BIO 754 - Lecture 09

27-04-2017

Contents


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

Testing differential expression using linear models

Once we have preprocessed our dataset (summarization, transformation, normalization, filtering), we will now be learning how to test differences among groups for each gene, also called **differential expression**. One common approach is to use linear models, like ANOVA.

Assumptions for linear models:

- data is normally distributed,
- data points are independent,
- variance among groups are equal (residuals are homogeneous),

We will start by building an ANOVA model of the species effect for the first gene's expression profile. We will be testing the null hypothesis that the means of the groups are equal (the samples derive from a population with the same mean). Let's first study the variation in the expression data.

```
# check the dataset
dim(nmat6)
```

```
## [1] 15595    18
```

```
str(nmat6)
```

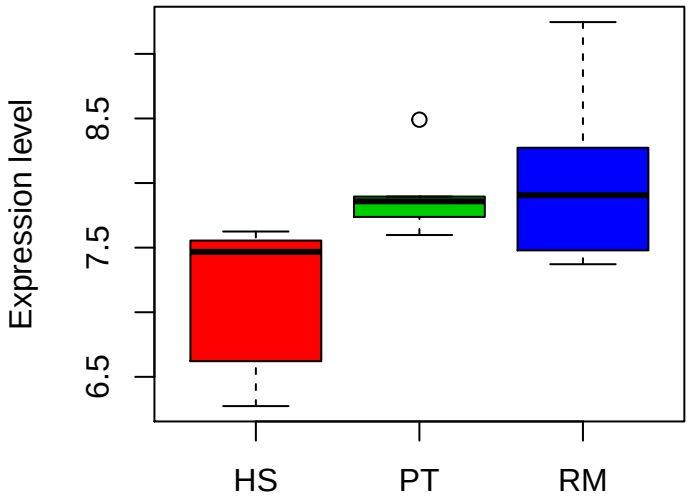
```
##  num [1:15595, 1:18] 6.27 0 4.65 6.19 3.34 ...
##  - attr(*, "dimnames")=List of 2
```

```
##   ..$ : chr [1:15595] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457" ...
##   ..$ : chr [1:18] "HSM1" "PTF1" "RMM1" "HSF1" ...
```

```
# choose the first gene
y = nmat6[1,]
y
```

```
##     HSM1     PTF1     RMM1     HSF1     PTM1     RMF1     RMF2     HSM2
## 6.273480 7.737963 7.478763 7.554903 7.865687 7.850765 7.961526 7.624893
##     PTF2     RMM2     HSF2     PTM2     RMM3     RMF3     HSM3     PTF3
## 7.895412 9.246079 7.456257 8.490531 8.273440 7.371782 7.478299 7.597779
##     PTM3     HSF3
## 7.851859 6.621084
```

```
boxplot( y ~ species2, col=2:4, ylab="Expression level")
```



split, tapply, and changing the order of factor levels

The boxplot function plots the quartiles. How about the means of the groups, which is what ANOVA is about. Let's calculate group means:

```
split( y, species2 )
```

```
## $HS
##     HSM1     HSF1     HSM2     HSF2     HSM3     HSF3
## 6.273480 7.554903 7.624893 7.456257 7.478299 6.621084
##
## $PT
##     PTF1     PTM1     PTF2     PTM2     PTF3     PTM3
## 7.737963 7.865687 7.895412 8.490531 7.597779 7.851859
##
## $RM
##     RMM1     RMF1     RMF2     RMM2     RMM3     RMF3
## 7.478763 7.850765 7.961526 9.246079 8.273440 7.371782
```

```
sapply( split( y, species2 ), mean )
```

```
##       HS       PT       RM
## 7.168153 7.906538 8.030392
```

We would also use a `tapply` loop, which does the same thing, separating data in its first argument by category levels in the second, and applying a function (the 3rd argument) to each set of values:

```
tapply( y, species2, mean)
```

```
##       HS       PT       RM
## 7.168153 7.906538 8.030392
```

Note that by default, boxplot and other functions sort the groups according to the order of the levels in the factor object (the categorical variable). We can change the order of a factor as follows:

```
# the original
species2
```

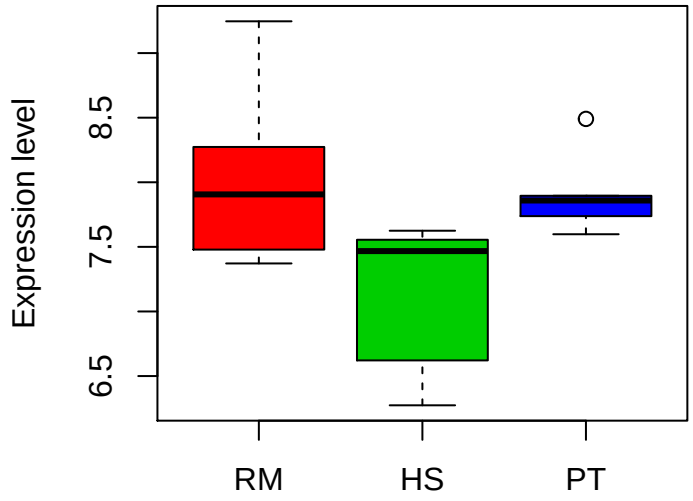
```
##  [1] HS PT RM HS PT RM RM HS PT RM HS PT RM RM HS PT PT HS
## Levels: HS PT RM
```

```
# redefine
species2 = factor(species2, levels = c("RM", "HS", "PT"))
species2
```

```
##  [1] HS PT RM HS PT RM RM HS PT RM HS PT RM RM HS PT PT HS
## Levels: RM HS PT
```

Now `RM` will be given priority in functions that use a list object:

```
boxplot(y ~ species2, col=2:4, ylab="Expression level")
```



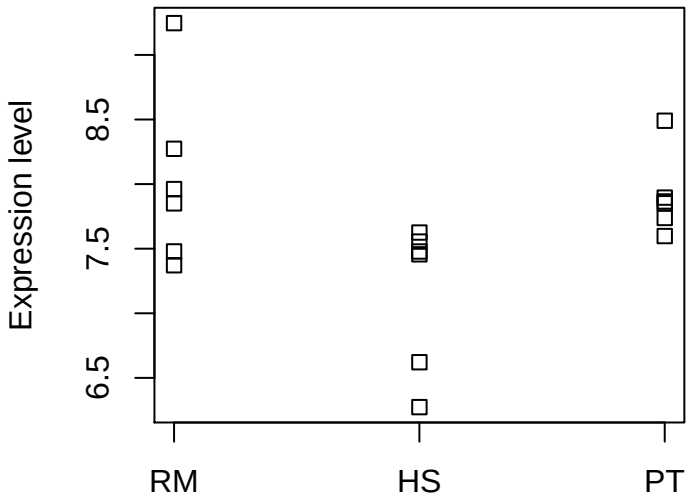

```
species_means = tapply(y, species2, mean)
species_means
```

```
##       RM       HS       PT
## 8.030392 7.168153 7.906538
```

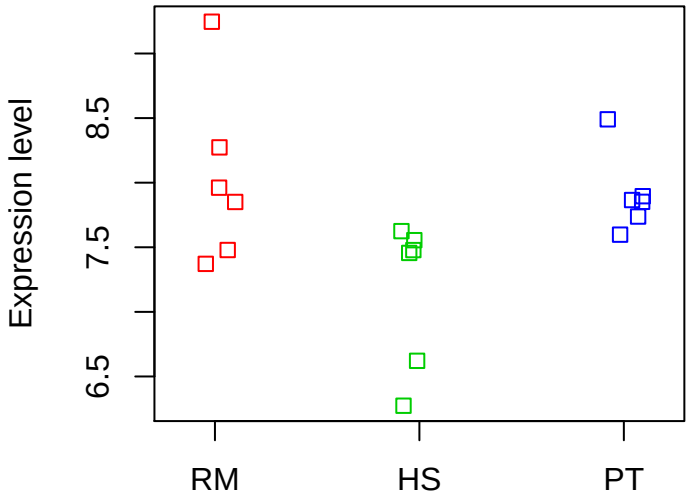
More on plotting: `stripchart` and `segments`, `arrows`,

One can also use `stripchart` to plot the data themselves. Using the "jitter" option is particularly useful when you have many data points that could overlap. It adds a random jitter of small magnitude to the points (which will be different each time you run the function).

