R/MAANOVA: An extensive R environment for the Analysis of Microarray Experiments

Hao Wu, Gary A. Churchill

April 25, 2007

Contents

1 Introduction

R/maanova is an extensible, interactive environment for the analysis of microarray experiments. It is implemented as an add-on package for the freely available statistical language R (www.r-project.org). The engine functions were written in C for better performance.

MAANOVA stands for MicroArray ANalysis Of VAriance. It provides a complete work flow for microarray data analysis including:

- Data quality checks and visualization
- Data transformation
- ANOVA model fitting for both fixed and mixed effects models
- Statistical tests including permutation
- Cluster analysis with bootstrapping

R/maanova can be applied to any microarray data but it is specially tailored for multiple factor experimental designs. Mixed effects models are implemented to estimate variance components and perform F and T tests for differential expressions.

2 Installation

2.1 System requirements

This package was developed under *R 1.8.0* in the *Linux Redhat 8.0* operating system. The programs have been observed to work under *Windows NT/98/2000*. The memory requirement depends on the size of the input data but a minimum of 256Mb memory is recommended.

2.2 Obtaining R

R is available in the Comprehensive R Archive Network (CRAN). Visit `http://cran.r-project.org` or a local mirror site. Source code is available for UNIX/LINUX, and binaries are available for Windows, MacOS, and many versions of Linux.

2.3 Installation - Windows(9x/NT/2000)

- Install *R/maanova* from Rgui

  1. Start Rgui
  2. Select Menu `Packages`, click `Install package from local zip file`. Choose the file `maanova_*.tar.gz` and click 'OK'.

- Install *R/maanova* outside Rgui

  1. Unzip the `maanova_*.tar.gz` file into the directory `$RHOME/library` (`$RHOME` is something like `c:/Program Files/R/rw1081`). Note that this should create a directory `$RHOME/library/maanova` containing the R source code and the compiled dlls.
  2. Start Rgui.
  3. Type `link.html.help()` to get the help files for the maanova package added to the help indices.

2.4 Installation - Linux/Unix

1. Go into the directory containing `maanova_*.tar.gz`.

2. Type `R CMD INSTALL maanova` to have the package installed in the standard location such like `/usr/lib/R/library`. You will have to be the superuser to do this. As a normal user, you can install the package in your own local directory. To do this, type `R CMD INSTALL -library=$LOCALRLIB maanova_*.tar.gz`, where

$LOCALRLIB is something like /home/user/Rlib/. Then you will need to create a file .Renviron in your home directory to contain the line R_LIBS=/home/user/Rlib so that R will know to search for packages in that directory.

3 Data and function list

The following is a list of the available functions. For more information about the function usage use the online help by typing "`?functionname`" in *R* environment or typing `help.start()` to get the html help in a web browser.

1. Sample data available with the package

   `abf1` A 18-array affymetric experiment. 500 hand picked genes are included in the data set

   `kidney` A 6-array kidney data set from CAMDA (Critical Assessment of Microarray Data Analysis).

   `paigen` A multiple factor 28-array experiment from Bev Paigen's lab in The Jackson Lab. Only 300 hand picked genes are included in this data set

2. File I/O

   `read.madata` Read microarray data from TAB delimited simplex text file.

   `write.madata` Write microarray data to a TAB delimited simple text file.

3. Data quality check

   `arrayview` View the layout of the arrays

   `riplot` Ratio intensity plot for arrays

   `gridcheck` Plot grid-by-grid data comparison for arrays

   `dyeswapfilter` Flag the bad spots in dye swap experiment Note that these data quality check functions only work for 2-dye arrays at this time.

4. Data transformation

   `createData` Create a data object with options to collapse the replicated spots and do log2 transformation

   `transform.madata` Data transformation with options to use any of several methods

5. ANOVA model fitting

   `makeModel` Make model object to represent the experimental design

   `fitmaanova` Fit ANOVA model

   `resiplot` Residual plot on a given ANOVA model

6. Hypothesis testing

`matest` F-test or T-test with permutation

`adjPval` Calculate FDR adjusted P values given the result of [`matest`]

`volcano` Volcano plot for summarizing F or T test results

7. Clustering

   `macluster` Bootstrap clustering.

   `consensus` Build consensus tree out of bootstrap cluster result

   `fom` Use figure of merit to determine the number of groups in K-means cluster

   `geneprofile` Plot the estimated relative expression for a given list of genes

8. Utility functions

   `fill.missing` Fill in missing data.

   `summary.madata` Summarize the data object.

   `summary.mamodel` Summarize the model object.

   `subset.madata` Subsetting the data objects.

   `exprSet2Rawdata` Convert an object of exprSet to an object of Rawdata.

4 Preparing the input files

Before using the package, the user must manually prepare a data file and a design file.

4.1 Preparing the data file

There is only ONE data file for all of the slides in an experiment. Most gridding software produces one file for each slide. Thus you will have multiple files for a multiple array experiment and you have to combine these files to one data file as the input for *R/maanova*.

The data file is a TAB delimited text file. Each rows corresponds to the data for a gene. In the first a few columns, you can put some gene information, e.g., the Clone ID, Gene Bank ID, etc. and the grid location of the spot. Note that some gridding softwares return Block numbers instead of metarow and metacol. Then you must manually compute metarow and metacol from block and put them in the file. After that you need to put the scanned data for all arrays in the rest of the columns. (You need to make the decision what data you want to use in analysis, e.g., mean versus median, background subtracted or not, etc.) For N-dye arrays, the N channel intensity data for one array need to be adjacent to each other (in consecutive columns). The order (dye1, dye2, ...) must be consistent across all of the slides. You can put the spot flag as a column after intensity data for each array. (Note that if you have flag, you will have N+1 columns of data for each array. Again, this must be consistent for all arrays in the data set.) If you have duplicated spots within one array, replicated measurements of the same clone on the same array should appear in adjacent rows. This can be easily done by sorting on cloneid. The number of replicates must be constant for all genes.

As an example, you have four slides for a 2-dye array experiment scanned by GenePix. Then you will have four output files. Following the steps to create your data file:

1. Open your favorite spread sheet editor, e.g., MS Excel and create a new file;

2. Paste your clone ID, Gene names, Cluster ID and whatever information you want to keep into the first several columns;

3. Open your first GenePix file in another window, copy the grid location into next 4 columns (you only need to do this once because they are all the same for four slides);

4. Copy the two columns of foreground mean value (if you want to use it) and one column of flag to the file in the order of Cy5, Cy3, flag;

5. Open your other 3 files and repeat step 4;

6. Select the whole file and row sort it according to Clone ID;

7. Save the file as tab delimited text file and you are done.

The data file must be “full”, that is, all rows have to have same number of columns. Sometimes leading and trailing TAB in the text file can cause problems, depending on the operating system. So the user should be careful about that. Sometimes the special characters in gene description can cause reading problem. I don’t encourage you to put the gene description in the data file. If you have to do that, you must be careful (sometimes you need to remove the special characters manually).

