BCB420 - Computational Systems Biology

Lecture 5 - Differential Expression

Ruth Isserlin

2020-03-22

Before we start

Assignment #1

- Due Today! @ 20:00

What to hand in?

- **html rendered RNotebook** - you should be able to submit this through quercus
- Make sure the notebook and all associated code is checked into your github repo as I will be pulling all the repos at the deadline and using them to compile your code. - Your checked in code must replicate the handed in notebook.
- **Do not check the data file into your repo!** - your code should download the data from GEO and generate a new, cleaned data file.
- Document your work and your code directly in the notebook.
- **Read the paper associated with your data!**
- You are allowed to use helper functions or methods but make sure when you source those files the paths to them are relative and that they are checked into your repo as well.

Differential Gene Expression Analysis

Where we left off from last week

- data from "Apoptosis enhancing drugs overcome innate platinum resistance in CA125 negative tumor initiating populations of high grade serous ovarian cancer"
- 10 ovarian tumours sorted by CA125+ve and -ve antibody
- we normalized it, we cleaned it, we made sure we had up to date identifers from ensembl.
- What's next?

First things first,

- Load the data

```
normalized_count_data <- read.table(file=file.path("data",
"GSE70072_finalized_normalized_counts.txt"),
                                    header = TRUE,sep = "\t",
                                    stringsAsFactors = FALSE,
                                    check.names=FALSE)
```

- Take a look at the data we just loaded.

```
kable(normalized_count_data[1:5,1:5], type="html")
```

ensembl_gene_id	hgnc_symbol	Pt.A.CA125-	Pt.A.CA125+	pt.B.CA125-
ENSG00000000003	TSPAN6	6.945591	6.6488678	7.1585772
ENSG00000000419	DPM1	5.912242	6.0789211	5.3233556
ENSG00000000457	SCYL3	4.046979	3.2375251	4.2441139
ENSG00000000460	C1orf112	3.927282	3.6138063	4.1747420
ENSG00000000938	FGR	0.000000	0.8434171	0.4502989

Create a numerical matrix that we can create a heatmap from

```
heatmap_matrix <-
normalized_count_data[,3:ncol(normalized_count_data)]
rownames(heatmap_matrix) <- normalized_count_data$ensembl_gene_id
colnames(heatmap_matrix) <-
colnames(normalized_count_data[,3:ncol(normalized_count_data)])
```

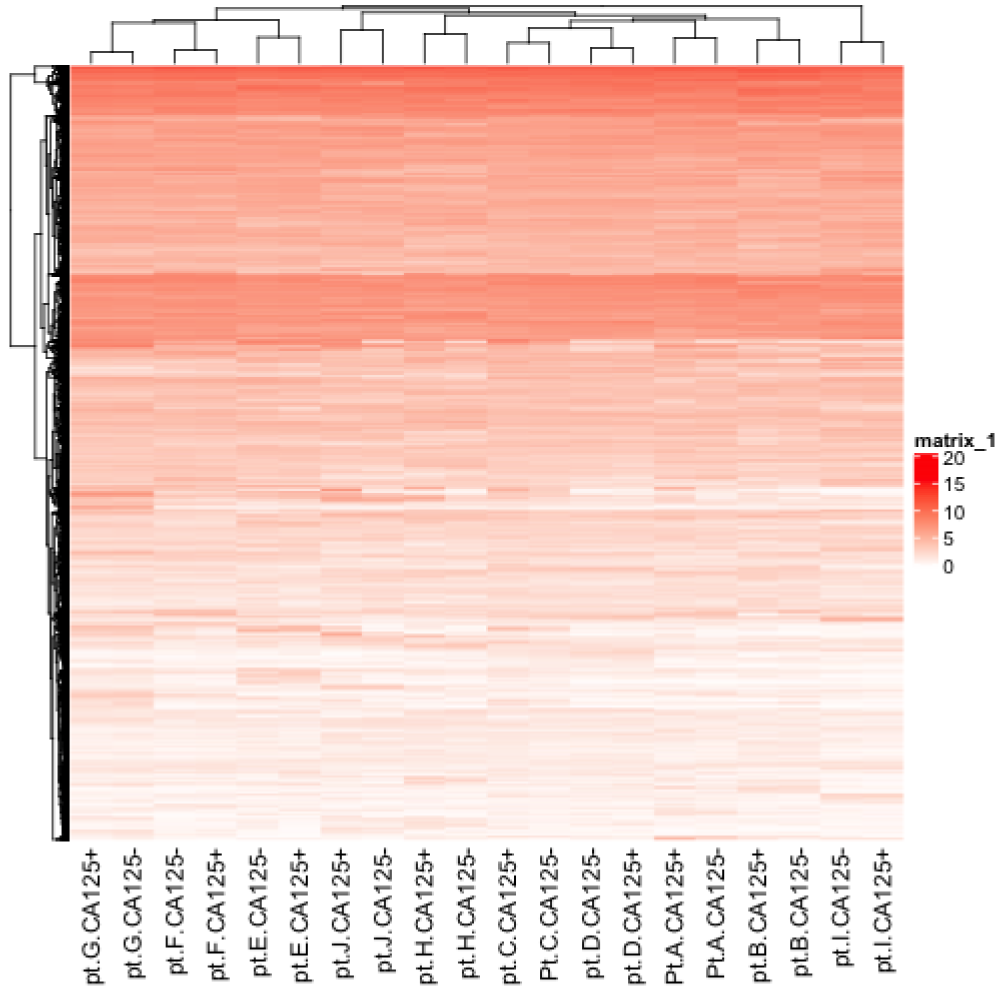
Create a Heatmap

What is a heatmap?

