Introduction R and exploratory data analysis

Lab work

In this first practical, you will explore R and RMarkDown and use it to perform an initial analysis of a microarray dataset. You will work in RStudio. Begin by starting it up and having a look at the various subwindows.

When you start working with new languages and environments it can help to keep a “cheat sheet” handy. For R, ggplot2, RStudio and RMarkDown, good cheat sheets can be found here.

R

Maths and variables

Use the console to answer the following questions.

a. Calculate the outcome of this equation: the modulus of 13 and the difference between 25 and the product of 3 and 6.

```
> 13%%(25-3*6)

## [1] 6
```

b. Calculate 2^10 + 2^6.

```
> 2^10+2^6

## [1] 1088
```

c. Create two variables x and y and assign them the values π and e. Calculate x to the power of y, rounded to the first decimal.

```
> x <- pi; y <- exp(1); round(x^y,1)

## [1] 22.5
```

Vectors

d. Create a vector v containing the first five prime numbers. What is range(v)?

```
> v <- c(2,3,5,7,11)
> range(v)

## [1]  2 11
```

e. Set the sixth element of v to 0/0. What is sum(v), and why?

```
> v[6] <- 0/0
> v[6]

## [1] NaN
```

```
> sum(v)

## [1] NaN
```

v[6] has become NaN, and as long as there is at least one NaN, summaries such as sum, mean, max etc. will be NaN.

f. How do you select only the valid numbers in v?

```
> v[!is.na(v)]

## [1]  2  3  5  7 11
```

There are likely other possibilities.

Matrices and packages

g. Install the "magic" library and load it, so it can be used.

```
> install.packages("magic");

> library(magic);

## Loading required package: abind
```

h. Generate a 3x3 magic square M (hint: ?magic).

```
> M <- magic(3)
> M

##      [,1] [,2] [,3]
## [1,]    2    7    6
## [2,]    9    5    1
## [3,]    4    3    8
```

magic(n) generates an n-by-n magic square, i.e. a matrix for which each column and each row add up to the same number.

i. What is the maximum value of the square of M?

```
> max(M%*%M)

## [1] 91
```

Loops, functions and scripts

j.Generate a vector f containing the first 10 Fibonacci numbers, using a for loop (hint: see Wikipedia).

```
> f = c(1,1)
> for (i in 3:10) {
+   f[i] = f[i-1] + f[i-2]
+ }
```

k. Write a function is.odd(v) that returns only the odd numbers in a vector v.

```
> is.odd <- function(v) {
+   return (v[v%%2==1])
+ }
```

l. Combine the answers to the two questions above into a script that outputs what percentage of the first 50 Fibonacci numbers is odd.

```
> is.odd <- function(v) {
+   return (v[v%%2==1])
+ }
>
> f = c(1,1)
> for (i in 3:50) {
+   f[i] = f[i-1] + f[i-2]
+ }
>
> length(is.odd(f))/length(f)

## [1] 0.68
```

RMarkDown

m. Write an RMarkDown report that outputs a magic square of size 5, without echoing the R code. Give the report a descriptive header. *Simply use the magic library you encountered earlier. Use r{echo=FALSE} to suppress echoing the code in RMarkDown.*

n. Now add the code for your function is.odd(v), without running it. *Create an R block with the option eval=FALSE.*

o. Finally, add scatterplots of the built-in iris dataset, using plot(iris), and generate a Word file.

Optional

p. Generate a vector v containing 4 4's, 8 8's and 12 12's, in that order, using rep().

```
> v <- rep(c(4,8,12),c(4,8,12))
```

q. Calculate the sum of $(2^i/i^2 + 3^i/i^3)$ for i = 10...20, using a for loop.

```
> s <- 0;
> for (i in 10:20) {
+   s <- s + (2^i/i^2 + 3^i/i^3)
+ }
> s

## [1] 721669.3
```

r. Calculate the sum of $(2^i/i^2 + 3^i/i^3)$ for i = 10...20, using a single command sequence.

```
> s <- sum(2^(10:20)/(10:20)^2 + 3^(10:20)/(10:20)^3)
> s

## [1] 721669.3
```

s. Create a vector l containing the labels "Sample 1" to "Sample 72", using paste().

```
> l <- paste("Sample", 1:72)
```

t. Use runif() and a rounding function to generate a vector r of 10,000 integers uniformly drawn in the range 1...3. Verify your answer using table().

```
> r <- floor(runif(10000)*3)+1
> table(r)

## r
##    1    2    3
## 3347 3281 3372
```

u. Create a 5x5 magic square matrix M as above and calculate the row sums and the column sums using the apply() function.

```
> M <- magic(5)
> apply(M,1,sum)

## [1] 65 65 65 65 65

> apply(M,2,sum)

## [1] 65 65 65 65 65
```

v. Solve $Mx = b$ for x, where M is a 3x3 magic square matrix and b = c(1,2,3), using solve(). Verify your outcome.

```
> M <- magic(3)
> x <- solve(M,c(1,2,3))
> x

## [1]  0.3 -0.2  0.3

> M%*%x

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

x <- solve(M,c(1,2,3)) gives x=c(0.3,-0.2,0.3); M%∗%x is indeed c(1,2,3).

Summary statistics

In this second part, you will work with a small, real-world microarray dataset. It contains 4 Affymetrix HG-U133A microarray samples, 2 each taken in 2 human T-cell development stages: single positive CD4+CD-8 and single positive CD4-CD8+ (for an overview of T-cell development, see Fig. 1 in http://www.nature.com/nri/journal/v2/n5/full/nri798.html). CDx+ means a cell expresses a T-cell receptor CDx, CDx- means it does not.

Dataset characteristics

a. Download and unzip the data for block 1 from Blackboard. The microarray data is stored in "SP48.txt". Inspect it using a text editor (Notepad or Word) and check what separator is used and whether it contains a header. What is the correct call to read.table() in R? Use it to read in the data.

