Block 2: Visualization and normalization

Huizhi Lin (88112518130) & Jan Orsel (950608630010)
November 8, 2017

First we loaded the data into dataframe (skin.df) then recreated the transposed format (tdata) like we did in the block 1 project.

```
# load in the data
skin.df <- read.table("get_normal_vs_tumor2_RAW_Skin.out",sep="
",header=TRUE)
# Create a reversed data frame
# Transpose just the gene expression
tdata <- data.frame(t(skin.df[,-2562]))
# Add sample type as column name
colnames(tdata)<-paste0(skin.df$tissue,1:72)
```

In order to get a general view of the data we used functions like dim(), str() names() summary() and quantile(). This resulted in a lot of information that can help interpreting the data set. In the block below for some functions the results have been cut of because of the first few lines indicate what the function returns and deleting the rest of the lines will increase the readability of this document. This is done for all results that show a lot of repetitive lines throughout this file.

```
dim(tdata)

## [1] 2561   72

str(tdata)

## 'data.frame':    2561 obs. of  72 variables:
##  $ tumor1  : num  16 82.6 366.4 1351.1 44 ...
##  $ tumor2  : num  15.3 86.7 344.5 1361.4 47.2 ...
ETC

names(tdata)

##  [1] "tumor1"   "tumor2"   "tumor3"   "tumor4"   "tumor5"   "tumor6"
##  [7] "tumor7"   "tumor8"   "tumor9"   "tumor10"  "tumor11"  "tumor12"
ETC

quantile(tdata, na.rm=TRUE)

##         0%        25%        50%        75%       100%
##     9.6200    40.2775   106.7700   323.1450 21891.7600

summary(tdata)

##      tumor1             tumor2             tumor3
##  Min.   :   10.40   Min.   :   10.32   Min.   :   10.39
```

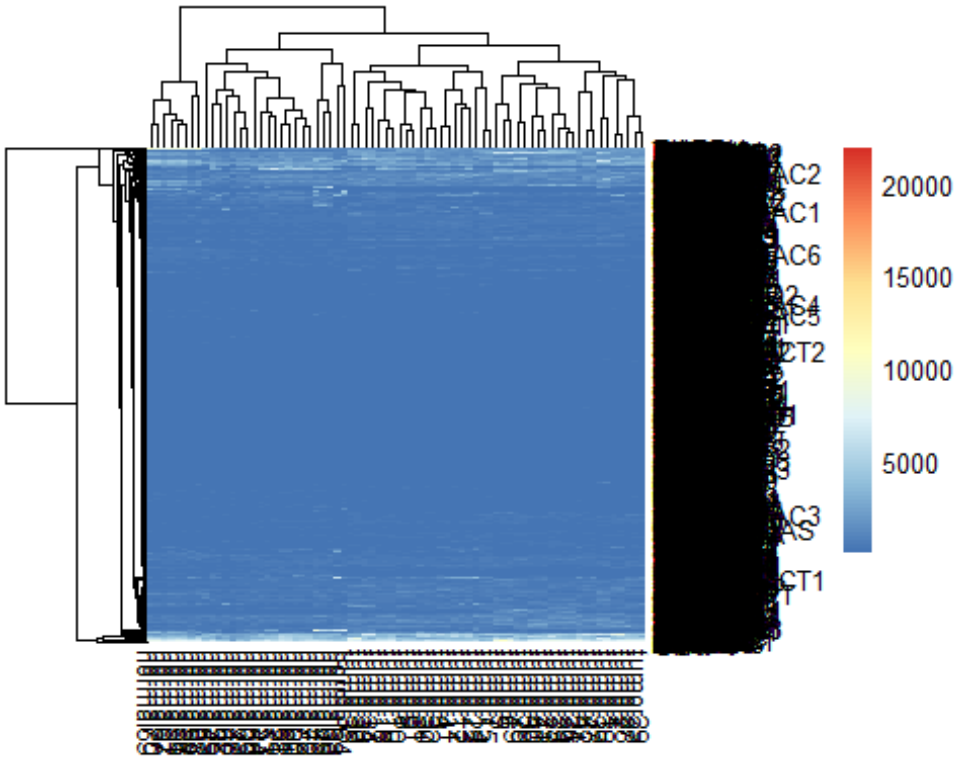
```
##  1st Qu.:   36.69   1st Qu.:   37.69   1st Qu.:   36.46
##  Median :  102.43   Median :   98.89   Median :  103.49
##  Mean   :  443.14   Mean   :  449.55   Mean   :  388.55
##  3rd Qu.:  387.14   3rd Qu.:  385.08   3rd Qu.:  358.07
##  Max.   :18651.45   Max.   :18504.65   Max.   :20513.70
ETC
```

The most important characteristics that can be derived from these functions are that the data frame has 2561 rows (genes), 72 column (samples), the maximal value is 21891.76, the minimal value is 9.62 and median is 106.77.