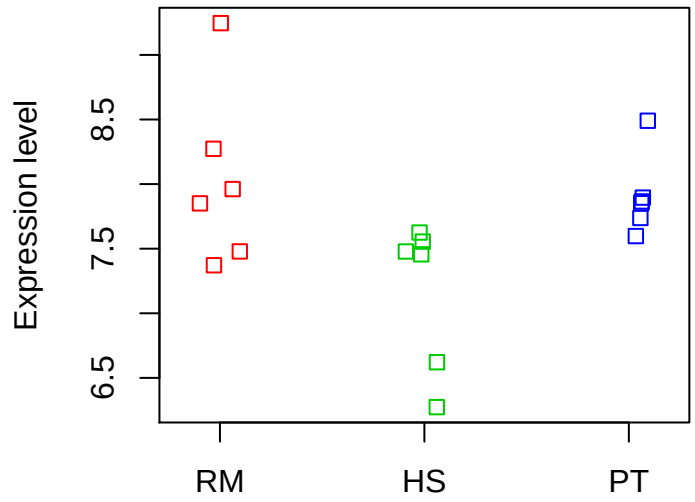
```
stripchart(y ~ species2, vertical = TRUE, ylab="Expression level")
```




```
stripchart(y ~ species2,
           vertical = TRUE, ylab="Expression level",
           method = "jitter", col=2:4)
```

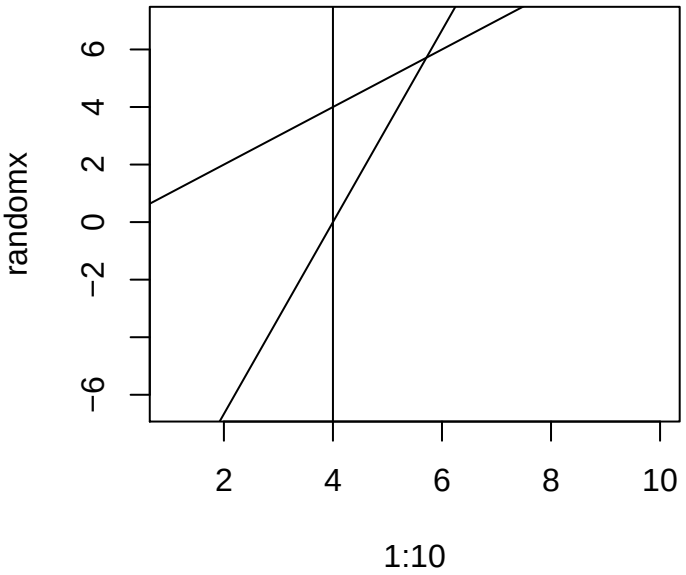



```
stripchart(y ~ species2,
           vertical = TRUE, ylab="Expression level",
           method = "jitter", col=2:4)
```



One can also add means of each group to the plot, using `segments`. A small example:

```
randomx = rnorm(10)*10
plot(1:10, randomx, type="n")
segments(x0 = 0, y0 = 0, x1 = 10, y1 = 10)
segments(x0 = 1, y0 = -10, x1 = 7, y1 = 10)
# you can also add arrows the same way
arrows(x0 = 4, y0 = -10, x1 = 4, y1 = 12, length = 0.2)
```

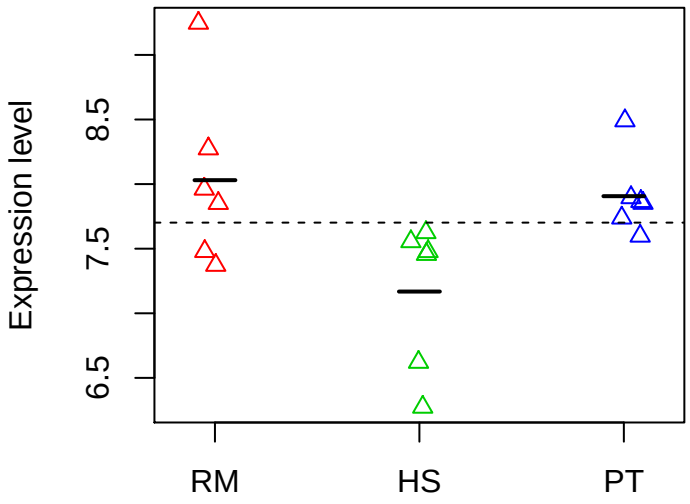


Adding the means:

```
stripchart(y ~ species2,
           vertical = TRUE, ylab="Expression level",
           method = "jitter", col=2:4, pch = 2)

segments(1:3-0.1, species_means, 1:3+0.1, species_means,
         lwd = 2)

abline(h = mean(y), lwd = 1, lty = 2)
```



Understanding the ANOVA table

How to calculate the ANOVA table, which appears as follows:

Source of Variation	d.f.	SS	MS	F_0
Factor A (between groups)	a-1	$SSA = \sum_{i=1}^{a} n_i \left(\bar{y}_{i.} - \bar{y}_{..}\right)^2$	$MSA = \frac{SSA}{(a-1)}$	$\frac{MSA}{MSE}$
Factor B (between groups)	b-1	$SSB = \sum_{j=1}^{b} n_j \left(\bar{y}_{.j} - \bar{y}_{..}\right)^2$	$MSB = \frac{SSB}{(b-1)}$	$\frac{MSB}{MSE}$
Error (within groups)	(a-1)(b-1)	$SSE = SST - SSA - SSB$	$MSE = \frac{SSE}{(a-1)(b-1)}$	
Total	N-1	$SST = \sum_{i=1}^{a} \sum_{j=1}^{n} \left(y_{ij} - \bar{y}_{..}\right)^2$		

?

We will now learn how the components here are calculated:

```
summary( aov(y ~ species2) )
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## species2     2  2.608  1.3040   4.427 0.0308 *
## Residuals   15  4.419  0.2946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What ANOVA does is to separate the total sum of squares (SS), which is the sum of squared differences between each point and the grand mean. It separates the total SS into SS among and within groups:

```
grand_mean = mean(y)
y - grand_mean
```

```
##        HSM1        PTF1        RMM1        HSF1        PTM1        RMF1
## -1.42821449  0.03626879 -0.22293197 -0.14679122  0.16399220  0.14906996
##        RMF2        HSM2        PTF2        RMM2        HSF2        PTM2
##  0.25983171 -0.07680128  0.19371697  1.54438481 -0.24543732  0.78883640
##        RMM3        RMF3        HSM3        PTF3        PTM3        HSF3
```

```
##  0.57174498 -0.32991249 -0.22339537 -0.10391512  0.15016418 -1.08061072
```

```
# total SS (sum of squares)
totalSS = sum( (y - grand_mean)^2 )
totalSS
```

```
## [1] 7.026745
```

```
# SS among groups, which is weighted by the group size
species_means = tapply(y, species2, mean)
species_means
```

```
##       RM       HS       PT
## 8.030392 7.168153 7.906538
```

```
diffamong = species_means - grand_mean
diffamong
```

```
##         RM         HS         PT
##  0.3286978 -0.5335417  0.2048439
```

```
diffamong^2
```

```
##         RM         HS         PT
## 0.10804227 0.28466678 0.04196102
```

```
6*(species_means - grand_mean)^2
```

```
##        RM        HS        PT
## 0.6482536 1.7080007 0.2517661
```

```
amongSS = sum( 6*(species_means - grand_mean)^2 )

# SS within groups (residual SS)
resi = tapply( y, species2, function(x) { x - mean(x)  } )
resi
```

```
## $RM
##        RMM1        RMF1        RMF2        RMM2        RMM3        RMF3
## -0.55162980 -0.17962788 -0.06886612  1.21568698  0.24304714 -0.65861032
##
## $HS
##       HSM1       HSF1       HSM2       HSF2       HSM3       HSF3
## -0.8946728  0.3867505  0.4567405  0.2881044  0.3101464 -0.5470690
##
## $PT
##        PTF1        PTM1        PTF2        PTM2        PTF3        PTM3
## -0.16857511 -0.04085171 -0.01112694  0.58399250 -0.30875903 -0.05467972
```

```
# it is a list, and we have to unlist
withinSS = sum( unlist( resi )^2 )
withinSS
```

```
## [1] 4.418724
```

This also does the same:

```
aov(y ~ species2)$residuals
```

```
##        HSM1        PTF1        RMM1        HSF1        PTM1        RMF1
## -0.89467276 -0.16857511 -0.55162980  0.38675051 -0.04085171 -0.17962788
```

