Plaid™ User’s Guide

Art B. Owen
Stanford University

April 2000

Abstract

This document describes the plaid program. This program is designed for analyzing gene expression data.

Plaid Model

The plaid model was introduced by Lazzeroni & Owen (2000). To fix ideas, we suppose that \( X \) is a matrix of gene expression data. The entry \( X_{ij} \) describes the level of expression of gene \( i \) in sample \( j \). The Plaid model approximates this data by a sum

\[
\sum_{k=1}^{K} \rho_{ik} \kappa_{jk} \theta_{ijk}
\]

(1)

where \( \rho_{ik}, \kappa_{jk} \in \{0, 1\} \) are membership indicator variables. The row \( i \) is said to be in layer \( k \) if and only if \( \rho_{ik} = 1 \). Similarly column \( j \) is in layer \( k \) if and only if \( \kappa_{jk} = 1 \). The values \( \theta_{ijk} \) represent Analysis of Variance (anova) models fit to the rows and columns in the layer. There are four types of anova model:

\[
\begin{align*}
\theta_{ijk} &= \mu_k \\
\theta_{ijk} &= \mu_k + \alpha_{ik} \\
\theta_{ijk} &= \mu_k + \beta_{jk} \\
\theta_{ijk} &= \mu_k + \alpha_{ik} + \beta_{jk}.
\end{align*}
\]

If the \( \alpha_{ik} \) term is present then it must satisfy \( \sum_{i} \rho_{ik} \alpha_{ik} = 0 \). A \( \beta_{jk} \) term must satisfy \( \sum_{j} \kappa_{jk} \beta_{jk} = 0 \).
The algorithm adds layers to the model one at a time. It searches for layer $K$ in the residual

$$Z_{ij} = X_{ij} - \sum_{k=1}^{K-1} \rho_{ik} \kappa_{jk} \theta_{ijk}$$

left over from the first $K - 1$ layers.

**Installation**

Plaid has been written to work under Windows 95\textsuperscript{TM}, Windows 98\textsuperscript{TM}, Windows NT\textsuperscript{TM}, and Windows 2000\textsuperscript{TM}. To install the Plaid program simply place a copy of the executable file into the folder containing the data to be analyzed. If there are two or more folders containing the data to be analyzed, then one can place the executable in one of the folders, and create shortcuts in the other folders.

**Data files**

To use the program, place your data into three files. The file names must have extensions .row, .col and .dat, and a common base name. For example, they could be called: MyData.row, MyData.col, and MyData.dat. The contents of these files are described below. When the program starts, it will give the prompt:

**Enter data set name:**

The user then types MyData at the prompt and presses Enter. A given folder can contain many .row, .col and .dat triples of files.

The file MyData.row should contain the names of the genes to be analyzed. There should be one gene per line of input. The gene names can contain spaces or tabs. A gene name can contain up to 999 characters.

The file MyData.col should contain the names of the samples. There should be one sample line per line of input and the sample names can be up to 999 characters long.

The gene and sample names are used to label output. The program does not do any computation with them. In particular it is possible for two or more of the gene (or sample) names to be identical. Before using these names, Plaid replaces spaces and tabs contained within the names with underscore _ characters. Thus “Mouse liver” is changed to “Mouse_liver”.

2
This makes it easier for other programs to read through the Plaid output. These changes are not written out to the .row and .col files.

The Plaid program counts the number of names in files MyData.row and MyData.col. Suppose that there are \( n \) row names and \( p \) column names. Then the file MyData.dat should contain an \( n \times p \) matrix of data values. There must be exactly \( n \) rows of data in this file, one row per line. Each row must contain exactly \( p \) numbers. Missing values are not allowed. They can be handled as described below. The numbers within a row should be separated by spaces or tabs (not commas).

The Plaid executable, or a shortcut to it, should be present in the same folder that the files are in.