Task 1

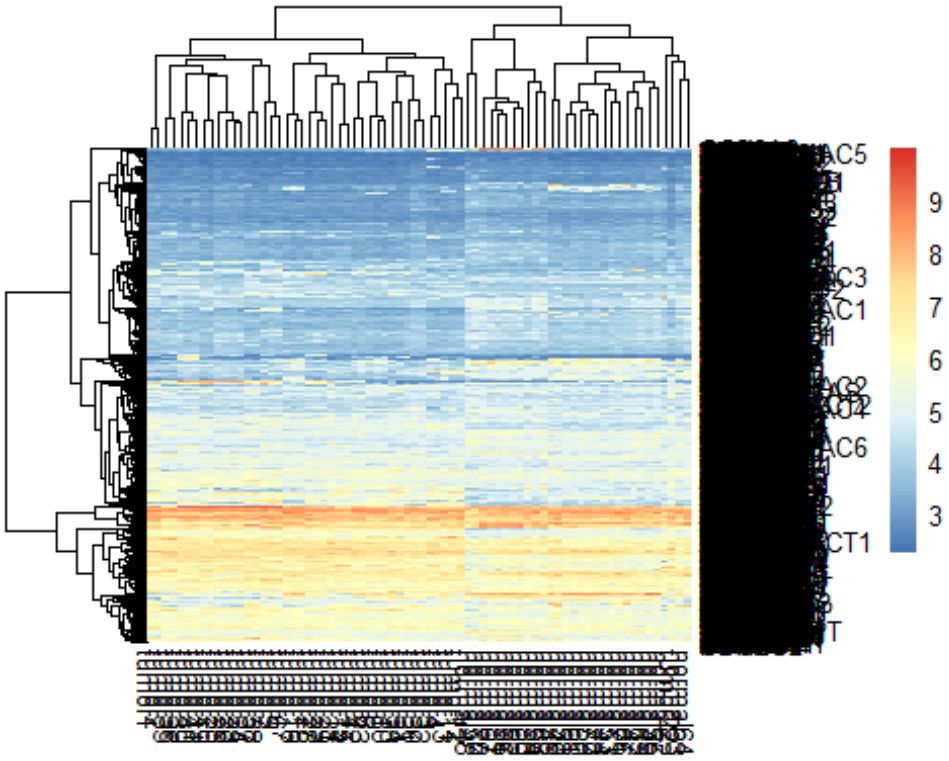
First we generate two heatmaps, one for the log-transformed data and one for the original data. This log function is used because all the variables are right-skew. This can be seen in the histograms created in the block one project. Because the data is right-skewed it can be influenced a lot by a few outliers. Taking the log will help by reducing the skew and therefore make differences easier to spot.

```
# Generate a heatmap for both log-transformed data with the original data.
library(pheatmap)

## Warning: package 'pheatmap' was built under R version 3.4.2

# for original data
pheatmap(tdata)
```



This heatmap is of course not very informative, there seems to be hardly any difference at all. When looking at the heatmap of the log of the data however, a clear distinction between tumour and normal samples is visible. The tree on the top of the plot nicely separates the samples in two groups

```
# for log-transformed data
pheatmap(log(tdata))
```

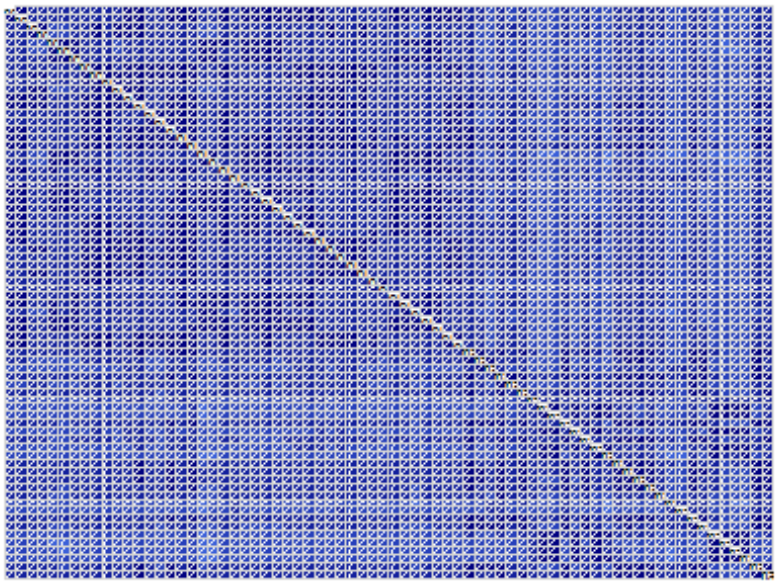


We visualized the sample correlation matrix and made correlation matrixes using the corrgram function.