```
> data <- read.table("~/SP48.txt", header=TRUE, sep="\t")
```

b. Use View(), class(), dim(), ncol(), nrow() and names() to inspect the data. What is the number of rows of the dataset?

```
> class(data)

## [1] "data.frame"

> dim(data)

## [1] 22283     7

> ncol(data)

## [1] 7

> nrow(data)

## [1] 22283

> names(data)

## [1] "Probeset"  "Gene"      "Type"      "SP4plus.1" "SP4plus.2"
"SP8plus.1"
## [7] "SP8plus.2"
```

The dataset contains 22,283 rows.

c. What do the rows correspond to? Hint: have a look at https://www.thermofisher.com/search/results?query=E14184&focusarea=Search%20All.

Probesets are (kind of) average intensities of a number of short probes interrogating a single sequence.

d. What is the number of columns?

7: probeset ID, gene name and type, and 4 expression values.

e. You can check your answers using str(), which summarizes the aspects of a dataset. Now inspect the first and last few rows of the data using head() and tail(), and get an overview using summary(). What do you think the range of the expression data is?

```
> str(data)

## 'data.frame':    22283 obs. of  7 variables:
##  $ Probeset : Factor w/ 22283 levels "1007_s_at","1053_at",..:
22217 22218 22216 22220 22219 22222 22221 22224 22223 22226 ...
##  $ Gene     : Factor w/ 13516 levels "---","A1CF","A2M",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ Type     : Factor w/ 2 levels "immunity","other": 2 2 2 2 2 2 2
2 2 2 ...
##  $ SP4plus.1: num  134 136 142 170 137 ...
##  $ SP4plus.2: num  106 114 113 144 120 ...
##  $ SP8plus.1: num  144 150 148 185 149 ...
##  $ SP8plus.2: num  80.2 84.8 88.3 104.4 83.4 ...
```

```
> head(data)
```

```
##           Probeset Gene  Type SP4plus.1 SP4plus.2 SP8plus.1
SP8plus.2
## 1  AFFX-BioB-5_at  --- other    133.87   105.690    143.57
80.161
## 2  AFFX-BioB-M_at  --- other    135.77   114.150    150.33
84.818
## 3  AFFX-BioB-3_at  --- other    141.79   113.050    147.75
88.270
## 4  AFFX-BioC-5_at  --- other    169.62   144.120    185.37
104.400
## 5  AFFX-BioC-3_at  --- other    136.83   119.890    149.31
83.403
## 6 AFFX-BioDn-5_at  --- other    122.14    98.845    134.10
75.307
```

```
> tail(data)
```

```
##           Probeset     Gene  Type SP4plus.1 SP4plus.2 SP8plus.1
SP8plus.2
## 22278   222379_at    KCNE4 other    153.85    159.91    186.34
106.720
## 22279 222380_s_at    PDCD6 other    233.61    196.56    323.45
138.990
## 22280   222381_at      --- other    161.01    141.61    190.81
106.490
## 22281 222382_x_at   NUP205 other    182.14    165.71    212.72
110.810
## 22282 222383_s_at   ALOXE3 other    146.51    113.30    155.69
80.713
## 22283   222384_at SMG7-AS1 other    133.00    106.02    136.09
86.931
```

```
> summary(data)
```

```
##        Probeset                                             Gene
##  1007_s_at:    1   ---                                        : 1058
##  1053_at  :    1   HFE                                        :   13
##  117_at   :    1   TCF3                                       :   12
##  121_at   :    1   CFLAR                                      :   11
##  1255_g_at:    1   LOC100506403 /// LOC101928269 /// RUNX1:   11
##  1294_at  :    1   FGFR2                                      :   10
##  (Other)  :22277   (Other)                                    :21168
##         Type          SP4plus.1          SP4plus.2          SP8plus.1
##  immunity: 2465   Min.   :  97.99   Min.   :  76.52   Min.   :
97.84
##  other   :19818   1st Qu.: 165.47   1st Qu.: 138.69   1st Qu.:
187.84
##                   Median : 212.99   Median : 184.91   Median :
```