```
##        RMF2        HSM2        PTF2        RMM2        HSF2        PTM2
## -0.06886612  0.45674046 -0.01112694  1.21568698  0.28810441  0.58399250
##        RMM3        RMF3        HSM3        PTF3        PTM3        HSF3
##  0.24304714 -0.65861032  0.31014636 -0.30875903 -0.05467972 -0.54706899
```

```
sum( aov(y ~ species2)$resi^2 )
```

```
## [1] 4.418724
```

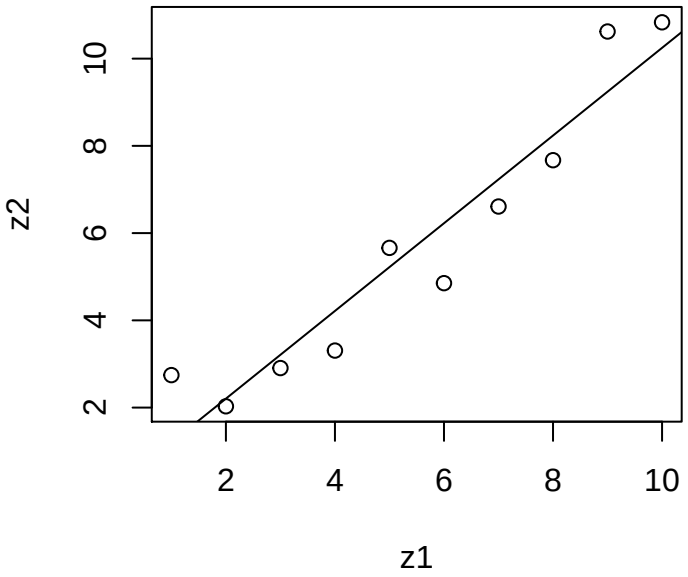
Residuals are the difference between each point and the expected (or fitted) values. The expected value is represented by the group mean here.

```
aov( y ~ species2 )$fitted.values  # the group mean for each individual
```

```
##     HSM1     PTF1     RMM1     HSF1     PTM1     RMF1     RMF2     HSM2
## 7.168153 7.906538 8.030392 7.168153 7.906538 8.030392 8.030392 7.168153
##     PTF2     RMM2     HSF2     PTM2     RMM3     RMF3     HSM3     PTF3
## 7.906538 8.030392 7.168153 7.906538 8.030392 8.030392 7.168153 7.906538
##     PTM3     HSF3
## 7.906538 7.168153
```

In regression, the fitted values would be represented by the regression line:

```
z1 = 1:10
z2 = rnorm(10) + z1
plot(z1, z2)
abline(lm(z2 ~ z1))  # fit a linear regression line
```




```
resid(lm(z2 ~ z1))
```

```
##          1          2          3          4          5          6
##  1.5442873 -0.1759248 -0.3042482 -0.9084539  0.4401437 -1.3736218
##          7          8          9         10
## -0.6219394 -0.5654537  1.3797676  0.5854431
```

Note that the `withinSS` and `amongSS` sum up to `totalSS`:

```
totalSS
```

```
## [1] 7.026745
```

```
amongSS + withinSS
```

```
## [1] 7.026745
```

The `withinSS` and `amongSS` are scaled using the corresponding degrees of freedoms, yielding the **mean SS** among and within.

```
summary( aov( y ~ species2 ) )
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## species2     2  2.608  1.3040   4.427 0.0308 *
## Residuals   15  4.419  0.2946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If the mean SS within is relatively large than mean SS among, there is little evidence for a difference among groups. But what is relatively large or small? How is the p-value calculated?

Hypothesis testing with ANOVA

The t-distribution, `pt` and `dt`

Let's first remind ourselves how the two-sample t-test works. It is used to test the null hypothesis (H0) that two samples are derived from populations with the same mean (i.e. the difference is `0`). In this case we calculate a difference between the means of the samples and scale it with an estimate of the population standard deviation, which is the t-statistic. We then compare the t-statistic with a t-distribution with **degrees of freedom** (df) `n-2`.

The t-distribution is similar to the normal: it is symmetric and bell-shaped. But it has fatter tails than the normal, which gets even more fat when the df is small. When the df is large, it converges to the standard normal.

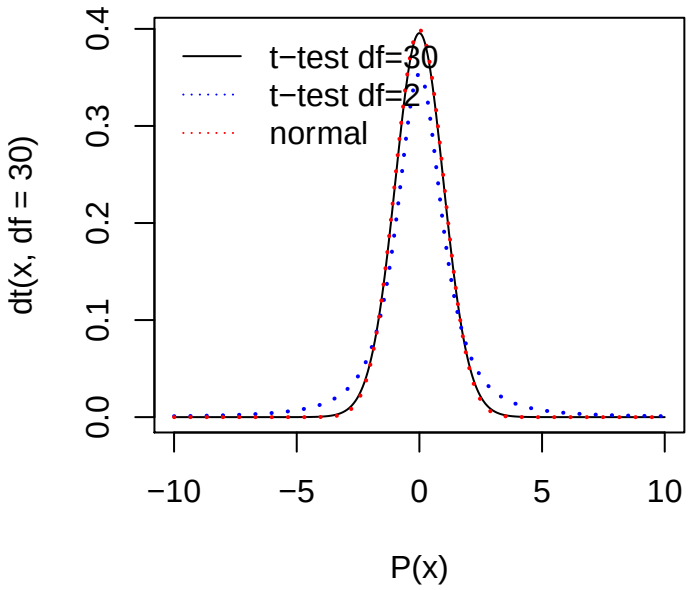
What is the advantage of the t-test compared to the Z-test, i.e. using the standard normal distribution? If our sample size is not large (`< 30`) and we do *not* know the population variance, using the normal distribution can lead to inflated Type I error rates. The fat tails of the t-distribution with small df corrects for this effect. When the sample size is large, the t-test approximates the Z-test.

We can study the shape of the t-distribution using the probability density function `dt`.

```
dt(0, df = 30)
```

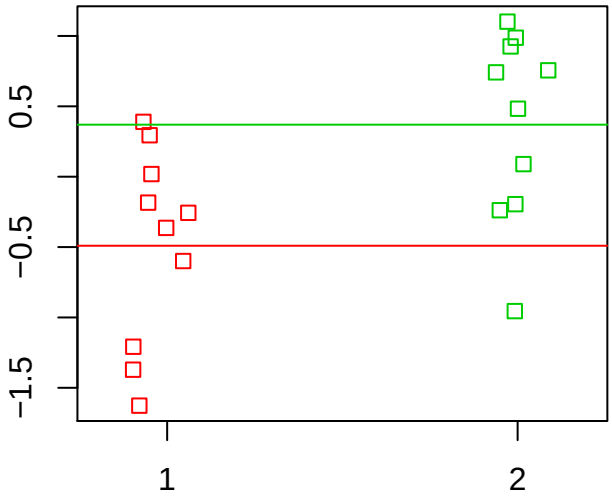
```
## [1] 0.3956322
```

```
x = seq(-10, 10, by=0.01)
plot(x, dt(x, df = 30), type = "l", col=1, lwd=1, lty=1, xlab = "P(x)")
lines(x, dt(x, df = 2), col=4, lwd=2, lty=3)
lines(x, dnorm(x), col=2, lwd=2, lty=3)
legend( "topleft", c("t-test df=30", "t-test df=2", "normal"), bty = "n", col = c(1,4,2), lty = c(1,3,3
```



How are p-values are calculated from these distributions? Let us check an example. We draw two random samples from the same population (the standard normal distribution) using `rnorm`, and compare the means.

```
set.seed(10)
z1 = rnorm(10)
z2 = rnorm(10)
stripchart(list( z1, z2 ), vertical = T, method = "jitter", col=2:3)
abline(h = c(mean(z1), mean(z2)), col=2:3)
```




```
# the Student t-test
t.test(z1, z2, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  z1 and z2
## t = -2.8063, df = 18, p-value = 0.01168
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.504269 -0.216228
```

```
## sample estimates:
##  mean of x  mean of y
## -0.4906568  0.3695915
```

```
# the Welch t-test with correction for unequal variance
t.test(z1, z2, var.equal = F)
```

```
##
##  Welch Two Sample t-test
##
## data:  z1 and z2
## t = -2.8063, df = 17.967, p-value = 0.01169
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.5043525 -0.2161442
## sample estimates:
##  mean of x  mean of y
## -0.4906568  0.3695915
```

Note that correction decreases the df and elevates the p-values.