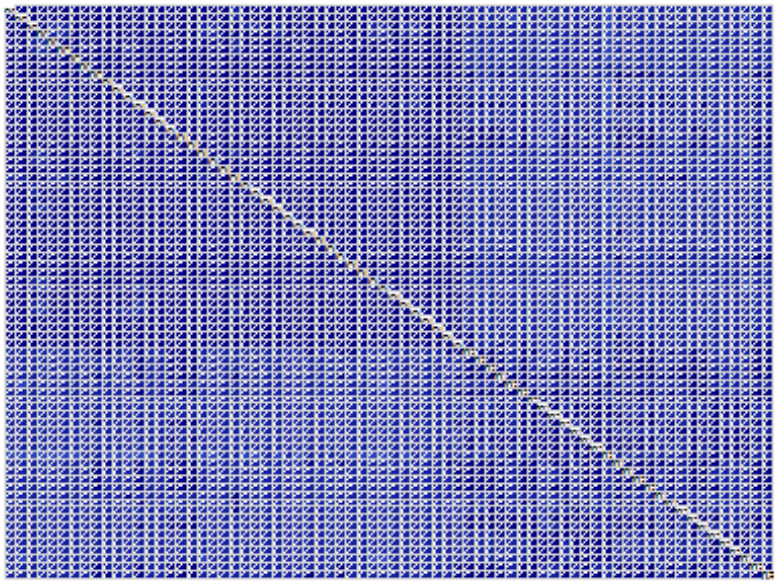
```
# create correlation matrix for both log-transformed data with the original
data.
cor(tdata)
```

```
##             tumor1    tumor2    tumor3    tumor4    tumor5    tumor6
## tumor1   1.0000000 0.9959747 0.8200759 0.8231901 0.8330841 0.8251239
## tumor2   0.9959747 1.0000000 0.8142107 0.8202314 0.8380307 0.8305948
## tumor3   0.8200759 0.8142107 1.0000000 0.9833068 0.7011787 0.6835361
## tumor4   0.8231901 0.8202314 0.9833068 1.0000000 0.7029619 0.6816784
## tumor5   0.8330841 0.8380307 0.7011787 0.7029619 1.0000000 0.9778747
## tumor6   0.8251239 0.8305948 0.6835361 0.6816784 0.9778747 1.0000000
## tumor7   0.8440737 0.8495522 0.7277355 0.7316856 0.9582441 0.9463439
ETC
```

```r
# visualize the correlation matrix
library(corrgram)

## Warning: package 'corrgram' was built under R version 3.4.2

corrgram(tdata)
```



```r
corrgram(log(tdata))
```



The second correlation matrix shows, though vaguely, a separation into two groups. It is not clear what those groups are though. The heatmaps we generated showed no white blocks so we assumed there are no missing values in the data. We confirmed this by using the code below.

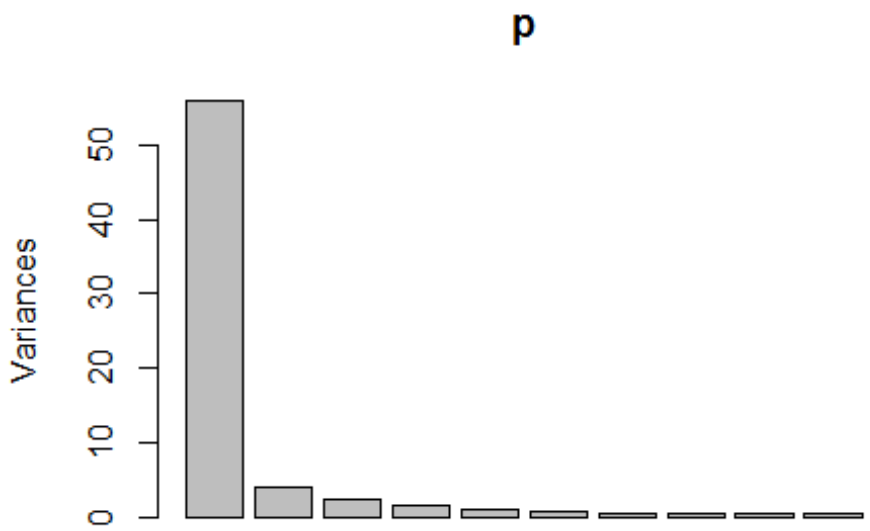
```
# Check missing values
any(is.na(tdata))

## [1] FALSE
```

Task 2

We used the prcomp() function to generate PCA. Using plot() and summary() this data was inspected. This was done for the original data and the log derived data.

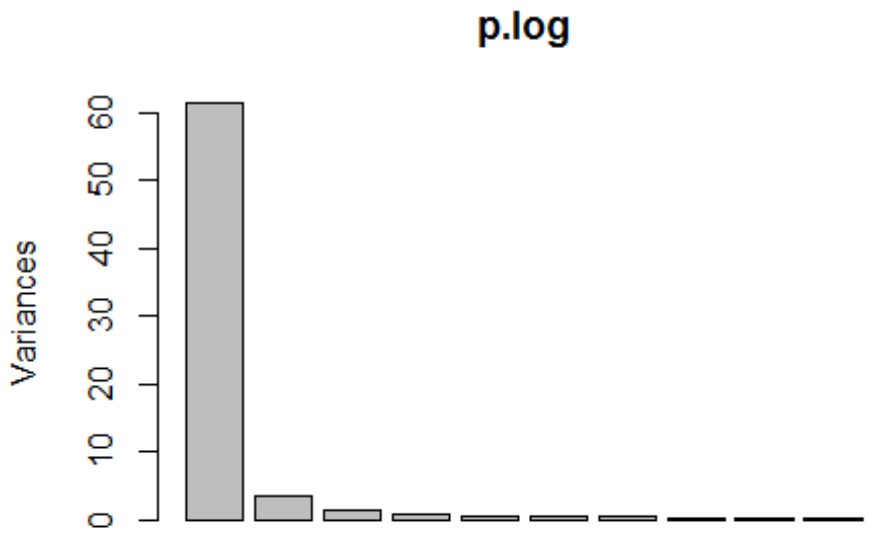
```
p <- prcomp(tdata,scale=TRUE)
summary(p)

## Importance of components%s:
##                            PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     7.4807 2.02059 1.48838 1.25235 0.92381 0.81535
## Proportion of Variance 0.7772 0.05671 0.03077 0.02178 0.01185 0.00923
## Cumulative Proportion  0.7772 0.83393 0.86470 0.88648 0.89833 0.90757
##                            PC7     PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.71572 0.69210 0.65840 0.62961 0.61872 0.6119
## Proportion of Variance 0.00711 0.00665 0.00602 0.00551 0.00532 0.0052
```

```
## Cumulative Proportion  0.91468 0.92133 0.92735 0.93286 0.93818 0.9434
ETC

plot(p)
```