```
256.95
##                      Mean   :  375.48   Mean   :  365.11   Mean   :
522.12
##                      3rd Qu.:  341.91   3rd Qu.:  330.18   3rd Qu.:
476.12
##                      Max.   :15745.00   Max.   :14667.00   Max.
:21249.00
##
##    SP8plus.2
##   Min.   :   63.17
##   1st Qu.:  100.78
##   Median :  129.53
##   Mean   :  248.14
##   3rd Qu.:  217.21
##   Max.   :10733.00
##
```

All values lie between approximately 60 and 21,500. The actual possible range is 0 to 65,535, i.e. 16 bits.

f. Estimate how many unique genes the dataset contains?

According to str(data), Gene is a factor with 13,516 levels; minus 1 for the '---' gives 13,515 genes.

g. Which gene is represented by most probesets?

According to summary(data), the HFE gene occurs 13 times (the '---' occurs 1058 times, but that is not likely to be a gene identifier).

h. How many NAs are there?

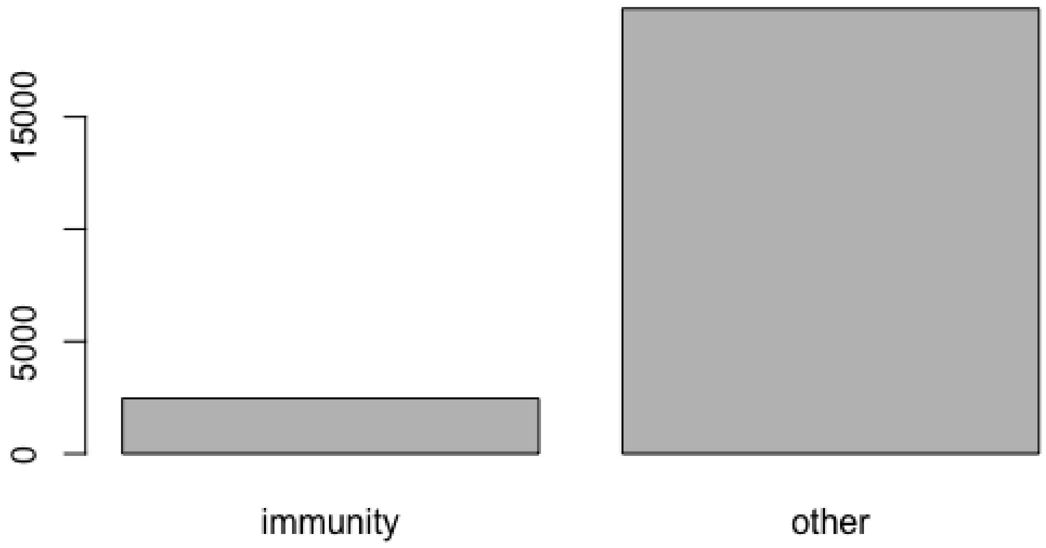
```
> length(which(is.na(data[,4:7])))

## [1] 0
```

There are no missing values: Affymetrix microarrays always yield a value for a probe, even though it may be very unreliable.

Visualization

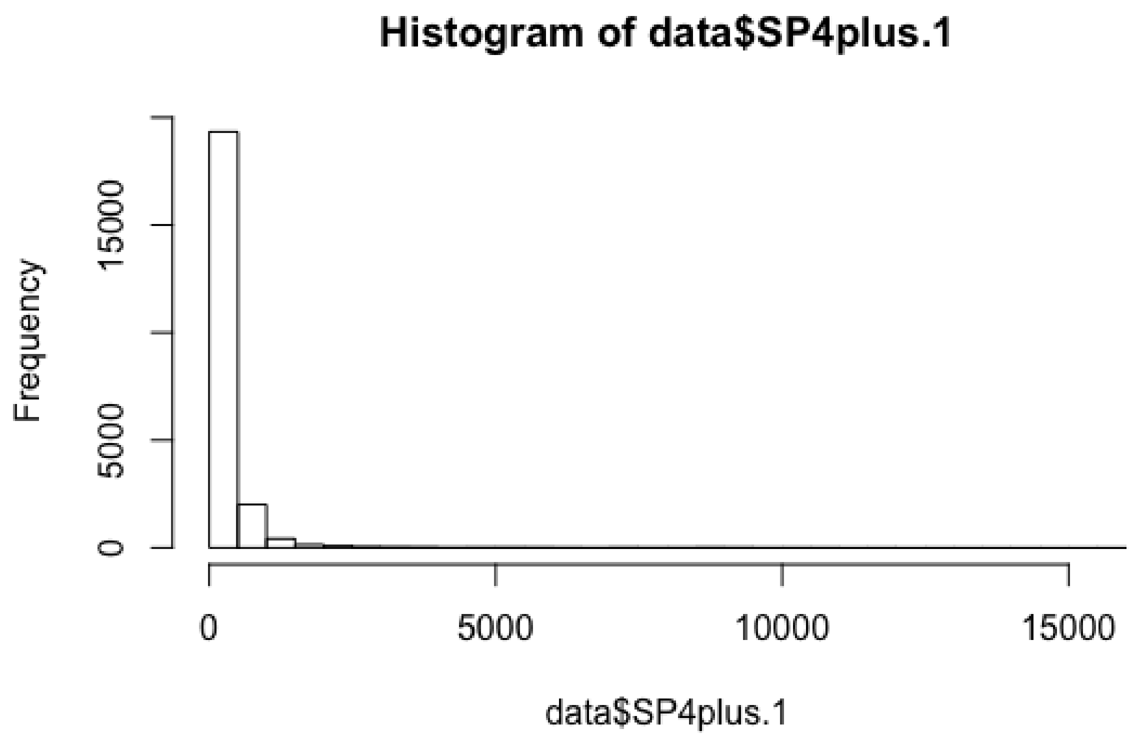
i. Use table() to learn how many immunity-related genes there are in the data and create a bar plot.