- data graph that translates numbers into a colour scale over many samples and measurements.
- Has multiple additional methods that we can use to restructure the format to highlight themes in the data.

```
library(ComplexHeatmap)
library(circlize)

if(min(heatmap_matrix) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix)), c(
"white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix), 0,
max(heatmap_matrix)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix),
                               show_row_dend = TRUE,
                               show_column_dend = TRUE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE
```

```
current_heatmap
```

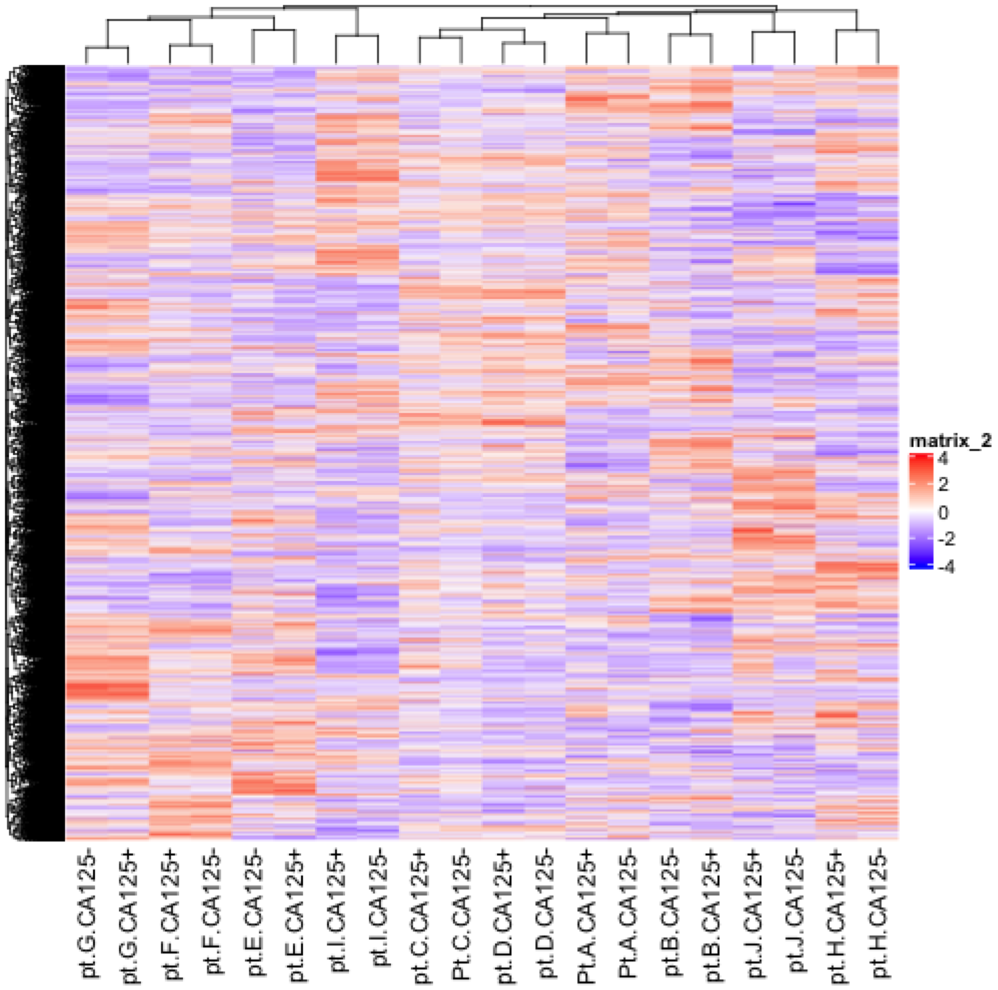


Let's try that again using Row - normalization

- scale each row and centre them around the mean.
- From each value we subtract the mean and divide by the standard deviation of the row to row normalize it.
- some other heatmap packages might have row normalization built in.

```r
heatmap_matrix <- t(scale(t(heatmap_matrix)))

if(min(heatmap_matrix) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix)), c(
"white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix), 0,
max(heatmap_matrix)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix),
                               show_row_dend = TRUE,
                               show_column_dend = TRUE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE
                               )
```

```
current_heatmap
```