4.2 Preparing the design file

The design file is another TAB delimited text file. The number of rows in this file equals the number of arrays times N(the number of dyes) plus one (for the column headers). The number of columns in this file depends on the experimental design. For example, you can have “Strain”, “Diet”, “Sex”, etc. in your design file. You must have the following columns in the design file (column headers are case sensitive):

- Array: for array name
- Dye: for dye name
- Sample: Sample ID number

Array and Dye columns are easy to understand. Sample column contains integers used to identify biological replicates and reference samples. Usually you should assign each biological individual a unique Sample number. Reference samples are represented by zero(0). Reference sample are treated differently. They will always be treated as fixed factor in the model and not involved in statistical tests.

You must not have the following columns in the design file:

- Spot: reserved for spot effect
- Label: reserved for labelling effect

Sample column in design file need to be continuous integer. All other columns can be either integers or characters.

You don’t have to USE all factors in design file. When making the model object in `makeModel`, the experimental design will be determined by the design and a formula. You can put all factors in design file but turn them on/off in formula.

5 A quick tour of the functions

This section will go through a few demo scripts distributed with the package to help the users understand the function syntax and capabilities. The binary data are distributed with the software. The original data file can be downloaded from:
`http://www.jax.org/staff/churchill/labsite/software/`

5.1 CAMDA kidney experiment

This is the kidney data from CAMDA (Critical Assessment of Microarray Data Analysis) originally described by Pritchard *et al*.

The website for CAMDA is `http://www.camda.duke.edu`. It is a 24-array double reference design. Six samples are compared to a reference with dye swapped and all arrays are duplicated. Flag for bad spots is included in the data.

Note that because BioConductor requires all contributed packages to be less than 1Mb in size, the binary data distributed with the package is only a small part of the real data set (first 300 genes). So you cannot reproduce the figures presented in following sections. The package with full data set, can be download from Gary Churchill's website at `http://www.jax.org/staff/churchill/labsite/software/`. You can also find the original data file (in text format) there.

1. First load data into the workspace

   ```
   R> data(kidney)
   ```

2. Then we do some data quality check

   ```
   R> gridcheck(kidney.raw)
   ```

   `gridcheck` is used to check the hybridization and gridding quality within and cross arrays. You should see near linear scatter plots in all grid for all arrays. This one is not great but acceptable. The grid check plot for the first array is shown in figure 1. The red dots are for the spots with flags.

   ```
   R> riplot(kidney.raw)
   ```

   `riplot` stands for ratio-intensity plot. It is also called MA plot. The riplot for the first array is shown in figure 2.

   ```
   R> arrayview(kidney.raw)
   ```

   `arrayview` is used to view the spatial pattern of the arrays. The standardized log ratios for all spots are shown in different colors. The arrayview for the first array is shown in figure 3.

   You will generate a lot of figures by doing gridcheck, riplot and arrayview. Use `graphics.off` to close all figures.

