BIO 754 - Lecture 05

23-03-2017

Contents

The primate liver dataset (continued)

We had started studying a **primate liver transcriptome** RNA-sequencing dataset (Blekhman et al 2010 Genome Research).

It is available at NCBI GEO: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE17274. We have already downloaded the raw read counts.

The dataset contains:

- 6 humans, 6 chimpanzees, 6 rhesus macaques
- 3 females and 3 males each
- 2 technical replicates from each sample

The dataset was prepared as follows, starting from *postmortem* liver data:

- Total RNA extraction -> polyA mRNA selection -> cDNA conversion -> random fragmentation -> library preparation,
- RNA-sequencing on a Illumina Genome Analyzer2 ,
- Using the 35-bp **single-end read** mode,
- About 6 million reads per sample.

How would experiments conducted today differ?

- Min. 10 million reads per sample for differential expression,
- Min. 30 million if you are to study differential splicing,
- Even more if you need to construct your transcriptome *de novo*.

The preprocessing steps applied in the published study:

i. use exons from the Ensembl database (http://www.ensembl.org/index.html),
ii. identify 1:1 orthologous exons among species,
iii. map reads to this orthologous exon set (corresponding to 20,000+ genes in total),
iv. choose un-ambiguously mapped reads,
v. sum number of reads mapped to all exons of a gene,
vi. divide by total number of reads that mapped to genes in that lane (X 1 million), to obtain relative gene expression levels,
vii. transform & analyze.

We start the analysis here from step (v).

Note that the dataset at step (v) does not include normalization for either total read depth nor for gene size, which are included in statistics such as TPM, RPKM and FPKM. Normalization for total read depth we will implement. Normalization for gene size we will not include, as we will *not* be comparing expression level estimates *among* genes.

What questions could we try to answer with such data?

- Does transcriptome divergence reflect common environments?
- Or does it reflect common descent (phylogeny)?
- Which genes will be **differentially expressed (DE)** among the species?
- Which genes are differentially expressed among females & males?
- Do DE gene sets correspond to specific functions?
- How is DE regulated?

Some of these will be questions we will tackle during the course.