Based on these results it is shown that the first six PCs are needed to explain at least 90% of the variation.

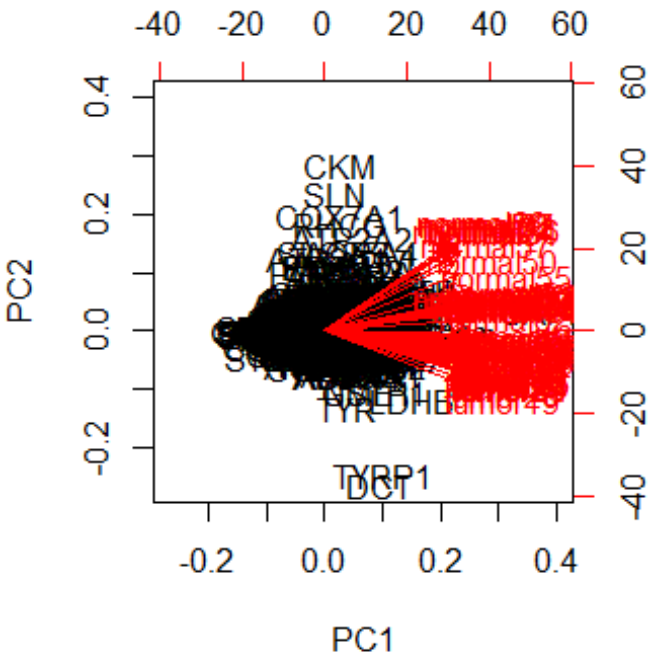
```
p.log <- prcomp(log(tdata),scale=TRUE)
summary(p.log)

## Importance of components%s:
##                            PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     7.8300 1.84983 1.14255 0.86721 0.71614 0.59917
## Proportion of Variance 0.8515 0.04753 0.01813 0.01045 0.00712 0.00499
## Cumulative Proportion  0.8515 0.89903 0.91716 0.92761 0.93473 0.93972
##                            PC7     PC8    PC9    PC10    PC11    PC12
## Standard deviation     0.5631 0.53948 0.5231 0.50293 0.47144 0.44985
## Proportion of Variance 0.0044 0.00404 0.0038 0.00351 0.00309 0.00281
## Cumulative Proportion  0.9441 0.94816 0.9520 0.95548 0.95856 0.96137
ETC

plot(p.log)
```



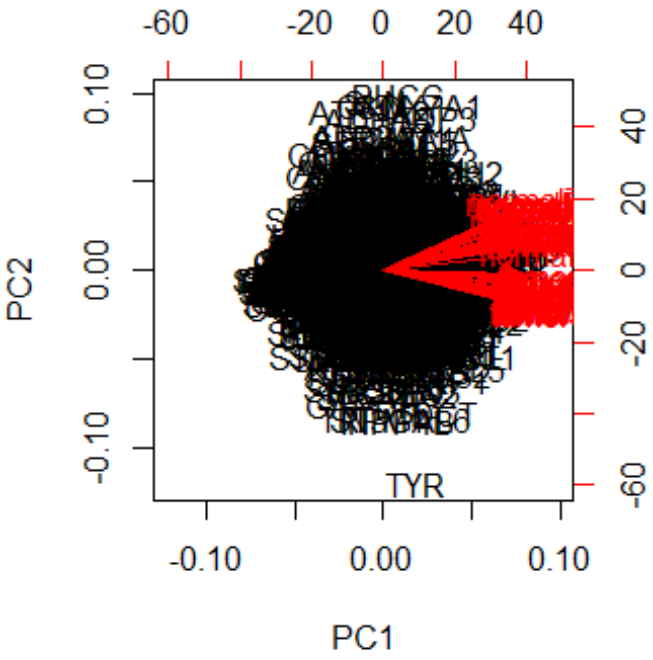
The proportion of variance is highly differentiates from the data derived from the original data. Now only the first three PCs are needed to explain at least 90% of the variation.