```
> table(data$Type)

##
## immunity    other
##     2465    19818

> barplot(table(data$Type))
```



j. Create histograms of the four microarray samples. What do they tell you about the intensity distribution?

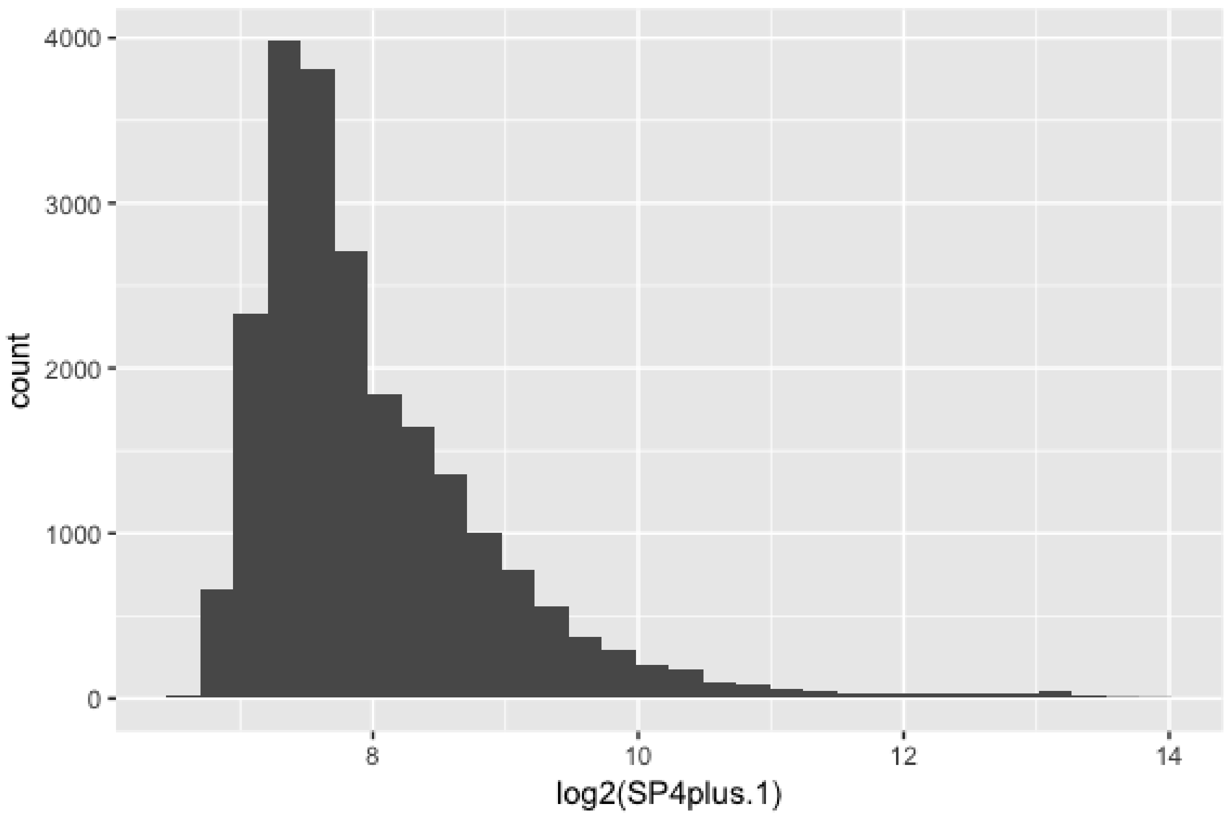
```
> hist(data$SP4plus.1,50)
```



... and so on. We use hist(...,50) to see more detail. But we can say little about the distribution, as it is very long-tailed: mostly low intensities, with a few outliers.

k. Now create histograms of the log2() of the microarray data, using ggplot2. What do these tell you?

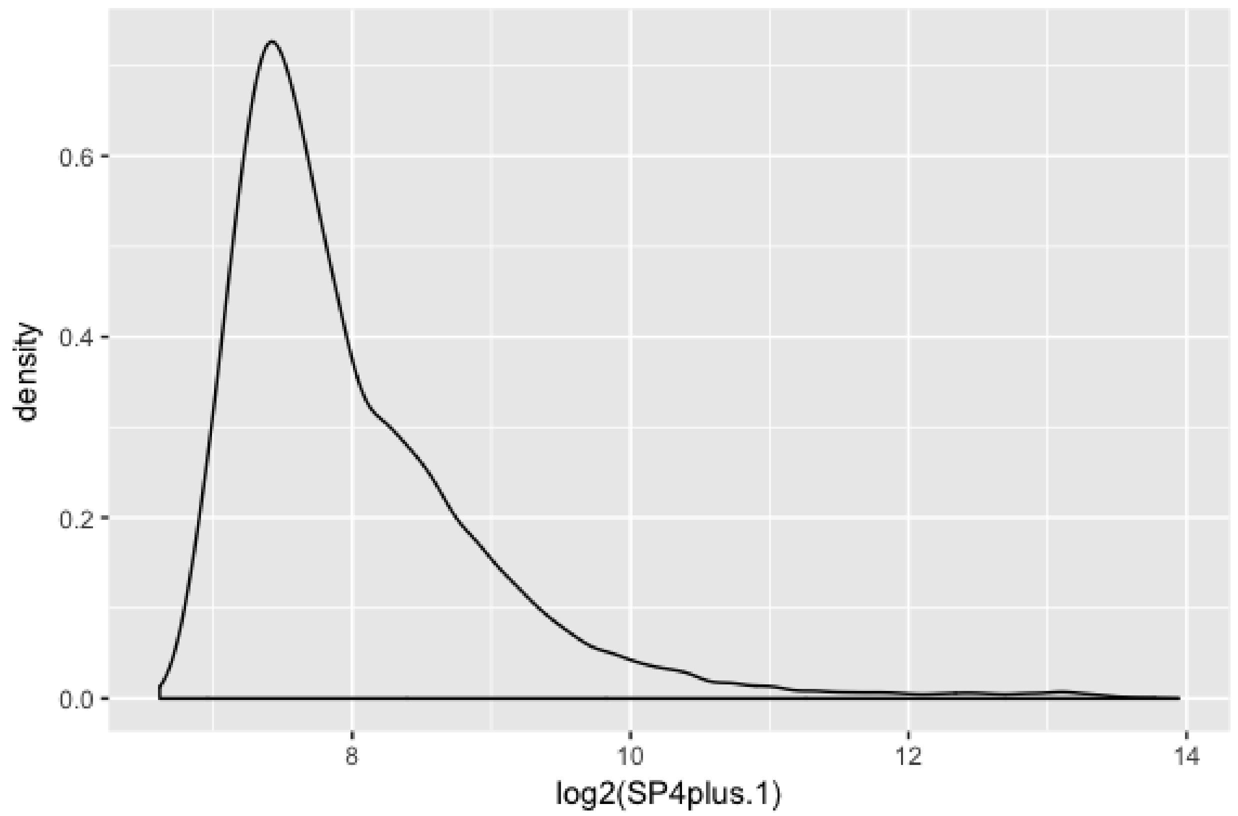
```
> library(ggplot2)
> qplot(log2(SP4plus.1), data=data, geom="histogram")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Of course you can also use the histogram() function like above. The distributions are still long-tailed to the right, but easier to inspect. They look mostly the same, but are shifted somewhat.