Read in the data and overview

```
# double check the dataset is already in the working directory
grep("GSE17274", list.files(), val=T)
```

```
## [1] "GSE17274_ReadCountPerLane.txt"
```

```r
# read in
mat = read.table(
  "./GSE17274_ReadCountPerLane.txt", row.names=1, head=T, sep="\t" )
# remember:
# head argument: first row contains column names
# row.names: first column contains row names
# sep: use tab as seperator (actually the default)

# now, take an overview of the data:
dim(mat)  # check the dimensions - is it what you expect?
```

```
## [1] 20689    36
```

```r
# does the data look fine?
head(mat)[,1:6] # samples in columns and genes in rows
```

```
##                 R1L1.HSM1 R1L2.PTF1 R1L3.RMM1 R1L4.HSF1 R1L6.PTM1
## ENSG00000000003        60       285       207       172       176
## ENSG00000000005         0         1         1         0         0
## ENSG00000000419        17        54        20        36        23
## ENSG00000000457        50        61        68        41       143
## ENSG00000000460         9         6         2         3         3
## ENSG00000000938        32        50        44        23        99
##                 R1L7.RMF1
## ENSG00000000003       259
## ENSG00000000005         0
## ENSG00000000419        38
## ENSG00000000457        42
## ENSG00000000460         2
## ENSG00000000938        46
```

```r
tail(mat)[,1:6]
```

```
##                 R1L1.HSM1 R1L2.PTF1 R1L3.RMM1 R1L4.HSF1 R1L6.PTM1
## ENSG00000221781         0         0         0         0         0
## ENSG00000221782         0         0         0         0         0
## ENSG00000221783         0         0         0         0         0
## ENSG00000221784         0         0         0         0         0
## ENSG00000221786         0         0         0         0         0
## ENSG00000221788         0         0         0         0         0
##                 R1L7.RMF1
## ENSG00000221781         0
## ENSG00000221782         0
## ENSG00000221783         1
## ENSG00000221784         0
## ENSG00000221786         0
## ENSG00000221788         0
```

```r
# two other approaches to obtain a general overview
str(mat)
```

```
## 'data.frame':    20689 obs. of  36 variables:
##  $ R1L1.HSM1: int  60 0 17 50 9 32 1726 99 155 13 ...
##  $ R1L2.PTF1: int  285 1 54 61 6 50 2617 135 211 35 ...
##  $ R1L3.RMM1: int  207 1 20 68 2 44 7207 109 455 52 ...
##  $ R1L4.HSF1: int  172 0 36 41 3 23 2262 155 323 19 ...
##  $ R1L6.PTM1: int  176 0 23 143 3 99 1155 114 137 34 ...
##  $ R1L7.RMF1: int  259 0 38 42 2 46 7168 150 572 39 ...
##  $ R2L2.RMF2: int  299 1 40 47 1 18 5986 68 500 25 ...
##  $ R2L3.HSM2: int  219 1 42 30 2 26 3278 118 446 16 ...
##  $ R2L4.PTF2: int  213 1 35 111 5 53 4248 141 172 42 ...
##  $ R2L6.RMM2: int  676 0 39 55 2 20 2577 45 324 19 ...
##  $ R2L7.HSF2: int  147 0 26 28 8 30 3473 118 377 15 ...
##  $ R2L8.PTM2: int  316 0 26 110 3 16 2361 65 269 36 ...
##  $ R3L1.RMM3: int  338 0 36 76 2 29 5515 61 842 26 ...
##  $ R3L2.HSF2: int  153 0 35 34 9 35 3752 133 360 15 ...
##  $ R3L3.PTM1: int  233 0 36 131 9 99 1413 140 161 35 ...
##  $ R3L4.RMF3: int  242 0 22 61 5 28 9509 193 571 45 ...
##  $ R3L6.HSM3: int  199 0 33 53 11 32 2945 147 240 13 ...
##  $ R3L7.PTF3: int  180 0 35 46 2 39 2155 161 427 13 ...
##  $ R3L8.RMM1: int  217 1 29 49 5 36 7029 92 418 45 ...
##  $ R4L1.HSM3: int  160 0 29 42 6 33 2601 102 219 15 ...
##  $ R4L2.HSF1: int  157 0 45 50 3 21 2503 142 307 17 ...
##  $ R4L3.RMM3: int  367 0 54 87 2 43 6319 79 910 35 ...
##  $ R4L4.PTF1: int  289 0 46 70 2 34 2765 170 206 28 ...
##  $ R4L6.PTM2: int  369 1 32 129 4 15 2658 86 300 38 ...
##  $ R4L7.RMF3: int  238 2 35 57 5 26 8738 186 539 63 ...
##  $ R4L8.HSM2: int  202 0 36 43 5 17 3105 116 538 17 ...
##  $ R5L1.RMF1: int  252 0 29 47 2 27 6398 142 490 23 ...
##  $ R5L2.HSM1: int  61 0 22 64 6 41 1874 101 181 16 ...
##  $ R5L3.PTF3: int  206 1 30 41 3 37 2180 152 458 12 ...
##  $ R5L4.RMM2: int  672 0 43 59 1 14 2619 43 340 16 ...
##  $ R5L8.RMF2: int  165 0 32 22 3 10 3978 38 286 9 ...
##  $ R6L2.PTM3: int  216 1 40 71 5 30 1269 162 113 14 ...
##  $ R6L4.PTM3: int  212 0 53 78 5 28 1384 169 105 27 ...
##  $ R6L6.PTF2: int  216 3 31 118 3 51 4083 160 156 38 ...
##  $ R8L1.HSF3: int  78 0 16 34 7 112 1665 79 151 16 ...
##  $ R8L2.HSF3: int  90 0 40 42 5 98 1740 110 155 20 ...
```

```
summary(mat)
```

```
##    R1L1.HSM1          R1L2.PTF1          R1L3.RMM1
##  Min.   :    0.00   Min.   :     0.0   Min.   :     0.0
##  1st Qu.:    0.00   1st Qu.:     0.0   1st Qu.:     0.0
##  Median :    3.00   Median :     5.0   Median :     5.0
##  Mean   :   65.08   Mean   :   128.9   Mean   :   128.4
##  3rd Qu.:   30.00   3rd Qu.:    49.0   3rd Qu.:    48.0
##  Max.   :91116.00   Max.   :235597.0   Max.   :304723.0
##    R1L4.HSF1          R1L6.PTM1          R1L7.RMF1        R2L2.RMF2
##  Min.   :     0.00   Min.   :    0.00   Min.   :     0   Min.   :     0
##  1st Qu.:     0.00   1st Qu.:    0.00   1st Qu.:     0   1st Qu.:     0
##  Median :     4.00   Median :    5.00   Median :     5   Median :     4
##  Mean   :    80.53   Mean   :   71.61   Mean   :   116   Mean   :   113
##  3rd Qu.:    35.00   3rd Qu.:   35.00   3rd Qu.:    47   3rd Qu.:    41
##  Max.   :276431.00   Max.   :70066.00   Max.   :225484   Max.   :373964
```

```
##    R2L3.HSM2          R2L4.PTF2           R2L6.RMM2
##  Min.   :     0.0   Min.   :     0.00   Min.   :     0.00
##  1st Qu.:     0.0   1st Qu.:     0.00   1st Qu.:     0.00
##  Median :     5.0   Median :     4.00   Median :     3.00
##  Mean   :   107.2   Mean   :    92.34   Mean   :    93.88
##  3rd Qu.:    42.0   3rd Qu.:    38.00   3rd Qu.:    28.00
##  Max.   :325834.0   Max.   :113411.00   Max.   :202945.00
##    R2L7.HSF2          R2L8.PTM2           R3L1.RMM3
##  Min.   :     0.00   Min.   :     0.00   Min.   :     0.0
##  1st Qu.:     0.00   1st Qu.:     0.00   1st Qu.:     0.0
##  Median :     3.00   Median :     4.00   Median :     3.0
##  Mean   :    78.31   Mean   :    77.73   Mean   :   102.5
##  3rd Qu.:    32.00   3rd Qu.:    34.00   3rd Qu.:    40.0
##  Max.   :241969.00   Max.   :179575.00   Max.   :135723.0
##    R3L2.HSF2          R3L3.PTM1           R3L4.RMF3
##  Min.   :     0.00   Min.   :    0.00   Min.   :     0.0
##  1st Qu.:     0.00   1st Qu.:    0.00   1st Qu.:     0.0
##  Median :     4.00   Median :    5.00   Median :     6.0
##  Mean   :    87.05   Mean   :   81.91   Mean   :   129.8
##  3rd Qu.:    36.00   3rd Qu.:   40.00   3rd Qu.:    59.0
##  Max.   :265900.00   Max.   :80305.00   Max.   :147554.0
##    R3L6.HSM3          R3L7.PTF3           R3L8.RMM1
##  Min.   :     0.00   Min.   :     0.00   Min.   :     0.0
##  1st Qu.:     0.00   1st Qu.:     0.00   1st Qu.:     0.0
##  Median :     5.00   Median :     4.00   Median :     4.0
##  Mean   :    95.42   Mean   :    88.85   Mean   :   121.1
##  3rd Qu.:    41.00   3rd Qu.:    39.00   3rd Qu.:    45.0
##  Max.   :215637.00   Max.   :158409.00   Max.   :290488.0
##    R4L1.HSM3          R4L2.HSF1           R4L3.RMM3
##  Min.   :     0.00   Min.   :     0.00   Min.   :     0.0
##  1st Qu.:     0.00   1st Qu.:     0.00   1st Qu.:     0.0
##  Median :     4.00   Median :     4.00   Median :     4.0
##  Mean   :    88.23   Mean   :    83.09   Mean   :   116.6
##  3rd Qu.:    37.00   3rd Qu.:    36.00   3rd Qu.:    46.0
##  Max.   :195288.00   Max.   :287958.00   Max.   :154804.0
##    R4L4.PTF1          R4L6.PTM2           R4L7.RMF3
##  Min.   :     0.0   Min.   :     0.00   Min.   :     0.0
##  1st Qu.:     0.0   1st Qu.:     0.00   1st Qu.:     0.0
##  Median :     5.0   Median :     5.00   Median :     5.0
##  Mean   :   129.4   Mean   :    94.08   Mean   :   122.5
##  3rd Qu.:    49.0   3rd Qu.:    42.00   3rd Qu.:    55.0
##  Max.   :238287.0   Max.   :209435.00   Max.   :138475.0
##    R4L8.HSM2          R5L1.RMF1         R5L2.HSM1
##  Min.   :     0.0   Min.   :     0   Min.   :     0.00
##  1st Qu.:     0.0   1st Qu.:     0   1st Qu.:     0.00
##  Median :     5.0   Median :     4   Median :     3.00
##  Mean   :   104.8   Mean   :   102   Mean   :    72.39
##  3rd Qu.:    41.0   3rd Qu.:    41   3rd Qu.:    33.00
##  Max.   :314901.0   Max.   :202623   Max.   :101436.00
##    R5L3.PTF3          R5L4.RMM2           R5L8.RMF2
##  Min.   :     0.00   Min.   :     0.00   Min.   :     0.00
##  1st Qu.:     0.00   1st Qu.:     0.00   1st Qu.:     0.00
##  Median :     4.00   Median :     3.00   Median :     2.00
##  Mean   :    87.68   Mean   :    95.44   Mean   :    74.14
```

```
##  3rd Qu.:    38.00   3rd Qu.:    28.00   3rd Qu.:    27.00
##  Max.   :161030.00   Max.   :208149.00   Max.   :251481.00
##    R6L2.PTM3           R6L4.PTM3           R6L6.PTF2
##  Min.   :    0.00   Min.   :    0.00   Min.   :     0.00
##  1st Qu.:    0.00   1st Qu.:    0.00   1st Qu.:     0.00
##  Median :    3.00   Median :    3.00   Median :     4.00
##  Mean   :   84.35   Mean   :   87.17   Mean   :    90.94
##  3rd Qu.:   34.00   3rd Qu.:   35.00   3rd Qu.:    38.00
##  Max.   :93337.00   Max.   :97961.00   Max.   :110345.00
##    R8L1.HSF3           R8L2.HSF3
##  Min.   :     0.00   Min.   :     0.00
##  1st Qu.:     0.00   1st Qu.:     0.00
##  Median :     5.00   Median :     5.00
##  Mean   :    67.37   Mean   :    70.11
##  3rd Qu.:    35.00   3rd Qu.:    36.00
##  Max.   :111548.00   Max.   :117717.00
```

```
class(mat)
```

```
## [1] "data.frame"
```

Data frames are useful for storing complex data, but our matrix is just numeric. We therefore prefer to convert to `matrix`. This is helpful because some functions for numeric data (e.g. `hist`) do not work on data frames but only numeric matrices:

```
mat = as.matrix(mat)
class(mat)
```

```
## [1] "matrix"
```

