Gene Set Enrichment Analysis

Chao-Jen Wong

Fred Hutchinson Cancer Research Center

January 28, 2010

1. **Hypergeometric Testing**

2. **Simple GSEA using Z-score and Permutation**

3. **GSEA using Linear Models**

Gene set enrichment analysis

- Unlike per-gene analysis ...
- Search for categories where the constituent genes show changes in expression level over the experimental conditions.
- Use predefined gene set such as KEGG pathways, GO classifications, chromosome bands, and protein complexes.
- No need to make a cutoff between genes that are differentially expressed and those that are not.
- Provided in the *GESABase*, *Category*, *GOstats* and *topGO*.

Outline

1. Hypergeometric Testing
2. Simple GSEA using Z-score and Permutation
3. GSEA using Linear Models

Hypergeometric testing

- Basic concept: Suppose there are N balls in an urn, n are white and m are black. Drawing k balls out of the urn without replacement, how many black balls do we expect to get? What is the probability of getting x black balls?
- Hypergeometric testing for under- and over-representation of GO terms.
- Inputs
  1. Gene universe, N.
  2. GO categories (categorize genes by GO terms).
  3. A list of interesting genes, I, (differentially expressed genes identified by limma or just simply t-test by rowttests).

Hypergeometric testing

	Interesting (Black)	Not (White)	
In GO term	n_{11}	n_{12}	K
Not in GO term	n_{21}	n_{22}	$N-K$
	I	$N-I$	N

Suppose there are j interesting genes in the GO term ($n_{11} = j$), compute

1. Probability of seeing j or more black balls in K draws.
2. Expected number of black balls seeing in K draws.

Data preparation

- Define gene universe (a vector of Entrez Gene IDs).
- Select a list of interesting genes (a vector of Entrez Gene ID).

Data preparation

- Define gene universe (a vector of Entrez Gene IDs).
- Select a list of interesting genes (a vector of Entrez Gene ID).

Code: gene selection via *t*-test

```
> library(genefilter)
> library(day2)
> library(hgu95av2.db)
> data(ALLfilt_bcrneg)
> ttests <- rowttests(ALLfilt_bcrneg, "mol.biol")
> ## select interesting genes
> smPV <- ttests[ttests$p.value < 0.005, ]
> selectedEntrezIds <- unlist(mget(rownames(smPV),
+                                   hgu95av2ENTREZID))
> entrezUniverse=unlist(mget(featureNames(ALLfilt_bcrneg),
+   hgu95av2ENTREZID))
```

Hypergeometric testing

- Create a GOHyperGParams object.

Code: GOHyperGParams

```
> library(GOstats)
> hgCutoff <- 0.001
> GOparams <- new("GOHyperGParams",
+                  geneIds=selectedEntrezIds,
+                  universeGeneIds=entrezUniverse,
+                  annotation="hgu95av2.db",
+                  ontology="BP",
+                  pvalueCutoff=0.001,
+                  conditional=TRUE,
+                  testDirection="over")
```

- Outputs and summary.

Code: hyperGTest

```
> hgOver <- hyperGTest(GOparams)
> class(hgOver)
> summary(hgOver)
```

- Outputs and summary.

Code: hyperGTest

```
> hgOver <- hyperGTest(GOparams)
> class(hgOver)
> summary(hgOver)
```

- Exercise: generate report using htmlReort.

```
> showMethods("htmlReport")
> htmlReport(hgOver, file="hgResult.html")
> browseURL("hgResult.hrml")
```

Lab activity

1. Chapter 14: read and do the exercises in Section 14.3 and 14.4.
2. Use the `topGenes` dataset (load the data using data(topGenes)) and find a subset of genes whose adj.P.Val are less than 0.01.
3. Repeat the conditional Hypergeometric testing to find under- and over-represented biological processes.
4. Generate html reports.

Outline

1. Hypergeometric Testing

2. **Simple GSEA using Z-score and Permutation**

3. GSEA using Linear Models

Simple GSEA

Consider two group comparison

- Start with data quality assessment.
- Compute per-gene t-statistics: t_k for each gene k.
- Null hypothesis: no difference in mean expression

$$H_o : Z_K = 0$$

$$Z_K = \frac{1}{\sqrt{|K|}} \sum_{k \in K} t_k \sim \mathcal{N}(0,1),$$

where K denotes the gene sets, and $|K|$ the number of genes in the gene set.