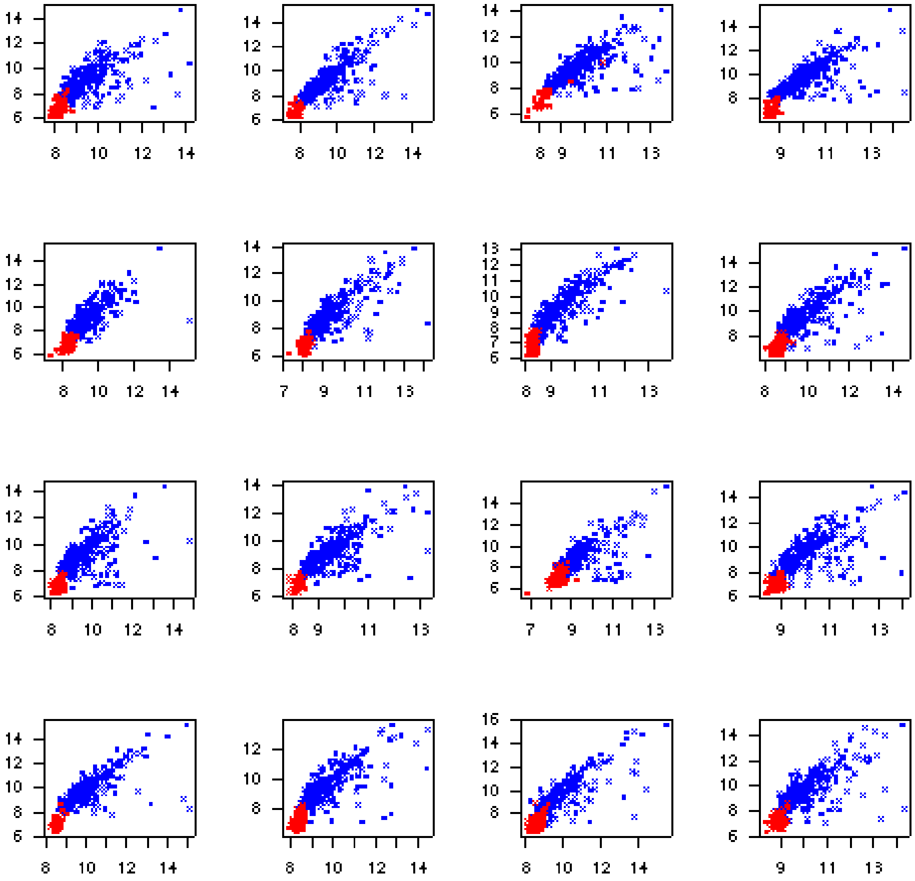


Figure 1: Grid check plot for the first array in kidney data

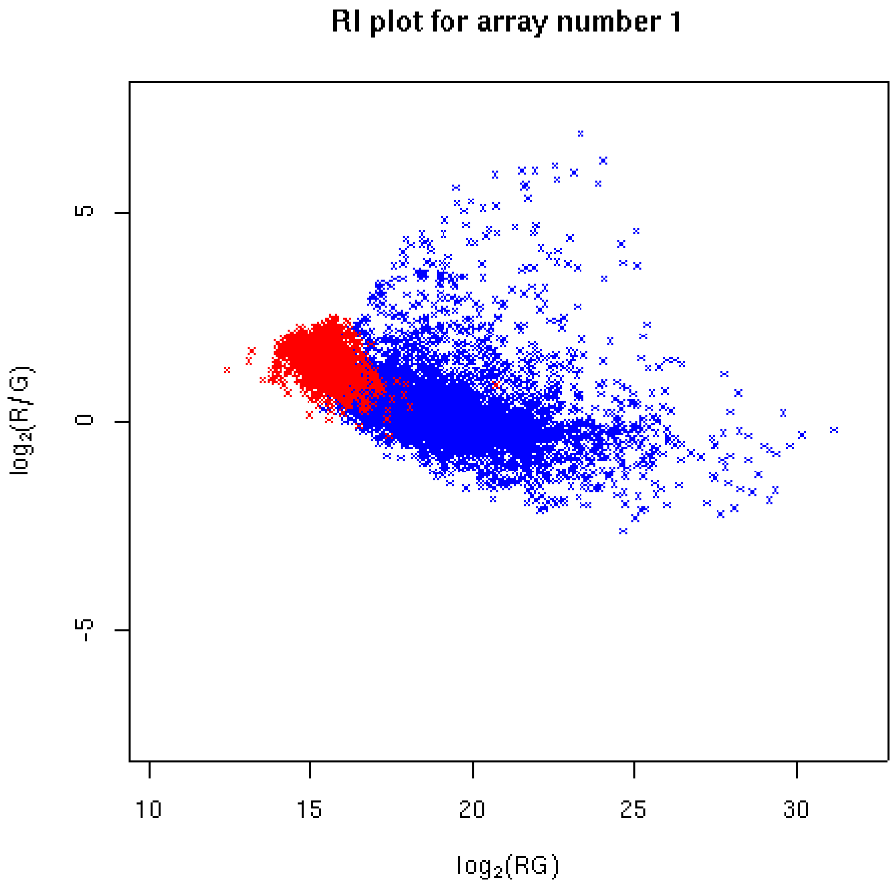


Figure 2: RI plot for the first array in kidney data

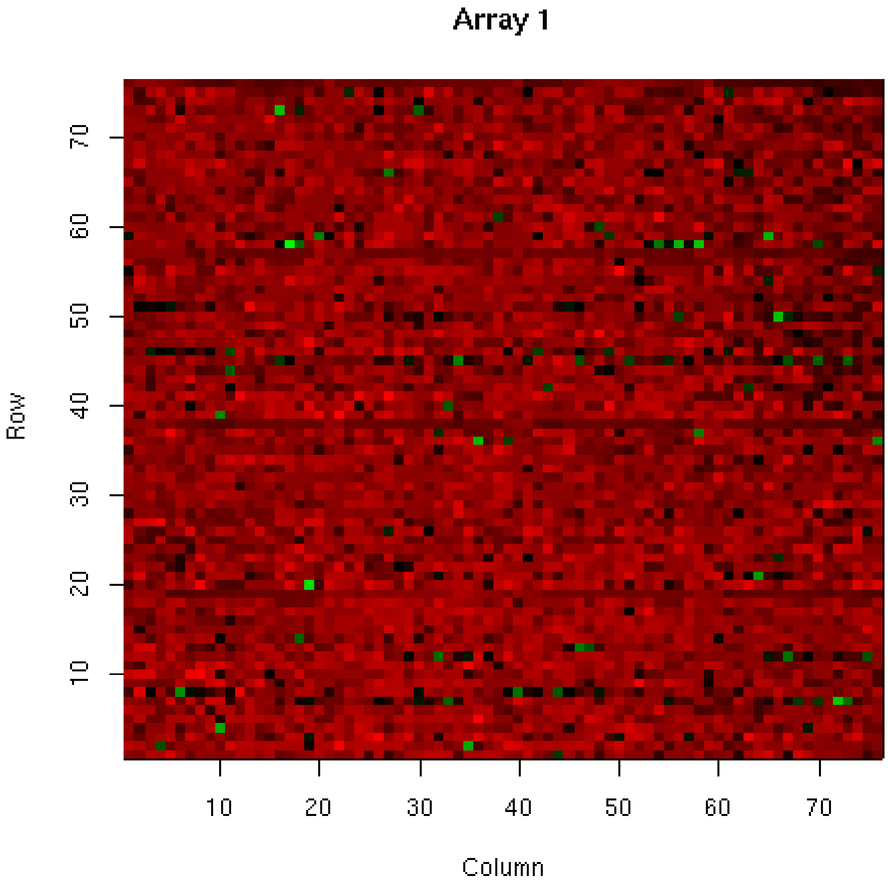


Figure 3: arrayview for the first array in kidney data

3. Now make an object of class `madata`. `madata` and `mamodel` are the key objects in *R/maanova*. Most of the functions work on them. A `madata` object stores the experimental data. It derived from the raw data. The `mamodel` object stores the experimental design information. We will discuss it a little later.

   ```
   R> kidney <- createData(kidney.raw)
   R> summary(kidney)
   ```

4. Transform the data using spatial-intensity joint loess.

   ```
   R> kidney <- transform.madata(kidney, method="rlowess")
   ```

   There are several data transformation method. Which method to use depends on the data. Read Cui *et al.*(2002) for detail. The transformation plot for the first array is shown in figure 4.

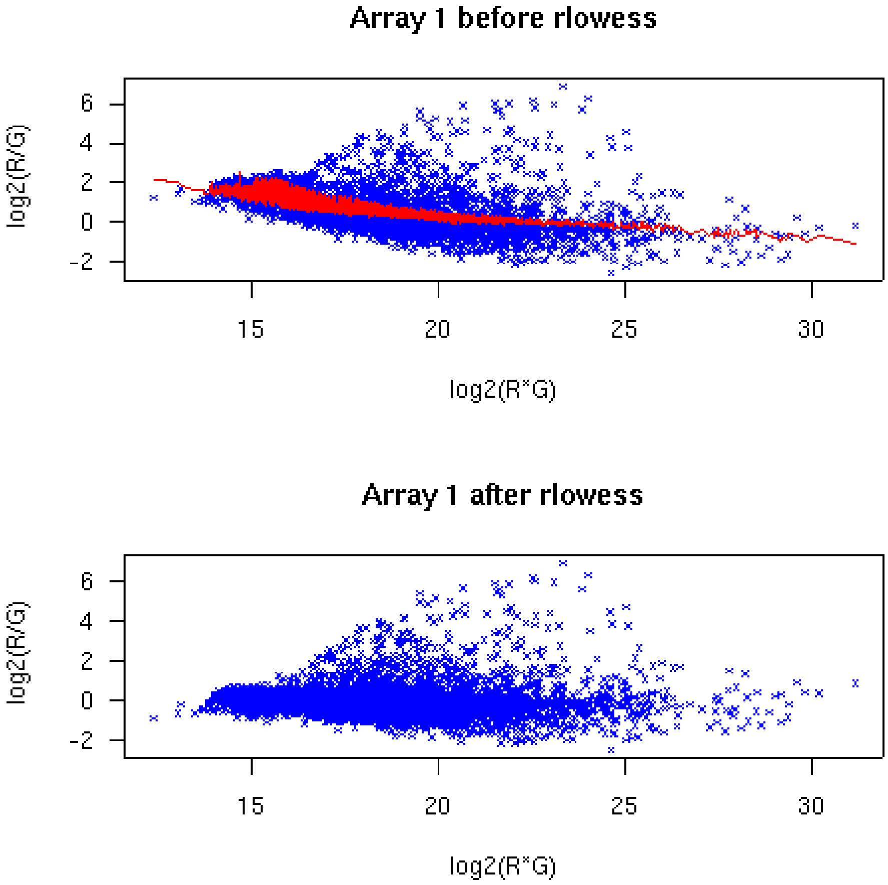


Figure 4: Joint lowess transformation on the first array for the kidney data

5. Make model object for fixed model

   ```
   R> model.fix <- makeModel(data=kidney, formula=~Dye+Array+Sample)
   R> summary(model.fix)
   ```

   A `mamodel` object store the experimental design information. `makeModel` function takes a data object and a R formula as the ANOVA model and make the design matrices. The input formula is an object of `formula`. It represents the ANOVA model. In the formula, you can put any combination of the factors in your design. Interaction between any two terms are allowed. At this point, 3 or higher way interactions are not taken by the program. `makeModel` takes another argument `random` for the random terms. `random` must be another formula. All terms in `random` must be included in `formula` as well.