What are the column names?

```
colnames(mat)
```

```
##  [1] "R1L1.HSM1" "R1L2.PTF1" "R1L3.RMM1" "R1L4.HSF1" "R1L6.PTM1"
##  [6] "R1L7.RMF1" "R2L2.RMF2" "R2L3.HSM2" "R2L4.PTF2" "R2L6.RMM2"
## [11] "R2L7.HSF2" "R2L8.PTM2" "R3L1.RMM3" "R3L2.HSF2" "R3L3.PTM1"
## [16] "R3L4.RMF3" "R3L6.HSM3" "R3L7.PTF3" "R3L8.RMM1" "R4L1.HSM3"
## [21] "R4L2.HSF1" "R4L3.RMM3" "R4L4.PTF1" "R4L6.PTM2" "R4L7.RMF3"
## [26] "R4L8.HSM2" "R5L1.RMF1" "R5L2.HSM1" "R5L3.PTF3" "R5L4.RMM2"
## [31] "R5L8.RMF2" "R6L2.PTM3" "R6L4.PTM3" "R6L6.PTF2" "R8L1.HSF3"
## [36] "R8L2.HSF3"
```

```
# and row names?
head( rownames(mat)  )
```

```
## [1] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457"
## [5] "ENSG00000000460" "ENSG00000000938"
```

Check if all the gene IDs are Ensembl, using `grep`, in one line:

```
length( grep ("ENSG", rownames( mat) ) ) == nrow( mat )
```

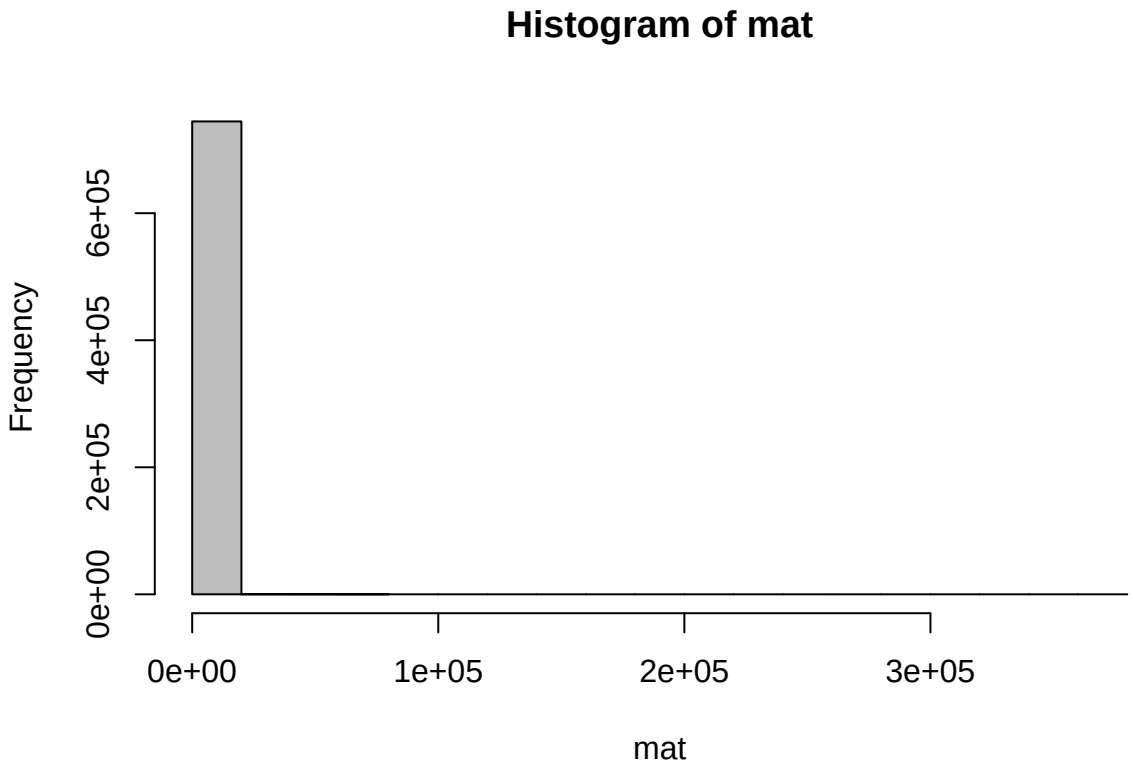
```
## [1] TRUE
```

```
# or you could also use
unique( substr (rownames( mat), 1, 4) )
```