Using the biplot() function a biplot is created for both the normal data and the log derived data.

```
# original data
biplot(p)
```




```
# log-transformed data
biplot(p.log)
```



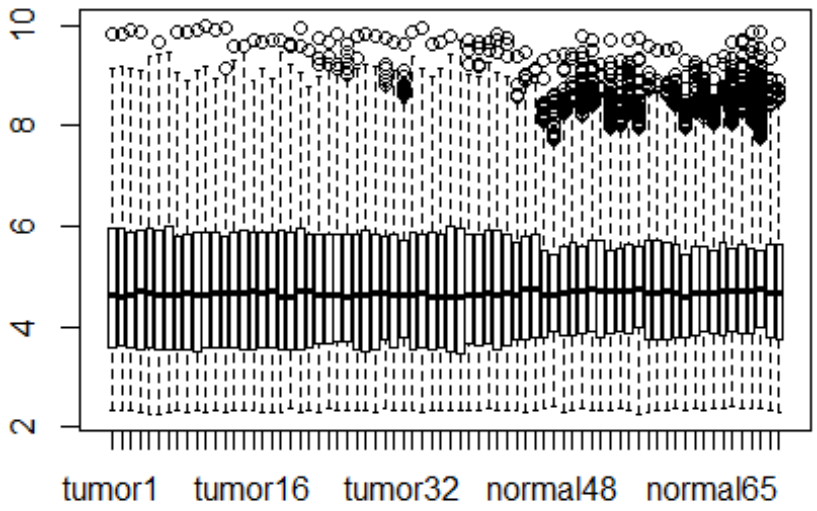
These biplots shows that most variation is captured by PC1. This can be stated because all of the arrows range from 0.1 to 0.4 over the PC1 axis and therefore all point in generally the same direction. PC2 shows more variation, about half of the samples are grouped on the positive side of the PC2 axis, these are all the samples derived from healthy tissue. The other half is grouped on the negative side of the PC2 axis, these are all the samples derived from tumor samples.

The biplot based on the log derived data shows these division of groups in PC2 much more clearly. Between the samples in the group seems to be less variation.

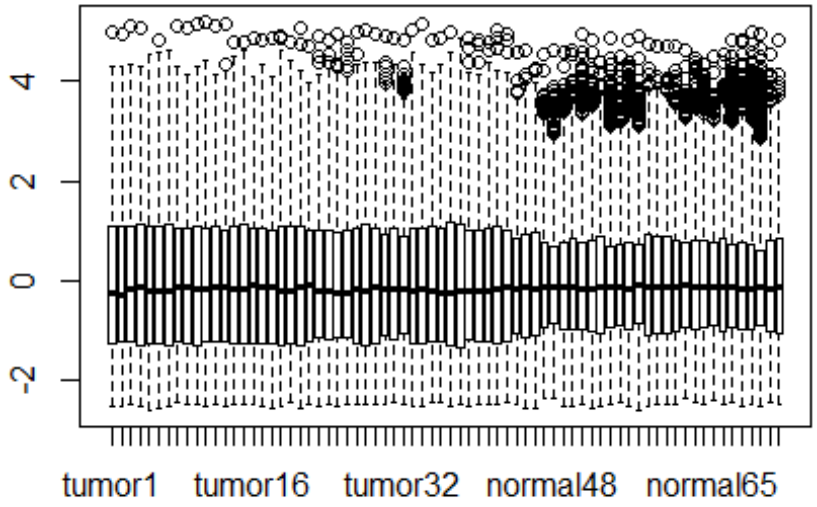
Task 3

Here we try different normalisation methods to help us decide which is best. The data was mean normalised (assigned to the mn_tdata variable) and mean/variance normalised (assigned to the mvn_tdata variable). Boxplots were created to visualise the difference between these normalisation methods. The first boxplot created in the following block of code produces a plot of the non-normalised log of the original data. This can be used to compare to the normalised plots and to visualise the difference between the not normalised and the normalised data.

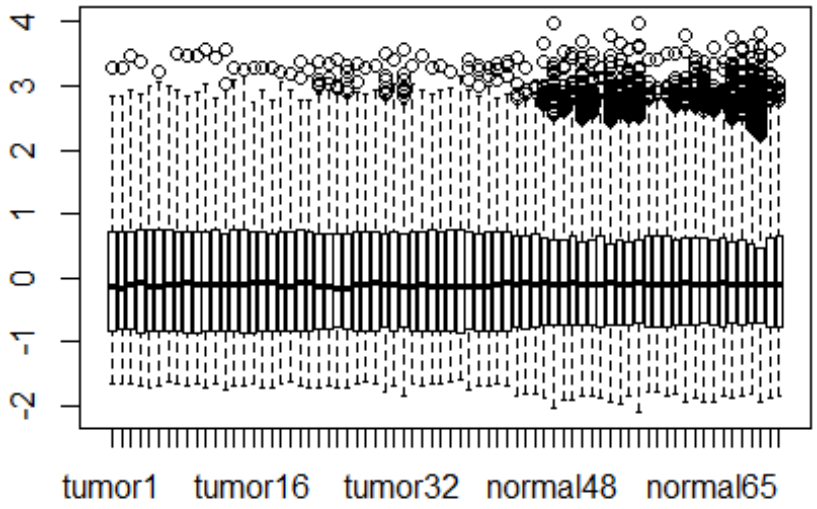
```
# mean normalisation
mn_tdata <- scale(log(tdata), center=TRUE, scale=FALSE)
# mean/variance normalisation
mvn_tdata <- scale(log(tdata), center=TRUE, scale=TRUE)
# boxplot on log-transformed data
boxplot(log(tdata))
```



```r
# boxplot on mean normalised data
boxplot(mn_tdata)
```