- Alternative approach: use permutation test to assess which gene sets have an unusually large absolute value of z_K.

Data preparation

ALLfill_bcrneg

```
> library(ALL)
> library(hgu95av2.db)
> data(ALL)
> bcell <- grep("^B", as.character(ALL$BT))
> types <- c("NEG", "BCR/ABL")
> moltyp <- which(as.character(ALL$mol.biol) %in% types)
> # subsetting
> ALL_bcrneg <- ALL[, intersect(bcell, moltyp)]
> ALL_bcrneg$BT <- factor(ALL_bcrneg$BT)
> ALL_bcrneg$mol.biol <- factor(ALL_bcrneg$mol.biol)
> # nonspecific filter: remove genes that does not
> ## show much variation across samples
> library(genefilter)
> filt_bcrneg <- nsFilter(ALL_bcrneg,
+                         var.cutoff=0.5)
> ALLfilt_bcrneg <- filt_bcrneg$eset
```

Using KEGG

- Data representation: create an incidence matrix `Am` where $a_{ij} = 1$ if gene j is in gene set i and $a_{ij} = 0$ otherwise.

```
> library(KEGG.db)
> library(GSEABase)
> gsc <- GeneSetCollection(ALLfilt_bcrneg,
+                          setType=KEGGCollection())
> Am <- incidence(gsc)
```

- `ExpressionSet` object retains only those features that are in the incidence matrix `Am`.

```
> nsF <- ALLfilt_bcrneg[colnames(Am), ]
```

Using KEGG

Exercise

1. How many gene sets and how many genes are represented by the incidence matrix `Am`?
2. How many gene sets have fewer than ten genes in them?
3. What is the largest number of gene sets in which a gene can be found?
4. What is the name of this gene set? (use KEGGPATHID2NAME)

Using KEGG

Exercise

1. How many gene sets and how many genes are represented by the incidence matrix `Am`?
2. How many gene sets have fewer than ten genes in them?
3. What is the largest number of gene sets in which a gene can be found?
4. What is the name of this gene set? (use KEGGPATHID2NAME)

Code

```
> dim(nsF)
> dim(Am)
> nGene <- rowSums(Am)
> rownames(Am)[nGene < 10]
> sort(nGene, decreasing=TRUE)[1]
> KEGGPATHID2NAME[["05200"]]
```

Using KEGG

- Compute the per-gene test statistics using the rowttests function.

```
> rtt <- rowttests(nsF, "mol.biol")
> names(rtt)
```

```
[1] "statistic" "dm"        "p.value"
```

```
> rttStats <- rtt$statistic
```

Using KEGG

- Compute the per-gene test statistics using the rowttests function.

```
> rtt <- rowttests(nsF, "mol.biol")
> names(rtt)
```

```
[1] "statistic" "dm"        "p.value"
```

```
> rttStats <- rtt$statistic
```

- Reduce the incidence matrix by removing all gene sets that have fewer than ten genes in them.

```
> selectedRows <- (rowSums(Am) > 10)
> Am2 <- Am[selectedRows, ]
```

Using KEGG

- Compute the per-gene test statistics using the rowttests function.

```
> rtt <- rowttests(nsF, "mol.biol")
> names(rtt)
```

```
[1] "statistic" "dm"         "p.value"
```

```
> rttStats <- rtt$statistic
```

- Reduce the incidence matrix by removing all gene sets that have fewer than ten genes in them.

```
> selectedRows <- (rowSums(Am) > 10)
> Am2 <- Am[selectedRows, ]
```

- Compute z_k for each pathway: $z_K = \frac{1}{\sqrt{|K|}} \sum_{k \in K} t_k$.

```
> tA <- as.vector(Am2 %*% rttStats)
> tAadj <- tA /sqrt(rowSums(Am2))
> names(tAadj) <- rownames(Am2)
```

Using KEGG

Exercise

1. Which pathways have remarkably low (< 5) and high aggregate statistics (> 5)?
2. What is the name the pathway that has the lowest z_k score?
3. Use KEGG2heatmap to plot a heatmap for the genes in this pathway.