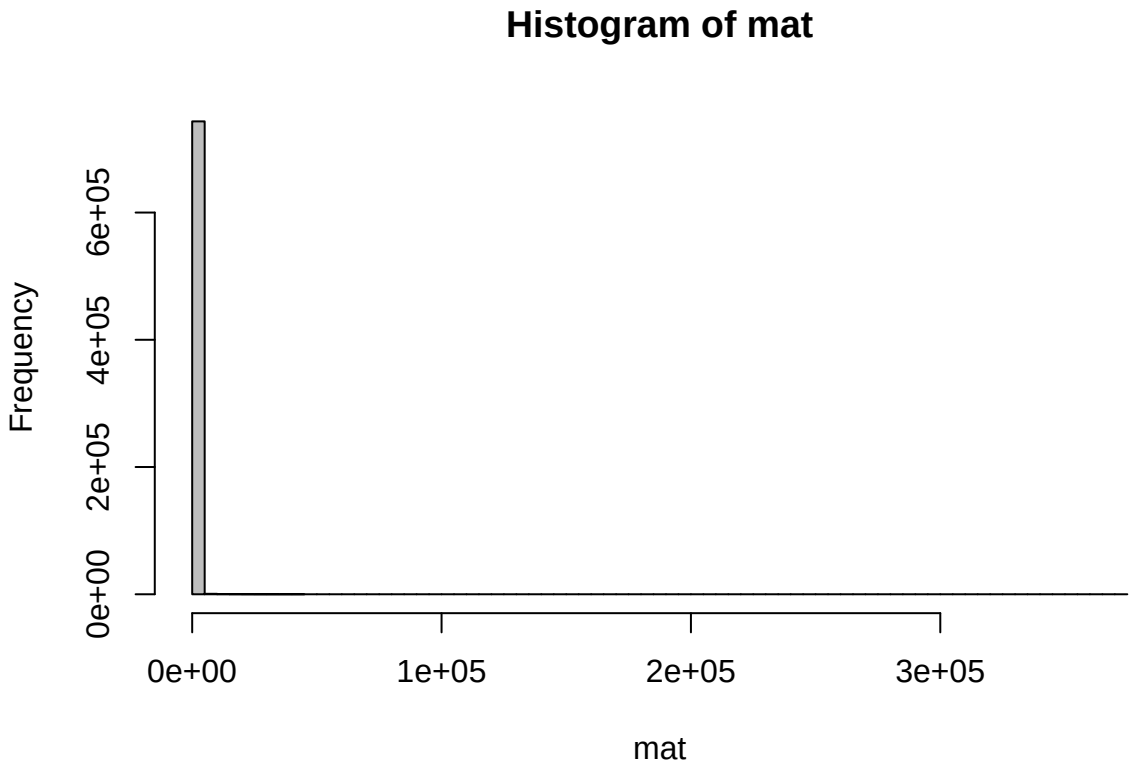
```
## [1] "ENSG"
```

Let's study the histogram:

```
hist( mat, col=8)
```



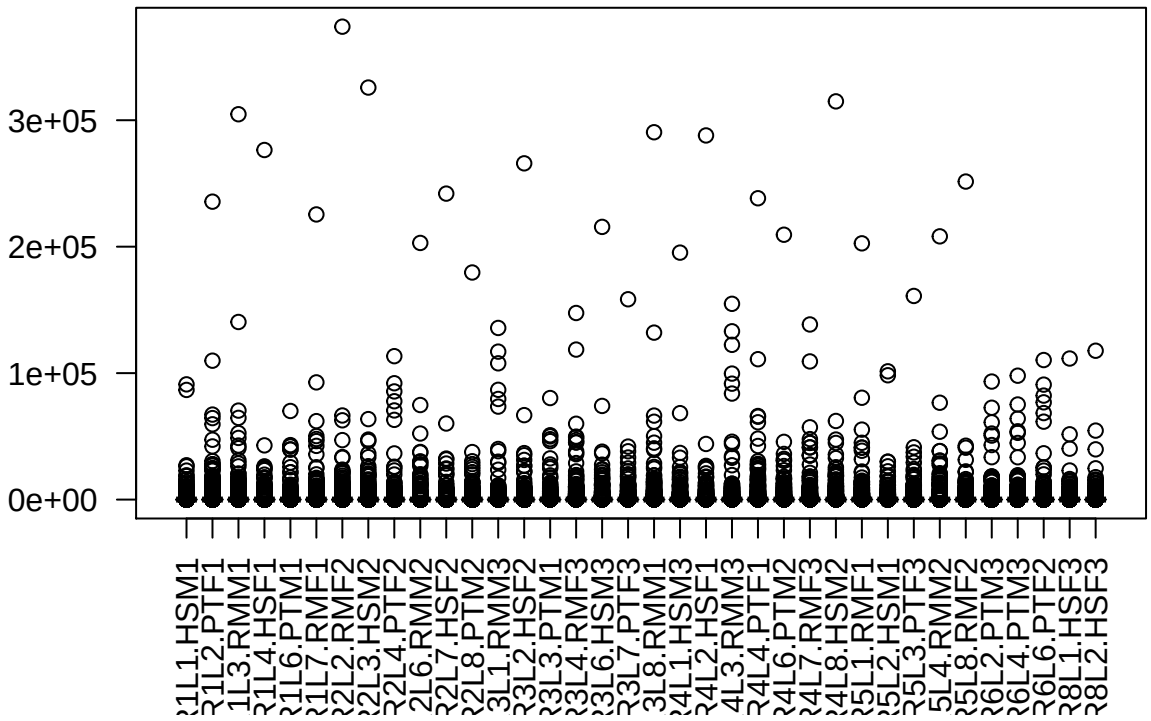

```
# a stick! is it about the number of breaks?
hist( mat, col=8, breaks = 100)
```




```
# not really, the data is just skewed
```

Now draw a boxplot for all the data, to see each column is distributed:

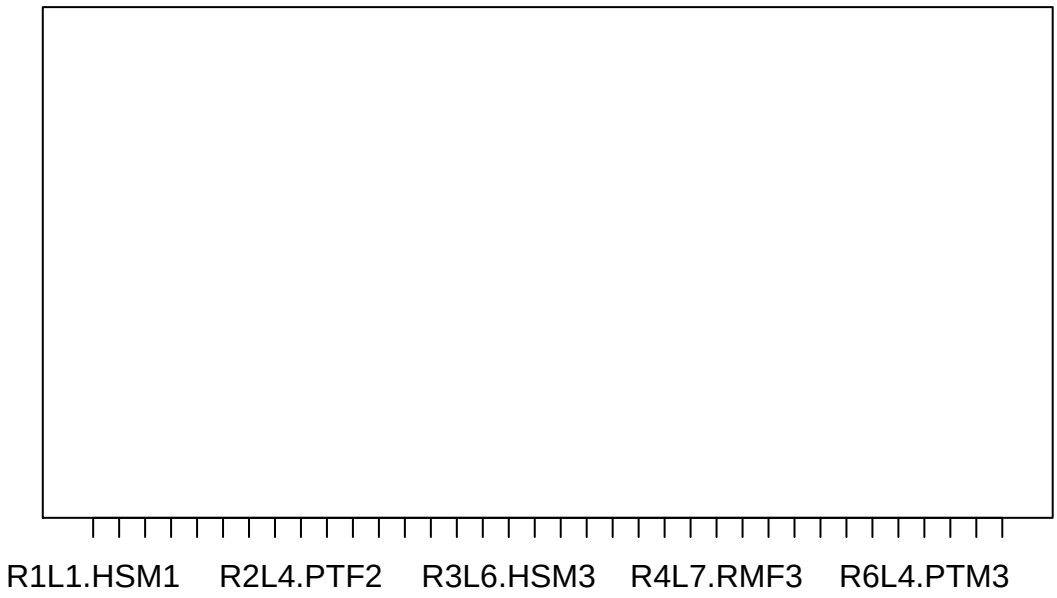
```
# automatically draws boxplots for the columns
# las=2 ensures vertical plotting of x-axis labels (default is horizontal)
boxplot( mat, col=8, las = 2)
```



Not very informative. First, we still need to know which sample belongs to which species, etc. We also probably need to transform - it looks very skewed. As a simple solution, can we plot the data in log?

```
# plot the y axis in log
boxplot( mat, log="y", col=8 )
```

```
## Warning in plot.window(xlim = xlim, ylim = ylim, log = log, yaxs = pars
## $yaxs): nonfinite axis limits [GScale(-inf,5.57283,2, .); log=1]
```



Doesn't work - why?

In this case `-Inf` values arise and boxplot cannot handle these (although other plotting functions, such as `hist`, automatically remove those).

Hierarchical clustering using `dist` and `hclust`

Indeed, the data look very skewed, overall. This is one thing we'll have to deal with.

To continue our overview, we can use data reduction techniques such as hierarchical clustering, k-means clustering, principle components analysis (PCA), multidimensional scaling (MDS), etc. These methods:

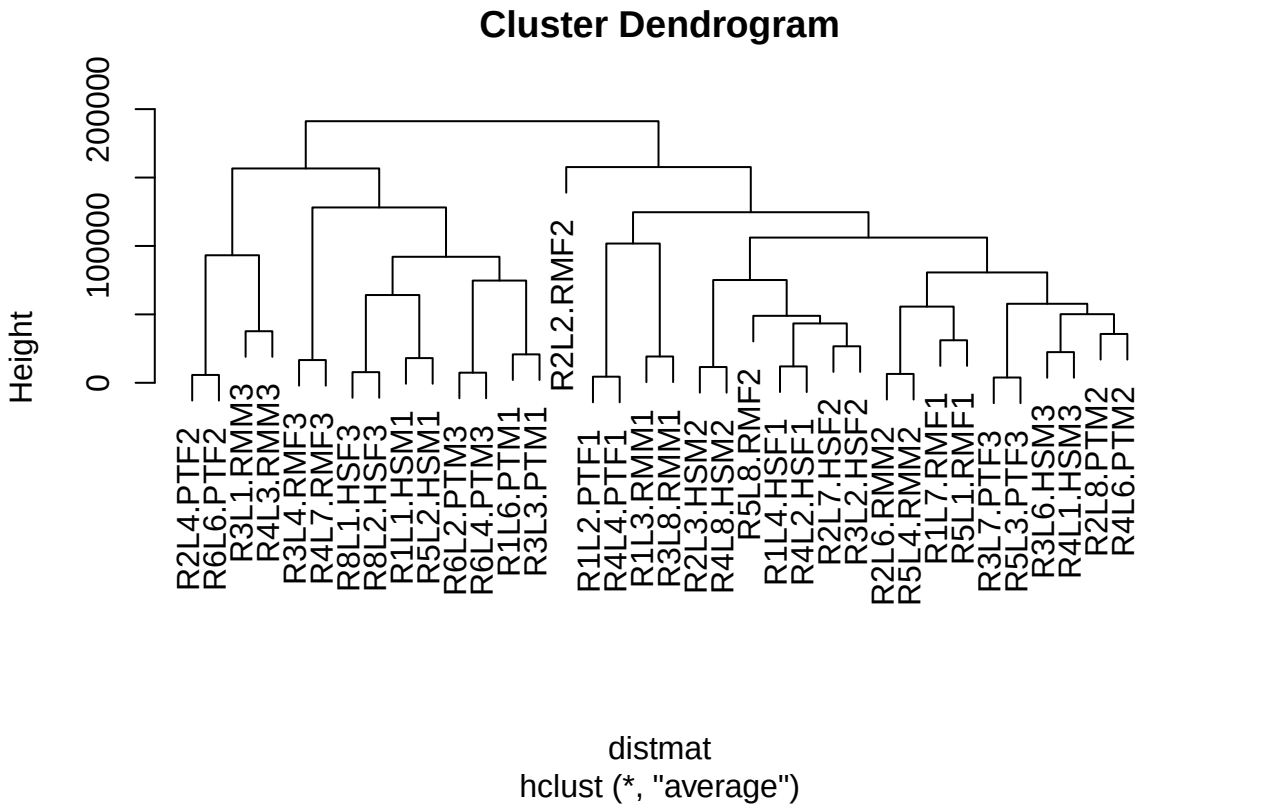
- provide a feeling of the **main sources of variance** in the data,
- can point out possible **outliers** and other unexpected effects.