6. Fit ANOVA model and do residual plot

   ```
   R> anova.fix <- fitmaanova(kidney, model.fix)
   R> resiplot(kidney, anova.fix)
   ```

7. Permutation F test. F-test function can test one or multiple terms in a given model object. User can do residual shuffling or sample shuffling for fixed effect models. For mixed effects models, only sample shuffling is available. Permutation test could be very time consuming, especially for mixed effects models. The permutation test function can run on linux clusters through message passing interface (MPI). For detailed information about F test and using computer cluster, please read appendix.

   Here we want to test Sample effect in the model:

   ```
   R> test.fix <- matest(kidney, model=model.fix, term="Sample",
          n.perm=500)
   ```

   Now `test.fix` contains the tabulated P values and permutaion P values. Sometimes we want to calculate FDR adjusted P values for the test result.

   ```
   R> test.fix <- adjPval(test.fix)
   ```

   After getting the F-test result, We can do volcano plot to visualize it. `volcano` function has options to choose the P-values to use and set up thresholds. We will use tabulated P values for F1 and FDR adjusted permutation P values for other tests here. In the plot, the orange dots above the horizontal line represent the significant genes. Note that the the flagged spot can be highlighted in the plot. You can turn if on by providing `highlight.flag=TRUE`.

   ```
   R> idx.fix <- volcano(test.fix,method=c("unadj", rep("fdrperm",3)),
           highlight.flag=FALSE)
   ```

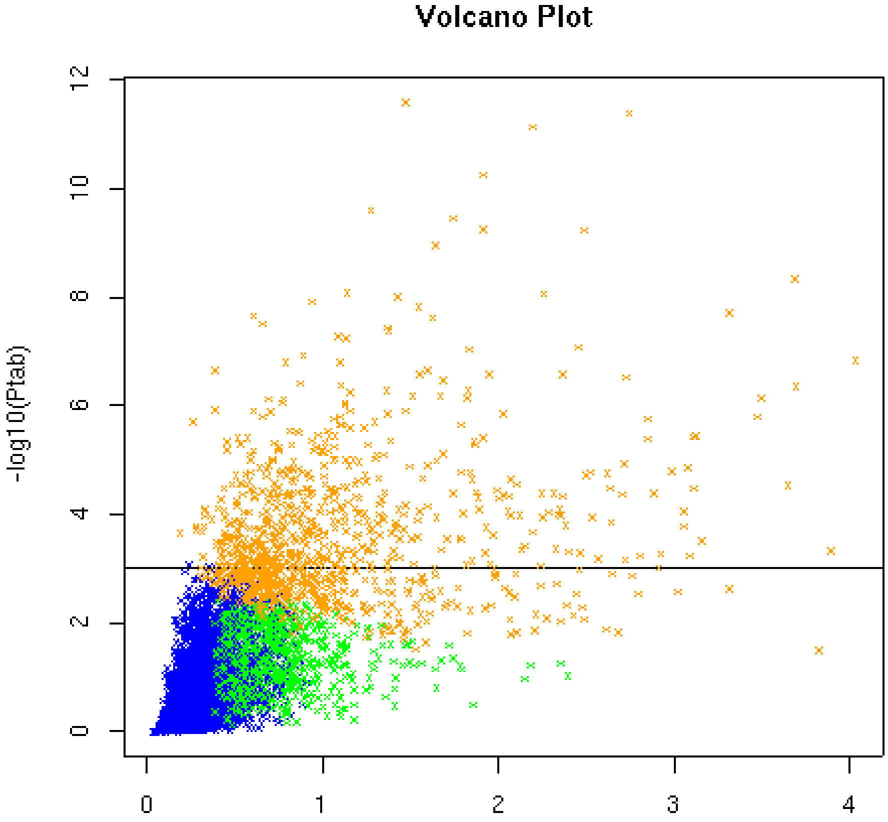


Figure 5: Volcano plot for kidney data - fixed effect model

The volcano plot is shown in figure 5. Note that the return variable of volcano contains the indices for significant genes.

8. Now we can do cluster bootstrapping and build consensus trees. Currently two cluster methods are implemented, hierarchical clustering and K-means. Hierarchical cluster could be very sensitive to bootstrap if you have too many leaves on the cluster. Some small disturbance on the data could change the whole tree structure. So if you have many genes, say, more than 50, and you want to build a consensus tree from 100 bootstrapped hierarchical trees. It is very likely that you get a comb, that is, all leaves are directly under root. But if you only have a few genes to cluster, it is working fine. So I suggest user use K-means to cluster the genes and use hierarchical to cluster the samples.

   `macluster` is the function to do cluster bootstrapping and `consensus` is used to build consensus trees (groups for K-means) from the bootstrap results.

   ```
   R> cluster.kmean <- macluster(anova.fix, ,term="Sample",
            idx.gene=idx.fix$idx.all,what="gene", method="kmean"
            kmean.ngroups=5, n.perm=100)
   R> con.kmean <- consensus(cluster.kmean, 0.7)
   ```

   An expression profile plot will be generated for the consensus K-means result. The plot is shown in figure 6.

   Now we can do hierarchical cluster on the samples. The consensus tree is shown in figure 7.

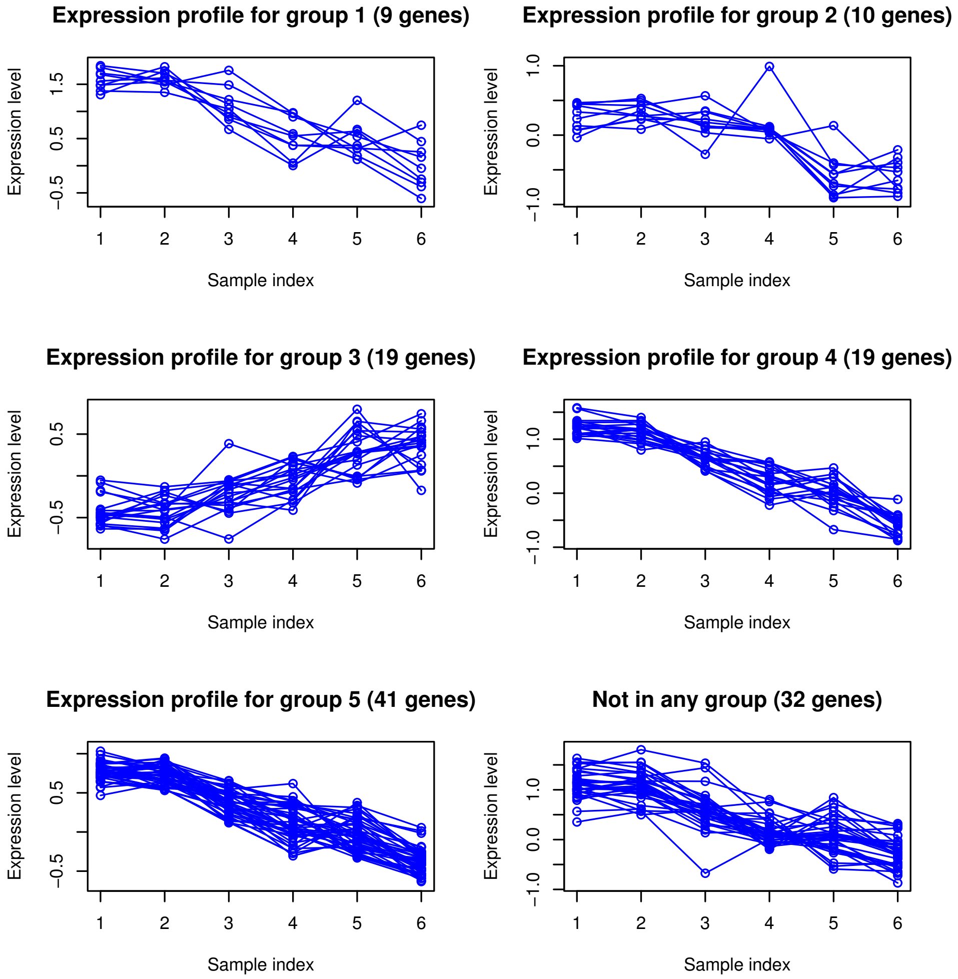
   ```
   R> cluster.hc <- macluster(anova.fix, term="Sample",
         idx.gene=idx.fix$idx.all,what="sample", method="hc", n.perm=100)
   R> con.hc <- consensus(cluster.hc)
   ```