I. Are densities more informative?

```
> qplot(log2(SP4plus.1), data=data, geom="density")
```

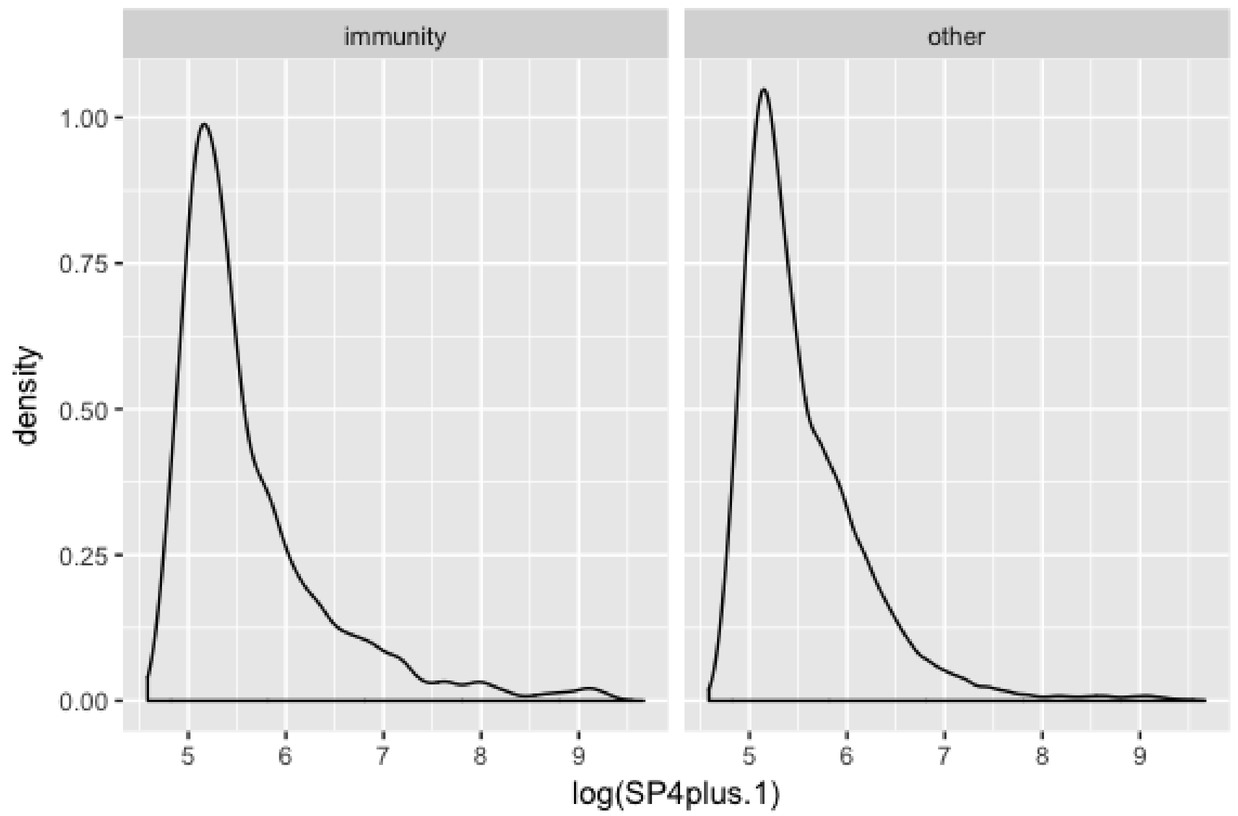


They look more smooth, but this is an approximation and does not necessarily give more information!

Data inspection

m. Plot densities for both immune genes and others, using ggplot(data=data, aes(x=log(SP4plus.1))) + geom_density() + facet_wrap(~Type). Repeat for the other arrays. Do you see a difference between the distributions of these two types of genes?