**Basic commands**

The Plaid program puts up a scrollable command window. The user enters commands into this window and most output is displayed into this window.

Typing quit or exit causes Plaid to terminate. Typing help or ? causes a list of the available commands to be typed out.

**sink filename** directs output to the named file, where it can be imported by other programs. Some further commands below are specifically geared to writing out raw data in a format suitable for producing color images.

**source filename** reads input from the named file. Keeping commands in an input file makes it easier to document and reuse data analyses. A source command without a filename returns the input focus to the console.

**Data adjustment**

Plaid has a limited facility to make adjustments on the data. Adjustments can be made to the rows or to the columns or to both. A location adjustment visits each entity (row or column or both) in turn, computes a value for that entity, and subtracts it from every data element in the entity. A scale adjustment divides each data element by a value. When both location and scale adjustments are requested, the location adjustment is done first then the scale adjustment.

The command adjust carries this out. Here are some examples of this command:

```
adjust row scale moment
```
adjust col location robust
adjust both location moment
adjust col both robust
adjust both both moment 10

The first argument to adjust describes whether rows, columns or both are to be adjusted. The second column describes whether the adjustment should be for location, or scale, or both. The third argument can be moment for which location and scale are mean and standard deviation, or robust for median and median absolute deviation, or extreme for minimum and maximum minus minimum.

A fourth argument, if present, describes the number of times to apply the adjustment if this number is to be larger than 1. For example, to make every row have the same scale and every column have the same scale may require iteration.

If two arguments are given, then one moment adjustment is made. If one argument is given, then one scale moment adjustment is made. If no arguments are given, then one column scale moment adjustment is made. The algorithm will refuse to divide any row or column by 0, and will instead print a warning. Any number of adjust commands can be made. They are applied in sequence.

Layer preferences

This section describes strategies for adding the next layer to the model.

The algorithm can be tuned to prefer $\mu_k + \alpha_{ik}$ to have the same sign as $\mu_k$. To set this option give the command unisignrow. To remove this option give the command unisignrowoff. Similarly, commands unisigncol and unisigncoloff set and unset preferences for $\mu_k + \beta_{jk}$ to have the same sign as $\mu_k$. The unisign features are treated as preferences, and not as constraints by the algorithm. By default, the algorithm starts without any unisign preferences.

The command seekms causes the algorithm to prefer small intense layers to larger diffuse ones. These small layers may not reduce the sum of squared errors as much as in the default (seekss) mode, but they can be more interpretable. The name seekms suggests that the sum of squared $\theta_{ijk}$ within such a layer is large compared to the number of parameters in the layer. seekms is implemented by changing the followup iterations. These are the iterations that take place after all $\rho_{ik}$ and $\kappa_{jk}$ for the new layer $k$ are either
0 or 1. \texttt{seeks} is implemented by following up the default search with a number of iterations that prefer small intense layers.

The algorithm can be tuned to release rows or columns that do not fit the present layer so well. The basic algorithm is a greedy search to reduce the sum of squared errors. As such rows or columns with very large entries can be selected to join the present layer. It may be better to release them and have them turn up in a subsequent layer where they fit better.

Setting \texttt{rowrel} .3 will cause the algorithm to release row \( i \) from the candidate layer \( k \) if

\[
\sum_{j=1}^{p} \kappa_{jk}(Z_{ij} - \theta_{ijk})^2 > (1 - .3) \sum_{j=1}^{p} \kappa_{jk}Z_{ij}^2.
\]

Of course, the argument to \texttt{rowrel} need not be 0.3. It can be interpreted as a minimum desired \( R^2 \) value for rows in the new layer. Row releases can be turned off by the command \texttt{rowreloff}. Commands \texttt{colrel} and \texttt{colreloff} apply analogously to columns. The algorithm starts in a default state with no row or column releases.