9. Now we are going to analyze the data using mixed effect model. First we make a model object with Array effect as random factor. Note that normally Array effect, Spot effect and Labeling effect should be treated as random. Since we don't have technical replicate here, Spot and Label cannot be fitted.

   ```
   R> model.mix <- makeModel(data=kidney, formula=~Dye+Array+Sample,
              random=~Array)
   R> summary(model.mix)
   ```

10. Then we can fit the ANOVA model. This will take quite a while to finish. EM algorithm is implemented in the engine function for solving MME. For details about MME and the EM algorithms, read Searle *et al.*.

    ```
    R> anova.mix <- fitmaanova(kidney, model.mix)
    ```



Expression profile for group 1 (9 genes)

Expression profile for group 2 (10 genes)

Expression profile for group 3 (19 genes)

Expression profile for group 4 (19 genes)

Expression profile for group 5 (41 genes)

Not in any group (32 genes)

Figure 6: Expression profile plot for bootstrap K-means result, kidney data

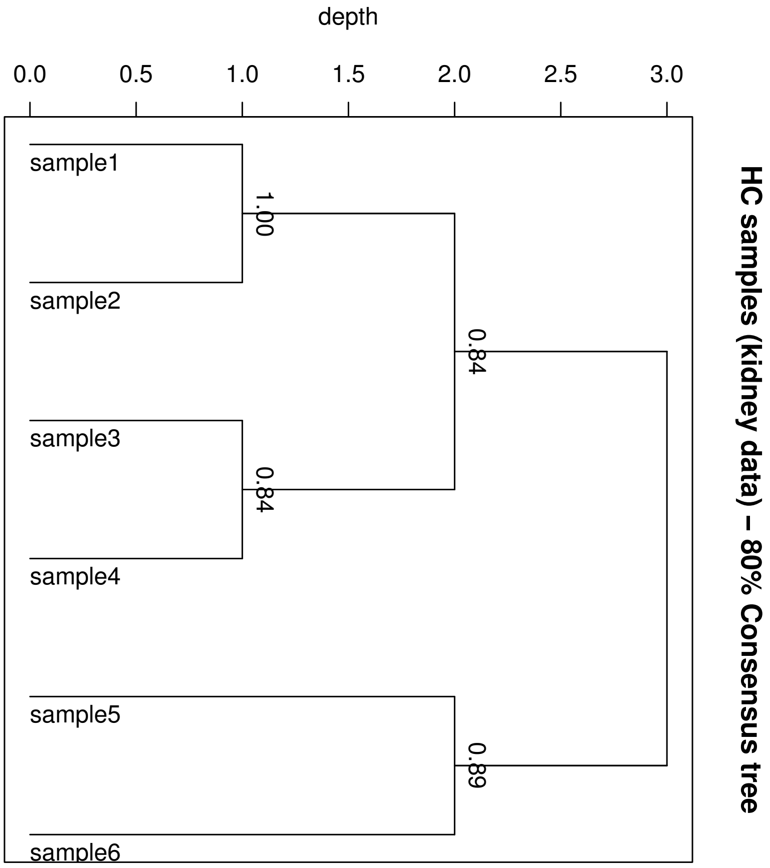


Figure 7: 80% Consensus tree for bootstrapping hierarchical cluster on the samples, kidney data

11. Now we test the sample effect. Again, permutation test for mixed effects model is very slow. You have better run it on computer clusters if possible.

    ```
    R> ftest.mix <- matest(data=kidney, model=model.mix, term="Sample",
           n.perm=100)
    ```

    We can do volcano plot for the mixed model result.

    ```
    R> idx.mix <- volcano(anova.mix, ftest.mix)
    ```

    The rest of the analysis (clustering and consensus trees) are skipped here.

5.2 Paigen's 300-gene experiment

This is a multiple factor 28-array experiment. The experiment is done in Beverly Paigen's Lab in The Jackson Lab. They took three strains of mice and feed them with two kind of diets. In that way you get six kind of mice. They picked two individuals in each group then you have totally 12 distinct mice. So in this experiment, you have strain, diet and biological replicates as the factors. You can test the effects from any factor or any combination of them. The experimental deign in shown in table 1 and figure 8.

	Hi fat diet	Low fat diet
Strain:Pera	A1, A2	E1, E2
Strain:I	B1, B2	F1, F2
Strain:DBA	C1, C2	D1, D2

Table 1: Mice used in Paigen's 28-array experiment

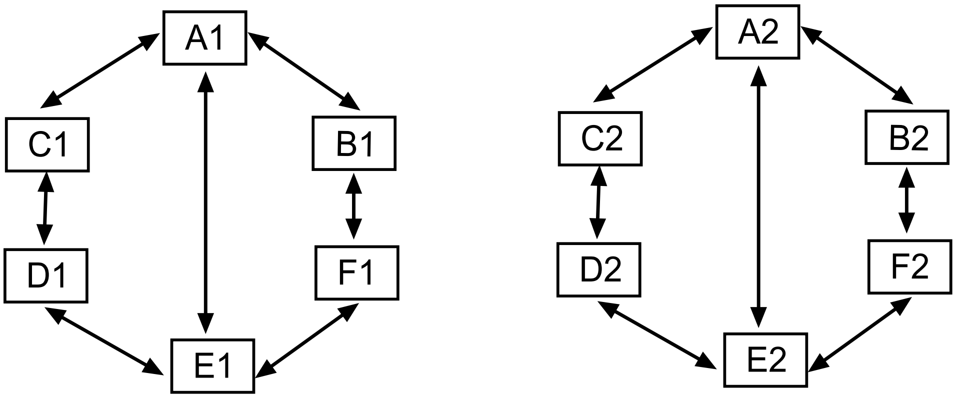


Figure 8: Array assignment in Paigen's 28-array experiment

We will skip the data quality check and transformation steps in this example. We will show you how to test different terms in the model in a multiple factor design. In this design, there are strain effect, diet effect, strain by diet interaction effect and biological replicate effect. The interaction effect is nested within the biological replicate effect.