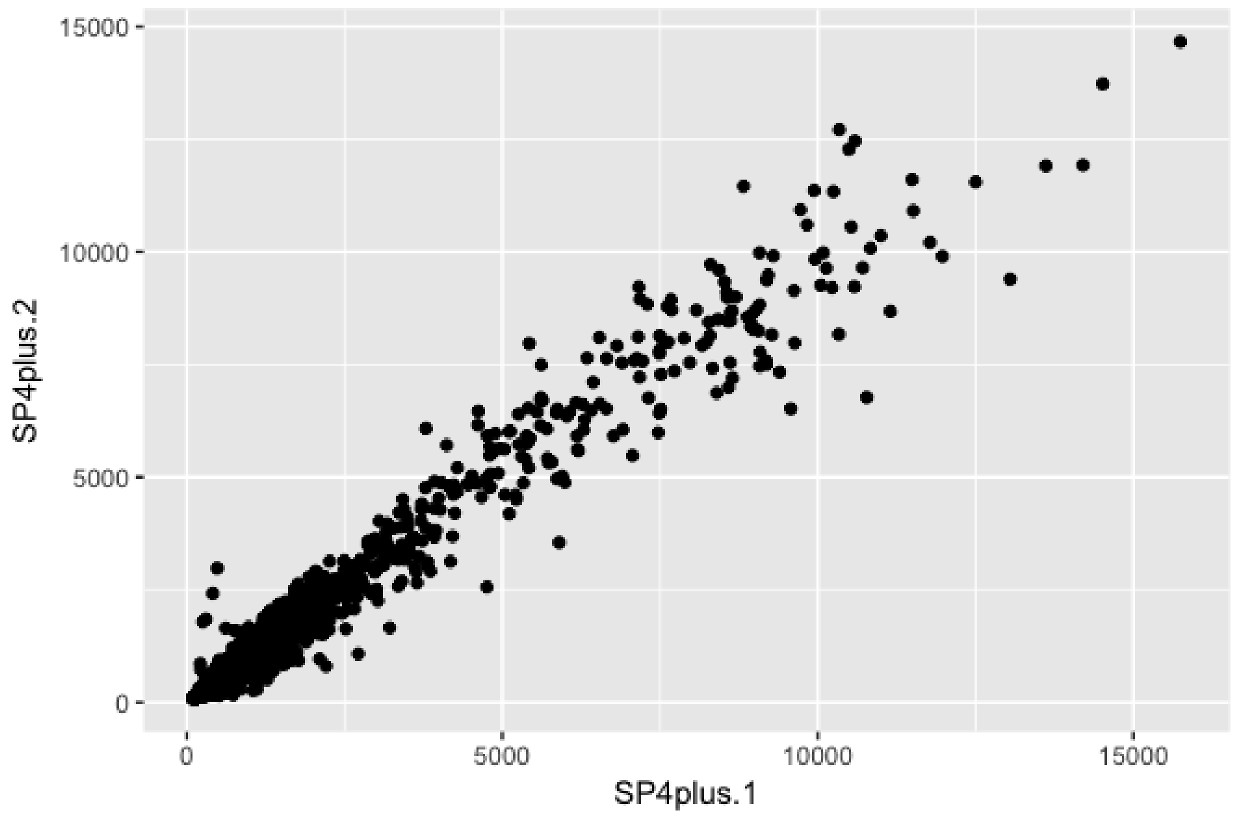
```
> ggplot(data=data, aes(x=log(SP4plus.1))) + geom_density() +
facet_wrap(~Type)
```



The main distributions are similar, but there does seem to be a 'bump' in the right tail for the immunity genes, i.e. a small number of genes are overexpressed.

n. Use ggplot() to make a scatterplot of the expression values measured in the two CD4+CD8- samples. Do the same for the CD4-CD8+ samples. What do you notice?

```
> ggplot(data, aes(x=SP4plus.1,y=SP4plus.2)) + geom_point()
```



Most expression values are correlated, i.e. lie on the line y=x, with some exceptions. However, the range of expression values differs, especially for the CD4-CD8+ samples.

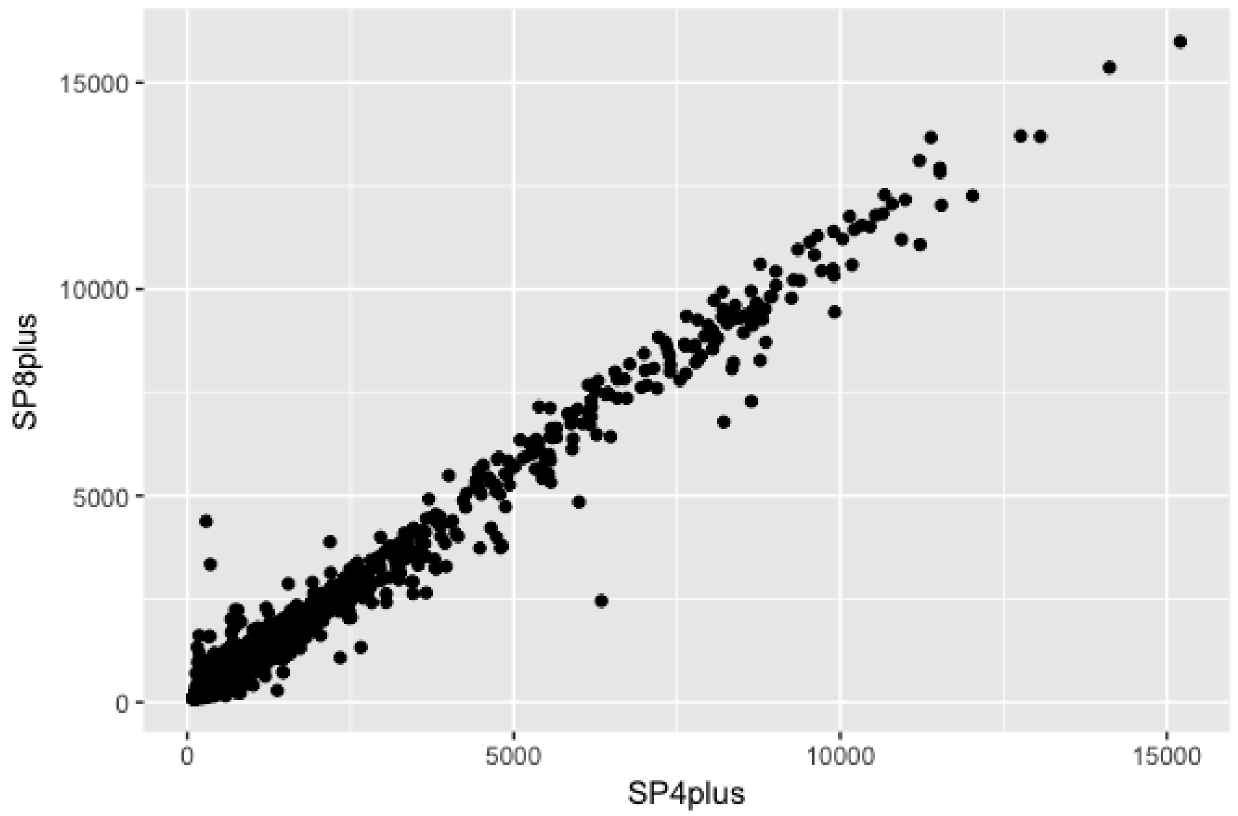
In order to visualize some aspects of the data, it is sometimes necessary to reshape your data or calculate intermediate results yourself. There is no standard recipe, but in general you can use the functions as.matrix(), as.numeric() and data.frame() to convert selected rows and colums to a matrix, matrix of numbers or a data frame, respectively.

