defined here. Once rendering is complete the recorder object is notified to save the image.

- **LimnRecorder** receives notification from the LimnRender component to begin processing the next frame. Once finished processing it notifies the LimnEngine that the cycle has completed.
Currently there are modules for each of these interfaces are as follows:

**LimnData**
- RAFData - will cache only sections of a data file when data from that section is needed.
- SimpleData - copies all the data into memory.

**LimnBasisFactory**
- GTBasisFactory - produces random seeded basis.
- PPHolesFactory - computes the ppholes index for each projection. This information can then be rendered within the animation, and can be used to set a rejection value for new projections.

**LimnBasis**
- SimpleBasis - contains active random basis and performs projection calculations against rows of data.

**LimnEngine**
- TourEngine - basic basis interpolation
- MatrixEngine - layout of bases into a matrix format

**LimnRender**
- RecordCanvas - Abstract class that provides the core functionality for providing the images to the recorder.
- ScatterPlotRender - extends RecordCanvas and as a 2d scatter-plot canvas, and creates index files for the plot image.

**LimnRecorder**
- QuicktimeRecorder - Provides methods for creating quicktime movies out of the animations.
- JPEGRecorder - saves images as JPEG images to be compiled by some external MPEG building utility.

**References**