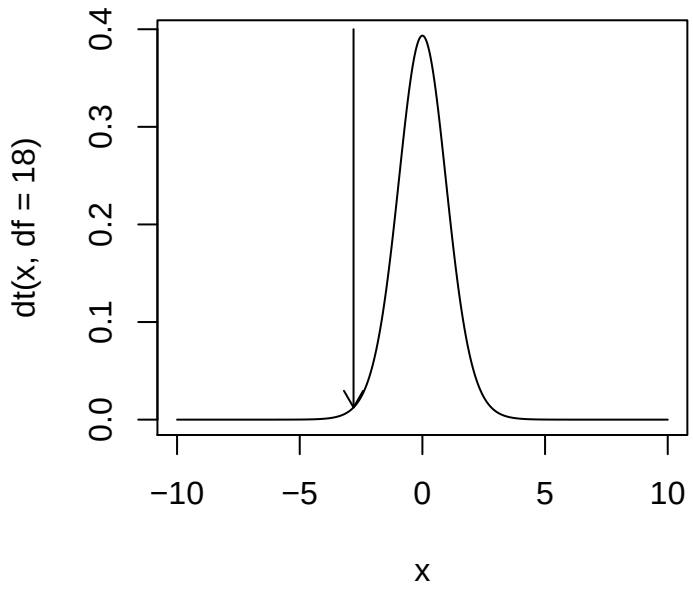
And where does the t-statistic come from?

```
# pooled and scaled variance estimate
v = (1/10 + 1/10)^0.5 * ((9* var(z1) + 9* var(z2))/18)^0.5
# mean difference scaled by variance
t = (mean(z1) - mean(z2))/v
t
```

```
## [1] -2.806301
```

How are the p-values calculated then? They represent the area under the curve, calculated by the `pt` function:

```
plot(x, dt(x, df = 18), type = "l", col=1, lwd=1, lty=1)
arrows(t, 0.4, t, dt(t, df = 18 ), length = 0.1)
```




```
# left tailed area
pt(t, df = 18)
```

```
## [1] 0.005838973
```

```r
# the two-tailed p-value
2 * pt(t, df = 18)
```

```
## [1] 0.01167795
```

```r
# the two-tailed p-value using the corrected df from Welch test
2 * pt(t, df = 17.967)
```

```
## [1] 0.01169236
```

```r
# if the df were higher, the same t-statistic would be more significant
2 * pt(t, df = 1000)
```

```
## [1] 0.005108689
```

```r
# if the df were lower, the same t-statistic would be less significant
2 * pt(t, df = 2)
```

```
## [1] 0.1069855
```

The F-distribution: `df` and `pf`

The null distribution in ANOVA is the F-distribution, representing the ratio of two chi-square variates (scaled differences between observed and expected), with degrees of freedom d1 and d2. The distribution has two parameters, d1 and d2. It ranges from `0` to `-Inf`. It can be used to represent the expected mean differences among groups which are random samples from the same population: i.e. the null of ANOVA.

You can learn more at: https://en.wikipedia.org/wiki/F-distribution

Using this distribution we can calculate p-values for the F-statistics in the ANOVA table:

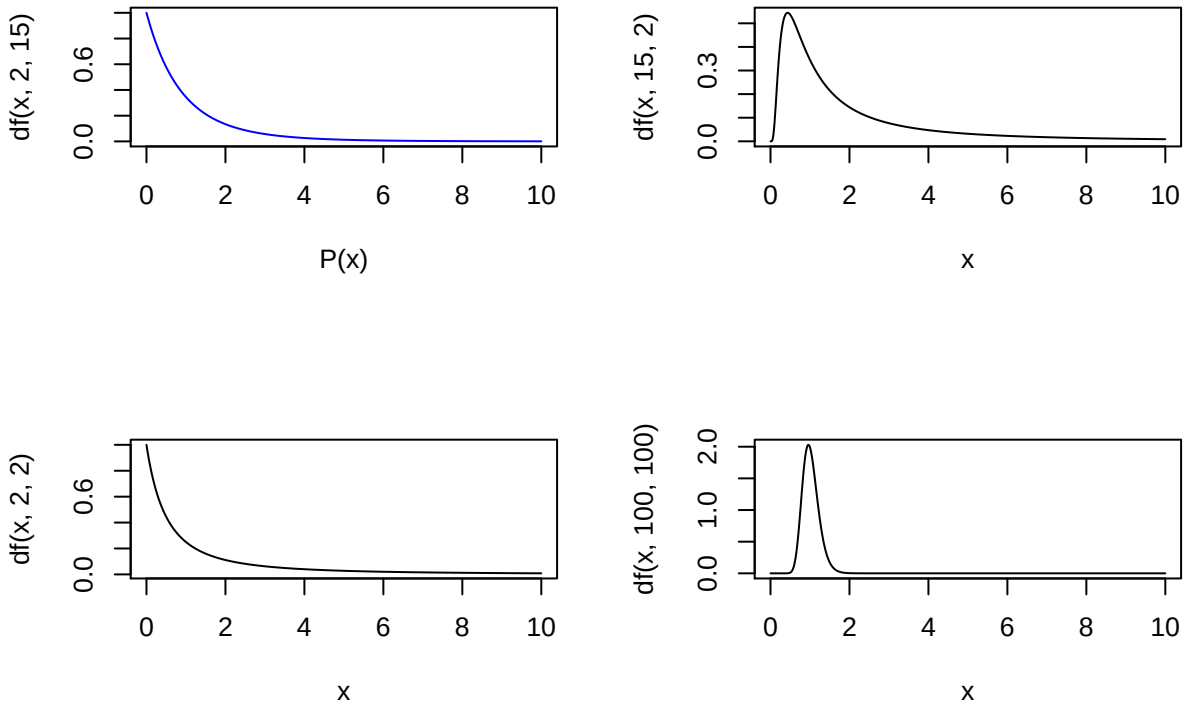
```r
anova(aov(y ~ species2))
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value  Pr(>F)
## species2   2 2.6080 1.30401  4.4267 0.03084 *
## Residuals 15 4.4187 0.29458
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

How does the F distribution look like?

We can study this using the `df` function:

```r
x = seq(0, 10, by=0.01)
par(mfrow=c(2,2))
plot(x, df(x, 2, 15), type="l", col = 4, xlab="P(x)")
plot(x, df(x, 15, 2), type="l")
plot(x, df(x, 2, 2), type="l")
plot(x, df(x, 100, 100), type="l")
```



What does the p-value stand for? In this case, the test is one-sided: the probability of finding an F-value that is greater than or equal to that observed.

```
f = anova(aov(y ~ species2))$F[1]
f
```

```
## [1] 4.426652
```

```
pf(f, 2, 15)
```

```
## [1] 0.9691631
```

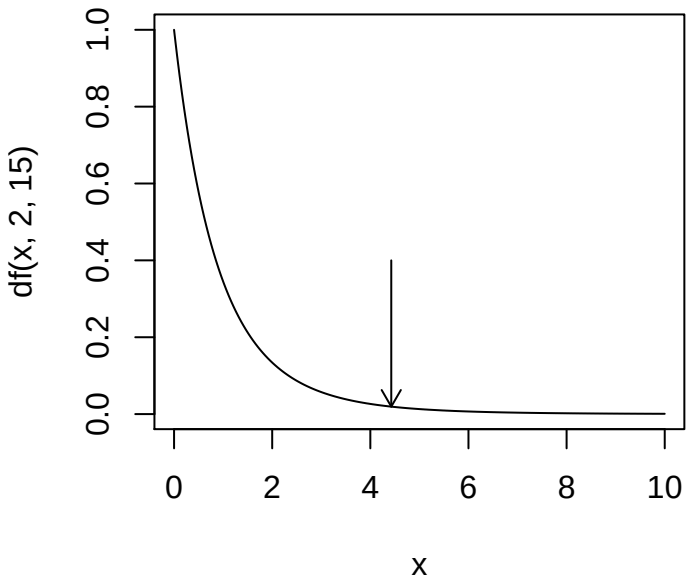
```
# the p-value
1 - pf(f, 2, 15)
```

