BIO 754 - Lecture 10

29-04-2017

Contents


```
load("liver_transcriptome_v1.Rdata")
load("liver_transcriptome_v2.Rdata")
load("liver_transcriptome_v3.Rdata")
load("liver_transcriptome_v4.Rdata")
```

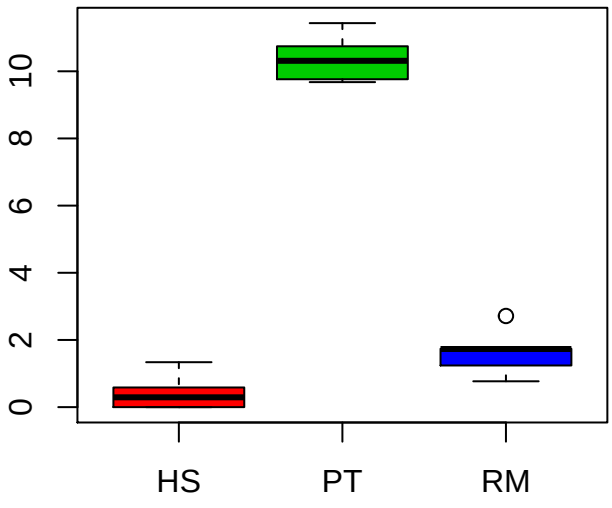
Permutation (randomization) tests

Is it possible to determine if any of the species differ from the others *without* using a theoretical distribution to compare with the calculated F-value? Actually, yes! We can run a simulation representing our H0 (i.e. the groups are coming from the same population, there is no difference between their means) and see how likely we have differences among groups.

Permutation test for mean difference of 2 groups

Calculate a permutation test p-value for difference between human and chimp expression for the gene with the most significant species effect:

```
y = nmat6[order( nmat6_aovp[ ,"species"] )[1], ]
boxplot(y ~ species2, col=2:4)
```



```r
# take the human and chimp subset
y2 = y[species2 %in% c("HS", "PT")]
# subset of the species vector
species2_2 = species2[species2 %in% c("HS", "PT")]
diff = mean(split( y2, species2_2 )$HS) - mean(split( y2, species2_2 )$PT)
# the mean difference
diff
```

```
## [1] -9.955744
```

```r
t.test( y2 ~ species2_2 )
```

```
##
##  Welch Two Sample t-test
##
## data:  y2 by species2_2
## t = -27.993, df = 9.4198, p-value = 2.177e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -10.754849  -9.156639
## sample estimates:
## mean in group HS mean in group PT
##        0.4179986       10.3737428
```

Now we can calculate a p-value by simulating no difference among groups, using the `sample` function:

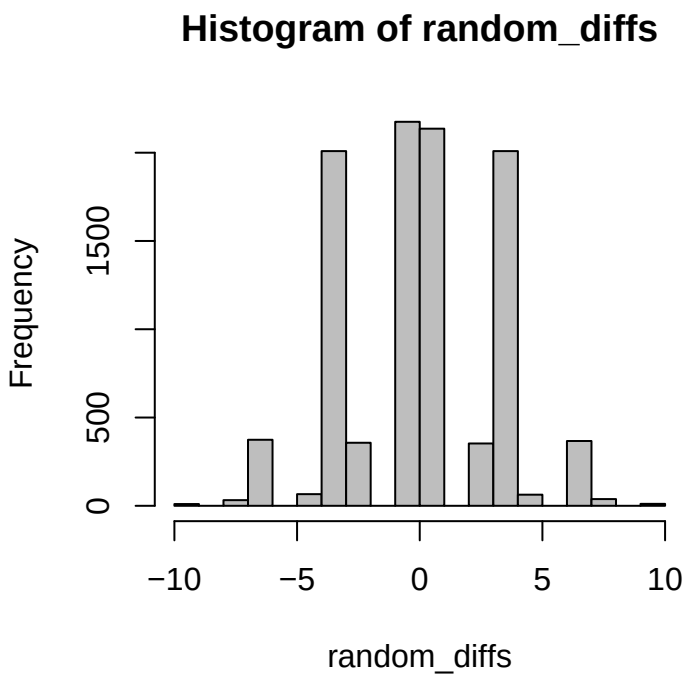
```r
s222 = sample(species2_2)  # a random grouping
random_diff = mean(split( y2, s222 )$HS) - mean(split( y2, s222 )$PT)
random_diff
```

```
## [1] -3.594158
```

```r
# try again
s222 = sample(species2_2)  # a random grouping
random_diff = mean(split( y2, s222 )$HS) - mean(split( y2, s222 )$PT)
random_diff
```

```
## [1] 3.267
```

```r
# now let's run this 10,000 times
random_diffs = sapply(1:10000, function(i) {
  s222 = sample(species2_2)
  random_diff = mean(split( y2, s222 )$HS) - mean(split( y2, s222 )$PT)
  return(random_diff)
})
hist( random_diffs, col=8 )
```




```
# the p-value
mean( abs(diff) <= abs(random_diffs) )
```

```
## [1] 0.0021
```

