1 Prerequisites

R and BioConductor

- R is an advanced statistical programming language and data analysis environment. From programming point of view, it bears many similarities with MATLAB.
- R is free software. You can obtain and install it from . If you run Linux, I recommend you install it from your package management system.
- BioConductor is a large collection of R libraries designed for analyzing high-throughput genomic data.

Installing BioConductor

Install BioConductor from R is simple

```
# source("http://bioconductor.org/biocLite.R")
# biocLite()
```

Once BioConductor is installed, you can install specific package by the same `biocLite()` function

```
# biocLite(c("ALL", "genefilter", "GOstats", "samr", "multtest", "GEOquery"))
```

2 Welcome to the world of *-omics data!

Gene expressions

- The most commonly used bioinformatics data are some form of gene expressions, measured by techniques such as microarrays (RNA/DNA microarrays, exonarray, beads array, etc) and RNA-seq (miRNA-seq, ribosomal profiling, etc).
- While microarray and RNA-seq technologies use very different approaches, at the end of the day, they both can report expression levels as a $m \times n$ dimensional matrix.
- Here m is the number of units such as genes or exons, n is the number of samples.
- Typically m is in the range of $20 \sim 50$ thousands, and n is in the range of $10 \sim 100$. This is a typical "large p, small n" scenario in statistical analysis.

*Other *-omics data (I)*

- Though we focus on gene expression data analysis in this talk, we would like to point out that many other *-omics data exist.
- PCR (Polymerase chain reaction) data. Low throughput (cheaper); high sensitivity; often used *after* high-throughput gene expression analysis to *confirm* differentiation of specific genes.
- Protein binding microarray (*a.k.a.* biochip, proteinchip). Records protein instead of DNA/RNA expressions. Good for identifying protein-protein interactions; transcription factor protein-activation; antibody measurements.

Microarray Data pre-processing

- Raw scanner level data on Affymetrix platform are images with "`.CEL`" suffixes. `R/BioConductor` provides a function `ReadAffy()` to read these CEL files into `R`.
- The default preprocessing method provided by `R/BioConductor` is RMA (Robust Multichip Average) which includes
  1. Background correction at the image level.
  2. Quantile normalization, which calibrates all arrays to the same scale (all quantiles must be the same).
  3. Summarization. Multiple $(15-25)$ probes are used to detect one target so they need to be summarized to one number.
  4. Variance stabilization transformation. Most common VST: log2 transformation[1].

RNA-seq data pre-processing

- Acquire the raw sequence data. For example, on Illumina platform, this initial step produces (huge) `FASTQ` files.
- Alignment. In the second step, these raw sequences are aligned to a well annotated reference genome. If one piece of sequence is aligned to a known genetic unit (exon, gene, or splice junctions), we count one "match". Tools: TopHat, BowTie2, etc. Output file: `BAM` or `SAM` file.
- `Rsamtools` to work with BAM files; `Rsubread` to turn these files into counting matrix.
- Normalization. Note that once you normalize the counts, most tools that are designed specifically for RNA-seq, such as `DESeq2`, no longer apply. Instead, you should use microarray-oriented tools.

3 Now we have the data. Which genes are "interesting"?

Load the ALL data

Let us load a sample data (`ALL`, Acute Lymphoblastic Leukemia) first.

```
## biocLite("ALL")
library(ALL)       #This is a data library
data("ALL")        #a 12,625 by 128 dim matrix
print(ALL)         #print out some useful information
```

Object: `ExpressionSet`; features (genes/probesets): 12625; sample size: 128.