```
## [1] 0.03083689
```

```
# alternatively, you can ask pf for the upper tail area
pf(f, 2, 15, lower.tail = F)
```

```
## [1] 0.03083689
```

```
plot(x, df(x, 2, 15), type="l")
arrows(f, 0.4, f, df(f, 2, 15), length = 0.1)
```



Linear models with `lm`

`lm` (linear model) works similar to `aov`. Running `lm` itself returns you the fitted values: The first group's mean, and the other groups' means' differences from that of the first group's:

```
lm( y ~ species2 )
```

```
##
## Call:
## lm(formula = y ~ species2)
##
## Coefficients:
## (Intercept)   species2HS   species2PT
##      8.0304      -0.8622      -0.1239
```

```
# compare with this:
species_means
```

```
##       RM       HS       PT
## 8.030392 7.168153 7.906538
```

```
species_means["HS"] - species_means["RM"]
```

```
##         HS
## -0.8622396
```

```
species_means["PT"] - species_means["RM"]
```

```
##         PT
## -0.1238539
```

You can also calculate the confidence intervals for the first group's mean being `0` (not so important), or the difference between the first group's and other groups' means being `0` (i.e. there being a difference among groups)

```
confint( lm( y ~ species2 ) )
```

```
##                 2.5 %     97.5 %
```

```
## (Intercept)  7.5581095  8.5026754
## species2HS  -1.5301485 -0.1943306
## species2PT  -0.7917629  0.5440550
```

Also, `summary.lm` calculates the total variance explained by the factor:

```
summary( lm( y ~ species2 ) )
```

```
##
## Call:
## lm(formula = y ~ species2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89467 -0.27648 -0.04777  0.30464  1.21569
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0304     0.2216  36.242 5.06e-16 ***
## species2HS   -0.8622     0.3134  -2.752   0.0148 *
## species2PT   -0.1239     0.3134  -0.395   0.6982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5428 on 15 degrees of freedom
## Multiple R-squared:  0.3712, Adjusted R-squared:  0.2873
## F-statistic: 4.427 on 2 and 15 DF,  p-value: 0.03084
```

Here **multiple R-squared** stands for the total variance explained in the first gene's expression values (`y`) with the species factor. It will range from `0` to `1`. The more the divergence among groups, and the smaller the divergence within groups, we obtain a smaller R-squared result.

Checking ANOVA assumptions

As we mentioned, in ANOVA we are **assuming normality** (can be not so critical if sample sizes are large) and equal variances among groups. Are these satistified? Let's learn how to check with the first gene:

```
ys = split( y, species2 )
ys
```

```
## $RM
##     RMM1     RMF1     RMF2     RMM2     RMM3     RMF3
## 7.478763 7.850765 7.961526 9.246079 8.273440 7.371782
##
## $HS
##     HSM1     HSF1     HSM2     HSF2     HSM3     HSF3
## 6.273480 7.554903 7.624893 7.456257 7.478299 6.621084
##
## $PT
##     PTF1     PTM1     PTF2     PTM2     PTF3     PTM3
## 7.737963 7.865687 7.895412 8.490531 7.597779 7.851859
```

```
# is the data normally distributed for human?
shapiro.test(ys[[1]])$p.val
```

```
## [1] 0.341577
```

```
# for all 3?
sapply(ys, function(x) shapiro.test(x)$p.val )
```

```
##         RM         HS         PT
## 0.34157700 0.04209703 0.08329061
```

```
# are the variances equal, between human and chimp?
var.test(ys$RM, ys$HS)
```

```
##
##  F test to compare two variances
##
## data:  ys$RM and ys$HS
## F = 1.4123, num df = 5, denom df = 5, p-value = 0.7141
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##   0.1976205 10.0926293
## sample estimates:
## ratio of variances
##           1.412271
```

```
# a test that runs overall comparisons
bartlett.test( y, species2)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  y and species2
## Bartlett's K-squared = 2.6935, df = 2, p-value = 0.2601
```

In real analysis, it would be useful to test this across all genes and check the overall trends.

`oneway.test`: Not assuming equal variance

You can also apply an ANOVA with the Welch correction, which does not assume equal variance.

```
oneway.test(y ~ species2 )
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  y and species2
## F = 4.0437, num df = 2.0000, denom df = 8.8528, p-value = 0.05656
```

Note that the p-value is higher than that in ANOVA.

Two-way ANOVA

You can also use ANOVA to study two factors at the same time:

```
aov( y ~ species2 + sex2 )
```

```
## Call:
##    aov(formula = y ~ species2 + sex2)
##
## Terms:
```

```
##                 species2     sex2 Residuals
## Sum of Squares  2.608020 0.357170  4.061555
## Deg. of Freedom        2        1        14
##
## Residual standard error: 0.5386196
## Estimated effects may be unbalanced
```

```
summary( aov( y ~ species2 + sex2 ) )
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## species2     2  2.608  1.3040   4.495 0.0311 *
## sex2         1  0.357  0.3572   1.231 0.2859
## Residuals   14  4.062  0.2901
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# and to obtain the p-values for species and sex:
anova( aov( y ~ species2 + sex2 ) )$Pr[1:2]
```

```
## [1] 0.03105696 0.28587969
```

Is the total sum of squares the same as in the one-way ANOVA model (only the species effect)?

```
sum( anova( aov( y ~ species2 + sex2 ) )$Sum )
```

```
## [1] 7.026745
```

```
sum( anova( aov( y ~ species2 ) )$Sum )
```

```
## [1] 7.026745
```

You can also use `lm` to estimate each level:

```
summary( lm( y ~ species2 + sex2 ) )
```

```
##
## Call:
## lm(formula = y ~ species2 + sex2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03554 -0.19209  0.02214  0.27923  1.07482
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.8895     0.2539  31.072 2.57e-14 ***
## species2HS   -0.8622     0.3110  -2.773    0.015 *
## species2PT   -0.1239     0.3110  -0.398    0.696
## sex2M         0.2817     0.2539   1.110    0.286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5386 on 14 degrees of freedom
## Multiple R-squared:  0.422,  Adjusted R-squared:  0.2981
## F-statistic: 3.407 on 3 and 14 DF,  p-value: 0.04752
```

Note that adding the sex factor has interesting effects on the table. First, even though sex is not significant, it still explains some part of the variance, and the SS of residuals decreases. In this case, so does mean SS residuals. The species SS remains the same. The species F-value increases. But now we compare the new

F-values with different F-distributions, with smaller d.f.:

```
pf(4.495, 2, 14, lower.tail = F)
```

```
## [1] 0.03105442
```

```
# this would not have been the same:
pf(4.495, 2, 15, lower.tail = F)
```

```
## [1] 0.0295432
```

Two-way ANOVA with interaction

```
summary( aov( y ~ species2 * sex2 ) )
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## species2       2  2.608  1.3040   4.229 0.0407 *
## sex2           1  0.357  0.3572   1.158 0.3030
## species2:sex2  2  0.361  0.1807   0.586 0.5718
## Residuals     12  3.700  0.3084
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary( lm( y ~ species2 * sex2 ) )
```

```
##
## Call:
## lm(formula = y ~ species2 * sex2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85400 -0.21404  0.05849  0.31949  0.91332
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.72802    0.32060  24.105 1.56e-11 ***
## species2HS        -0.51728    0.45340  -1.141    0.276
## species2PT         0.01569    0.45340   0.035    0.973
## sex2M              0.60474    0.45340   1.334    0.207
## species2HS:sex2M -0.68993    0.64120  -1.076    0.303
## species2PT:sex2M -0.27910    0.64120  -0.435    0.671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5553 on 12 degrees of freedom
## Multiple R-squared:  0.4734, Adjusted R-squared:  0.254
## F-statistic: 2.158 on 5 and 12 DF,  p-value: 0.1277
```

Note that the species effect is now slightly less significant. Adding non-significant terms into the model will influence the overall result.

This is reflected in the **"Adjusted R-squared"** term. While "Multiple R-squared" increases with all new terms we test, the "Adjusted R-squared" does not. It actually decreases. This is because it is a **penalized** version of the proportion of the variance explained for including non-significant terms.

What do the estimates mean now? The effect of each term on the average mean: each species, each sex, and each species-sex term:

```r
summary(lm(y ~ species2 * sex2))
```

```
##
## Call:
## lm(formula = y ~ species2 * sex2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85400 -0.21404  0.05849  0.31949  0.91332
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.72802    0.32060  24.105 1.56e-11 ***
## species2HS        -0.51728    0.45340  -1.141    0.276
## species2PT         0.01569    0.45340   0.035    0.973
## sex2M              0.60474    0.45340   1.334    0.207
## species2HS:sex2M -0.68993    0.64120  -1.076    0.303
## species2PT:sex2M -0.27910    0.64120  -0.435    0.671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5553 on 12 degrees of freedom
## Multiple R-squared:  0.4734, Adjusted R-squared:  0.254
## F-statistic: 2.158 on 5 and 12 DF,  p-value: 0.1277
```

```r
yc = summary(lm(y ~ species2 * sex2))$coef[,1]
yc
```

```
##      (Intercept)       species2HS       species2PT            sex2M
##       7.72802431      -0.51727615       0.01569382       0.60473621
## species2HS:sex2M species2PT:sex2M
##      -0.68992684      -0.27909550
```

```r
# e.g. how do I retrieve the human male average?
yc["(Intercept)"] +  # this is female macaques
  yc["species2HS"] +
  yc["sex2M"] +
  yc["species2HS:sex2M"]  # the interaction term for being human and male
```

```
## (Intercept)
##    7.125558
```

Testing differential expression

Let's run the full test with interaction on all expressed genes:

```r
nmat6_aovp = t( apply(nmat6, 1, function(y)
  anova( aov( y ~ species2 * sex2 ) )$Pr[1:3] ) )
```

```r
dim(nmat6_aovp)
```

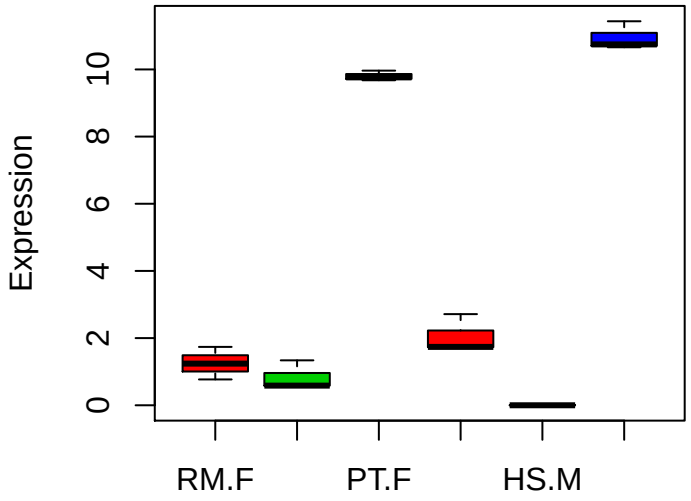
```
## [1] 15595     3
```

```r
colnames( nmat6_aovp ) = c('species', 'sex', 'int')
head(nmat6_aovp)
```

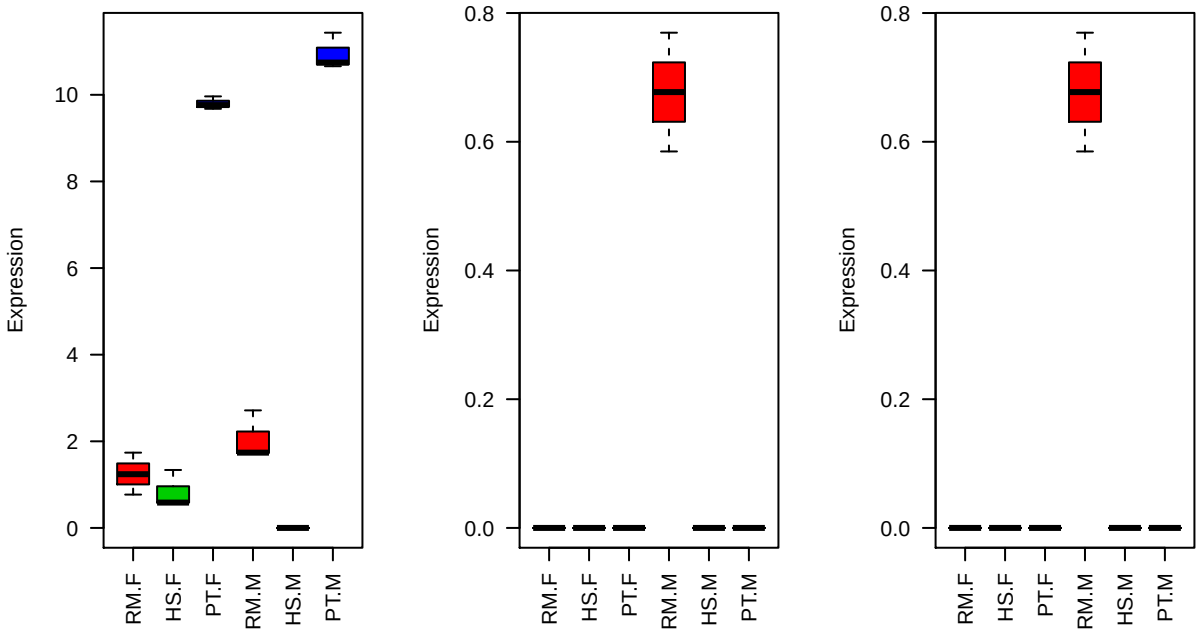
```
##                     species        sex       int
## ENSG00000000003 0.040730044 0.30297619 0.5717540
## ENSG00000000005 0.099684058 0.43790571 0.4568374
## ENSG00000000419 0.844576926 0.93365422 0.5792549
## ENSG00000000457 0.008606455 0.02528147 0.5430024
## ENSG00000000460 0.017986444 0.43213919 0.7036096
## ENSG00000000938 0.377448044 0.81321773 0.5495795
```

Now check the most significant gene for the sex effect by plotting the expression values in a boxplot, to get a feeling if it worked:

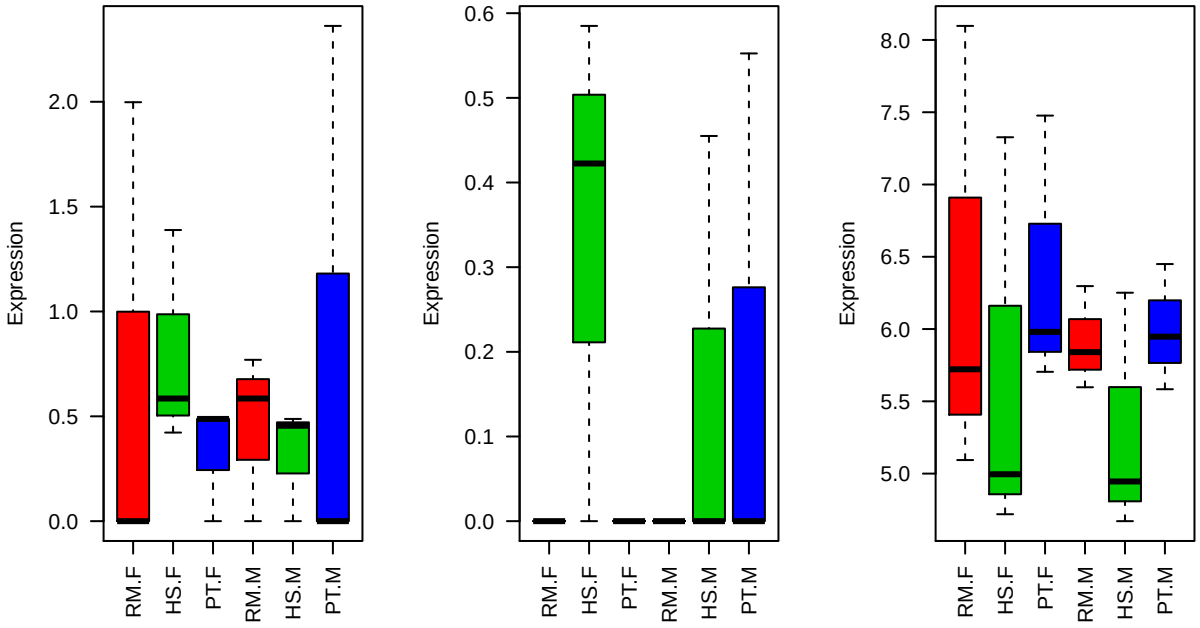
```
i = which.min ( nmat6_aovp[,'species'] )
y = nmat6 [i, ]
boxplot ( y ~ species2 * sex2, col=2:4, ylab='Expression' )
```