Traditionally, low scale experiments are designed to compare the expression of a single gene or maybe an handful of genes.

```
ca125_pos_samples <- grep(colnames(normalized_count_data),
                          pattern="\\+")
ca125_neg_samples <- grep(colnames(normalized_count_data),
                          pattern="\\-")

gene_of_interest <- which(normalized_count_data$hgnc_symbol ==
"MUC16")
```

```
muc16_neg_samples <-
t(normalized_count_data

[gene_of_interest,

ca125_neg_samples])
colnames(muc16_neg_samples) <-
c("neg_samples")
muc16_neg_samples
```

```
##              neg_samples
## Pt.A.CA125-   10.723796
## pt.B.CA125-    8.850579
## Pt.C.CA125-    9.153117
## pt.D.CA125-    9.013810
## pt.E.CA125-    7.072799
## pt.F.CA125-    7.666300
## pt.G.CA125-    8.847846
## pt.H.CA125-    9.392200
## pt.I.CA125-    7.536275
## pt.J.CA125-    6.190249
```

```
muc16_pos_samples <-
t(normalized_count_data

[gene_of_interest,

ca125_pos_samples])
colnames(muc16_pos_samples) <-
c("pos_samples")
muc16_pos_samples
```

```
##              pos_samples
## Pt.A.CA125+    9.909214
## pt.B.CA125+    8.587325
## pt.C.CA125+    8.340244
## pt.D.CA125+    8.710818
## pt.E.CA125+    7.085022
## pt.F.CA125+    7.861518
## pt.G.CA125+    8.140352
## pt.H.CA125+    8.884911
## pt.I.CA125+    7.785848
## pt.J.CA125+    5.596313
```

Is MUC16 differentially expressed in our samples?

- Using a simple t.test compare this individual gene.
- The null hypothesis of the two sample t-test is that there is **no** difference in means of each sample
- It assumes that both sample A and sample B are normally distributed.

```
t.test(x=t(muc16_pos_samples),y=t(muc16_neg_samples))
```

```
## 
##      Welch Two Sample t-test
## 
## data:  t(muc16_pos_samples) and t(muc16_neg_samples)
## t = -0.63961, df = 17.695, p-value = 0.5306
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.5205410  0.8114598
## sample estimates:
## mean of x mean of y 
##  8.090156  8.444697
```

```
muc16_neg_samples <-
t(normalized_count_data[gene_of_
colnames(muc16_neg_samples) <-
c("neg_samples")
muc16_neg_samples
```

```
##                 neg_samples
## Pt.A.CA125-   10.723796
## pt.B.CA125-    8.850579
## Pt.C.CA125-    9.153117
## pt.D.CA125-    9.013810
## pt.E.CA125-    7.072799
## pt.F.CA125-    7.666300
## pt.G.CA125-    8.847846
## pt.H.CA125-    9.392200
## pt.I.CA125-    7.536275
## pt.J.CA125-    6.190249
```

```
muc16_pos_samples <-
t(normalized_count_data[gene_of_
colnames(muc16_pos_samples) <-
c("pos_samples")
muc16_pos_samples
```

```
##                 pos_samples
## Pt.A.CA125+    9.909214
## pt.B.CA125+    8.587325
## pt.C.CA125+    8.340244
## pt.D.CA125+    8.710818
## pt.E.CA125+    7.085022
## pt.F.CA125+    7.861518
## pt.G.CA125+    8.140352
## pt.H.CA125+    8.884911
## pt.I.CA125+    7.785848
## pt.J.CA125+    5.596313
```

How can we account for these variables?

- There are many different packages that try and control for these variables. We are going to go through two of them:
    - Limma - LInear Models of MircroArray
    - orginallay published in 2004 for use with microarrays
    - updated and improved over the years to also include rnaseq data.
    - edgeR
    - Suite of methods specialized for Bulk RNAseq analysis
    - contains multiple methods to compute differential expression including a similar general linear method to the limma package.

Limma

- LInear Models of MircroArray
- The premise of the limma approach is the use of linear models to define differential expression.
- **Linear Models** - "describe a continuous response variable as a function of one or more predictor variables."^1
- Linear regression involves finding an linear model to explain the data. Often described as fitting a line to a set of data points.
- for our example, we have a set of measurements and we want to figure out the function that best describes it.
- Using empirical bayes to compute the odds of any gene being differentially expressed given its contrasts.

[1]https://www.mathworks.com/discovery/linear-model.html

If you remember from last week we used an MDSPlot to look at how our samples are clustering. We used the plotMDS from the edgeR package but we can just as easily use the plotMDS function from the the limma package.

```r
limma::plotMDS(heatmap_matrix,
               col = rep(c("darkgreen","blue"),10))
```

Another way to look at the exact same plot is to color by patient

```r
pat_colors <- rainbow(10)
pat_colors <- unlist(lapply(pat_colors,FUN=function(x){rep(x,2)}))

limma::plotMDS(heatmap_matrix,
               col = pat_colors )
```

Model

Define the groups

- From the above plot we know that which samples/patient the data comes from is important to determining its value.
- We also have hypothesized that CA125 status will also contribute to the differential.

```
#get the 2 and third token from the column names
samples <- data.frame(
        lapply(colnames(normalized_count_data)[3:22],
        FUN=function(x){
          unlist(strsplit(x, split = "\\."))[c(2,3)]}))
colnames(samples) <- colnames(normalized_count_data)[3:22]
rownames(samples) <- c("patients","cell_type")
samples <- data.frame(t(samples))
```

```
samples[1:5,]
```

```
##              patients cell_type
## Pt.A.CA125-         A    CA125-
## Pt.A.CA125+         A    CA125+
## pt.B.CA125-         B    CA125-
## pt.B.CA125+         B    CA125+
## Pt.C.CA125-         C    CA125-
```

Model - cont'd

- function to create a linear model in R - model.matrix
- creates a design matrix

```
model_design <- model.matrix(~ samples$cell_type)
kable(model_design, type="html")
```

(Intercept)	samples$cell_typeCA125+
1	0
1	1
1	0
1	1
1	0
1	1
1	0
1	1
1	0
1	1

Create our data matrix

- similar to what we used last week when we were using the edgeR package but slightly different

```
expressionMatrix <- as.matrix(normalized_count_data[,3:22])
rownames(expressionMatrix) <- normalized_count_data$ensembl_gene_id
colnames(expressionMatrix) <- colnames(normalized_count_data)[3:22]
minimalSet <- ExpressionSet(assayData=expressionMatrix)
```

Fit our data to the above model

```
fit <- lmFit(minimalSet, model_design)
```

Apply empircal Bayes to compute differential expression for the above described model.