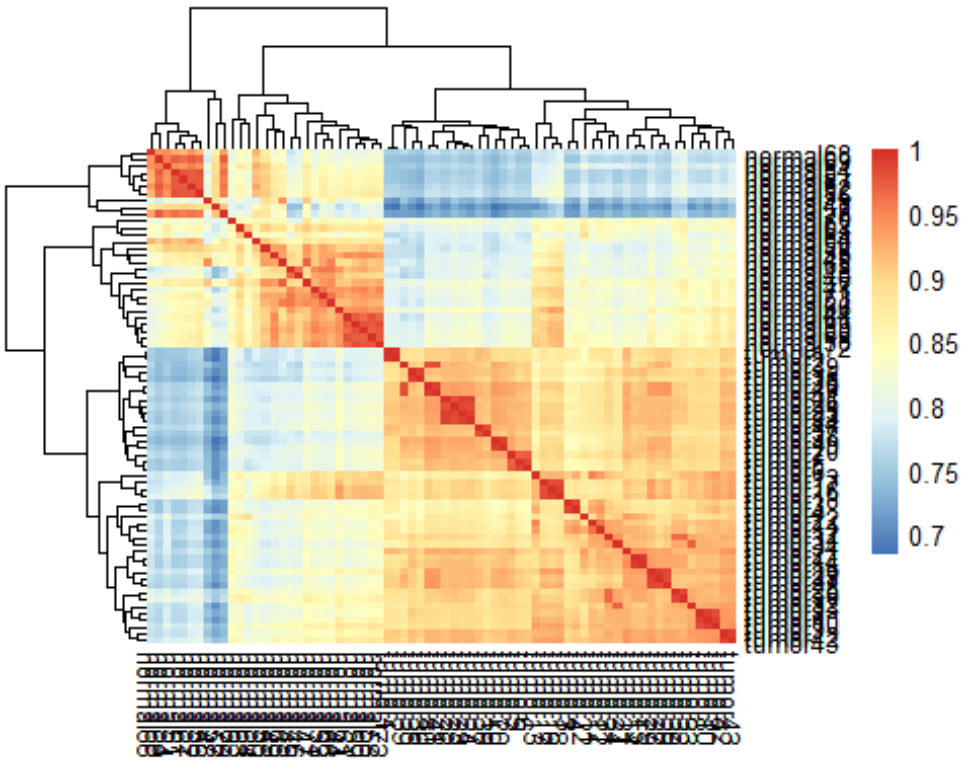

```r
# boxplot on mean/variance normalised data
boxplot(mvn_tdata)
```



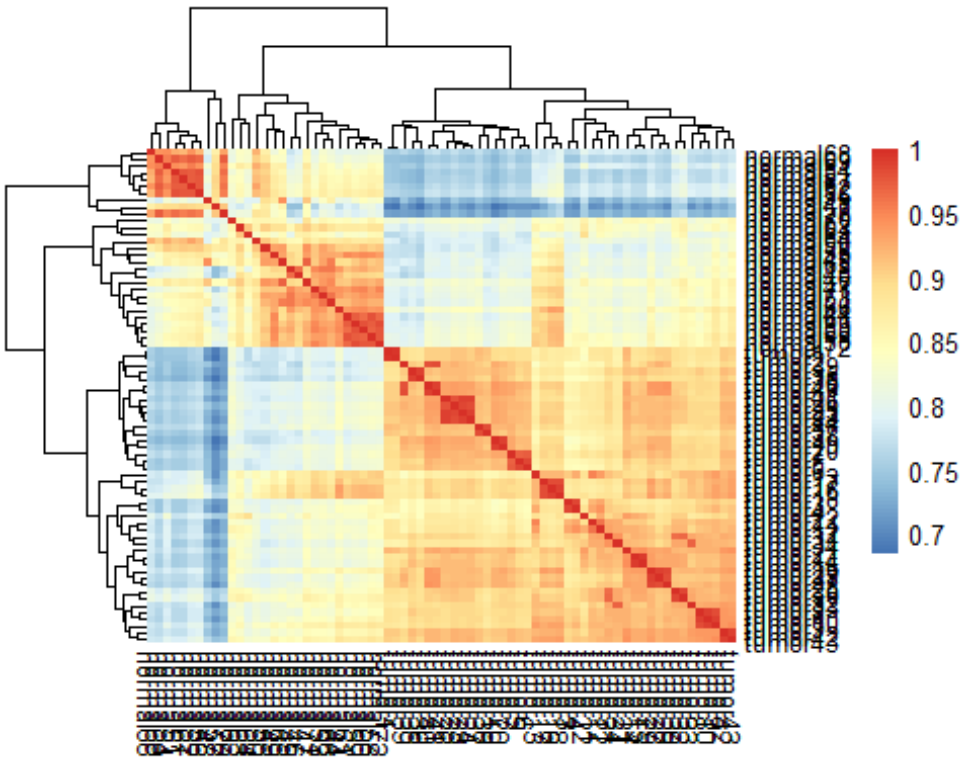
In our opinion, the mean normalisation is in this case the better option. Because the purpose of this project is to analyse human transcriptome variation across healthy tissues and diseased (cancer) tissues. Therefore we expect there would be differences between tumour and normal samples. When using the mean/variance normalisation we could remove the biological variation we are looking for. Whilst the mean normalisation minimalizes the technical variation it will not interfere with the potentially interesting biological variation. We therefore decided to keep working with the mean normalised data