We'll start by using **hierarchical clustering**. This is a simple algorithm, applied in the R function `hclust`. We can calculate distance among the samples (columns) in the expression matrix. In this case, using euclidean distances is common. In the clustering we use the "average", or UPGMA (https://en.wikipedia.org/wiki/UPGMA) algorithm:

```
distmat = dist( t( mat ), method = "euclidean")
dim( distmat )
```

```
## NULL
```

```
# this is the UPGMA method
plot( hclust( distmat, method = "average") )
```



Can we detect any pattern on the tree? Or was the experiment a waste?

Creating a `dist` matrix

We can also create a tree that represents DNA divergence among the 3 species (human-chimp 1%, hominid-macaque: 7%). For this we can build a distance matrix representing these numbers:

This could also have worked:

```
m = matrix(c(0,1,7,1,0,7,7,7,0), 3, 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    7
## [2,]    1    0    7
## [3,]    7    7    0
```

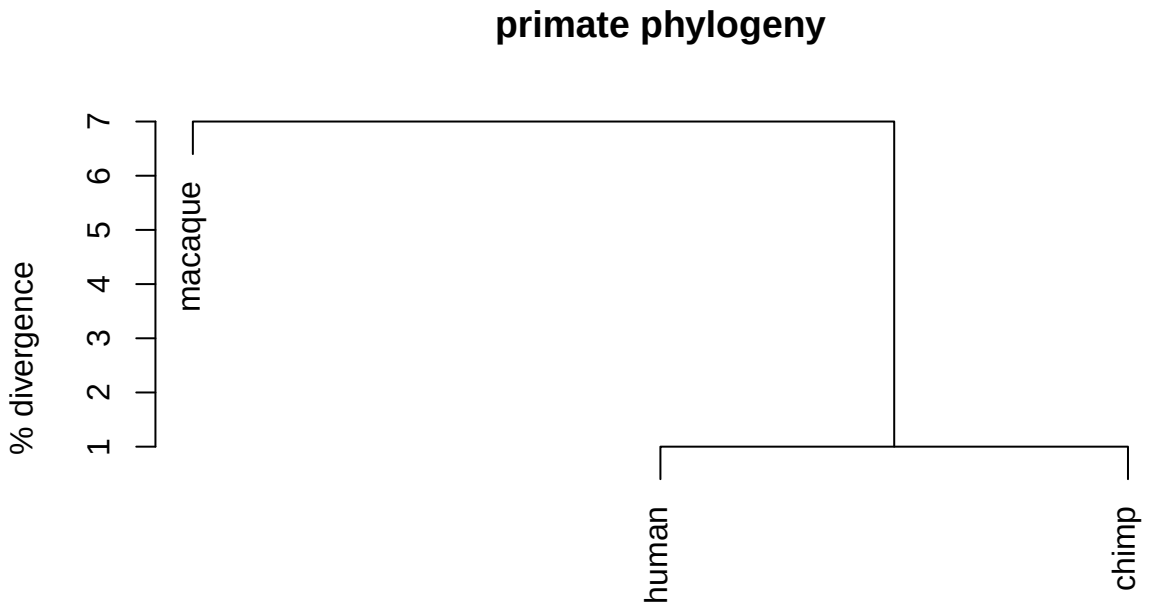
```
colnames(m) = c("human", "chimp", "macaque")
rownames(m) = c("human", "chimp", "macaque")
m
```

```
##         human chimp macaque
## human       0     1       7
## chimp       1     0       7
## macaque     7     7       0
```

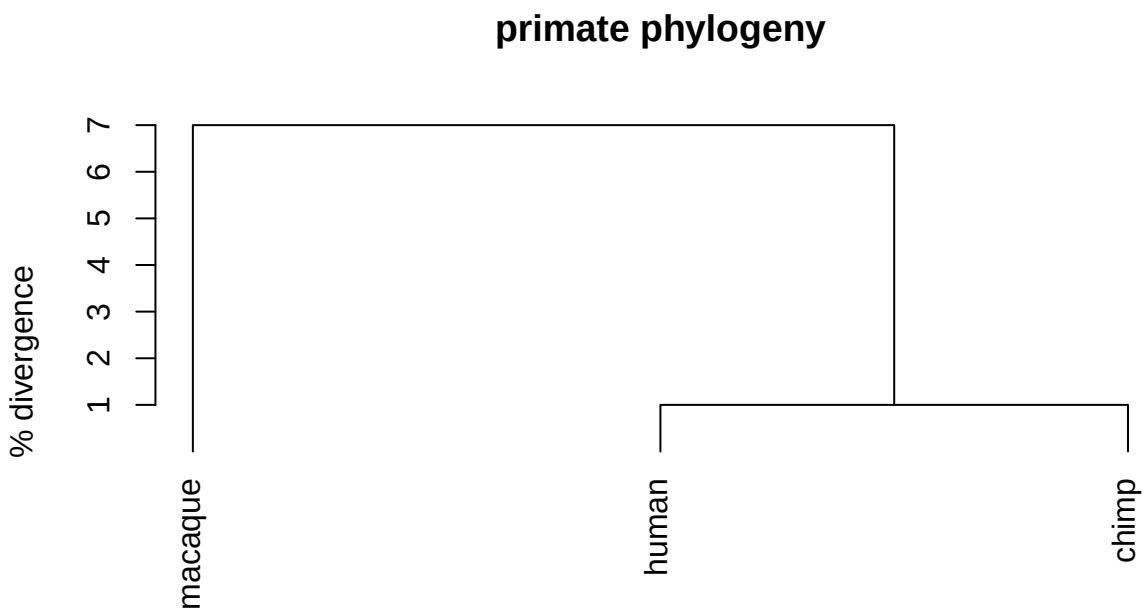
```
distx2 = as.dist(m)
distx2
```

```
##         human chimp
## chimp       1
## macaque     7     7
```

```
plot( hclust( distx2, method = "average"), xlab = "", sub = "",
      main = "primate phylogeny", ylab = "% divergence")
```



```
# another representation
plot( hclust( distx2, method = "average"), xlab = "", sub = "",
      main = "primate phylogeny", hang = -1, ylab = "% divergence")
```



Exercise: Creating a pseudogenome

You could also create a pseudogenome of 200 bases, and introduce mutations on each lineage to create 2 and 14 differences, respectively.

```
# create blank genomes
human = chimp = macaque = rep(0, 200)
# introduce mutations
human[1:6] = chimp[1:6] = 1
human[7] = chimp[8] = 1
macaque[9:15] = 1
mean(abs(human - chimp))
```

```
## [1] 0.01
```

```
mean(abs(human - macaque))
```

```
## [1] 0.07
```

```
pseudogen = rbind(human, chimp, macaque)
pseudogen[,1:20]
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## human       1    1    1    1    1    1    1    0    0     0     0     0
## chimp       1    1    1    1    1    1    0    1    0     0     0     0
## macaque     0    0    0    0    0    0    0    0    1     1     1     1
##          [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
```