- The parameter trend=TRUE is specific to RNA-seq data. (exclude for microarray data)

```
fit2 <- eBayes(fit,trend=TRUE)
```

```
topfit <- topTable(fit2,
                   coef=ncol(model_design),
                   adjust.method = "BH",
                   number = nrow(expressionMatrix))

#merge hgnc names to topfit table
output_hits <- merge(normalized_count_data[,1:2],
                     topfit,
                     by.y=0,by.x=1,
                     all.y=TRUE)

#sort by pvalue
output_hits <- output_hits[order(output_hits$P.Value),]
```

```r
kable(output_hits[1:10,],type="html")
```

	ensembl_gene_id	hgnc_symbol	logFC	AveExpr	t	P.Value	adj.P.V
7460	ENSG00000144824	PHLDB2	1.2098391	3.5114343	2.860563	0.0089748	0.99997
5910	ENSG00000134013	LOXL2	1.2123611	2.7201330	2.703285	0.0128421	0.99997
7076	ENSG00000141753	IGFBP4	1.3157584	4.7001042	2.650939	0.0144467	0.99997
2481	ENSG00000103241	FOXF1	0.8909267	0.5938013	2.624033	0.0153431	0.99997
5248	ENSG00000128578	STRIP2	0.5782501	1.6851401	2.623044	0.0153770	0.99997
5657	ENSG00000132031	MATN3	0.7304186	0.5863327	2.620737	0.0154564	0.99997
13151	ENSG00000187479	C11orf96	1.5050456	1.6884847	2.592236	0.0164698	0.99997
7458	ENSG00000144810	COL8A1	1.5478954	4.5217668	2.591839	0.0164843	0.99997
4132	ENSG00000117152	RGS4	1.6802024	2.8692665	2.585316	0.0167250	0.99997
18391	ENSG00000261335		-0.4209006	0.4253192	-2.556786	0.0178171	0.99997

How many gene pass the threshold p-value < 0.05?

```
length(which(output_hits$P.Value < 0.05))
```

```
## [1] 87
```

How many genes pass correction?

```
length(which(output_hits$adj.P.Val < 0.05))
```

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

Correction?

- Referring to multipole hypothesis testing. As the number of tests performed increases the liklihood that a positive results will occur simply by chance increases. We need to control for this
- Multiple hypothesis testing will come up for differential expression, pathways analysis and for any analysis where there are multiple tests being performed
- Control for family-wise error rate or for false discovery rate
- There are a range of different methods to correct for this:
    1. Bonferonni - considered to be overly stringent by many. p-values are multiplied by the number of comparisons
    2. Benjamni - hochberg
    3. Benjamini - Yekutieli

```
p.adjust.methods
```

```
## [1] "holm"         "hochberg"     "hommel"       "bonferroni"  "BH"
## [6] "BY"           "fdr"          "none"
```

Can we improve our results if we account for the patient variability?

Model - cont'd

- function to create a linear model in R - model.matrix
- creates a design matrix

```
model_design_pat <- model.matrix(
  ~ samples$patients + samples$cell_type)
kable(model_design_pat,type="html")
```

(Intercept)	samples$patientsB	samples$patientsC	samples$patientsD	samples$patientsE	sam
1	0	0	0	0	
1	0	0	0	0	
1	1	0	0	0	
1	1	0	0	0	
1	0	1	0	0	
1	0	1	0	0	
1	0	0	1	0	
1	0	0	1	0	

Fit our data to the above model

```
fit_pat <- lmFit(minimalSet, model_design_pat)
```

Apply empircal Bayes to compute differential expression for the above described model.

- The parameter trend=TRUE is specific to RNA-seq data. (exclude for microarray data)

```r
fit2_pat <- eBayes(fit_pat,trend=TRUE)
```

```r
topfit_pat <- topTable(fit2_pat,
                   coef=ncol(model_design_pat),
                   adjust.method = "BH",
                   number = nrow(expressionMatrix))

#merge hgnc names to topfit table
output_hits_pat <- merge(normalized_count_data[,1:2],
                         topfit_pat,by.y=0,by.x=1,all.y=TRUE)

#sort by pvalue
output_hits_pat <- output_hits_pat[order(output_hits_pat$P.Value),]
```

```r
kable(output_hits_pat[1:10,],type="html")
```