To plot the average expression of genes in the CD4+CD8- samples against those in the CD4-CD8+ samples, you can do the following:

```
> # Calculate the mean CD4+CD8- expression
> mn4 <- rowMeans(data[,4:5])
> # Calculate the mean CD4-CD8+ expression
> mn8 <- rowMeans(data[,6:7])
> # Concatenate into data frame
> mndata <- data.frame(SP4plus=mn4,SP8plus=mn8)
```

o. Use the code above and ggplot() to make a scatterplot of the average expression values in the CD4+CD8- samples vs. those in the CD4-CD8+ samples. What do you see?

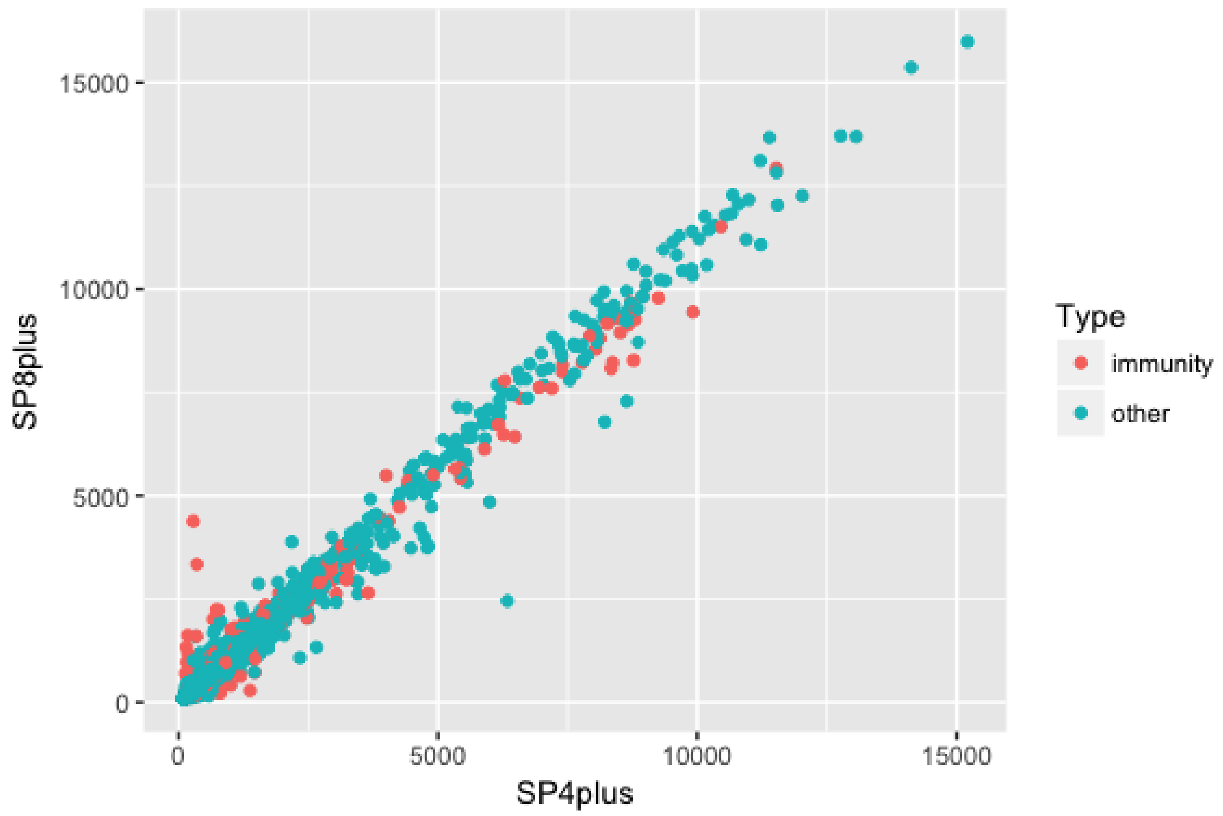
```
> ggplot(mndata,aes(x=SP4plus,y=SP8plus))+geom_point()
```