Select a subset of arrays

Assume that we want to select only sample from B-cell lymphomas harboring the BCR/ABL translocation and from lymphomas with no observed cytogenetic abnormalities (NEG).

```
## BT is the variable of ALL which distinguish B cells
## from T cells.
## Show in another window: as.character(ALL[["BT"]])
bcell <- grep("^B", as.character(ALL[["BT"]]))
```

[1] Average in the log-scale is the *geometric mean*; multiplicative noise becomes additive in the log-scale, etc.

```
## select molecular types of BCR/ABL and NEG
molbiol <- as.character(ALL[["mol.biol"]])
moltype <- which(molbiol %in% c("BCR/ABL", "NEG"))

data1 <- ALL[, intersect(bcell, moltype)]
## drop unwanted levels of mol.biol
data1[["mol.biol"]] <- factor(data1[["mol.biol"]])
```

The result, `data1`, has 79 arrays (42 NEG, 37 BCR/ABL).

Nonspecific filtering

- Filter out genes *before* statistical analysis. The filtering criteria often include basic quality control characteristics such as minimum expression level and standard deviation (or interquartile range).

```
library(genefilter)
## filter based on expression levels
filter1 <- rowMax(exprs(data1))>=4.0

## filter based on standard deviation
filter2 <- rowSds(exprs(data1))>=0.25
data2 <- data1[filter1 & filter2, ]
```

- The result, `data2`, has 79 arrays and 8411 genes.

Hypothesis testing

In principle, any parametric or nonparametric statistic designed for single hypothesis testing may be useful in microarray. In practice, the following test statistics are wided used in microarray data analysis:

- Two sample student t-statistic
- Wilcoxon rank-sum statistic
- Kolmogorov-Smirnov statistic
- Cramér-von Mises statistic
- N-statistic

Two sample student t-statistic

- Two sample student t-statistic is a parametric statistic. In order to compute the p-value associated with a t-statistic from the standard t-distribution, we need to assume the underlying distribution of expression levels are normally distributed.
- Remark: You can still compute t-statistic when data are not normally distributed. But you can not obtain a valid p-value by looking up the *standard t-distribution*.
- In fact, a permutation version of t-test is valid even for non-normal data.

Two sample student t-statistic

$$t_i = \frac{\bar{\mathbf{x}}_i^{(A)} - \bar{\mathbf{x}}_i^{(B)}}{\sqrt{s_i^2(\frac{1}{n_A} + \frac{1}{n_B})}}. \tag{1}$$

where $\bar{\mathbf{x}}_i^{(A)}$, $\bar{\mathbf{x}}_i^{(B)}$ represents the mean expressions of the ith gene in two phenotypic groups (A and B); n_A, n_B are the number of arrays of each phenotype; s_i^2 is an estimated variance pooled from both groups.

Fold difference, and why it is bad

Fold difference[2] is simply the difference between $\bar{x}$ and $\bar{y}$.

Data 1 $\mathbf{x}^{(A)} = \{1,2,3\}$, $\mathbf{x}^{(B)} = \{4,5,6\}$,

Data 2 $\mathbf{x}^{(A)} = \{10,20,30\}$, $\mathbf{x}^{(B)} = \{40,50,60\}$.

Fold differences: 3 for Data 1, 30 for Data 2. t-statistics: -3.6742 for both data sets. p-values: 0.02131 for both data sets. [7, 5]

SAM (significant analysis of Microarrays)

A compromise: the SAM (significant analysis of Microarrays) approach [8].

$$t_{i,\mathrm{SAM}} = \frac{\bar{\mathbf{x}}_i^{(A)} - \bar{\mathbf{x}}_i^{(B)}}{\sqrt{(s_i^2 + s_0^2)(\frac{1}{n_A} + \frac{1}{n_B})}}. \tag{2}$$

where s_0^2 is estimated from the other genes and can be considered as a) an estimate of the variance of background noise; b) a way to stabilize the signal-to-noise ratio.

Wilcoxon rank-sum statistic

- The Wilcoxon rank-sum statistic is a nonparametric alternative to t-statistic:

$$W_i = \sum_{j=1}^{n_1} r(x_{ij}^{(A)}) \tag{3}$$

- Where $r(x_{ij}^{(A)})$ is the rank of $x_{ij}^{(A)}$ in the combined sample.
- The key point: only ranking information are needed, so
  1. p-value computed from this statistic is valid even when the distribution of the expression levels are not normal. [1]
  2. outliers have less effect to inference.