So the permutation test may be less sensitive, but has the advantage of not having any assumptions regarding the data distribution.

Permutation test for the species effect

As the null hypothesis in the ANOVA states that the 3 samples are from the same population, pooling our observed 3 groups together, forming 3 random groups from the pooled data, calculating residual SS values according to that grouping, and repeating the whole procedure many times will yield a null distribution.

For this, we can use the `sample` function in R, and create new random groups, the same size as the real species samples: 3 X 6. We can then run the test again, and store the p-values. We could also have used the F-statistic or the among-group SS in our simulations to compare with the observed data. They would be equivalent, as the higher the among-group SS, the higher will be the F-statistic, and the lower the p-value.

To test the H0 of **no species effect**, we will randomly reshuffle the `species2` vector.

```
# let's try on the first gene
y = nmat6[1,]
# the real result
anova( aov( y ~ species2 * sex2 ) )$Pr[1:3]
```

```
## [1] 0.04073004 0.30297619 0.57175398
```

```
# using sample, randomize the species ID vector
randomsp = sample( species2 )
# and run the test
anova( aov( y ~ randomsp * sex2 ) )$Pr[1:3]
```

```
## [1] 0.2492844 0.1043578 0.1074088
```

```r
# randomize and test, once again
randomsp = sample( species2 )
anova( aov( y ~ randomsp * sex2 ) )$Pr[1:3]
```

```
## [1] 0.8205669 0.3380961 0.4091145
```

```r
# randomize and test, once again
randomsp = sample( species2 )
anova( aov( y ~ randomsp * sex2 ) )$Pr[1:3]
```

```
## [1] 0.06258399 0.73538046 0.33973158
```

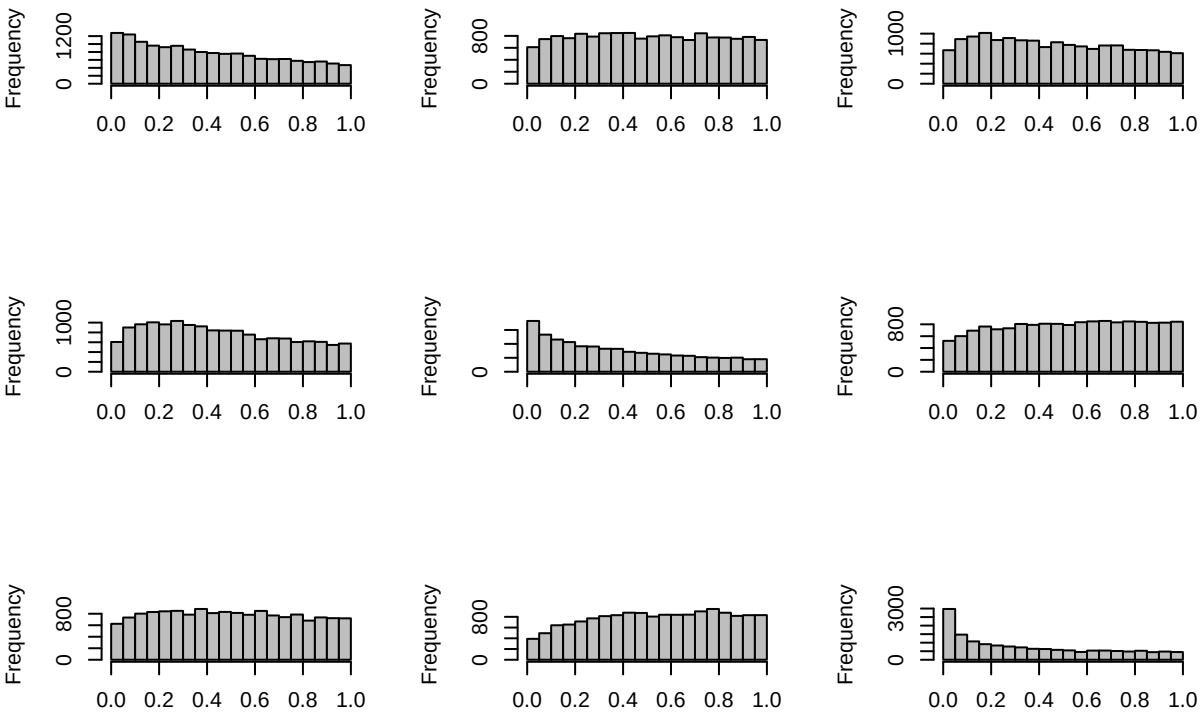
Now repeat this `300` times, and check how many times you find as many genes with `p<0.05`.