	ensembl_gene_id	hgnc_symbol	logFC	AveExpr	t	P.Value	adj.P.Va
12450	ENSG00000182752	PAPPA	1.2179144	1.681068	5.419384	0.0000782	0.442091
415	ENSG00000026508	CD44	1.1072045	7.848369	5.198156	0.0001182	0.442091
2402	ENSG00000102755	FLT1	0.7659613	1.645930	4.849620	0.0002293	0.442091
5117	ENSG00000126878	AIF1L	-0.6347014	6.355584	-4.795010	0.0002547	0.442091
1506	ENSG00000085552	IGSF9	-0.7312699	2.432940	-4.786149	0.0002591	0.442091
12397	ENSG00000182326	C1S	0.9220917	7.579617	4.745380	0.0002803	0.442091
8036	ENSG00000150938	CRIM1	0.6230945	5.673619	4.741666	0.0002823	0.442091
13268	ENSG00000188295	ZNF669	-0.7028924	3.272620	-4.724103	0.0002921	0.442091
13314	ENSG00000188641	DPYD	0.8558769	3.122403	4.549019	0.0004107	0.442091
19113	ENSG00000272796	RP1-74M1.3	-0.6786633	1.269677	-4.539777	0.0004182	0.442091

How many gene pass the threshold p-value < 0.05?

```r
length(which(output_hits_pat$P.Value < 0.05))
```

```
## [1] 1713
```

How many genes pass correction?

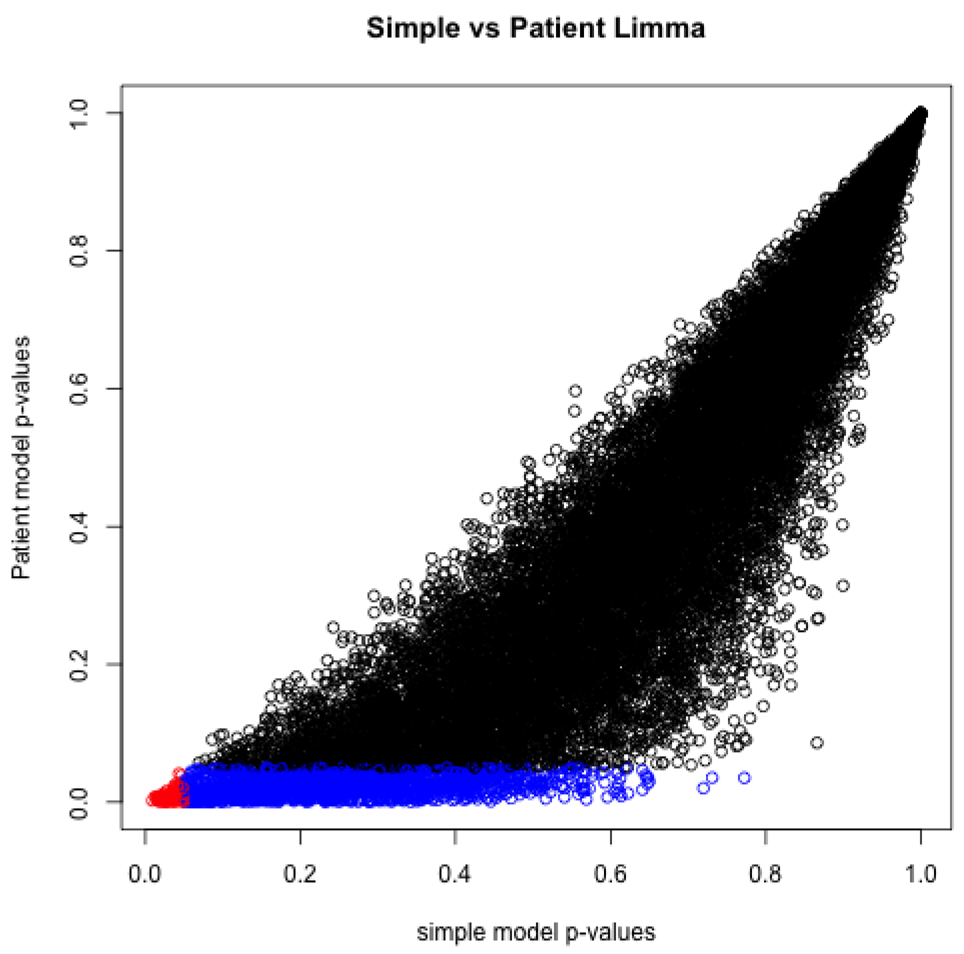
```r
length(which(output_hits_pat$adj.P.Val < 0.05))
```