[2]Fold difference is similar but not identical to fold change due to log2 transformation.

Wilcoxon rank-sum statistic

- For both Data 1 and Data 2, $W = 6$, $p = 0.1$. Notice that, this p-value is greater than the one computed from the t-statistic (0.02131).
- Generally speaking, nonparametric tests are less powerful than their parametric counterparts, because we assume that we have some *a priori* information about the underlying distribution in the parametric case.

Other advanced statistics

- Kolmogorov-Smirnov statistic: a distribution free statistic which can be thought of as a *distance* between two empirical distribution functions. Its parametric counterpart is the χ^2-statistic.
- Cramér-von Mises statistic: a distribution free statistic which was first designed to test the normality of a given empirical distribution, it can also be used as a *distance* between two empirical distribution functions.
- The granularity of this statistic (which causes inaccuracy of the corresponding p-values) is smaller than that for the Kolmogorov-Smirnov statistic, but this statistic and its associate p-value is much harder to compute. [10]

Permutation N-test

- Another nonparametric test based on the N-distance between two empirical distribution functions.
- it is a family of distribution functions depending on the *kernel* being used. We can use kernels that are more sensitive to a certain kind of departure to get better power.
- For instance, by using L^1 kernel, this statistic is more sensitive to a departure in shifting. [4].
- This is also used in differential association studies [3, 2, 6].

Hypothesis testing (using t-statistics)

```
tt <- rowttests(data2, "mol.biol")
```

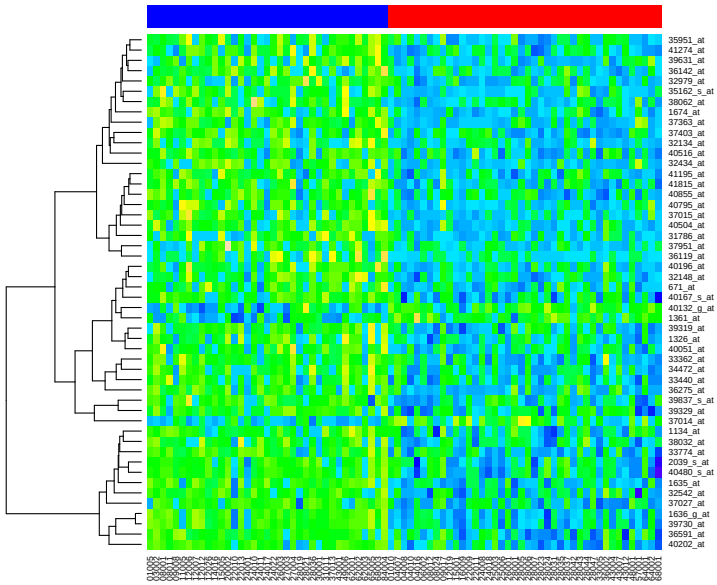
Report top 5 differentially expressed probe sets and their gene symbols.

```
tt.sorted <- tt[order(tt[["p.value"]]), ]
top5 <- rownames(tt.sorted)[1:5]
library("hgu95av2.db")
top5.table <- cbind(links(hgu95av2SYMBOL[top5]),tt.sorted[1:5,])
print(xtable(top5.table), file="results/table-top5.tex")
```

Top differentially expressed genes

	probe_id	symbol	statistic	dm	p.value
1	1635_at	ABL1	9.26	1.10	0.00
2	1636_g_at	ABL1	8.69	1.15	0.00
3	1674_at	YES1	7.28	1.20	0.00
4	39730_at	ABL1	6.90	1.43	0.00
5	40504_at	PON2	6.57	1.18	0.00