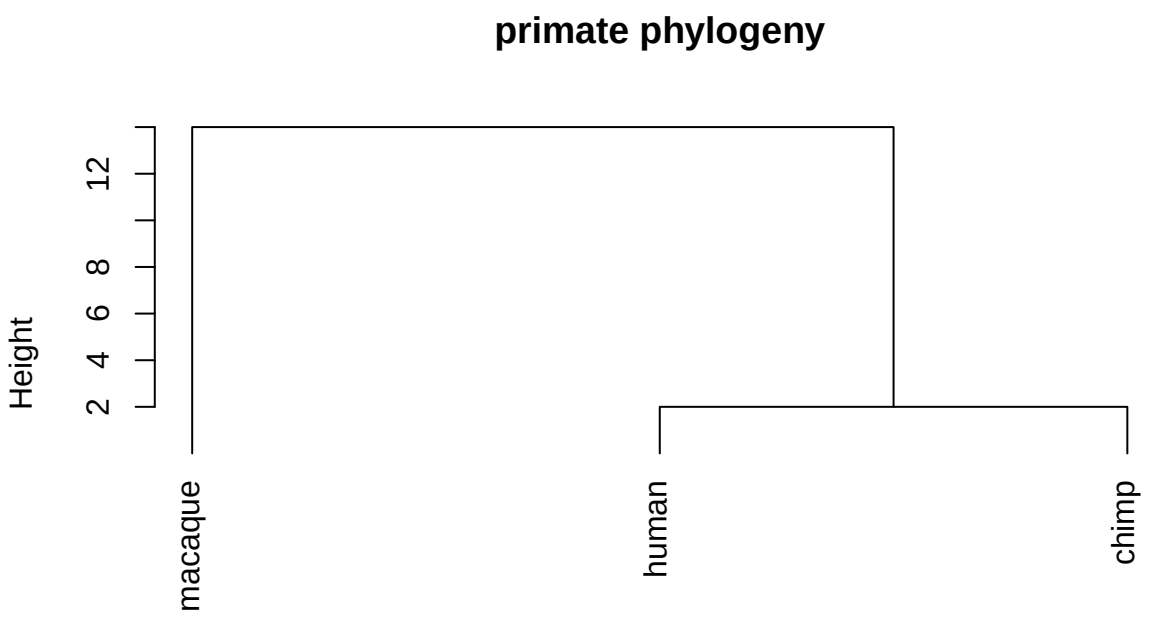
```
## human       0     0     0     0     0     0     0     0
## chimp       0     0     0     0     0     0     0     0
## macaque     1     1     1     0     0     0     0     0
```

Now calculate **manhattan distances**, which are the sum of absolute differences between elements of two vectors:

```
distx3 = dist(pseudogen, method = "manhattan")
distx3
```

```
##         human chimp
## chimp       2
## macaque    14    14
```

```
plot( hclust( distx3, method = "average" ), xlab = "", sub = "",
      main = "primate phylogeny", hang = -1)
```

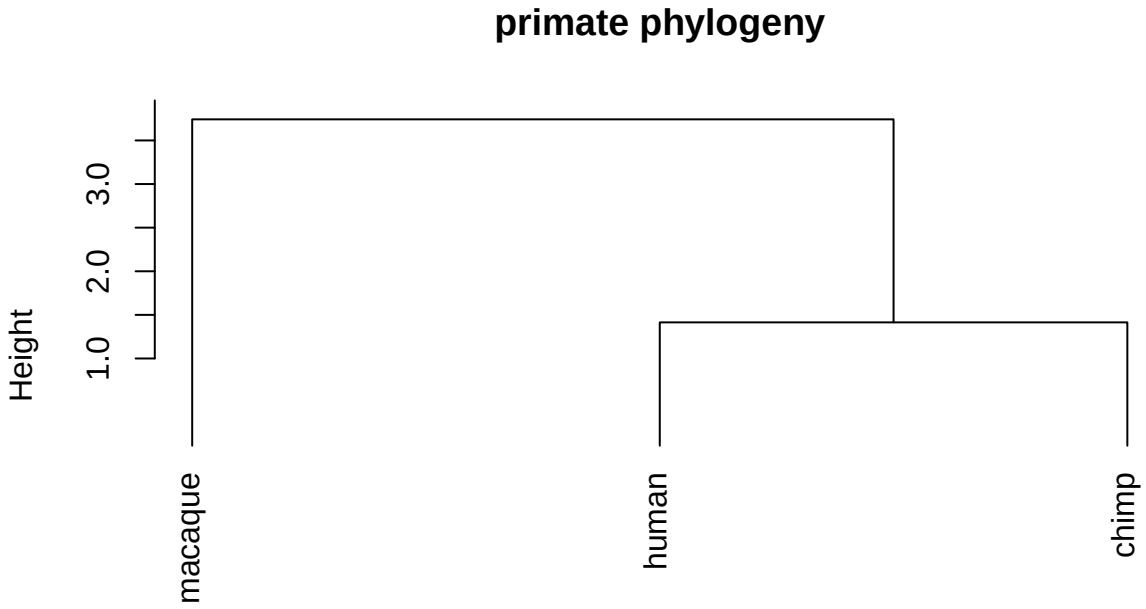


Note that using euclidean distances on the matrix would *not* have produced the same result:

```
distx4 = dist(pseudogen, method = "euclidean")
distx4
```

```
##            human    chimp
## chimp   1.414214
## macaque 3.741657 3.741657
```

```
plot( hclust( distx4, method = "average"), xlab = "", sub = "",
      main = "primate phylogeny", hang = -1)
```



Large differences are underweighted using euclidean compared to manhattan.

Categorical factors and the `factor` class

To further study the data, we need to know which column corresponds to which species, sex, sequencing run. All the information encoded in column names, e.g.:

`**R1L1.HSM1**`

This indicates, respectively:

- sequencing run no.,
- sequencing lane no.,
- species (HS: Homo sapiens, PT: Pan troglodytes, RM: rhesus macaque),
- sex,
- indv. ID.

How to extract that info? We'll now repeat the homework:

```
# substr runs loops along vectors
species = substr ( colnames(mat), 6, 7 )
# check that it worked
species
```

```
##  [1] "HS" "PT" "RM" "HS" "PT" "RM" "RM" "HS" "PT" "RM" "HS" "PT" "RM" "HS"
## [15] "PT" "RM" "HS" "PT" "RM" "HS" "HS" "RM" "PT" "PT" "RM" "HS" "RM" "HS"
## [29] "PT" "RM" "RM" "PT" "PT" "PT" "HS" "HS"
```

```
# frequencies
table( species )
```

```
## species
## HS PT RM
## 12 12 12
```

```
sex = substr(colnames(mat), 8, 8)
table( sex )
```

```
## sex
##  F  M
## 18 18
```

```
run = substr(colnames(mat), 2, 2)
table( run )
```

```
## run
## 1 2 3 4 5 6 8
## 6 6 7 7 5 3 2
```

```
indv = substr(colnames(mat), 6, 9)
table( indv )
```

```
## indv
## HSF1 HSF2 HSF3 HSM1 HSM2 HSM3 PTF1 PTF2 PTF3 PTM1 PTM2 PTM3 RMF1 RMF2 RMF3
##    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
## RMM1 RMM2 RMM3
##    2    2    2
```

```
length(unique( indv ) )
```

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

Now use `summary` and `cbind` to check the properties of the new R vectors:

```
summary( cbind( species, indv, run, sex ) )
```

```
##  species      indv    run   sex
##  HS:12   HSF1   : 2   1:6   F:18
##  PT:12   HSF2   : 2   2:6   M:18
##  RM:12   HSF3   : 2   3:7
##          HSM1   : 2   4:7
##          HSM2   : 2   5:5
##          HSM3   : 2   6:3
##          (Other):24   8:2
```

Balance in experimental design

We can also study the frequency of combinations, to check, for example, how *balanced* the experiment was with respect to biological and technical factors.

```
table( species, sex )
```

```
##         sex
## species F M
##      HS 6 6
##      PT 6 6
##      RM 6 6
```

```
table( species, run )
```

```
##         run
## species 1 2 3 4 5 6 8
##      HS 2 2 2 3 1 0 2
##      PT 2 2 2 2 1 3 0
##      RM 2 2 3 2 3 0 0
```

```
table( species, run, sex )
```

```
## , , sex = F
##
##         run
## species 1 2 3 4 5 6 8
##      HS 1 1 1 1 0 0 2
##      PT 1 1 1 1 1 1 0
##      RM 1 1 1 1 2 0 0
##
## , , sex = M
##
##         run
## species 1 2 3 4 5 6 8
##      HS 1 1 1 2 1 0 0
##      PT 1 1 1 1 0 2 0
##      RM 1 1 2 1 1 0 0
```

It is important that the number of runs are comparable across both species and sexes, as they could introduce significant bias.