1. Load in data

```
R> data(paigen)
```

2. Make data object with replicates collapsed. Note that if we donot collapse replicates, we need to put Spot in the model as a random terms. That will make the calculation even slower. Based on our experience, collapsing replicates gives similar results as not collapsing and fit Spot effect as random.

```
R> paigen <- createData(rawdata, 2, 1)
```

3. Make several model object and fit ANOVA. First make full model object with Strain, Diet, interaction and biological replicate effects.

   ```
   R> model.full.mix <- makeModel(data=paigen,
        formula=~Dye+Array+Strain+Diet+Strain:Diet,
        random=~Array+Sample)
   R> anova.full.mix <- fitmaanova(paigen, model.full.mix)
   ```

   We can do a variance component plot on ANOVA result.

   ```
   R> varplot(anova.full.mix)
   ```

   Then make a model object without interaction effect and fit ANOVA model

   ```
   R> model.noint.mix <- makeModel(data=paigen,
      formula=~Dye+Array+Strain+Diet+Sample, random=~Array+Sample)
   R> anova.noint.mix <- fitmaanova(paigen, model.noint.mix)
   ```

4. F test on different effects. First we want to test the interaction effect. The interaction effect must be tested in the full model. Note that because the interaction effect is nested within biological replicates, we have to use biorep to test the interaction.

   ```
   R> ftest.int.mix <- matest(data=paigen, model=model.full.mix,
             term="Strain:Diet")
   R> idx.int.mix <- volcano(anova.full.mix, ftest.int.mix)
   ```

   Then we want to test the Strain and Diet effect. We use the model without interaction to test the main effect.

   ```
   R> ftest.strain.mix <- matest(data=paigen, model=model.noint.mix,
             term="Strain")
   R> ftest.diet.mix <- matest(data=paigen, model=model.noint.mix,
             term="Diet")
   ```

   We can do volcano plot to visualize the F-test results and pick significant genes. That step is skipped here.

5. Now we can do T-test on any comparison on a given term. The result still contains the F values (with numerator's degree of freedom equals 1). To to do all pairwise comparison on Strain, first make a comparison matrix:

   ```
   R> C <- matrix(c(1,-1,0,1,0,-1, 0,1,-1), nrow=3, byrow=T)
   ```

   Users need to be careful about specifing comparsion matrix. The comparison need to be the linear combination of the rows of the design matrix for fixed terms. Otherwise the comparision will be non-estimable.

   To do the test:

```
R> ttest.strain <- matest(paigen, model.noint.mix, term="Strain",
             Contrast=C, n.perm=500)
```

Because we have three comparisons here, doing volcano will generate three plots.

```
R> volcano(ttest.strain)
```

Again, the clustering analysis is skipped here.

5.3 An affymetric experiment

This is a 18-array affymetric experiment. It used 3 mouse strains, three individuals out of each strain (so there are nine mice) and did two affymetric arrays for each individual. The experimental design is shown in figure 9. I hand picked 500 genes out of it to make the calculation runs faster.

Note that the data visualization and transformation functions in R/maanova are not applicable to one-dye data (at this time). Model fitting and statistical testing functions will work.

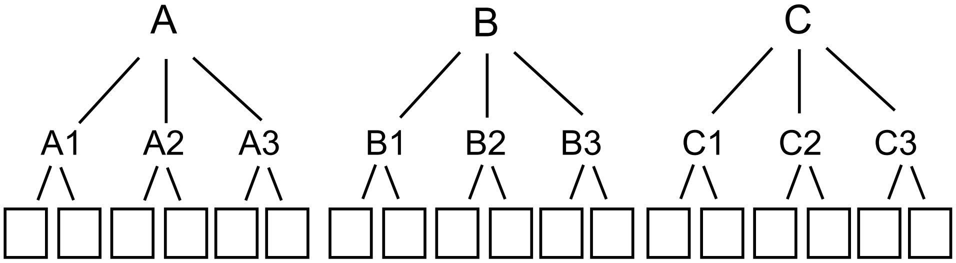


Figure 9: 18-array affymetric experiment design

1. Load in data

```
R> data(abf1)
```

2. Make data object. There is no data transformation function for affymetric array in R/maanova at this time. So this data set was pre-transformed using **Affy** package in **BioConductor**. So we should not take log transformation in making data object.

```
R> abf1 <- createData(abf1.raw, 1, log.trans=F)
```

3. Make model object and fit ANOVA model. We will treat the biological replicates as random effect. Note that you cannot fit Array and Dye effect for one-dye arrays.

```
R> model.full.mix <- makeModel(data=abf1, formula=~Strain+Sample,
         random=~Sample)
R> anova.full.mix <- fitmaanova(abf1, model.full.mix)
```

4. Do F-test on strain and generate volcano plot

```
R> test.Strain.mix <- matest(abf1, model.full.mix, term="Strain",
          n.perm=100)
R> idx.mix <- volcano(test.Strain.mix)
```

References

Y. Benjamin and Y. Hochberg. The control of the false discovery rate in multiple tesing under dependency. *Ann. Stat.*, 29:1665–8, 2001.

G.A. Churchill. Fundamentals of experimental design for cDNA microarrays. *Nat Genet*, 32(Suppl 2):490–5, 2002.

XQ Cui and G.A. Churchill. Statistical Tests for Differential Expression in cDNA Microarray Experiments. *Genome Biology*, 4:201, 2003.

XQ Cui, M.K. Kerr, and G.A. Churchill. Transformation for cDNA Microarray Data. *Statistical Applications in Genetics and Molecular Biology*, 2(1, article 4), 2003.

XQ Cui, G. Hwang, J. Qiu, N. Blades, and G.A. Churchill. Improved Statistical Tests for Differential Gene Expression by Shrinking Variance Components. *Biostatistics*, 6: 59–75, 2005.

J. Felsenstein. Confidence limits on phylogenies - An approach using the bootstrap. *Evolution*, 39:783, 1985.

SAS institute Inc. *SAS/STAT User's Guide.* SAS institute Inc., 1999.

M.K. Kerr and G.A. Churchill. Experimental design for gene expression microarrays. *Biostatistics*, 2:183, 2001a.

M.K. Kerr and G.A. Churchill. Bootstrapping cluster analysis: Assessing the reliability of conclusion from microarray experiments. *PNAS*, 98:8961, 2001b.

M.K. Kerr, M. Martin, and G.A. Churchill. Analysis of variance for gene expression microarray data. *J Computational Biology*, 7:819, 2000.

M.K. Kerr, C.A. Afshari, L. Bennett, P. Bushel, J. Martinez, N. Walker, and G.A. Churchill. Statistical analysis of a gene expression microarray experiment with replication. *Statistica Sinica*, 12:203, 2001.

M.K. Kerr, E.H. Leiter, L. Picard, and G.A. Churchill. *Computational and Statistical Approaches to Genomics*, chapter Sources of Variation in Microarray Experiments. Kluwer Academic Publishers, 2002.