Using KEGG

Exercise

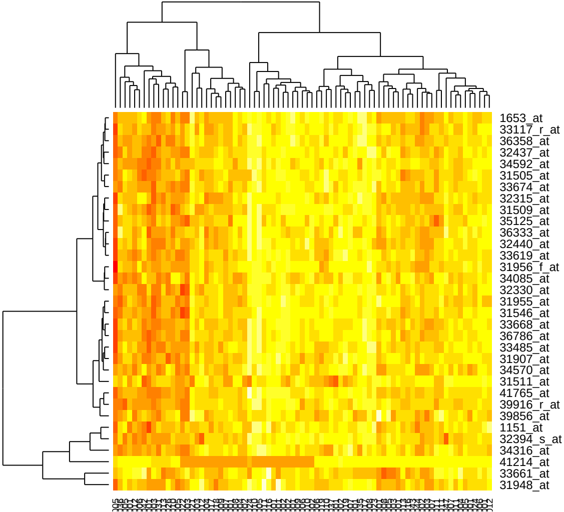
1. Which pathways have remarkably low (< 5) and high aggregate statistics (> 5)?
2. What is the name the pathway that has the lowest z_k score?
3. Use KEGG2heatmap to plot a heatmap for the genes in this pathway.

Code

```
> smPW <- tAadj[tAadj < -5]
> mget(names(smPW),KEGGPATHID2NAME)
> lgPW <- tAadj[tAadj > 5]
> mget(names(lgPW), KEGGPATHID2NAME)
```

KEGG2heatmap

```
> KEGG2heatmap("03010", nsF, "hgu95av2")
```



Permutation testing

- Assess the significant gene sets with respect to a reference distribution build by a number of permutations.
- gseattperm: permute the sample labels.
- Return *p*-value w.r.t. to a reference distribution:
  - `Lower`: proportion of permutation *t*-statistics that were smaller than the observed *t*-statistics
  - `Upper`: proportion of permutation *t*-statistics that were larger than the observed *t*-statistics

Code: using gseattperm

```
> library(Category)
> set.seed(123)
> pvals <- gseattperm(nsF, nsF$mol.biol, Am2, 1000)
> pvalCut <- 0.05
> lowC <- rownames(pvals)[pvals[, 1] <= pvalCut]
> unlist(getPathNames(lowC), use.names=FALSE)

[1] "Glycerophospholipid metabolism"
[2] "Ribosome"
```

Outline

1. Hypergeometric Testing
2. Simple GSEA using Z-score and Permutation
3. GSEA using Linear Models

Chromosome bands

- Use the mapping of genes to chromosome bands.
- To answer whether there are anomalies in the pattern of gene expression that related to chromosome bands.
- Use GSEA linear models.

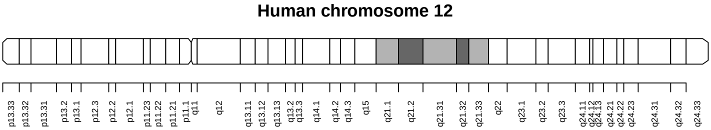


Figure: Ideogram for human chromosome 12. The shaded bands together represent 12q21. Notice that the chromosome bands are hierarchically nested, and they almost form a partition. (D. Sarker et. al. 2007)

Reference

"Using Categories defined by Chromosome Bands" by D. Sarker et. al.

Data preparation

- Consider the comparison of BCR/ABL and NEG groups.
- Use `ALL_bcrneg` object.
- Use nsFilter to remove probes with no Entrez Gene ID and no mapping to a chromosome band. Ensure that each Entrez Gene ID maps to exactly one probeset which has the highest IQR. Also remove probes with lack of variation ($\text{var} < 0.5$).

Data preparation

- Consider the comparison of BCR/ABL and NEG groups.
- Use `ALL_bcrneg` object.
- Use nsFilter to remove probes with no Entrez Gene ID and no mapping to a chromosome band. Ensure that each Entrez Gene ID maps to exactly one probeset which has the highest IQR. Also remove probes with lack of variation (var < 0.5).