Exercise: manipulating strings of variable length

What if the character strings were not of fixed length? How to extract "run" information from two strings of variable length: "R22L444.HSM2" and "R33333L88.HSF3"?

```
varstring = c("R22L444.HSM2", "R33333L88.HSF3")
# [] needed for strsplit() to treat L and R as different characters
strsplit( varstring, "[LR]" )
```

```
## [[1]]
## [1] ""          "22"        "444.HSM2"
##
## [[2]]
## [1] ""         "33333"    "88.HSF3"
```

```
# the result is embedded in a list
# we can retrieve the elements using sapply()
sapply( strsplit( varstring, "[LR]" ), function(x) {
  x[2]
})
```

```
## [1] "22"    "33333"
```

How about extracting the sex info from the same two strings using **substr**, assuming the part of the comma is fixed?

```
# we can use nchar
sapply(varstring, function(x) {
  substr(x, nchar(x)-1, nchar(x)-1)
})
```

```
##   R22L444.HSM2 R33333L88.HSF3
##            "M"            "F"
```

```
# this would return the character positions inside the string
gregexpr("[FM]", varstring)
```

```
## [[1]]
## [1] 11
## attr(,"match.length")
## [1] 1
## attr(,"useBytes")
## [1] TRUE
##
## [[2]]
## [1] 13
## attr(,"match.length")
## [1] 1
## attr(,"useBytes")
## [1] TRUE
```

Converting vectors to factors

Now convert the vectors to R class **factor**. Factors are useful to define categories & running ANOVAs, plotting boxplots, etc.

```
class( sex )
```

```
## [1] "character"
```

```
# define factors
sex = factor(sex)
sex
```

```
##  [1] M F M F M F F M F M F M M F M F M F M M F M F M F M F M F M M F F
## [36] F
## Levels: F M
```

```
class( sex )
```

```
## [1] "factor"
```

```
# note the difference
levels(sex) # similar to `unique`
```

```
## [1] "F" "M"
```

Factor type objects also retain information about the `levels` of a factor. E.g.:

```
in2sex = sex[2]
in2sex
```

```
## [1] F
## Levels: F M
```

```
class ( in2sex )
```

```
## [1] "factor"
```

```
length( in2sex )
```

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

```
levels( in2sex )
```

```
## [1] "F" "M"
```

Convert the other vectors into factor as well:

```
species = factor(species)
run = factor(run)
indv = factor(indv)
# check
species
```

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

```
run
```

```
##  [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 6 6 6 8
## [36] 8
## Levels: 1 2 3 4 5 6 8
```

```
sex
```

```
##  [1] M F M F M F F M F M F M M F M F M F M M F M F M F M F M F M F M M F F
## [36] F
## Levels: F M
```

Exercise: Redrawing trees

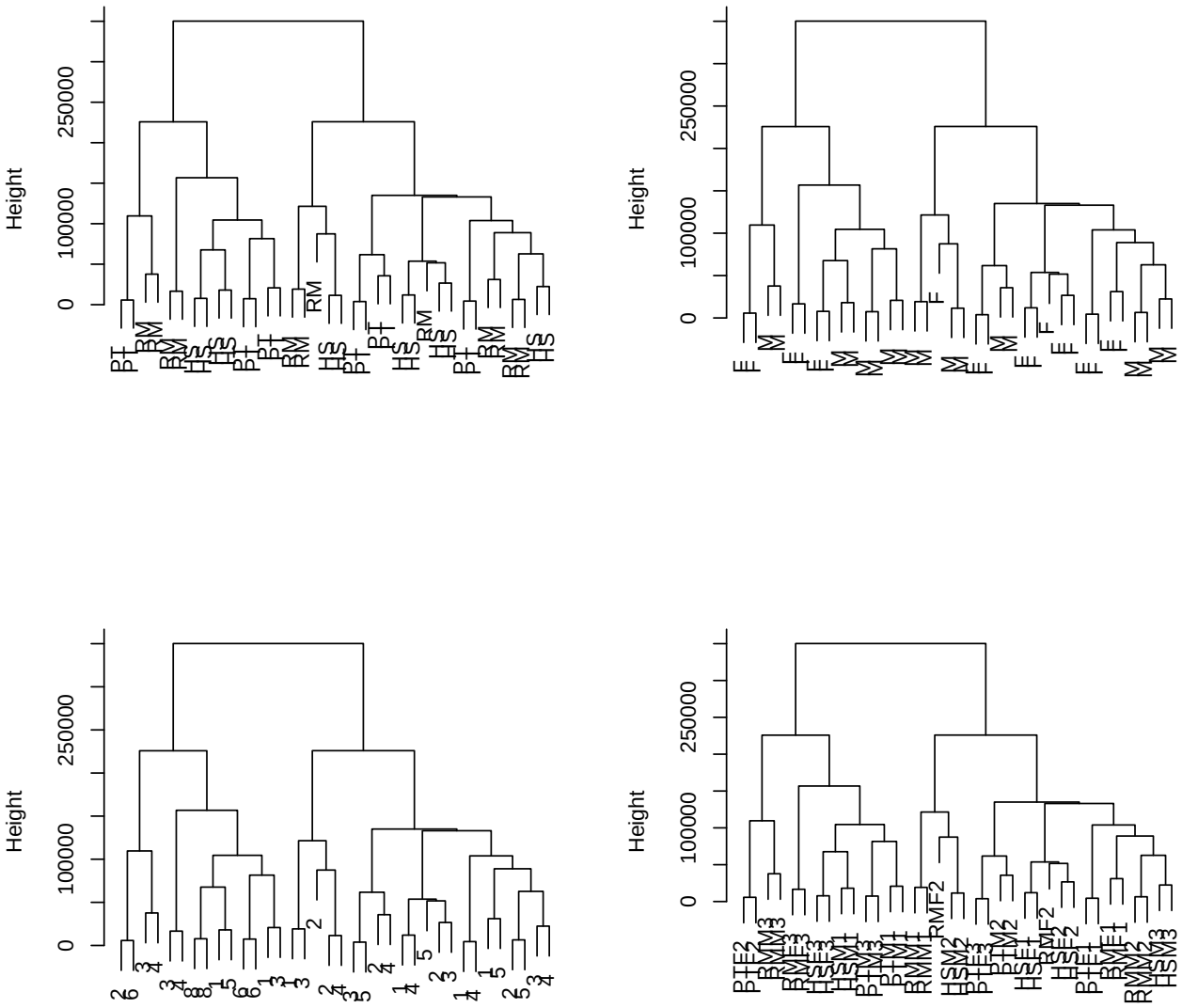
Now check if in the tree you draw, samples cluster according to species, individuals, sex, or run?

Here use a loop, and use the `labels` parameter of `plot`. But for that, you have to first create a list to run over:

```
myvariables = list( species, sex, run, indv )
names(myvariables) = c("species", "sex", "run", "indv")

# define a 2X2 window
par(mfrow=c(2,2))

# and run your loop to fill the window
for( varx in myvariables) {
  plot( hclust( distmat ), labels = varx,
        sub = "", xlab = "", main = "")
}
```



So again, no obvious clustering w.r.t. sex, individual, or run. But only w.r.t. individual, i.e. clustering of technical replicates.

ANOVA: Variation in sequencing depth

Before going further, we wish to study one trait of the dataset: Are all samples sequenced to the same depth?

