Statistics for Bioinformatics
Practicals - Student test

Introduction

Golub et al. (1999) measured the level of expression of human genes in 38 patients suffering from two cancer types: acute lymphoblastic leukemia (ALL, 27 patients) and acute myeloid leukemia (AML, 11 patients).

The aim of such experiments is to detect a subset of genes which can be used for diagnostic tests, in order to asses whether a new patient suffers from either of these cancer types.

For this practical, we will use a pre-normalized and pre-filtered subset of 3051 genes, and select those which show a significant difference in expression between AML and ALL patients.

We will thus apply a Student test to each gene *g* independendly, to test the null hypothesis.

H_0: $m_{g,ALL} = m_{g,AML}$

where

- $m_{g,ALL}$ is the mean expression for gene *g* in ALL patients,
- $m_{g,AML}$ is the mean expression for gene *g* in AML patients,

Theory

This tutorial is an application of concepts seen in the following chapters of the course:

1. Hypothesis testing
2. Conformity tests
3. Multitesting

Prerequisite

This tutorial assumes you already executed the script `config.R` as described in the configuration page.

Tutorial

Loading the data

The pre-normalized data can be loaded from the web site of the course.

```
dir.golub.data <- file.path(dir.data, 'gene_expression','golub_1999')
file.golub <- file.path(dir.golub.data, 'Golub_1999_train_Z.tab')
golub.expr <- read.table(file.golub,sep='\t', header=T)
print(dim(golub.expr))
```

The variable `golub.expr` contains the normalized and standardized microarray data, with 38 chips (one per column) and 3051 genes. Each chip has been centered (on the median) and scaled (the standard deviation was estimated on the basis of the inter-quartile range).

We also need to define the cancer type for each chip.

```
n.ALL <- 27
n.AML <- 11
cancer.type <- c(rep("ALL", n.ALL), rep("AML", n.AML))

## Add the cancer type to the column name, for the display
names(golub.expr) <- paste(names(golub.expr), cancer.type,sep="_")
```

Applying a single Student test

R contains a specific package for hypothesis testing, which of course includes the classical Student test. We will start by arbitrary selecting one gene, and testing whether this particular gene is differentially expressed between ALL and AML patients.

Exercise

Select the gene on the 347th row of the golub dataset, and test the following null hypothesis.

H_0: $m_{347,ALL}=m_{347,AML}$

where

- $m_{347,ALL}$ is the mean of expression for the gene number 347 in ALL patients,
- $m_{347,AML}$ is the mean of expression for the gene number 347 in AML patients,

1. Calculate the sample means, standard deviation, *t.obs*, P.value, and E.value with standard **R** functions.
2. How do you interpret the results of the Student test ? Is the gene *g* differentially expressed between AML and ALL patients ? Give an interpretation for the P-value and E-value.
3. Apply the Student test with the the t.test() function implemented in **R**.
4. Apply the Welch test with the the t.test() function implemented in **R**.
5. Discuss about the meaning of the different attributes, between students and with the teacher.
6. Compare the results obtained with the Student-Fischer and the Welch test, respectively.

Solution

We will first select a gene, and get the expression values for patients of the type ALL (sample 1) and AML (sample 2), respectively.

```
## ################################################################
## t.test with a single gene

g <- 347

## Alternatively, you can select a gene randomly
## g <- sample(1:nrow(golub.expr),1)
g.profile <- as.vector(as.matrix(golub.expr[g,]))

## Draw a barpplot with color-coded cancer type
plot.col <- c('ALL'='#4444BB', 'AML'='#FFFF88')
x11(width=12,height=4)
barplot(g.profile,main=paste("Golub (1999), gene", g), col=plot.col[cancer.type])
legend('topright', c("ALL","AML"),col=plot.col[c("ALL","AML")],pch=15,bty="o",bg='white')
legend('topright', c("ALL","AML"),col='black',pch=22,bty="n")

## separate data in two vectors
sample.ALL <- g.profile[cancer.type=="ALL"]
sample.AML <- g.profile[cancer.type=="AML"]
```