A heatmap of the top 50 DEGs



SAM example

```
## The phenotypic information contained in data2
pheno2 <- pData(data2)
## organize the expressions, grouping, genenames, whether
## data have been log2 transformed into one list.
input=list(x=exprs(data2), y=pheno2[["mol.biol"]],
    geneid=as.character(1:nrow(data2)),
    genenames=paste("g",as.character(1:nrow(data2)),sep=""),
    logged2=TRUE)
## Please use 1000 permutations in a real application
samr.obj <- samr(input, resp.type="Two class unpaired", nperms=100)
## detect significant genes
delta.table <- samr.compute.delta.table(samr.obj,
                    min.foldchange=0.1, nvals=200)
siggenes.table <- samr.compute.siggenes.table(samr.obj,
                  del=0, input, delta.table, all.genes=TRUE)
sum(as.numeric(siggenes.table$genes.up[, "q-value(%)"])<5)
sum(as.numeric(siggenes.table$genes.lo[, "q-value(%)"])<5)
```

RNA-seq data, SAM

```
## please load pasilla.Rdata from the net-drive
load("pasilla.Rdata")
head(pasillaExp)
pasillaPheno
### Note that we don't need logged2=TRUE
input=list(x=pasillaExp, y=pasillaPheno,
    geneid=as.character(1:nrow(pasillaExp)),
    genenames=rownames(pasillaExp))
## Notice the assay.type arguement
samr.obj<-samr(input,  resp.type="Two class unpaired",
               assay.type="seq", nperms=100)
delta.table <- samr.compute.delta.table(samr.obj)
siggenes.table<-samr.compute.siggenes.table(samr.obj, del=0, input, delta.table)
```

RNA-seq, DESeq2

```
# object construction
dds <- DESeqDataSetFromMatrix(pasillaExp, DataFrame(pasillaPheno), ~ pasillaPheno)
# standard analysis
dds <- DESeq(dds)
res <- results(dds)
head(res)
sum(res[, "padj"]<0.05, na.rm=TRUE)
```

4 How to control false discoveries for 20,000 comparisons

Type I, II error, testing power

	Accept H_0	Reject H_0
H_0	True Negative	False Positive, type I error)
H_1	False Negative, type II error	True Positive)

Per-Comparison Error Rate (PCER)

Per-comparison error rate, or PCER is the probability of making the type I error in a single test. Under most common scenario, this is precisely determined by the threshold of p-value to reject H_0 (the significance level of the test).

4.1 Family-wise Error Rate

Definition of FWER

From now on we assume that we have m hypotheses (whether a given gene is diff. exp. or not) to be tested simultaneously. By definition, FWER is the probability of producing one or more false positives

$$FWER = Pr(FP \geq 1) \tag{4}$$

Why controlling FWER is important

- controlling FWER at a significance level α gives us $1-\alpha$ confidence to say that there is no mistake in the outcome.
- Suppose you have done t-test for 20,000 genes. At significance level $\alpha = 0.05$, how many genes will be identified as DEG *by chance*?

Bonferroni adjustment

Bonferroni adjusted p-value: $p_i^{bon} = \min(m \times p_i, 1)$ For example, suppose raw p-value is 0.0428 and $m = 20$, then the Bonferroni adjusted p-value is 0.856. A statistical test based on the Bonferroni adjusted p-values rather than the unadjusted p-values is called the Bonferroni procedure.

Relationship to PFER

A related quantity is the per-family error rate (PFER), the expected number of false positives.

$$PFER = E(FP) \tag{5}$$

PFER is always greater or equal to FWER. So any procedure (including Bonferroni) which controls PFER controls FWER.

Sidak, Holm, etc

Some other less conservative FWER controlling procedures exist. These procedures usually adjust p-values step-wisely, i.e., they order the p-values first, then adjust p_i one by one according to the rank of p_i. They may further divided into step-down procedures and step-up procedures.

Correlation and FWER

- If we know the relationship of the hypotheses, we may be able to utilize this information to get a better procedure (that is, more powerful procedure which controls the same nominal FWER level).
- Westfall and Young procedure uses permutation to generate the null distribution. Thus keeps the correlation structure of hypotheses. It is time consuming but more powerful than Bonferroni. See [9].