Here we try to visualise the difference in correlation before and after our normalisation efforts. This however turned out to be a bit more difficult than we expected because the difference was so small.

```
#Comparing correlation
cor.matrix.log <- cor(log(tdata))
cor.mn_tdata <- cor(mn_tdata)
pheatmap(cor.matrix.log)
```

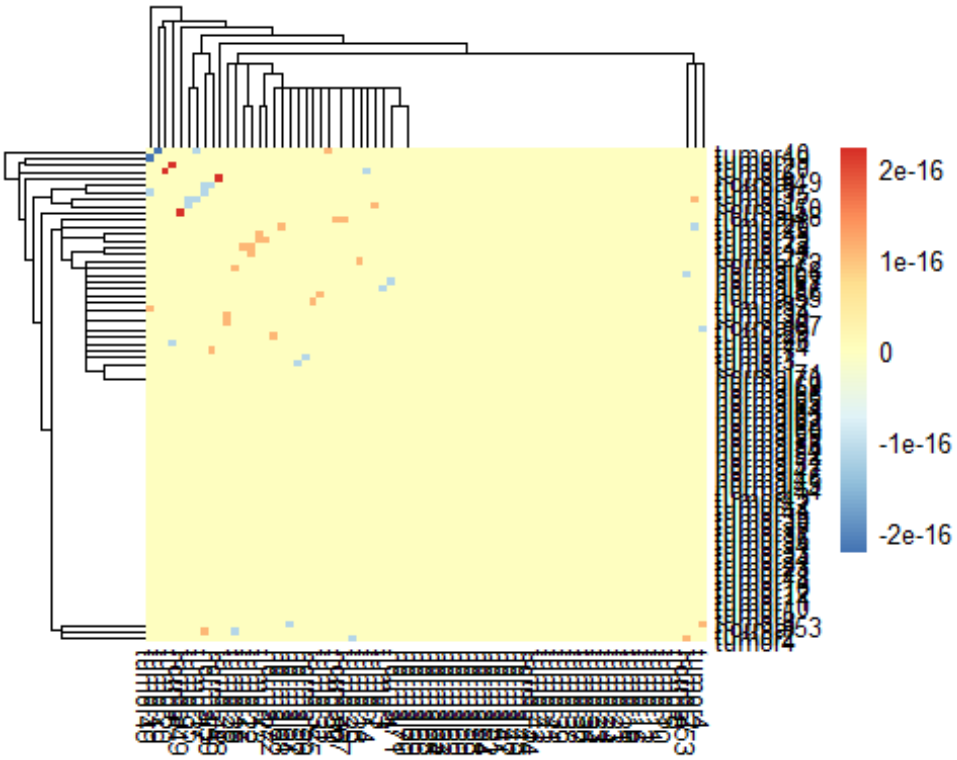



```
pheatmap(cor.mn_tdata)
```



When comparing these two plots no difference is visible. Therefore we decided to subtract the correlation of the normalised data from the original data and make a heatmap of the result. When we did this it became clear why no change was visible on the previous two plots. The difference in correlation is very small