In order to familiarize ourselves with the parameters of a Student test, we will first perform manually all the steps to compute the Student statistics (t_{obs}).

```
## Estimate the population means
mean.est.ALL <- mean(sample.ALL)
mean.est.AML <- mean(sample.AML)

## Compute the sample sd
## Don't forget that the sd() function automatically computes the
## estimate corrected with sqrt(n-1)
sample.sd.ALL <- sd(sample.ALL) * sqrt((n.ALL-1)/n.ALL)
sample.sd.AML <- sd(sample.AML) * sqrt((n.AML-1)/n.AML)

## Estimate the population standard deviations
## Don't forget that the sd() function automatically computes the
## estimate corrected with sqrt(n-1)
sd.est.ALL <- sd(sample.ALL)
sd.est.AML <- sd(sample.AML)
```

```
## Estimate the standard errors on the means
sd.err.est.ALL <- sd(sample.ALL) / sqrt(n.ALL)
sd.err.est.AML <- sd(sample.AML) / sqrt(n.AML)

## Estimate the standard deviation of the difference between two means,
## according to Student's formula
diff.sd.est <- sqrt((n.ALL*sample.sd.ALL^2 + n.AML*sample.sd.AML^2) * (1/n.ALL + 1/n.AML) /(n.ALL+n.AML-2))

## Compute t.obs
d <- abs(mean.est.ALL - mean.est.AML)
t.obs.Student <- d/diff.sd.est

## Compute the P-value.
## Since we perform the two-tail test, the single-tail probability has
## to be multiplied by 2 in order to obtain the alpha risk.
P.val.Student <- 2*pt(q=t.obs.Student, df=n.ALL+n.AML-2,lower.tail=F)
```

Actually, there is no need to perform all those steps manually, since **R** contains a predefined function `t.test()` which computes the Student as well as the Welch test.

```
## Apply the Student-Fischer t-test
## (this assumes that the two populations have an equal variance)
t.student <- t.test(sample.ALL,sample.AML, var.equal=TRUE)
print(t.student)

## Apply the Welch t-test (
## (this does not assumes that the two populations
## have an equal variance)
t.welch <- t.test(sample.ALL,sample.AML, var.equal=FALSE)
print(t.welch)
```

Interpretation

TO BE WRITTEN

Multiple testing

One simple way to select differentially expressed genes would be to apply the previous test to each gene successively. This would however be very inefficient in terms of calculation time. You can easily check this by inserting the previous code in a loop. If it takes too much time, press the *Esc* key to interrupt the calculation.

```
t.statistics <- vector()
P.values <- vector()
for (g in 1:nrow(golub.expr)) {
#  print(paste("Testing gene", g))
  g.profile <- as.vector(golub.expr[g,])
  sample.ALL <- g.profile[cancer.type=="ALL"]
  sample.AML <- g.profile[cancer.type=="AML"]
  t <- t.test(sample.ALL,sample.AML)
  t.statistics <- append(t.statistics, t$statistic)
  P.values <- append(P.values, t$p.value)
}
print(P.values)
```

Multi-testing

Multiple testing with the function `t.test.multi()`

In general, loops are not very efficient in R, and should be relplaced, whenever possible, by directly applying computations to either columns or rows of a matrix, with the function `apply()`. If you want to know more about this function, call its help with `help(apply)`.

As a matter of illustration, we will test a **R** program that uses the function `apply()` in order to run Welch's t-test on all rows of a table. All calculations are performed in parallel, by using the **R** `apply` function. This script can be loaded from the R scripts directory, with the following command.

```
source(file.path(dir.util, "util_student_test_multi.R"))
```

For the sake of curiosity, you can also have a look at the R code.

We will now apply the t-test to each gene of the data set, and store the result in a variable called `golub.t.result`.

```
golub.t.result <- t.test.multi(golub.expr, cancer.type)
dim(golub.t.result)
names(golub.t.result)
```

The computation takes ~4 seconds for 3051 genes on my laptop. The result of `t.test.multi` is a data frame with 3051 rows (one per gene) and 9 columns (one per statistics).

As is always the case with multiple testing, nominal P-values have to be interpreted with caution: since we applied the same test to 3051 genes, a P-value of 0.01 is expected to occur 30 times by chance alone. One classical way to correct this effect is to calculate and E-value, i.e. the product of the P-value by the number of tests. This is done automatically by `t.test.multi`, and the result is stored in the column "E-value".

We can now select genes according to the result of the t-test. Let us chose a threshold of E-value ≤ 1.

```
## Count the number of genes with E-value <= 1
sum(golub.t.result$E.value <= 1)

## Select this subset
golub.expr.E.1 <- golub.expr[golub.t.result$E.value <= 1,]
```

Naming the samples

In pevious tutorials, the gene expression profiles were analyzed in a "blind" way, without explicitly taking into accout the nature of the samples (patient types, cell types, ...).

The selection of differentially expressed genes relies on the exitence of different subgroups of samples (e.g. cell types, subtypes).