Code: nonspecific filtering

```
> ALLfilt <- nsFilter(ALL_bcrneg, require.entez=TRUE,
+                             remove.dupEntrez=TRUE,
+                             require.CytoBand=TRUE,
+                             var.func=IQR,
+                             var.cutoff=0.5)$eset
```

Data preparation

- Compute per-gene *t*-statistics using limma.

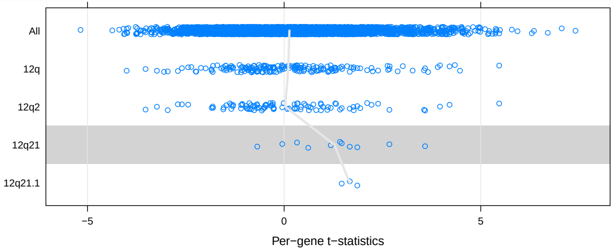
Data preparation

- Compute per-gene *t*-statistics using limma.

Code: moderate *t*-statistics

```
> library(limma)
> design <- model.matrix(~0 + ALLfilt$mol.biol)
> colnames(design) <- c("BCR/ABL", "NEG")
> contr <- c(1, -1)
> fit1 <- lmFit(ALLfilt, design)
> fit2 <- contrasts.fit(fit1, contr)
> fit3 <- eBayes(fit2)
> tlimma <- topTable(fit3, number=nrow(fit3),
+                    adjust.method="none")
> ## annotation
> entrezUniverse <- unlist(mget(tlimma$ID,
+                               hgu95av2ENTREZID))
> tstats <- tlimma$t
> names(tstats) <- entrezUniverse
```

Linear models



- Fitting linear model with per-gene *t*-statistics: for each category *j*,

$$y_i = \beta_0 + \beta_1 a_{ij} + \varepsilon_i,$$

where $a_{ij} = 1$ if gene i is associated with category j, and 0 otherwise. The index i may range over from universal genes to a subset of genes.

- $\beta_1 \sim \mathcal{N}(0,1)$

Linear models

- Create a ChrMapLinearMParams object.

Code: instance of class ChrMapLinearMParams

```
> library(Category)
> params <- new("ChrMapLinearMParams",
+               conditional=FALSE,
+               testDirection="up",
+               universeGeneIds=entrezUniverse,
+               geneStats=tstats,
+               annotation="hgu95av2",
+               pvalueCutoff=0.01,
+               minSize=4L)
```

Calling the `linearMTest` function

- linearMTest: compute the *p*-values for detecting up- or down-regulation of predefined gene sets.

Code: linearMTest

```
> lman <- linearMTest(params)
> lman
> summary(lman)
```

Exercise

1. Get familiar with the structure of `ChrMapLinearMParams` class? ChrMapLinearMParams or help(“ChrMapLinearMParams-class”)
2. Perform conditional GSEA linear models to find interesting chromosome bands that are up-regulated.
3. Summarize the result of the conditional test using summary.

Exercise

1. Get familiar with the structure of `ChrMapLinearMParams` class? ChrMapLinearMParams or help(“ChrMapLinearMParams-class”)
2. Perform conditional GSEA linear models to find interesting chromosome bands that are up-regulated.
3. Summarize the result of the conditional test using summary.

Code: conditional test

```
> slotNames(params)
> paramsCond <- params
> paramsCond@conditional <- TRUE
> lmanCond <- linearMTest(paramsCond)
> summary(lmanCond)
```

Summary

1. Basic idea behind GSEA.
2. Simple GSEA: *t*-tests and permutation.
3. Using KEGG categories.
4. Linear models and chromosome band categories.
5. Hypergeometric testings on GO BP terms.

Reference

- Assaf P. Oron et. al., Gene set enrichment analysis using linear models and diagnostics, *Bioinformatics*, vol. 24 no. 22, pp. 2566-2591, 2008.
- Florian Hahne et. al., *Bioconductor Case Studies*, chapter 13-14, Springer, 2008.
- Deepayan Sarker et. al., *Using Categories defined chromosome bands*, Bioconductor Category package vignette.
- D. Sarker et.al., Modeling gene expression data via chromosome bands, *Bioinformatics*, 2007.