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

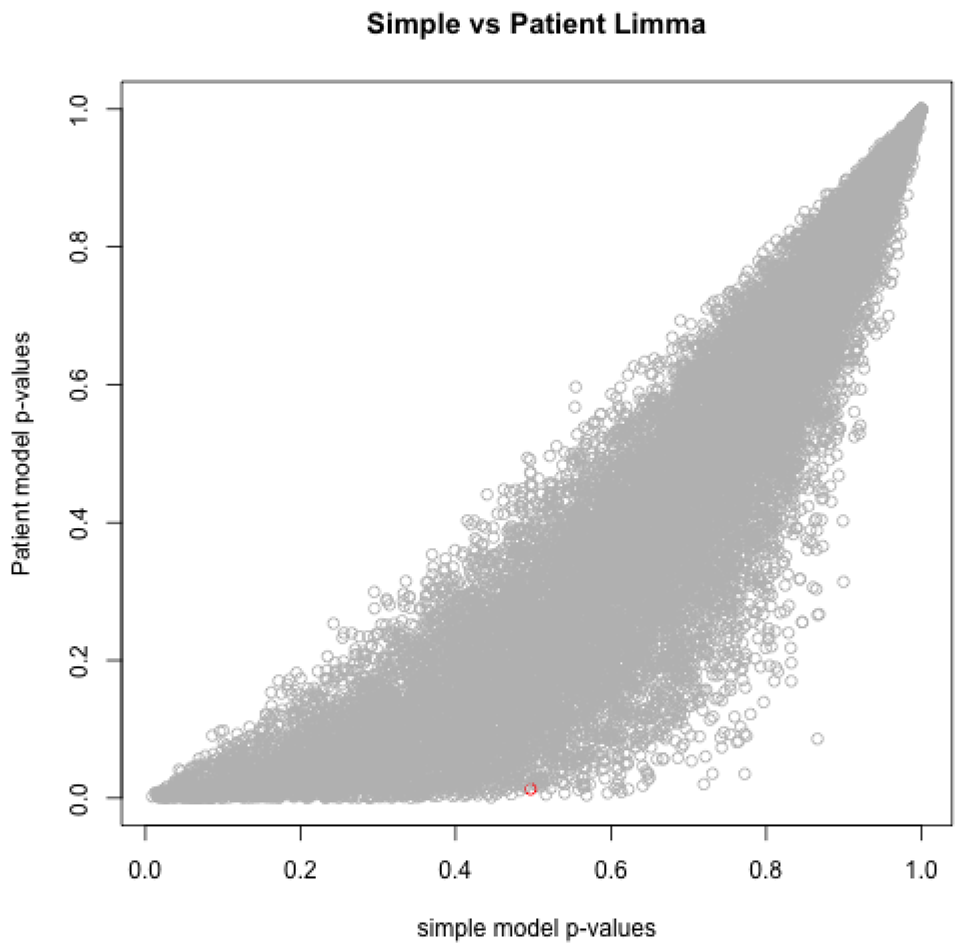
Compare the results from the two different models

```
simple_model_pvalues <- data.frame(ensembl_id =
output_hits$ensembl_gene_id,

simple_pvalue=output_hits$P.Value)
pat_model_pvalues <-  data.frame(ensembl_id =
output_hits_pat$ensembl_gene_id,
                                 patient_pvalue =
output_hits_pat$P.Value)
two_models_pvalues <- merge(simple_model_pvalues,
                            pat_model_pvalues,by.x=1,by.y=1)

two_models_pvalues$colour <- "black"
two_models_pvalues$colour[two_models_pvalues$simple_pvalue<0.05] <-
"orange"
two_models_pvalues$colour[two_models_pvalues$patient_pvalue<0.05]
<- "blue"
two_models_pvalues$colour[two_models_pvalues$simple_pvalue<0.05 &
two_models_pvalues$patient_pvalue<0.05] <- "red"
```

```
plot(two_models_pvalues$simple_pvalue,two_models_pvalues$patient_pval
     col = two_models_pvalues$colour,
     xlab = "simple model p-values",
     ylab ="Patient model p-values",
     main="Simple vs Patient Limma")
```


Simple vs Patient Limma
Patient model p-values
simple model p-values
0.0
0.2
0.4
0.6
0.8
1.0

```
ensembl_of_interest <- normalized_count_data$ensembl_gene_id[
  which(normalized_count_data$hgnc_symbol == "MUC16")]

two_models_pvalues$colour <- "grey"
two_models_pvalues$colour[two_models_pvalues$ensembl_id==ensembl_of_i
 <- "red"

plot(two_models_pvalues$simple_pvalue,two_models_pvalues$patient_pval
     col = two_models_pvalues$colour,
     xlab = "simple model p-values",
     ylab ="Patient model p-values",
      main="Simple vs Patient Limma")
```


Simple vs Patient Limma
Patient model p-values
simple model p-values
0.0
0.2
0.4
0.6
0.8
1.0