Bonferroni

Bonferroni is simple but conservative.

```
pvals <- tt[["p.value"]]
pvals.bonf <- p.adjust(pvals, "bonferroni")
sum(pvals.bonf<0.05)

## [1] 26
```

It finds 26 DEGs.

Westfall and Young

Westfall and Young permutation test is a more powerful alternative

```
library(multtest)
cl <- data1[["mol.biol"]]=="NEG" #class labels
rr <- mt.maxT(exprs(data2), cl, B=1000)

## to get back to the original ordering
ord <- order(rr$index)

## W-Y adjusted p-values
pvals.wy <- rr$adjp[ord]
sum(pvals.wy<0.05)
```

It finds 31 DEGs.

4.2 False Discovery Rate

Definition of the False Discovery Rate (FDR)

the False Discovery Rate (FDR) is defined as (roughly) the expected value of the ratio of the false positives among all positives.

$$FDR = E(Q) \tag{6}$$

Where $Q = \frac{FP}{FP+TP}$ when there is at least one positive, or zero otherwise (to avoid $0/0$).

Example

Suppose we have m hypotheses to be tested. The associated p-value for gene i is denoted as p_i. Then a test designed to control FDR at level 0.05 guarantees that out of 20 discoveries, only one or less is a false discovery on average.

FDR v. FWER/FCER

Like PFER, FDR is an *expected number*. While both FWER and FCER based procedures controls certain *probability* and fit well with the traditional significance testing framework, FDR controlling procedures controls an expected value that has a very different statistical meaning.

Instability of FDR

FDR is the expectation of a ratio. When we expect the denominator (total discoveries) to be small, small variation in the numerator/denominator can result in huge variation of the quotient.

False discovery rate (example)

Using the Benjamini-Hochberg procedure to control FDR.

```
pvals.fdr <- p.adjust(pvals, "BH")
sum(pvals.fdr<0.05)

## [1] 199
```

It finds 199 DEGs.

5 Machine learning techniques

Discriminant analysis (supervised machine learning)

- Select a manageable set of features. Usually done by statistical significant test.
- Standardize these features to the same scale.
- Split the data into two sets, one (larger) for training and one (smaller) for testing.
- Train a discriminant function on the training set, and then apply the formula to the testing set.
- Repeat for many times.

Linear discriminant analysis

- The simplest approach is linear discriminant analysis.
- Underline assumptions:
  1. the expressions of DEGs follow two multivariate normal distributions with the in different phenotypic groups.
  2. covariance structures are the same in these phenotypic groups.
  3. mean expressions are different

$$\mathbf{x}^{(A)} \sim MVN\left(\boldsymbol{\mu}^{(A)}, \Sigma\right), \quad \mathbf{x}^{(B)} \sim MVN\left(\boldsymbol{\mu}^{(B)}, \Sigma\right). \tag{7}$$

- Under these assumptions, the best data driven method to separate these two distributions is a hyperplane (a linear subspace).

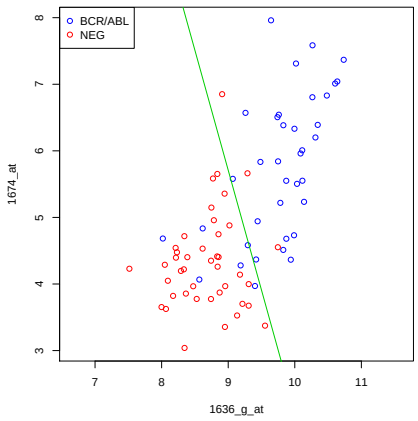
Linear discriminant analysis example

```
## The correlations between the 1st, 2nd, and 3rd top DEGs
## are too high. So I use the 1st and 4th genes for
## better visualization

top2 <- t(exprs(top50set)[c(1,4),])
rr.lda <- lda(top2, top50set[["mol.biol"]])
```

	BCR/ABL	NEG
BCR/ABL	29	8
NEG	3	39

Table 1: Results of linear discriminant analysis. Cross-validation is used for evaluate true/false predictions.