In this case each set of results is a matrix. So we can use `lapply` instead of `sapply`, which stores the results as a list (rather than coercing them into a matrix).

```r
nmat6_aovp_perm = lapply(1:300, function(i) {
  print(i)  # counter to follow the progress
  randomsp = sample(species2)
  nmat6_aovp_random = t(apply(nmat6, 1, function(y) {
    anova(aov(y ~ randomsp * sex2))$Pr[1:3]
  }))
  nmat6_aovp_random
})
length(nmat6_aovp_perm)
# this is a valuable object, so let's save
save(nmat6_aovp_perm, "liver_transcriptome_v5.Rdata")
```

Note that, if you had to run this e.g. 10,000 times, for real analysis, you might consider **parallelising** your jobs. E.g. running 100 packages of 100 simulations in parallel on a server. For this, you could use the `runif` function to assign unique names to your objects and your files (as you don't wish all your jobs to be written under the same name!)

```r
# check the first 9 permutation cases
# only the species effect p-values
par(mfrow=c(3,3))
for (i in 1:9) {
  hist( nmat6_aovp_perm[[i]][,1], col=8, main="", xlab="")
}
```



Global p-value

We can use the simulation results in different ways. First, we can use the simulations to calculate a global, or **transcriptome-wide p-value** for the species effect. This will be a significant species differences are across all genes, which will also be:

- Independent of ANOVA assumptions (because any violation of assumptions are also represented in the simulations),
- That takes into account dependence among genes (because the genes are also dependent in the simulations).

One possible approach is to use the observed proportion of genes with `p<0.05` in ANOVA as a statistic, and compare this with the simulations.

Check what proportion of genes had `p<0.05` in the real data and compare with the simulations:

```
# the observed proportion
obs = mean(nmat6_aovp[,"species"]<0.05)
obs
```

```
## [1] 0.5273485
```

```
# the expected proportions under H0
exp = sapply(nmat6_aovp_perm, function(x) mean( x[,1]<0.05 ) )
length(exp)
```