R.C. Littell, G.A. Milliken, W.W. Stroup, and R.D. Wolfinger. *SAS system for mixed models*. SAS institute Inc., 1996.

T Margush and F.R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43:239, 1981.

R.A. McLean, W.L. Sanders, and W.W. Stroup. A Unified Approach to Mixed Linear Models. *The American Statistician*, 45:54–64, 1991.

C.C. Prichard, L. Hsu, J. Delrow, and P.S. Nelson. Project normal: defining normal variance in mouse gene expression. *PNAS*, 98:13266, 2001.

S.R. Searle, G. Casella, and C.E. McCulloch. *Variance Components*. John Wiley and sons, Inc., 1992.

V. Witkovsky. MATLAB algorithm mixed.m for solving Henderson's mixed model equations. *Technical Report, Inst. of Measurement Science, Slovak Academy of Science*, 2001.

R.D. Wolfinger, G. Gibson, E.D. Wolfinger, L. Bennett, H. Hamadeh, P. Bushel, C. Ashfari, and R.S. Paules. Assessing gene significance from cDNA microarray expression data via mixed models. *J Comp Biol*, 8:625, 2001.

Benjamin Y. and Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J Royal Statistical Society*, Series B(57):289, 1995.

Y.H. Yang, S. Dudoit, P. Luu, D.M. Lin, V. Peng, J. Ngai, and T.P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30:e15, 2002.

APPENDIX

A Running R/maanova on computer cluster

Due to the intensive computation needed by the permutation test (especially for mixed effects models), parallel computing is implemented in R/maanova. The permutation tests can run on multiple computer nodes at the same time and the toal computational time will be greatly reduced.

Here I provide some tips for system and software configurations for running R/maanova on clusters. My system is a 32-node beowulf cluster running Redhat linux 8.0 and R 1.8.0. You might encounter problems on different system following the steps.

A.1 System requirements

The computer cluster need to be Unix/Linux cluster with LAM/MPI installed. LAM/MPI can be download from `http://www.lam-mpi.org`. R and `R/maanova` need to be installed on all nodes (or they can be exported from a network file system).

The following R packages are required (you can obtain them from `http://cran.r-project.org`:

- SNOW (Simple Network Of Workstations). Obtain this one from `http://www.stat.uiowa.edu/~luke/R/cluster/cluster.html`.
- Rmpi
- serialize (this is required by Rmpi)

A.2 System setup

You might need to ask your system administrator to do this. But basically, you need to install LAM/MPI software in your system with correct configuration. Here are some tips for using rsh as remote shell program. If you want to use ssh, things will be slightly different. I assume LAM/MPI has been installed on your system.

1. First thing to do is to check if you can "rsh" to other machine without being asked for a password. If it does, add a file called ".rhosts" in your home directory. Each line of the file should be mahine names and your user name. For a 2-node system, that file could look like (assume my account name is hao):
   node1 hao
   node2 hao
   Then set the mode of this file to be 644 by doing "chmod 644 .rhosts". After setting it up, do "rsh node1". If it still asks for password, contact your system administrator.

2. Configurate LAM to use rsh as the remote shell program by adding the following lines in your startup script (.bashrc, .cshrc, etc.)
   LAMRSH=”rsh”
   export LAMRSH
   (I found it’s okay if you don’t do this!)

3. Start LAM/MPI by typing ”lamboot -v”. You should see something like:
   LAM 6.5.6/MPI 2 C++/ROMIO - University of Notre Dame
   Executing hboot on n0 (node1 - 1 CPU)...
   Executing hboot on n1 (node2 - 1 CPU)...
   topology done

   If you see some error messages, contact your system administrator.

A.3 Install and test clusters in R

Here are the steps to install and use clusters in R.

1. Install all required package in R.

2. Start R, load in snow by doing ”library(snow)”.

3. To test if cluster is working, do the following:

   ```
   R> library(snow)
   R> cl <- makeCluster(2) # to make a cluster with two nodes
   R> # to see if your cluster is correct
   R> clusterCall(cl, function() Sys.info()[c("nodename","machine")])
   ```

   You’re supposed to see the node names and machine types like what I got here:

   ```
   [[1]]
   nodename               machine
   "node1"                "i686"
   [[2]]
   nodename           machine
   "node2"            "i686"
   ```

4. Check the random number generations by running

   ```
   clusterCall(cl, runif, 3)
   ```

   If there is correlation problem, you must install rsprng.

That’s it! Now you can start to enjoy the parallel computing.

B Basic Algorithms

I will briefly introduce the core algorithms implemented in `R/maanova`. For the details please read related references.

B.1 ANOVA model for microarray experiment

B.1.1 Model

The ANOVA model for microarray experiment was first proposed in Kerr *et al* (2000). Wolfinger *et al* (2001) proposed the 2-stage ANOVA model, which is the one used in `R/maanova`.

Basically an ANOVA model for microarray experiment can be specified in two stages. The first stage is the normalization model

$$y_{ijkgr} = \mu + A_i + D_j + AD_{ij} + r_{ijkgr} \tag{1}$$

The term μ captures the overall mean. The rest of the terms capture the overall effects due to arrays, dyes, and labeling reactions. The residual of the first stage will be used as the inputs for the second stage.

The second stage models gene-specific effects:

$$r_{ijkr} = G + AG_i + DG_j + VG_k + \epsilon_{ijkr} \tag{2}$$

Here G captures the average effect of the gene. AG captures the array by gene variation. DG captures the dye by gene variation. For one dye system there will be no DG effect in the model. VG captures the effects for the experimental varieties. In the multiple factor experiment such like Paigen's 28-array experiment, VG should be further decoupled into several factors.

B.1.2 Fixed model versus Mixed model

The fixed effect model assumes independence among all observations and only one source of random variation. Although it is applicable to many microarray experiments, the fixed effects model does not allow for multiple sources of variation nor does it account for correlation among the observations that arise as a consequence of different layers of variation.

The mixed model treats some of the factors in an experimental design as random samples from a population. In other words, we assume that if the experiment were to be repeated, the same effects would not be exactly reproduced, but that similar effects would be drawn from a hypothetical population of effects. Therefore, we model these factors as sources of variance. Usually Array effect (AG) should be treated as random factor in ANOVA model. If you have biological replicates, it should be treated as random as well.