Logistic Regression

- Alternatively we can use multiple logistic regression analysis.
- Logistic regression is an example of *generalized linear regression*.
- For more information you can check out the `glm()` function in `R`.

Nonlinear discriminant analysis

- For *nonlinear* discriminant analysis, support vector machine (SVN) and K-nearest neighbor methods are the most popular choices.
- They are implemented in packages `svn` and `MLinterfaces`.
- Another alternative is random forest, but it has a tendency to *overfit* a continuous model.

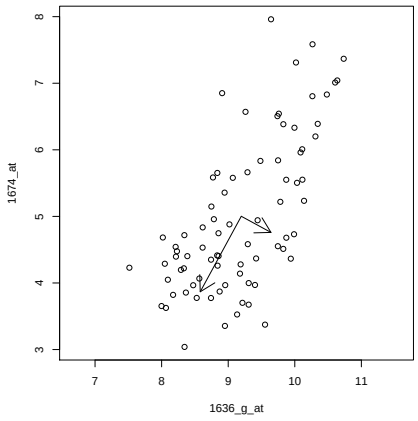
Principal component analysis (unsupervised learning)

- Genes are known to form functionally related sets, which means they could be highly correlated.
- PCA uses an orthogonal transformation to convert correlated gene expressions into a set of *uncorrelated* variables called principal components.
- Since PCs are uncorrelated, we can divide the total variance into variance components explained by each PC.
- Empirical evidence shows that just a few PCs can explain 95% of total variance.
- Geometrically speaking, this means the observations fit a low dimensional hyperplane very well.

To compute the principal components are easy

```
rr.pca <- prcomp(top2)
```

The explained standard deviation (shown as the length of vectors): 1.291 (first PC); 0.508 (second PC).



Cluster analysis (unsupervised learning)

- Feature selection. You can use the top M genes based on p-value, or the first few principal components computed from all significant genes. The bottom line: no more than 10 features otherwise you run the risk of overwhelming the cluster algorithms.
- Feature standardization.
- Distance/similarity measure. K-means uses the Euclidean distance. SK-means uses spherical distance.
- Determine how many clusters. AIC/BIC
- Run the actual cluster analysis.

```
## compute PCs
pcs <- prcomp(t(exprs(top50set)))

## take only the first 2 PCs as features
pc2 <- pcs[["rotation"]][, 1:2]

## feature standardization
pc2b <- scale(pc2)

## $K$-means clustering
rr.kmeans <- kmeans(pc2b, 2)
```

```
## Mclust() is a model based clustering method.
rr.mclust <- Mclust(pc2b)
```

6 Beyond analyzing probe sets

Gene annotation (I)

Gene annotation is tricky because

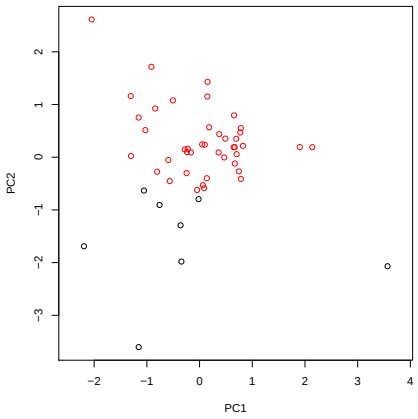


Figure 1: *K*-means clustering

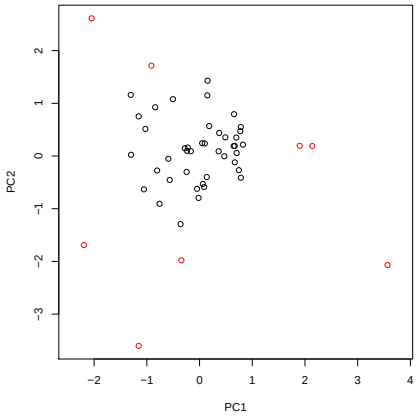


Figure 2: Mclust clustering

- Once significant probe sets are identified, we need to map them to the corresponding genes and obtain useful information about these genes.
- This mapping is nontrivial because
  1. Multiple probe sets per gene.
  2. Organism level information are grouped in different (and numerous) libraries.
  3. Platform dependency.