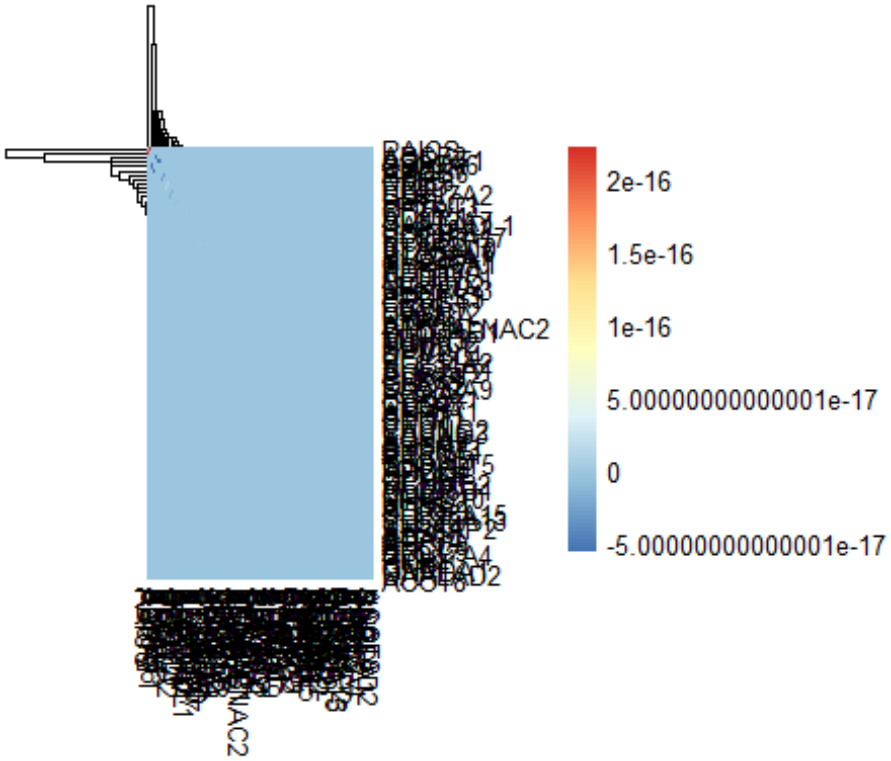
```
# calculate changes
change <- cor.matrix.log - cor.mn_tdata
pheatmap(change)
```



We also inspected the correlation between normalized and original values across first 100 genes and visualized the change induced by the normalization using a heatmap. Here too the change was minimal.

```
# select first 100 genes out of the tdata
genes <- skin.df[,1:100]
cor.genes <- cor(log(genes))
#normalise the genes data frame
mn_genes <- scale(log(genes),center=TRUE, scale=FALSE)
# calculate the difference and visualize the change
cor.mn_genes <- cor(mn_genes)
genes.diff <- cor.genes - cor.mn_genes
summary(genes.diff)

##     NAALAD2      NAALADL1              ACOT8        GNPDA1
##  Min.   :0   Min.   :0.000e+00   Min.   :0   Min.   :0
```

```
##  1st Qu.:0   1st Qu.:0.000e+00   1st Qu.:0   1st Qu.:0
##  Median :0   Median :0.000e+00   Median :0   Median :0
##  Mean   :0   Mean   :4.337e-21   Mean   :0   Mean   :0
##  3rd Qu.:0   3rd Qu.:0.000e+00   3rd Qu.:0   3rd Qu.:0
##  Max.   :0   Max.   :4.337e-19   Max.   :0   Max.   :0
##      KCNE3                 GNE         HCN4        PIGK      SLC17A4
##  Min.   :-5.421e-20   Min.   :0   Min.   :0   Min.   :0   Min.   :0
##  1st Qu.: 0.000e+00   1st Qu.:0   1st Qu.:0   1st Qu.:0   1st Qu.:0
##  Median : 0.000e+00   Median :0   Median :0   Median :0   Median :0
##  Mean   :-5.421e-22   Mean   :0   Mean   :0   Mean   :0   Mean   :0
##  3rd Qu.: 0.000e+00   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0   3rd Qu.:0
##  Max.   : 0.000e+00   Max.   :0   Max.   :0   Max.   :0   Max.   :0
ETC

pheatmap(genes.diff)
```



Task 4

We mean normalized all three datasets , and created the transposed format for all the datasets using a variation of the same code used to load in the original data. We saved the results for using later on in the project.

```
# open all the datasets
RAW.all <- read.table("get_normal_vs_tumor_RAW.out", header = TRUE, sep = "
", stringsAsFactors = FALSE)
```

```
RAW.breast <- read.table("get_normal_vs_tumor2_RAW_Breast.out", header =
TRUE, sep = " ", stringsAsFactors = FALSE)
RAW.FeReSy <-
read.table("get_normal_vs_tumor2_RAW_Female.Reproductive.System.out", header
= TRUE, sep = " ", stringsAsFactors = FALSE)

# create the transposed format
tRAW.all <- data.frame(t(RAW.all[,-2562]))
colnames(tRAW.all)<-paste0(RAW.all$tissue,1:2132)
tRAW.breast <- data.frame(t(RAW.breast[,-2562]))
colnames(tRAW.breast)<-paste0(RAW.breast$tissue,1:503)
tRAW.FeReSy <- data.frame(t(RAW.FeReSy[,-2562]))
colnames(tRAW.FeReSy)<-paste0(RAW.FeReSy$tissue,1:130)

# normalize all the datasets
mn.tRAW.all <- scale(log(tRAW.all), center=TRUE, scale=FALSE)
mn.tRAW.breast <- scale(log(tRAW.breast), center=TRUE, scale=FALSE)
mn.tRAW.FeReSy <- scale(log(tRAW.FeReSy), center=TRUE, scale=FALSE)
```

Conclusion

Using the log of a data set that is steeply skewed makes it easier to work with. For us this became particulary clear when comparing the first two heatmaps and the biplot of the original data to the biplot of the log derived data. The biplot generated from the log derived data showed a clearer difference between the tumour and normal sample whilst it looked like there was less difference between the groups. Visualising the difference of the normalisation efforts turned out to be harder than originally expected, the heatmap that displays the change between the data before and after normalisation only shows a small amount of difference.