let's come back to the initial heatmap representation of the data

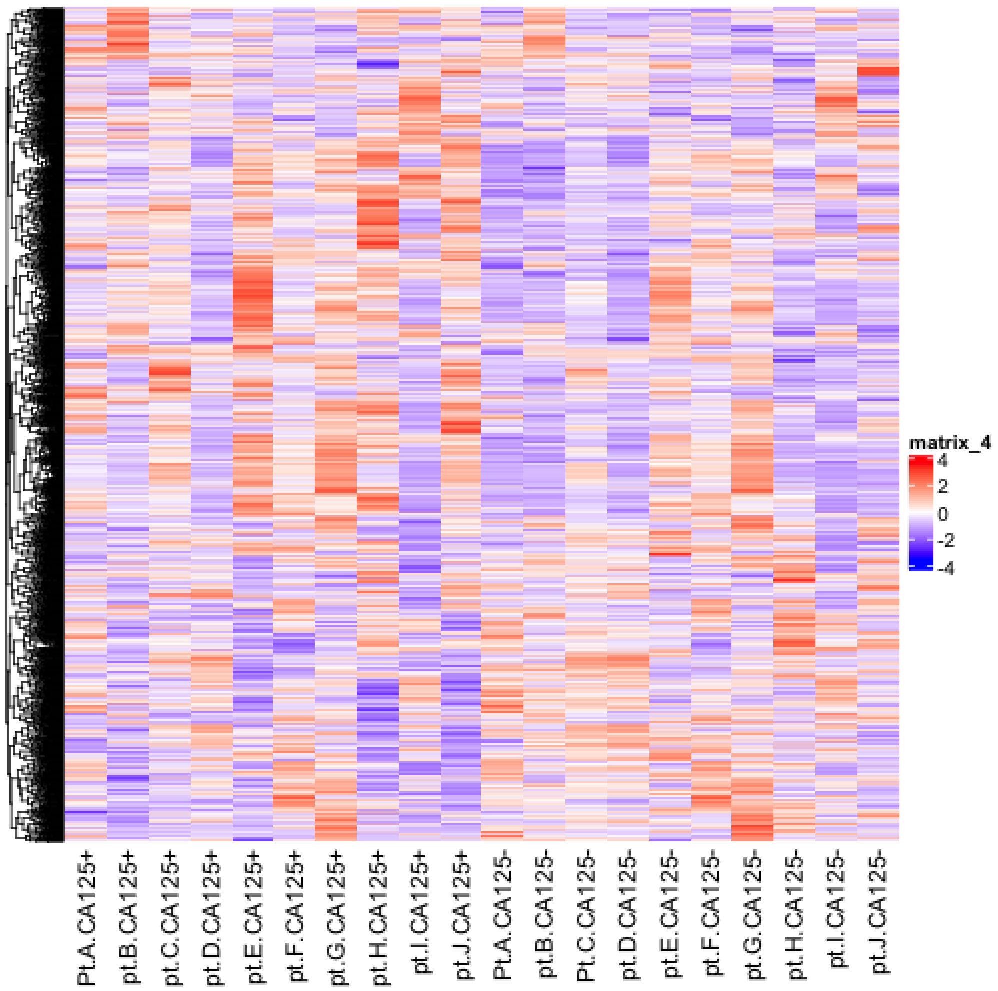
```
top_hits <-
output_hits_pat$ensembl_gene_id[output_hits_pat$P.Value<0.05]
heatmap_matrix_tophits <- t(
  scale(t(heatmap_matrix[
    which(rownames(heatmap_matrix) %in% top_hits),])))

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                           cluster_rows = TRUE,
                           cluster_columns = TRUE,
                               show_row_dend = TRUE,
                               show_column_dend = TRUE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE,
                               )
```

Heatmap of top hits using Limma (accounting for patient variability) -

- p-value < 0.05

```
current_heatmap
```

```
heatmap_matrix_tophits<- heatmap_matrix_tophits[,
                     c(
                     grep(colnames(heatmap_matrix_tophits),pattern
= "\\+"),
grep(colnames(heatmap_matrix_tophits),pattern = "\\-")
                      )]

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                           cluster_rows = TRUE,
                           cluster_columns = FALSE,
                               show_row_dend = TRUE,
                               show_column_dend = TRUE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE,
                               )
```