The Golub data set comes along with an information table describing some features of each sample.

```
## Load the information of each sample (includes training + testing set)
file.golub.info <- file.path(dir.golub.data, 'sample_info.tab')
sample.info <- read.table(file.golub.info, sep='\t', as.is=T, quote=NULL, header=T)
print(sample.info[1:10,])

## Extract cancer type and cell type from the sample info table
train.type <- sample.info[sample.info$Set=="train", "ALL.AML"]
train.cell <- sample.info[sample.info$Set=="train", "T.B.cell"]
train.cell <- sub("-cell","",train.cell)
train.cell[is.na(train.cell)] <- "M"

print(train.type)
print(train.cell)
```

We will rename the samples in order to explicitly indicate their cell types:

- "M" for myeloid cells (corresponding to the AML samples)
- "T" for lymphoblastic T-cells (subset of the ALL samples)
- "B" for lymphoblastic B-cells (subset of the ALL samples)

```
## Previous names
print(names(golub.expr.E.1))

## New names
names(golub.expr.E.1) <- paste("train", sample.info$Sample[sample.info$Set=="train"], train.cell, sep="_")
print(names(golub.expr.E.1))

head(golub.expr.E.1)
```

Storing the result

We will store the selection of 243 differentially expressed genes in a text file. These selected genes will be used for the tutorial on clustering.

```
## Create a directory for exporting the data
dir.golub <- file.path(dir.results, "microarrays", "golub_1999")
dir.create(dir.golub,showWarning=F,recursive=T)

## Save the expression values in a text file
write.table(golub.expr.E.1,file=file.path(dir.golub,"golub_Evalue_1.tab"),quote=F,sep='\t',col.names=F)
```

```
list.files(dir.golub)
```

Open the exported file with a text editor to check its content. There shoul be 38 columns (one per sample), and 243 rows (one per selected gene).

Exercise

Use the `t.test` function to check the results obtained with `t.test.multi` for the genes number 1, 2, 317 and 3051.

T-test with robust estimators

As mentionned above, one drawback of the t-test is that the average and variance of the two samples can be drastically affected by outliers. The function `t.test.multi` includes an option `robust.est` to use robust estimators of central tendency (the median) and dispersion (inter-quartile-range). We will use this option and compare the results with the standard Welch test.

```
golub.t.result.robust <- t.test.multi(golub.expr, cancer.type, robust.est=TRUE)
```

Note that this calculation is slower than with the standard estimators. On my laptop it takes 11 seconds instead of 4. We will now compare the results.

```
## Count the number of genes with E-value <= 1, with robust estimators
sum(golub.t.result.robust$E.value <= 1)

## Count the number of genes with E-value <= 1, with standard estimators
sum(golub.t.result$E.value <= 1)

## Count the number of genes at the intersection
sum(golub.t.result$E.value <= 1 & golub.t.result.robust$E.value <= 1)


## Draw 4 plots per page (2 rows, 2 columns)
par(mfrow=c(2,2))

## Draw plots to compare the mean and median for each of the two groups.
plot(golub.t.result$mean.ALL,
     golub.t.result.robust$mean.ALL,
     xlab="classical mean estimate (sample mean)",
     ylab="Robust mean estimate (median)",
     main="Estimates of central tendency (Sample 1)")
grid(col="blue")
abline(a=0,b=1,col="blue",lwd=1)

plot(golub.t.result$mean.AML,
     golub.t.result.robust$mean.AML,
     xlab="classical mean estimate (sample mean)",
     ylab="Robust mean estimate (median)",
     main="Estimates of central tendency (Sample 2)")
grid(col="blue")
abline(a=0,b=1,col="blue",lwd=1)


## Draw a plot to compare the estimates of variance (first group only).
plot(golub.t.result$var.est.ALL,
     golub.t.result.robust$var.est.ALL,
     main="Estimates of dispersion (Sample 1)",
     xlab="Classical estimate (sd)",
     ylab="Robust estimate (IQR)")
grid(col="blue")
abline(a=0,b=1,col="blue",lwd=1)

## Draw a plot to compare the E.value, on a logarithmic scale.
plot(golub.t.result$E.value,
     golub.t.result.robust$E.value,
     main='E-values',
     xlab="E-value with classical estimates (log axis)",
     ylab="E-value with robust estimates (log axis)",
     log="xy")
grid(col="blue")
abline(a=0,b=1,col="blue",lwd=1)
abline(v=1,col="blue",lwd=1)
abline(h=1,col="blue",lwd=1)
```

```
## Restore the standard drawing parameter (one plot per page)
par(mfrow=c(1,1))

## Draw plots to compare the means of the two groups, and color significant genes
## Note that the selected genes are not always those with the larges difference
plot(golub.t.result$mean.ALL, golub.t.result$mean.AML,pch=20,col='#888888')
lines(golub.t.result[golub.t.result$E.value <= 1,"mean.ALL"],
      golub.t.result[golub.t.result$E.value <= 1,"mean.AML"],
      pch=20,col='#0000FF',type='p')

## Same plot with the robust estimators
x11()
plot(golub.t.result.robust$mean.ALL, golub.t.result.robust$mean.AML,pch=20,col='#888888')
lines(golub.t.result.robust[golub.t.result.robust$E.value <= 1,"mean.ALL"],
      golub.t.result.robust[golub.t.result.robust$E.value <= 1,"mean.AML"],
      pch=20,col='#0000FF',type='p')

################################################################
## Volcano plot for the test with classical estimators (mean and stdev)
x11()
plot(golub.t.result$mean.diff,golub.t.result$sig,
     pch=20,col='#888888',
     panel.first=grid(col='#000000'),
     main='volcano plot - standardization with mean and sd')

lines(golub.t.result[golub.t.result$E.value <= 1,"mean.diff"],
      golub.t.result[golub.t.result$E.value <= 1,"sig"],
      pch=20,col='#0000FF',type='p')
abline(h=0,col='#FF0088')
abline(v=0,col='#FF0088')

################################################################
## Volcano plot for the test with robust estimates
x11()
plot(golub.t.result.robust$mean.diff,
     golub.t.result.robust$sig,
     pch=20,col='#888888',
     panel.first=grid(col='#000000'),
     main='volcano plot - standardization with median and IQR')

lines(golub.t.result.robust[golub.t.result.robust$E.value <= 1,"mean.diff"],
      golub.t.result.robust[golub.t.result.robust$E.value <= 1,"sig"],
      pch=20,col='#0000FF',type='p')
abline(h=0,col='#FF0088')
abline(v=0,col='#FF0088')

## Draw the genes selected with mean and df o the robust plot,
## to highlight common genes and differences
lines(golub.t.result.robust[golub.t.result$E.value <= 1,"mean.diff"],
      golub.t.result.robust[golub.t.result$E.value <= 1,"sig"],
      pch=20,col='#FF0000',type='p')
lines(golub.t.result.robust[golub.t.result$E.value <= 1
      & golub.t.result.robust$E.value <= 1,"mean.diff"],
      golub.t.result.robust[golub.t.result$E.value <= 1
      & golub.t.result.robust$E.value <= 1,"sig"],
      pch=20,col='#008800',type='p')

################################################################
## Compare the significance between the two standardization methods
x11()
plot(golub.t.result$sig,
     golub.t.result.robust$sig,
     pch=20,
     col='#0088FF',
     panel.first=grid(col='#000000'),
     main='Significance comparison',
     xlab="standardized with mean and sd",
     ylab="standardized with median and IQR")
```