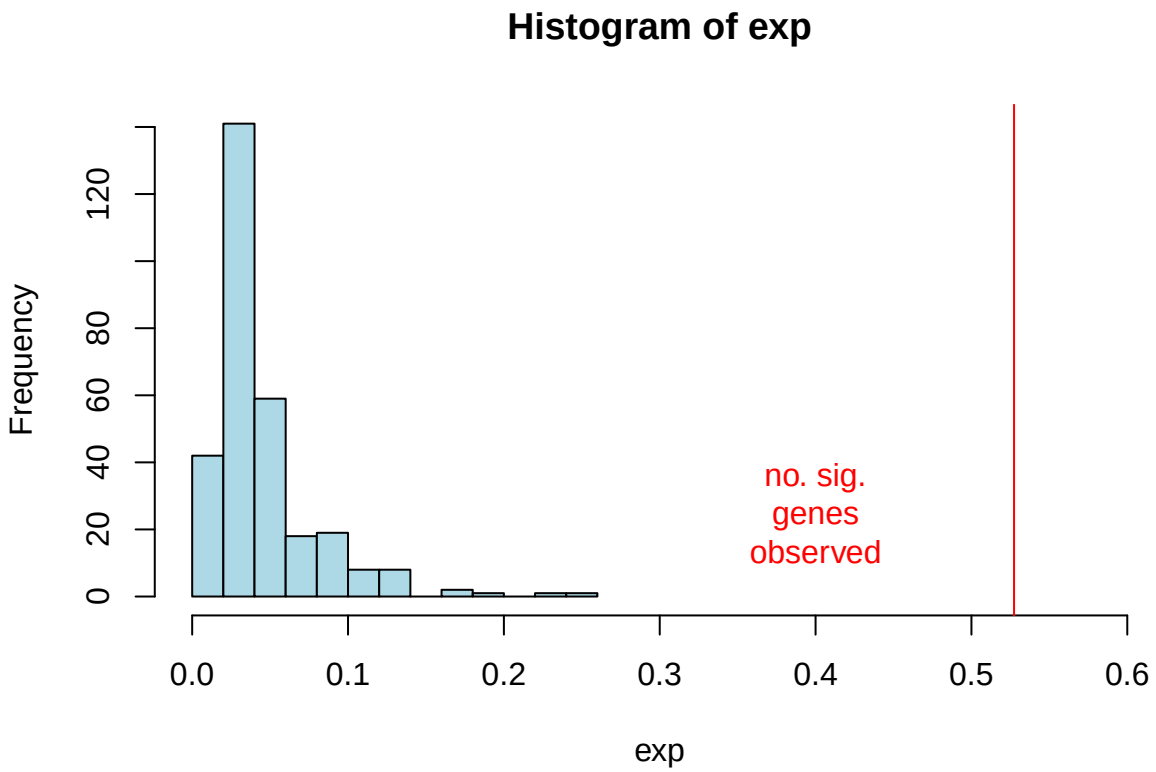
```
## [1] 300
```

```
summary(exp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01077 0.02435 0.03328 0.04506 0.05388 0.25620
```

We can also check the histogram, and add our observation on the plot:

```
hist(exp, col="light blue", xlim=c(0, 0.6))
abline(v=obs, col="red")
text(0.4, 25, "no. sig.\ngenes\nobserved", col="red")
```



The proportion of the area of bars on the left of observed value to the total area of the bars is the simulated p-value. It can be calculated by dividing number of cases which had smaller number of genes with `p<0.05` than observed.

```
global_perm_p = mean(exp >= obs)
global_perm_p
```

```
## [1] 0
```

False discovery rate

Thus we can report the global permutation test p-value as `p<0.01` (i.e. no case in 300 trials). We can also calculate the false discovery rate for the number of genes at `p<0.05` using the median of the simulated values:

```
median(exp)
```

```
## [1] 0.0332799
```

```
perm_fdr = median(exp)/obs
perm_fdr
```

```
## [1] 0.06310798
```

We therefore expect c. `0.07` of the genes with `p<0.05` to be false positives.

Functional analysis with Gene Ontology

We can focus on the species effect, and study the genes differentially expressed for species. We could use all 8000+ genes at ANOVA p-value <0.05, given that according to the permutation test, the FDR is $< 10\%$.

We could also choose genes at BY-corrected p<0.05, which chooses less genes (apparently in this case, is more stringent):

```
spDEGenes = rownames(nmat6)[nmat6_aovq[,1] < 0.05]
length(spDEGenes)
```

```
## [1] 3097
```

```
no_spDEGenes = rownames(nmat6)[nmat6_aovq[,1] >= 0.05]
length(no_spDEGenes)
```

```
## [1] 12498
```

```
length(intersect( spDEGenes, no_spDEGenes) )
```

```
## [1] 0
```