Gene annotation (II)

- Probe set IDs (`sig.probes <- rownames(top50set)` or `sig.probes <- featureNames(top50set)`).
- Entrez IDs: `links(hgu95av2ENTREZID[sig.probes])`.
- Unigene IDs: `links(hgu95av2UNIGENE[sig.probes])`.
- Gene Symbols: `links(hgu95av2SYMBOL[sig.probes])`.
- GO IDs: `links(hgu95av2GO[sig.probes])`.
- KEGG pathway IDs: `links(hgu95av2PATH[sig.probes])`.
- Chromosome start and stop locs: **CHRLOC** and **CHRLOCEND**.

Functional Enrichment Analysis

- Genes are grouped into sets, such as biological pathways or GO terms.
- The rationale of the hypergeometric test.
- The implementation. See the next slide.

Enrichment analysis example

```
## sig.probes are the significant probe sets
library(GOstats)
sig.probes <- rownames(tt)[pvals.fdr<0.1]
id.table <- links(hgu95av2ENTREZID[sig.probes])
sig.ids <- unique(id.table[,"gene_id"])

## entrezUniverse is the set of all (DEGs and NDEGs)
## probe sets.
entrezUniverse <- unlist(mget(rownames(tt),
                              hgu95av2ENTREZID))

params <- new("GOHyperGParams", geneIds=sig.ids,
              universeGeneIds=entrezUniverse,
              annotation="hgu95av2.db",
              ontology="BP",
              pvalueCutoff=0.001,
              conditional=FALSE,
              testDirection="over")
```

```r
## Run the hyper-geometric test for significance.
rr.gsea <- hyperGTest(params)

## Run a reasonable multiple testing procedure
pp.gsea <- pvalues(rr.gsea)
pp.gsea.bh <- p.adjust(pp.gsea, "BH")

## Output a table as report.
nn <- sum(pp.gsea.bh<0.05)
tab.gsea <- summary(rr.gsea,pvalue=pp.gsea[nn+1])
write.csv(tab.gsea, file="results/table-gsea.csv")
```

63 pathways are significant. Show the `table-gsea.csv` file.

Other advanced techniques

- Gene set enrichment analysis. Its relationship to enrichment analyses based on discrete tests such as hypergeometric test. Package: `limma`, function `camera()`.
- Network reconstruction. Bayesian network; boolean network.
- The use of ODE in network analysis for time-course data.
- Differential association analysis.

Bibliography

References

[1] Jean Dickinson Gibbons and Subhabrata Chakraborti, *Nonparametric statistical inference*, Marcel Dekker, Inc, 1992.

[2] Rui Hu, Xing Qiu, and Galina Glazko, *A new gene selection procedure based on the covariance distance*, Bioinformatics **26** (2010), no. 3, 348.

[3] Rui Hu, Xing Qiu, Galina Glazko, Lev Klebanov, and Andrei Yakovlev, *Detecting intergene correlation changes in microarray analysis: a new approach to gene selection*, BMC Bioinformatics **10** (2009), no. 1, 20.

[4] L. Klebanov, A. Gordon, Y. Xiao, H. Land, and A. Yakovlev, *A permutation test motivated by microarray data analysis*, Computational Statistics and Data Analysis (2005).

[5] Lev Klebanov, Xing Qiu, Stephen Welle, and Andrei Yakovlev, *Statistical methods and microarray data.*, Nat Biotechnol **25** (2007), no. 1, 25–6; author reply 26–7.