We can calculate & compare total counts among samples.

In general we expect that difference in total counts represents **technical variance**, not biological. This is because experiments start from the same RNA amount. If there exists differences in the total, this is assumed to be technical, and we should make sure that it is removed, and does not confound biological effects.

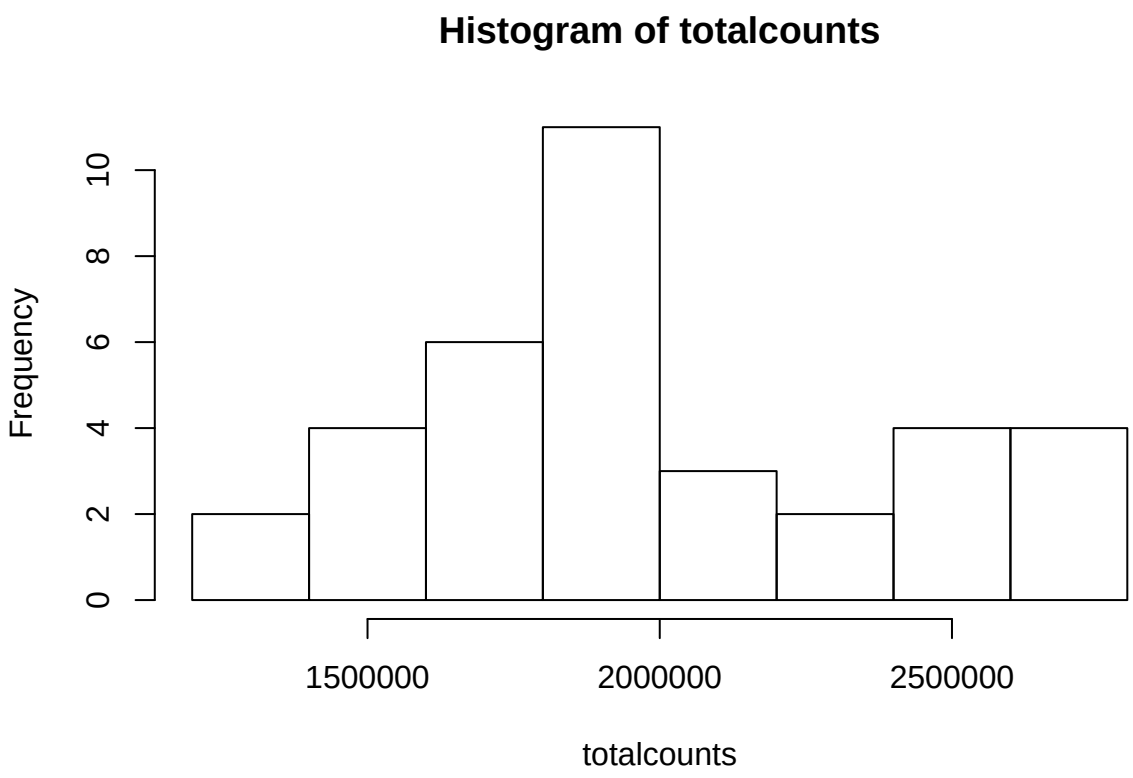
First create a vector of column sums to see how different they are:

```
totalcounts = apply( mat, 2, sum )
totalcounts
```

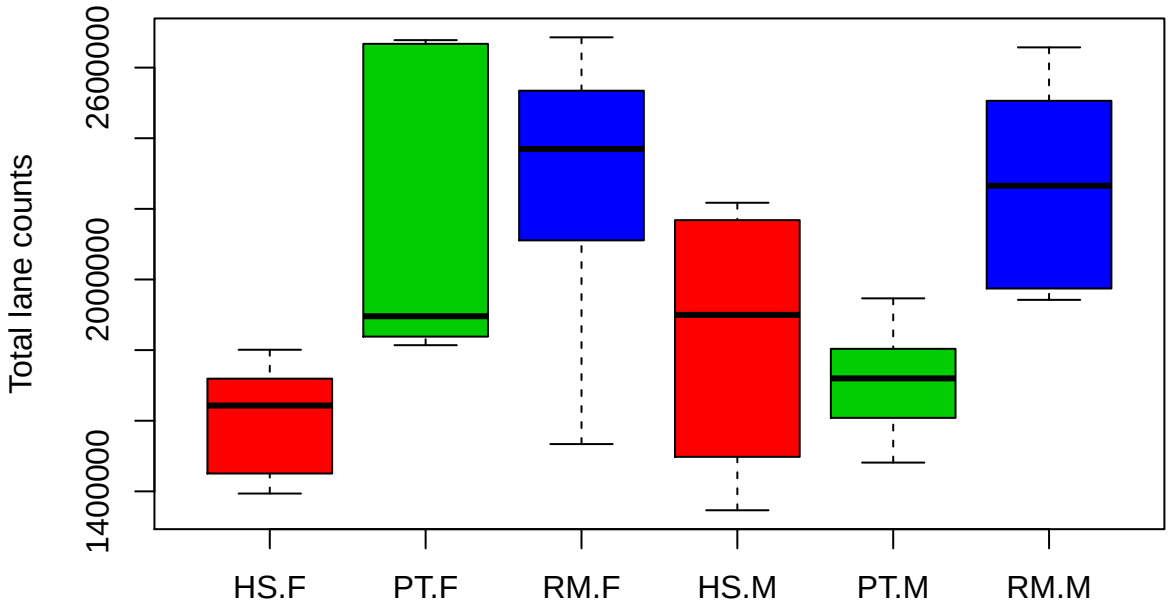
```
## R1L1.HSM1 R1L2.PTF1 R1L3.RMM1 R1L4.HSF1 R1L6.PTM1 R1L7.RMF1 R2L2.RMF2
##   1346515   2667264   2657274   1665987   1481536   2400660   2338433
```

```
## R2L3.HSM2 R2L4.PTF2 R2L6.RMM2 R2L7.HSF2 R2L8.PTM2 R3L1.RMM3 R3L2.HSF2
##   2217235   1910402   1942296   1620189   1608138   2119496   1801009
## R3L3.PTM1 R3L4.RMF3 R3L6.HSM3 R3L7.PTF3 R3L8.RMM1 R4L1.HSM3 R4L2.HSF1
##   1694688   2685655   1974228   1838275   2505941   1825373   1719125
## R4L3.RMM3 R4L4.PTF1 R4L6.PTM2 R4L7.RMF3 R4L8.HSM2 R5L1.RMF1 R5L2.HSM1
##   2411707   2677771   1946512   2534595   2167994   2110806   1497738
## R5L3.PTF3 R5L4.RMM2 R5L8.RMF2 R6L2.PTM3 R6L4.PTM3 R6L6.PTF2 R8L1.HSF3
##   1813918   1974502   1533906   1745188   1803555   1881431   1393867
## R8L2.HSF3
##   1450604
```

```
# check the overall distribution first
hist( totalcounts )
```




```
# plot against species
boxplot( totalcounts ~ species + sex,
         ylab='Total lane counts',
         col=2:4)
```



What do the boxes represent? And the two points?

Are the species differences significant? How to test?

```
# we could use ANOVA
summary( aov( totalcounts ~ species ) )
```

```
##             Df    Sum Sq   Mean Sq F value  Pr(>F)
## species      2 1.823e+12 9.113e+11   8.078 0.00139 **
## Residuals   33 3.723e+12 1.128e+11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD( aov( totalcounts ~ species ) )
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = totalcounts ~ species)
##
## $species
##           diff         lwr      upr     p adj
## PT-HS 199067.8 -137394.272 535529.9 0.3268430
## RM-HS 544617.3  208155.144 881079.4 0.0010361
## RM-PT 345549.4    9087.311 682011.5 0.0431198
```

```
# but ANOVA assumes equal variance and normality, which might be violated
```