The biomaRt package

```
library(biomaRt)
listMarts()
```

```
##                biomart               version
## 1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 88
## 2   ENSEMBL_MART_MOUSE      Mouse strains 88
## 3     ENSEMBL_MART_SNP  Ensembl Variation 88
## 4 ENSEMBL_MART_FUNCGEN Ensembl Regulation 88
## 5    ENSEMBL_MART_VEGA                  Vega
```

```
# connect to the database
ens = useMart("ensembl", dataset = "hsapiens_gene_ensembl")
head(listAttributes(ens), 10)
```

```
##                    name              description         page
## 1        ensembl_gene_id           Gene stable ID feature_page
## 2  ensembl_transcript_id     Transcript stable ID feature_page
## 3     ensembl_peptide_id        Protein stable ID feature_page
## 4        ensembl_exon_id           Exon stable ID feature_page
## 5            description         Gene description feature_page
## 6        chromosome_name Chromosome/scaffold name feature_page
## 7         start_position          Gene Start (bp) feature_page
## 8           end_position            Gene End (bp) feature_page
## 9                 strand                  Strand feature_page
## 10                  band           Karyotype band feature_page
```

```
grep("GO", listAttributes(ens)[,2], val=T)
```

```
## [1] "GO term accession"       "GO term name"
## [3] "GO term definition"      "GO term evidence code"
## [5] "GO domain"               "GOSlim GOA Accession(s)"
## [7] "GOSlim GOA Description"
```

```
listAttributes(ens)[grep("GO", listAttributes(ens)[,2]),]
```

```
##                      name             description         page
## 38                  go_id       GO term accession feature_page
## 39              name_1006            GO term name feature_page
## 40        definition_1006      GO term definition feature_page
## 41        go_linkage_type   GO term evidence code feature_page
## 42         namespace_1003               GO domain feature_page
## 43   goslim_goa_accession GOSlim GOA Accession(s) feature_page
## 44 goslim_goa_description  GOSlim GOA Description feature_page
```

We can obtain GO information. There are thousands of GO categories defined, but here we'll use only the GO Slim categories, which are the most common ~150 biological processes:

```
ens_goslim = unique(getBM(attributes=c('ensembl_gene_id', 'goslim_goa_accession'), values=T, mart=ens))
# save the object, as it takes long time
save(ens_goslim, file="ens_goslim.RData")
```

```
load("ens_goslim.RData")
```

```
dim(ens_goslim)
```

```
## [1] 297775      2
```

```
head(ens_goslim)
```

```
##   ensembl_gene_id goslim_goa_accession
## 1 ENSG00000198888           GO:0003674
## 2 ENSG00000198888           GO:0016491
## 3 ENSG00000198888           GO:0005575
## 4 ENSG00000198888           GO:0005622
## 5 ENSG00000198888           GO:0005623
## 6 ENSG00000198888           GO:0005737
```

```
# many genes are repeated multiple times
length( unique( ens_goslim[,1] ))
```

```
## [1] 21255
```

```
# number of unique GO Slim categories
length( unique( ens_goslim[,2] ))
```

```
## [1] 143
```

Genes associated with catabolism

Now we can test the hypothesis that genes assigned to the catabolism function may be more differentially expressed among species, than non-DE genes. We obtain the GO ID for "catabolic process" at this website: http://www.ebi.ac.uk/QuickGO

Specifically: http://www.ebi.ac.uk/QuickGO/GTerm?id=GO:0009056

```
cat_genes = unique(ens_goslim[ens_goslim[,2] == 'GO:0009056', 1])
no_cat_genes = setdiff( unique(ens_goslim[,1]), cat_genes)
length(no_cat_genes)
```

```
## [1] 19133
```

```
length(no_cat_genes) + length(cat_genes)
```

```
## [1] 21255
```

```
length( intersect(no_cat_genes, cat_genes) )
```

```
## [1] 0
```

Contingency analysis using the Fisher's exact test

We first need to calculate the length of the intersection between two vectors x and y:

```
length( intersect( spDEGenes, cat_genes ))
```

```
## [1] 341
```

```
length( intersect( spDEGenes, no_cat_genes ))
```

```
## [1] 2412
```

```
length( intersect( no_spDEGenes, cat_genes ))
```

```
## [1] 1251
```

```
length( intersect( no_spDEGenes, no_cat_genes ))
```

```
## [1] 9576
```

```
x = c(length( intersect( spDEGenes, cat_genes )),
  length( intersect( spDEGenes, no_cat_genes )),
  length( intersect( no_spDEGenes, cat_genes )),
  length( intersect( no_spDEGenes, no_cat_genes )) )
x
```

```
## [1]  341 2412 1251 9576
```

```
matx = matrix(x, 2, 2)
fisher.test(matx, alternative = "greater")
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  matx
## p-value = 0.1197
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.9695234       Inf
## sample estimates:
## odds ratio
##   1.082194
```