[6] Mark Needham, Rui Hu, Sandhya Dwarkadas, and Xing Qiu, *Hierarchical parallelization of gene differential association analysis*, BMC Bioinformatics **12** (2011), no. 1, 374.

[7] Leming Shi, Laura H Reid, Wendell D Jones, Richard Shippy, Janet A Warrington, Shawn C Baker, Patrick J Collins, Francoise de Longueville, Ernest S Kawasaki, Kathleen Y Lee, Yuling Luo, Yongming Andrew Sun, James C Willey, Robert A Setterquist, Gavin M Fischer, Weida Tong, Yvonne P Dragan, David J Dix, Felix W Frueh, Frederico M Goodsaid, Damir Herman, Roderick V Jensen, Charles D Johnson, Edward K Lobenhofer, Raj K Puri, Uwe Schrf, Jean Thierry-Mieg, Charles Wang, Mike Wilson, Paul K Wolber, Lu Zhang, Shashi Amur, Wenjun Bao, Catalin C Barbacioru, Anne Bergstrom Lucas, Vincent Bertholet, Cecilie Boysen, Bud Bromley, Donna Brown, Alan Brunner, Roger Canales, Xiaoxi Megan Cao, Thomas A Cebula, James J Chen, Jing Cheng,

Tzu-Ming Chu, Eugene Chudin, John Corson, J Christopher Corton, Lisa J Croner, Christopher Davies, Timothy S Davison, Glenda Delenstarr, Xutao Deng, David Dorris, Aron C Eklund, Xiao hui Fan, Hong Fang, Stephanie Fulmer-Smentek, James C Fuscoe, Kathryn Gallagher, Weigong Ge, Lei Guo, Xu Guo, Janet Hager, Paul K Haje, Jing Han, Tao Han, Heather C Harbottle, Stephen C Harris, Eli Hatchwell, Craig A Hauser, Susan Hester, Huixiao Hong, Patrick Hurban, Scott A Jackson, Hanlee Ji, Charles R Knight, Winston P Kuo, J Eugene LeClerc, Shawn Levy, Quan-Zhen Li, Chunmei Liu, Ying Liu, Michael J Lombardi, Yunqing Ma, Scott R Magnuson, Botoul Maqsodi, Tim McDaniel, Nan Mei, Ola Myklebost, Baitang Ning, Natalia Novoradovskaya, Michael S Orr, Terry W Osborn, Adam Papallo, Tucker A Patterson, Roger G Perkins, Elizabeth H Peters, Ron Peterson, Kenneth L Philips, P Scott Pine, Lajos Pusztai, Feng Qian, Hongzu Ren, Mitch Rosen, Barry A Rosenzweig, Raymond R Samaha, Mark Schena, Gary P Schroth, Svetlana Shchegrova, Dave D Smith, Frank Staedtler, Zhenqiang Su, Hongmei Sun, Zoltan Szallasi, Zivana Tezak, Danielle Thierry-Mieg, Karol L Thompson, Irina Tikhonova, Yaron Turpaz, Beena Vallanat, Christophe Van, Stephen J Walker, Sue Jane Wang, Yonghong Wang, Russ Wolfinger, Alex Wong, Jie Wu, Chunlin Xiao, Qian Xie, Jun Xu, Wen Yang, Liang Zhang, Sheng Zhong, Yaping Zong, and William Jr Slikker, *The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements.*, Nature Biotechnology **24** (2006), no. 9, 1151–61.

[8] V. G. Tusher, R. Tibshirani, and G. Chu, *Significance analysis of microarrays applied to the ionizing radiation response.*, Proc Natl Acad Sci U S A **98** (2001), no. 9, 5116–5121.

[9] P.H. Westfall and S. Young, *Resampling-based multiple testing*, Wiley, New York, 1993.

[10] Yuanhui Xiao, Alexander Gordon, and Andrei Yakovlev, *A c++ program for the cramér-von mises two-sample test*, Journal of Statistical Software **17** (2007), no. 8.