The \texttt{rowrel} and \texttt{colrel} filters are applied after the algorithm has identified a layer, so that every \( \kappa_{jk} \) and \( \rho_{ik} \) is 0 or 1. The releases are applied in parallel to all rows and or columns, and then the \( \theta_{ijk} \) are updated. It is possible that all rows or that all columns get released producing a candidate layer with no data. A nonparallel algorithm that alternated between releasing only a few (possibly one) of the rows or columns most deserving to be released and then updated the layer might yield fewer empty layers, but could be very slow on large data sets.

Whatever choices are made for the unisign or release features, the \( \mu_k \), \( \alpha_{ik} \), and \( \beta_{jk} \) values are always recomputed for the current \( \rho_{ik} \) and \( \kappa_{jk} \) values. This can cause the desired features to be violated.

The command \texttt{strategy} will print out the present state of all the layer seeking options.

\section*{Layer finding}

The commands to find layers are

\begin{verbatim}
findm
findma
findmb
findmab
\end{verbatim}
and they fit layers of the form

\[
\begin{align*}
\rho_{ik} \kappa_{jk} & \left[ \mu_k \right] \\
\rho_{ik} \kappa_{jk} & \left[ \mu_k + \alpha_{ik} \right] \\
\rho_{ik} \kappa_{jk} & \left[ \mu_k + \beta_{jk} \right] \\
\rho_{ik} \kappa_{jk} & \left[ \mu_k + \alpha_{ik} + \beta_{jk} \right]
\end{align*}
\]

respectively, where \( \rho_{ik}, \kappa_{jk} \in \{0, 1\} \).

The commands above determine the number of rows or columns as well as which ones to include. The user can also influence the number of rows or columns in the layer. The commands

\[
\begin{align*}
\text{findmab} & \ r \ c \\
\text{findmab} & \ 0 \ c \\
\text{findmab} & \ r \ 0
\end{align*}
\]

are for finding layers with exactly \( r \) rows and \( c \) columns. Setting \( r = 0 \) tells the algorithm to choose the number of rows, setting \( c = 0 \) tells the algorithm to choose the number of columns, and setting them both to 0 is the same as not providing \( r \) and \( c \). The commands \text{findm}, \text{findma}, and \text{findmb} also have optional arguments \( r \) and \( c \). Either none or both of these arguments must be provided, as the program cannot tell whether a single argument should refer to rows or to columns.

User specified \( r \) and \( c \) values can conflict with user specified options for unisign or releases as described above. In such cases it is possible for the algorithm to find a smaller layer than the desired size.

When a layer has been found, the program prints out a short description of it, giving the number of rows, number of columns, and an anova table for the layer. If the user is satisfied with the candidate layer, the user can give the \texttt{accept} command and the layer will be inserted in the model. Then the current values of \( Z_{ij} \) will be replaced by \( Z_{ij} = \rho_{ik} \kappa_{jk} \theta_{ij} \). Once the first layer has been accepted it is no longer possible to apply location or scale adjustments to the data.

The command \texttt{shuffle} repeats the most recent layer search on a shuffled version of the data. For \( i = 1, \ldots, n \) let \( \pi_i \) be a random permutation of the values \( 1, \ldots, p \) and for \( j = 1, \ldots, p \) let \( \tau_j \) be a random permutation of the values \( 1, \ldots, n \). A row shuffle turns \( Z_{ij} \) into

\[ \tilde{Z}_{ij} = Z_{i\pi_i(j)}. \]

Then a column shuffle turns \( \tilde{Z}_{ij} \) into

\[ \tilde{Z}_{ij} = \tilde{Z}_{\tau_j(i)j}. \]
The command **shuffle** $k$ repeats the most recent layer search on shuffled data $k$ times.

The command **backfit** $b$ causes the program to go through $b$ rounds of “backfitting”. In a round of backfitting, every accepted layer is revisited in order from first to most recent. When a layer is revisited its parameters $\mu_k$, $\alpha_{ik}$, and $\beta_{jk}$ are recomputed and the residual $Z_{ij}$ is correspondingly adjusted. Backfitting does not change the values of $\rho_{ik}$ or $\kappa_{jk}$. Backfitting can introduce violations of the unisign or row release criteria described above.