B.2 Hypothesis tests

For statistical test in the fixed ANOVA model, there is only one variance term and all factors in the model are tested against this variance. In mixed model ANOVA, there are multiple levels of variances (biological, array, residual, etc.). The test statistics need to be constructed based on the proper variations. I will skip the details here and the interested reader can read Searle *et al.*

B.2.1 F test in matrix notation

Mathematically speaking, a Microarray ANOVA model for a gene can be expressed as the following mixed effect linear model (without losing generality):

$$y = X\beta + Zu + e \tag{3}$$

Here, β and u are vectors for fixed-effects and random-effects parameters. X and Z are design matrices. Note that fixed effects model is a special case for mixed effects model with Z and u empty. An assumption is that u and e are normally distributed with

$$E\begin{bmatrix} u \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4}$$

$$Var\begin{bmatrix} u \\ e \end{bmatrix} = \begin{bmatrix} G & 0 \\ 0 & R \end{bmatrix} \tag{5}$$

G and R are unknown variance components and are estimated using restricted maximum liklihood (REML) method. The estimated $\hat{\beta}$ and $\hat{u}$ are called **best linear unbiased estimator** (BLUE) and **best linear unbiased predictor** (BLUP) respectively. Using estimated G and R, the variance-covariance matrix of $\hat{\beta}$ and $\hat{u}$ can be expressed as

$$\hat{C} = \begin{bmatrix} X'\hat{R}^{-1}X & X'\hat{R}^{-1}Z \\ Z'\hat{R}^{-1}X & Z'\hat{R}^{-1}Z + \hat{G}^{-1} \end{bmatrix}^{-} \tag{6}$$

where $^{-}$ denotes generalized inverse. After getting $\hat{C}$, test statistics can be obtained through the following formula. For a given hypothesis

$$H : L\begin{bmatrix} \beta \\ u \end{bmatrix} = 0 \tag{7}$$

F-statistic is:

$$F = \frac{\begin{bmatrix} \hat{\beta} \\ u \end{bmatrix}' L'(L\hat{C}L')^{-}L\begin{bmatrix} \hat{\beta} \\ u \end{bmatrix}}{q} \tag{8}$$

Here q is the rank of L matrix.

Notice that for a fixed effect model where Z and u are empty, equation 8 will become

$$F = \frac{(L\hat{\beta})'(L(X'X)^{-}L')^{-1}L\hat{\beta}}{q\sigma_e^2} \tag{9}$$

Here σ_e^2 is the error variance.

F approximately follows F-distribution with q numerator degrees of freedom. The calculation of denominator degrees of freedom can be tricky and I will skip it here. If there is not enough degree of freedom, rely on tabulated P-values can be risky and a permutation test is recommended.

B.2.2 Building L matrix

In equation 8, L must be a matrix that $L\beta$ is estimable. Estimability requires that the rows of L must be the linear combinations of the rows of X. In `R/maanova`, the program will automatically build L matrix for the term(s) to be tested. Normally in a mixed effects model, only the fixed effect terms can be tested so the u portion of L contains all 0s.

Building L matrix can be difficult. If the term to be tested is orthogonal to all other terms, L should contain all pairwise comparision for this term. That is, if the term has N levels, L should be a $(N-1)$ by N matrix. If the tested term is confounded with any other term, things will become complicated. `R/maanova` does the following to compute L matrix:

1. calculate a generalized inverse of $X'X$ in such a way that the dependent columns in X for the tested term are set to zero.

2. calculate $(X'X)^{-}(X'X)$. The result matrix span the same linear space as X and it contains a lot of zeros.

3. Take the part of $(X'X)^{-}(X'X)$ corresponding to the tested term, remove the rows with all zeros and the left part is L.

B.2.3 Four flavors of F-tests

`R/maanova` offers four F statistics (called F_1, F_2, F_3 and F_s). Users have the option to turn on or off any of them. The difference among them is the calculation of $\hat{C}$ matrix in equation 8. Or in another word, the way to estimate the variance components $\hat{R}$ and $\hat{G}$.

Briefly speaking, F_1 computes $\hat{C}$ matrix based on the variance components of a single gene. It does not assume the common variance among the genes. F_3 assumes the common variance among the genes. The $\hat{C}$ matrix will be calculated based on the global variance (mean variance of all genes). F_2 test is a hybrid of F_1 and F_3. It uses a weighted combination of global and gene-specific variance to compute the $\hat{C}$ matrix. F_s uses the James-Stein estimator for $\hat{R}$ and $\hat{G}$ and computes $\hat{C}$ matrix. For details about four F-tests, please read Cui (2003b) and Cui (2003c).

B.3 Data shuffling in permutation test

Choosing proper data shuffling method is crucial to permutation test. `R/maanova` offers two types of data shuffling, residual shuffling and sample shuffling.

Residual shuffling works only for fixed effects models. It is based on the assumption of homogeneous error variance. It does the following:

1. Fit a null hypothesis ANOVA model and get the residuals
2. Shuffle residuals globally and make a new data set
3. Compute test statistics on the new data set
4. Repeat step 2 and 3 for a certain times

Residual shuffling is incorrect for mixed effect models, where you have multiple random terms. For a mixed effects model, `R/maanova` will automatically choose sample shuffling, which does the following:

1. For the given tested term, check if it is nested within any random term. If not, go to step 4.
2. Choose the lowest random term (among the ones nesting with the tested term) as the base for shuffling. If there is only one random term nesting with the tested term, this random term will be the shuffle base.
3. Shuffle the sample names for the tested term in such a way that the same shuffle base will corresponds to the same sample name.
4. If there is no nesting, shuffle the sample names for the tested term freely, keeping all other terms unchanged. For multiple dye arrays, if Array and/or Dye effects are included in the model, the array structure should be perserved, e.g., the sample names on the same array should be shuffled together.
5. repeat step 3 or 4 for a certain times

Note that if the experiment size is small, the number of possible permutations will not be sufficient. In that case, users will have to rely on the tabulated values.

C Frequently Asked Questions

1. Can R/maanova run on Windows XP?
   R/maanova runs under R. So if there is a version of R for your operating system, you can run R/maanova.

2. Can R/maanova handle missing data?
   No. R/maanova does not tolerant missing, zero or negative intensity data. Missing data will bring many problems. A gene with missing data will have a different experimental design from others. That will cause problem in permutation tests. So if you have any data missing, you must manually remove all data associated with that gene and all its replicates. We suggest you use non-background subtracted data as input.

3. Can R/maanova handle affymetric arrays?
   Yes. The newest version of R/maanova (0.97) works for N-dye system. But the data visualization and transformation functions are only working for 2-dye arrays at this time. So for affymetric data, I suggest you to use some other packages (such like affy package in BioConduction) to do the data transformation before loading into R/maanova.

4. Can R/maanova handle unbalanced design?
   Yes.

5. Can R/maanova output an ANOVA table for a model fitting?
   Not at this time. We used to output an ANOVA table in the older version of R/maanova. But as things getting complicated (with multiple factor design, mixed effects models, etc.), it becomes more and more difficult to build an ANOVA table. We might think about it in the future.

6. What does R/maanova do with flagged genes?
   We did not do anything special to the flagged genes. They will be involved in data transformation and analysis. We just keep the flag in there so that if some of your significant genes were flagged, you need to put a question mark on it because the significance could be from a scratch on the slide.

7. Why I have problems reading in data?
   First of all data reading could behave differently in different operating system. Your data file need to be simple (ANSI) text file. Sometimes special characters (such like #, TAB, etc.) in the data file will cause problems. If you encounter that problem, remove the columns contains a lot of special characters (usually gene description) and try again. If you still have problem, contact the software author.

8. Why the estimated effects for a term don't sum to zero?
   When we build the design matrices we didn't put sum to zero constraints so that

the estimated effects for a term will not sum to zero. But the relative values of the estimates (which is the one we care about) will be correct.

9. Is there a graphical user interface for `R/maanova`?
   Yes. We are developing a GUI for `R/maanova` using Java. It will be available in the near future.