```
par(mfrow=c(1,3))
# most significant genes
for (i in 1:3)  {
  y = nmat6 [which.min ( nmat6_aovp[,i] ), ]
  boxplot ( y ~ species2 * sex2, col=2:4, ylab='Expression', las=2)
}
```




```
par(mfrow=c(1,3))
# least significant genes
for (i in 1:3)  {
  y = nmat6 [which.max ( nmat6_aovp[,i] ), ]
  boxplot ( y ~ species2 * sex2, col=2:4, ylab='Expression', las=2)
}
```



order vs. rank

We could also check the e.g. 4 lowest p-values, using `order`:

```
# the indexes of the elements starting with the lowest value
order( c(20, 15, 5, 10) )
```

```
## [1] 3 4 2 1
```

```
# compare with the ranks
rank( c(20, 15, 5, 10) )
```

```
## [1] 4 3 1 2
```

```
head(order( nmat6_aovp[ ,"species"] ))
```

```
## [1] 14539 15439 14537 14518 10975  4621
```

```
which.min( nmat6_aovp[ ,"species"] )
```

```
## ENSG00000208587
##           14539
```

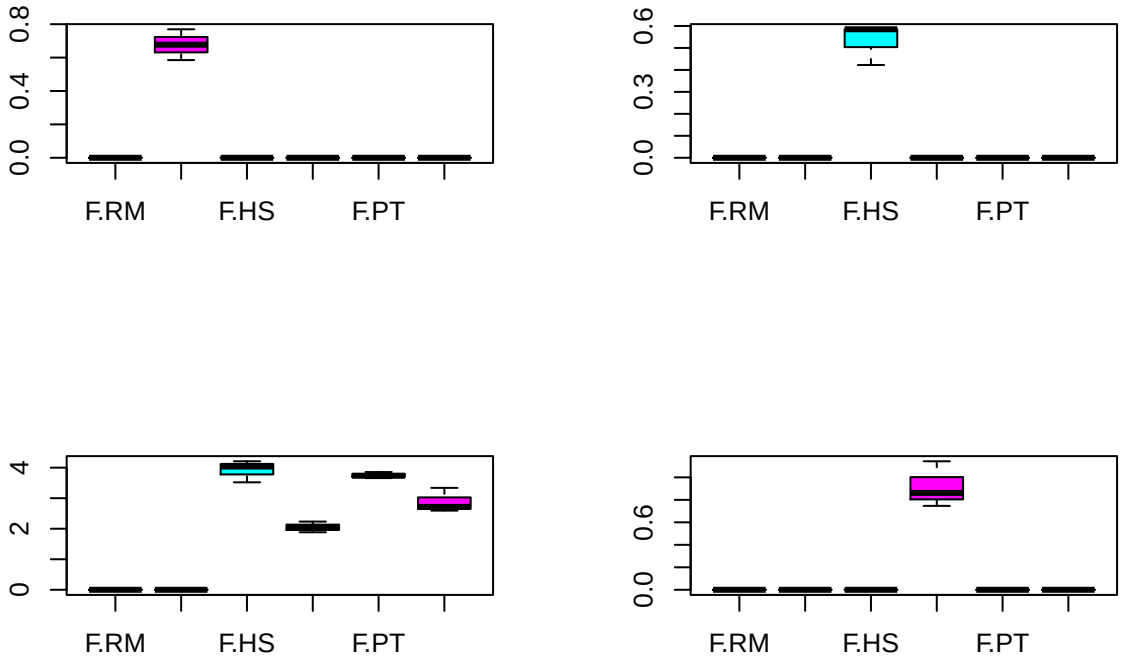
```
# the second lowest's index
minx2 = order( nmat6_aovp[ ,"species"] )[2]
minx2
```

```
## [1] 15439
```

```
# the second lowest's gene name
minx2 = names( sort( nmat6_aovp[ ,"species"] )[2] )
minx2
```

```
## [1] "ENSG00000219802"
```

```
# plot the lowest
par(mfrow=c(2,2))
for (i in 1:4)
boxplot( nmat6[ order( nmat6_aovp[ ,"sex"] )[i], ] ~ sex2 + species2, col=5:6)
```



Number of nominally significant genes

How many such genes are there?

```
# number and proportion of with cases p<0.05
colSums(nmat6_aovp < 0.05)
```

```
## species     sex     int
##    8224     867     743
```

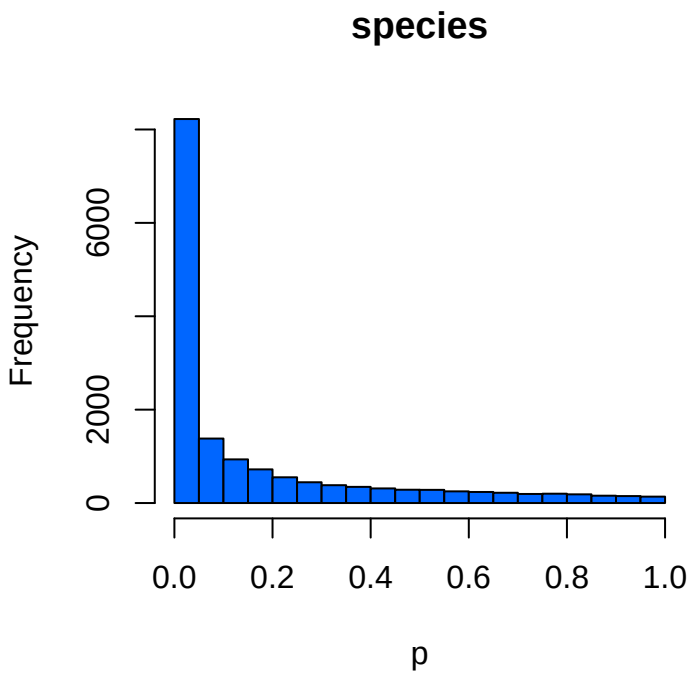
```
colMeans(nmat6_aovp < 0.05)
```

```
##    species        sex        int
## 0.52734851 0.05559474 0.04764348
```

```
# genes which are significant for >1 tests
table(rowSums(nmat6_aovp < 0.05))
```

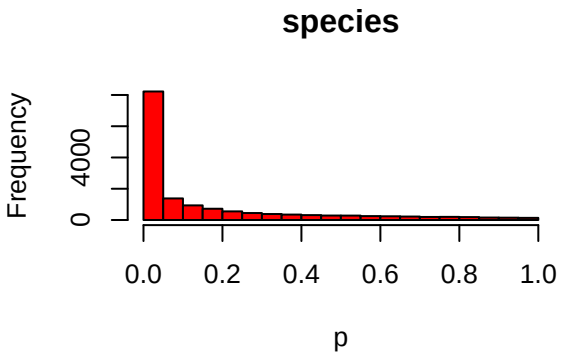
```
##
##    0    1    2    3
## 6910 7670  881  134
```

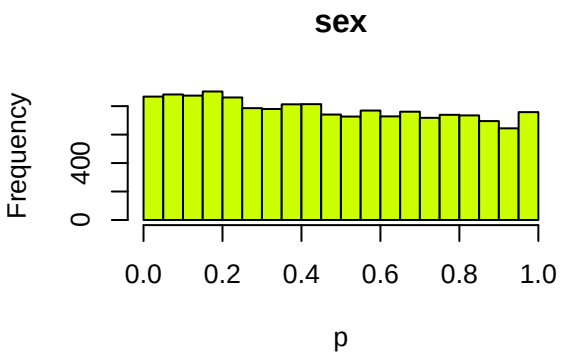
```
# histogram of the data
hist( nmat6_aovp[,"species"], xlab='p', col=rainbow(5)[i], main="species")
```

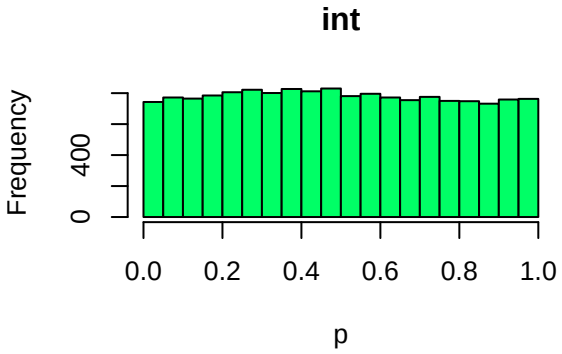


For all the p-values:

```
par( mfrow=c(2,2))
for (i in 1:3) {
    hist( nmat6_aovp[,i], xlab='p', col=rainbow(5)[i],
          main=colnames(nmat6_aovp)[i])
}
```







Multiple testing correction

One question is whether these distributions really meaningful, i.e. **non-random**?

Let's try a t-test on random data, and study the p-values:

```
set.seed(10)
t.test(rnorm(10), rnorm(10))$p.val
```

```
## [1] 0.01169219
```

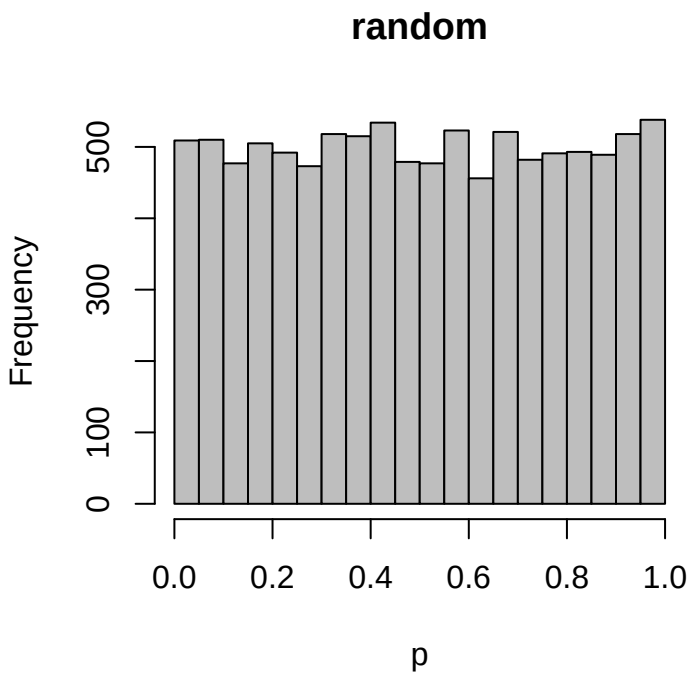
```
t.test(rnorm(10), rnorm(10))$p.val
```

```
## [1] 0.4233697
```

```
ranp = sapply( 1:10000, function(i) {   # here i is just a counter, no role inside the loop
  t.test(rnorm(10), rnorm(10))$p.val
})
mean( ranp < 0.05 )
```

```
## [1] 0.0509
```

```
hist( ranp, xlab='p', col=8, main="random")
```



As we had discussed in earlier classes, p-values from random data are already expected to be uniformly distributed. What can you conclude at this point on the effects of the 3 factors?

The false discovery rate

There are two ways we can do this, either using an algorithm implemented in the `p.adjust` function, such as Bonferroni, Benjamini-Hochberg (BH), or Benjamini-Hochberg-Yekutieli (BY). Bonferroni is the most conservative - multiplies the p-values by the number of tests. BH is commonly used, but assumes independence. You can find a description here: https://en.wikipedia.org/wiki/False_discovery_rate#Benjamini.E2.80.93Hochberg_procedure

BH converts p-values into so-called q-values, which indicate the false discovery rate at the cutoff. A q-value of `< 0.01` indicates that the set of genes with such statistic are expected to contain 1% false positives.

However, BH assumes independence among tests. Because gene expression patterns among genes are not independent in a dataset (genes are co-regulated, and one being significantly differentially expressed usually means other related genes are also significant), this assumption may not be valid.

The Benjamini-Hochberg-Yekutieli (BY) correction does **not assume independence**, and thus may be preferable in our case: https://en.wikipedia.org/wiki/False_discovery_rate#Benjamini.E2.80.93Hochberg.E2.80.93Yekutieli_procedure

Examples for correction:

```
p.adjust(c(0.01, 0.05, 1), method = "bonferroni")
```

```
## [1] 0.03 0.15 1.00
```

```
p.adjust(c(0.01, 0.05, 1), method = "BH")
```

```
## [1] 0.030 0.075 1.000
```

```
p.adjust(c(0.01, 0.05, 1), method = "BY")
```

```
## [1] 0.0550 0.1375 1.0000
```

Now on the real p-values:

```
head(nmat6_aovp)
```

```
##                     species        sex       int
## ENSG00000000003 0.040730044 0.30297619 0.5717540
## ENSG00000000005 0.099684058 0.43790571 0.4568374
## ENSG00000000419 0.844576926 0.93365422 0.5792549
## ENSG00000000457 0.008606455 0.02528147 0.5430024
## ENSG00000000460 0.017986444 0.43213919 0.7036096
## ENSG00000000938 0.377448044 0.81321773 0.5495795
```

```
nmat6_aovq = apply(nmat6_aovp, 2, function(x) {
  p.adjust( x, 'BY')
})
dim(nmat6_aovq)
```

```
## [1] 15595     3
```

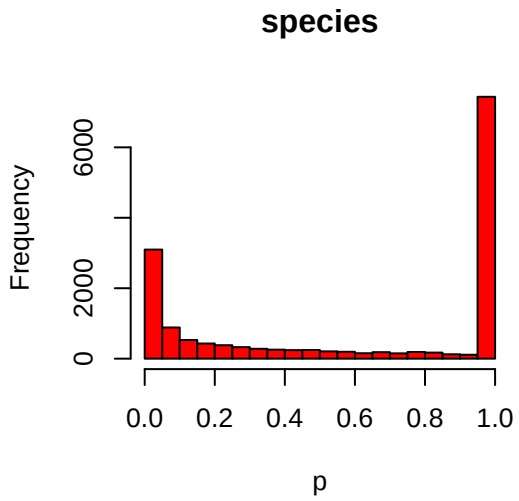
```
head(nmat6_aovq)
```

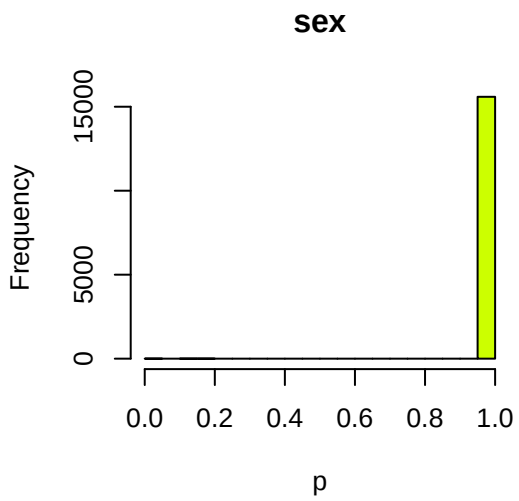
```
##                     species sex int
## ENSG00000000003 0.8291890   1   1
## ENSG00000000005 1.0000000   1   1
## ENSG00000000419 1.0000000   1   1
## ENSG00000000457 0.2559279   1   1
## ENSG00000000460 0.4471874   1   1
## ENSG00000000938 1.0000000   1   1
```

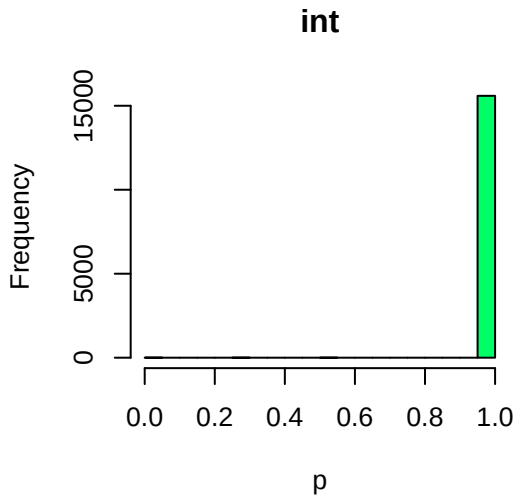
```
# how many significant left now?
apply(nmat6_aovq < 0.05, 2, mean)
```

```
##      species          sex          int
## 0.1985892914 0.0001282462 0.0001923693
```

```
par(mfrow=c(2,2))
for (i in 1:3) {
  hist( nmat6_aovq[,i], xlab='p', col=rainbow(5)[i], main=colnames(nmat6_aovq)[i])
}
```







This implies that 20% of the transcriptome shows a species effect, but the sex or interaction effects are practically not significant after correcting for multiple testing.

Save your results:

```
save(nmat6_aovp, nmat6_aovq, file="liver_transcriptome_v4.Rdata")
```