Heat map of the selected gene profiles

We can now draw a heat map to visualize the profiles of expression of the 243 genes selected as significant in our multiple Student test.

```
## Select the significant genes (as detected by the Student test)
x <- golub.expr[golub.t.result$E.value < 1,]
```

```
## A simple image of the profiles
image(t(as.matrix(x)))

## Same image, but with a clustering of the rows (genes)
heatmap(as.matrix(x), Colv=NA)

## Same image, but with a clustering of the rows (genes) and columns
## (patients)
heatmap(as.matrix(x))
```

Principal component analysis of the selected profiles

```
## PCA transformation
golub.pca <- prcomp(t(x))

## Display the percentage of variance associated to the first components
plot(golub.pca)

## Plot the patients in the plane of the 2 first compenents
biplot(golub.pca)
```

Discussion

Discuss the results, between students first, and then with the teacher.

- Why do we observe such differences between the t.test results depending on the choice of estimators ?
- How confident can one be in the results ?
- Which strategies could be envisaged ?

Additional material

- A more detailed analysis of Golub's data sets is described in the Bioconductor basics tutorial (Dudoit and Gentelman, 2002), which can be downloaded from the Bioconductor web site (www.bioconductor.org).

- Other tutorials on gene filtering are listed on the Bioconductor web site.

- The manual of the **multtest** package also contains examples of t-test filtering, based on a pre-processed subset of the Golub traning set.

- Sandrine Dudoit's slides on multiple testing.

- Anestis Antoniadis' slides on differential analysis.

References

1. DeRisi JL, Iyer VR, Brown PO (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. Science. Oct 24;278(5338):680-6. PMID: 9381177.
2. Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, Brown PO (2000). Genomic expression programs in the response of yeast cells to environmental changes. Mol Biol Cell. Dec;11(12):4241-57. PMID: 11102521.
3. Cui, X, and Churchill, G.A. (2003). Statistical tests for differential expression in cDNA microarray experiments. Genome Biology **4**: 210.1-210.10. PMID:12702200
4. Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. & Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439), 531-7. PMID:10521349

Jacques van Helden (jvhelden@ulb.ac.be)