Most expression values have not changed much, i.e. they lie on the line y=x. There are however some exceptions, which could point to differentially expressed genes.

p. Modify the R code above so that mndata also contains the Type column found in the original data, and use this to colour the points according to the gene type. Can you spot a difference between the two sets of genes? *Optional: retrieve the names of the two immune-related genes that stand out.*

```
> mndata <- data.frame(SP4plus=mn4,SP8plus=mn8,Type=data$Type)
> ggplot(mndata,aes(x=SP4plus,y=SP8plus,colour=Type))+geom_point()
```



To find the names of the genes, you could select the genes by hand, for example:

```
> data$Gene[intersect(which(mndata$SP4plus<1000),
+                     which(mndata$SP8plus>3000))]

## [1] CD8A                   CD8B /// LOC100996919
## 13516 Levels: --- A1CF A2M A4GALT A4GNT AAAS AACS AADAC AAGAB AAK1
... ZZZ3

> data$Gene[intersect(which(mndata$SP4plus>5000),
+                     which(mndata$SP8plus<3000))]

## [1] ITM2A
## 13516 Levels: --- A1CF A2M A4GALT A4GNT AAAS AACS AADAC AAGAB AAK1
... ZZZ3
```

find the two points that clearly stand out above the bulk of the data, and the point that stands out below it. As to be expected, CD8A and CD8B are overexpressed in SP8Pplus; whereas ITM2A, a membrane protein, is overexprssed in SP4plus.

q. Find the two genes coding for the receptors, CD4 and CD8A and plot their expression. Does it concur with your ideas?

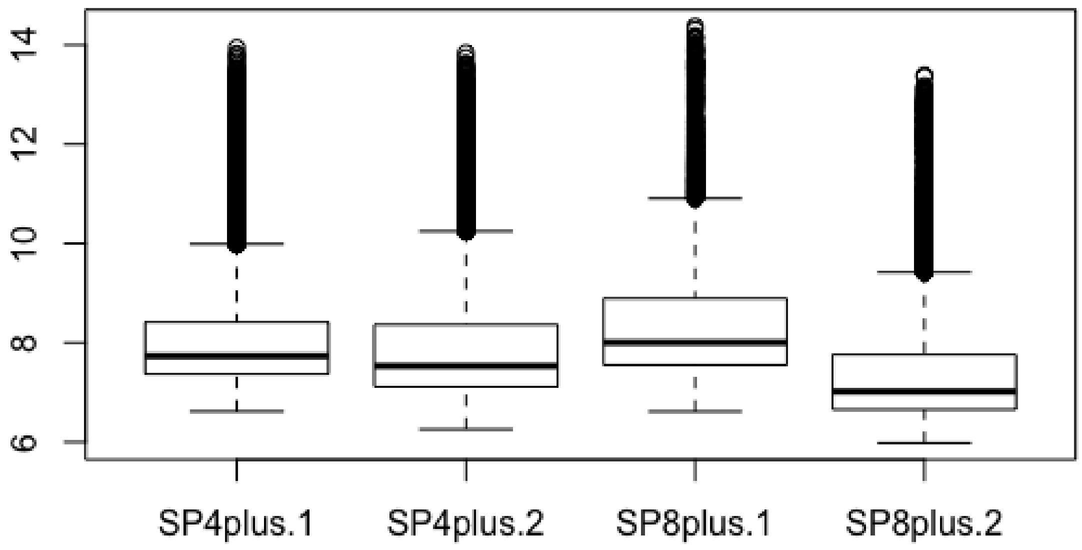
```
> data[data$Gene=="CD4",4:7]

##       SP4plus.1 SP4plus.2 SP8plus.1 SP8plus.2
## 3614    1369.90   1388.40    384.12   179.200
## 16335    134.44    114.23    144.94    88.963
```

```
> data[data$Gene=="CD8A",4:7]

##       SP4plus.1 SP4plus.2 SP8plus.1 SP8plus.2
## 5825     315.31    260.42    5913.2    2842.3
```

This shows that one probeset for CD4 (line 3614) and one for CD8A (line 5825) do indeed show higher expression in the SP4+ resp. SP8+ samples.

r. Create boxplots of the four microarrays using boxplot(log2(data[,4:7])). What potential problem do you notice?

```
> boxplot(log2(data[,4:7]))
```



The distributions look similar, but shifted, where you would expect replicates to be similar.

s. What could cause this potential problem and how could you correct for it?

It could be due to technical variation, for example when different amounts of RNA were hybridized to the array. It can be solved using normalization, which will be discussed in block 2.

<u>Optional</u>

t. The data you were supplied with was already pre-processed: probe values were combined into overall expression values for probesets BioConductor library contains numerous packages to work directly with the raw data, in this case Affymetrix CEL files.

Install the affyQCReport package and its dependencies:

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("affyQCReport")
```

Then generate an Affymetrix quality report using:

```
> library(affyQCReport)
> QCReport(ReadAffy(celfile.path="<MYFOLDER>"))
```

where you replace by the name of the folder that contains the CEL files ("SP4+_1na.CEL" ..., "SP8+_2na.CEL"). This function places a file AffyQCReport.pdf in the same folder. Open it - does it corroborate your findings above?