**Model inspection**

The command **dataanova** prints an anova table of the data. This anova reflects any location or scale adjustments that have been made to the data. The command **resanova** prints an anova table for the current set of residuals. The command **show** prints an anova table for the candidate layer.

The command **resid** prints the residual matrix in the output window. The **resid** command is useful for small data sets, but when the matrix is large a better alternative is the command **fileres**. This will write the residuals as an $n$ by $p$ matrix to a .res file. For example if the data were in **MyData.dat** the residuals will overwrite the file **MyData.res**. The previous contents of **MyData.res** are lost. These files can be very large, and so overwriting them saves disk space. The user should move, copy, or rename any residual files of permanent interest. Such file saving can be done through the operating system in a separate window.

The residual file can then be viewed in an image plot, in some other program than Plaid. Matlab<sup>TM</sup> or Splus<sup>TM</sup> are able to make color image plots from flat files of data. Both are able to run concurrently with Plaid.

The command **filelayer** $k$ writes the $k$'th layer in the model to a .lyr file, overwriting the previous contents. If the data were in **MyData.dat** the layer will overwrite the file **MyData.lyr**. Only the rows with $\rho_{ik} = 1$ and the columns with $\kappa_{jk} = 1$ are written out. Color image plots can be made of these layer files. Layer files of permanent interest must be moved, copied, or renamed.

The command **describe** prints a description of the model layer by layer. It shows the type, dimensions and sum of squares for each layer. It also shows the sign of $\mu_k$ for each layer.

The commands

```
compare rows
compare columns
```
compare data
compare

describe the extent to which the layers in the model overlap. The first three commands each produce a table with one row and one column for each layer. The entry in the table shows the percent of rows (respectively columns or data) appearing in the layer at the left margin that are also in the layer at the top margin. The fourth command produces all three comparisons.

The command print prints a description of all the layers. The description includes the row names, the column names, $\mu_k$, and any applicable $\alpha_{ik}$ and $\beta_{jk}$ values.

The command printa k prints the $\alpha_{ik}$ values if any in layer $k$ along with the names of the rows in layer $k$. The command printb k prints the $\beta_{jk}$ values if any in layer $k$ along with the names of the columns in layer $k$.

The command rowcases prints out three columns for each layer in the model. The first column gives the layer number. The second columns gives the index $i$ between 1 and $n$ for each row in the layer. The third column gives $\mu_k + \alpha_{ik}$ if the layer has $\alpha_{ik}$ values, and $\mu_k$ otherwise. The point of printing row indices instead of row names is that these values can be read into other software and used to identify the rows directly. The command rowcases k prints out these three columns only for layer $k$. The commands colcases and colcases k print column indices $j$ in layer $k$ and, if applicable, values of $\mu_k + \beta_{jk}$.

The command rowins i prints the data values for row $i$ along with the contribution to this row from each layer containing it, and the present residual for row $i$. The command colins j prints analogous information for column $j$.

The command latexlayer k produces a textual representation of layer $k$ that can, with minor editing, be used as a Table in \LaTeX\ articles. Underscores are removed from the row and column names that appear in the \LaTeX\ table.

Missing values

The way to handle missing values is to replace each missing value by an estimate computed outside of Plaid. The estimate could be a neutral background value, or it could be taken from a simple model such as a global anova on the whole data set. In this case one might replace a missing value $X_{ij}$ by $\bar{X}_i + \bar{X}_j - \bar{X}$. Here $\bar{X}_i$ is the average of non-missing $X_{ij}$ values in
row $i \bar{X}_j$ is defined similarly, and $\bar{X}_n$ is the average of all non-missing data values in the array.

References