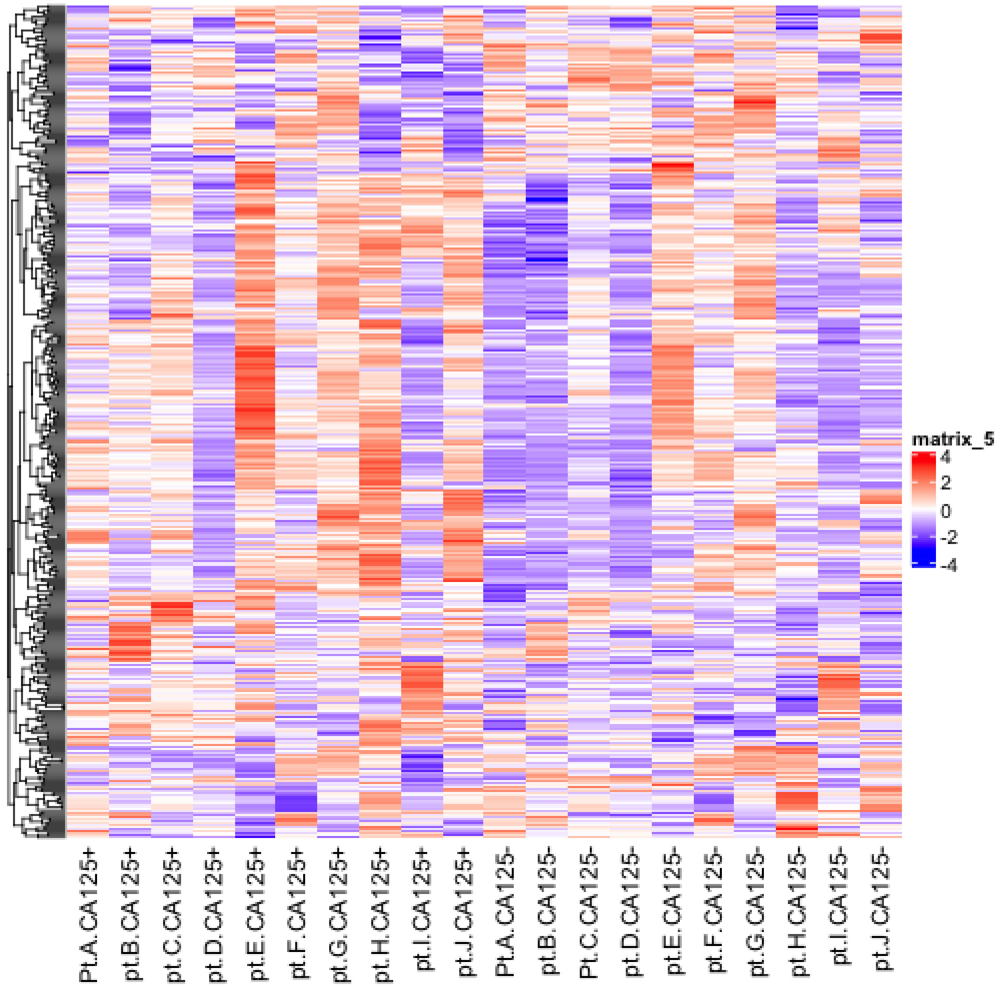

matrix_4
4
2
0
-2
-4
Pt.A.CA125+
pt.B.CA125+
pt.C.CA125+
pt.D.CA125+
pt.E.CA125+
pt.F.CA125+
pt.G.CA125+
pt.H.CA125+
pt.I.CA125+
pt.J.CA125+
Pt.A.CA125-
pt.B.CA125-
Pt.C.CA125-
pt.D.CA125-
pt.E.CA125-
pt.F.CA125-
pt.G.CA125-
pt.H.CA125-
pt.I.CA125-
pt.J.CA125-

Try for a slightly cleaner picture.

```
top_hits <-
output_hits_pat$ensembl_gene_id[output_hits_pat$P.Value<0.01]
heatmap_matrix_tophits <- t(
  scale(t(heatmap_matrix[which(rownames(heatmap_matrix) %in%
top_hits),])))

heatmap_matrix_tophits<- heatmap_matrix_tophits[,
       c(grep(colnames(heatmap_matrix_tophits),pattern = "\\+"),
grep(colnames(heatmap_matrix_tophits),pattern = "\\-"))]

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                  cluster_rows = TRUE,  show_row_dend = TRUE,
                  cluster_columns = FALSE,show_column_dend = FALSE,
                  col=heatmap_col,show_column_names = TRUE,
                  show_row_names = FALSE,show_heatmap_legend =
TRUE)
```

Heatmap of top hits using Limma (accounting for patient variability) -

- p-value < 0.05
- Columns ordered by cell type.



EdgeR

- Analysis package designed for the processing of RNASeq data.
- Interestingly, the Limma guide direct users to use edgeR up to the point of calculating differential expression.
- And limma and edgeR are all written by the same people though...
- There are many different models available in edgeR that can be used for differential expression.
    - exactTest - used for models that only have one factor
    - Quasi liklihood - used for more complicated models and is highly recommended for bulk RNASeq experiments. (glmQLFTest)
    - liklihood ratio test - can be useful for some experiments with limit number of samples or single sample RNA Seq.. (glmLRTest)

Review from last class: Set up our edgeR objects

```
d = DGEList(counts=filtered_data_matrix, group=samples$cell_type)
```

Estimate Dispersion - our model design.

```
d <- estimateDisp(d, model_design_pat)
```

Fit the model

```
fit <- glmQLFit(d, model_design_pat)
```

```r
kable(model_design_pat[1:10,1:5], type="html") %>%
  row_spec(0, angle = -45)
```

(Intercept)	samples$patientsB	samples$patientsC	samples$patientsD	samples$patientsE
1	0	0	0	0
1	0	0	0	0
1	1	0	0	0
1	1	0	0	0
1	0	1	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	1	0
1	0	0	0	1
1	0	0	0	1

Calculate differential expression using the Quasi liklihood model

```
qlf.pos_vs_neg <- glmQLFTest(fit, coef='samples$cell_typeCA125+')
kable(topTags(qlf.pos_vs_neg), type="html")
```

	logFC	logCPM	F	PValue	FDR
ENSG00000182752	2.2775281	2.7763253	35.42360	0.0000425	0.3654997
ENSG00000198804	-0.5969046	13.5106123	34.18749	0.0000507	0.3654997
ENSG00000240864	5.0704267	-0.7698429	55.33012	0.0000571	0.3654997
ENSG00000237973	-0.5913714	11.7910172	30.98434	0.0000818	0.3705585
ENSG00000102755	1.3577028	2.5717883	29.92806	0.0000965	0.3705585
ENSG00000198695	-0.4370418	8.9070099	26.08487	0.0001832	0.5859799
ENSG00000211625	2.9089985	1.8364482	31.99077	0.0002658	0.6229630
ENSG00000188641	1.0969060	3.9514876	23.53251	0.0002909	0.6229630
ENSG00000249119	-0.4797346	6.5308891	23.17139	0.0003114	0.6229630
ENSG00000026508	1.0787591	8.4785833	22.56846	0.0003495	0.6229630

x
BH

x
samples$c

Get all the results

```r
qlf_output_hits <- topTags(qlf.pos_vs_neg,sort.by = "PValue",
                           n = nrow(normalized_count_data))
```

How many gene pass the threshold p-value < 0.05?

```r
length(which(qlf_output_hits$table$PValue < 0.05))
```

```
## [1] 1360
```

How many genes pass correction?

```r
length(which(qlf_output_hits$table$FDR < 0.05))
```

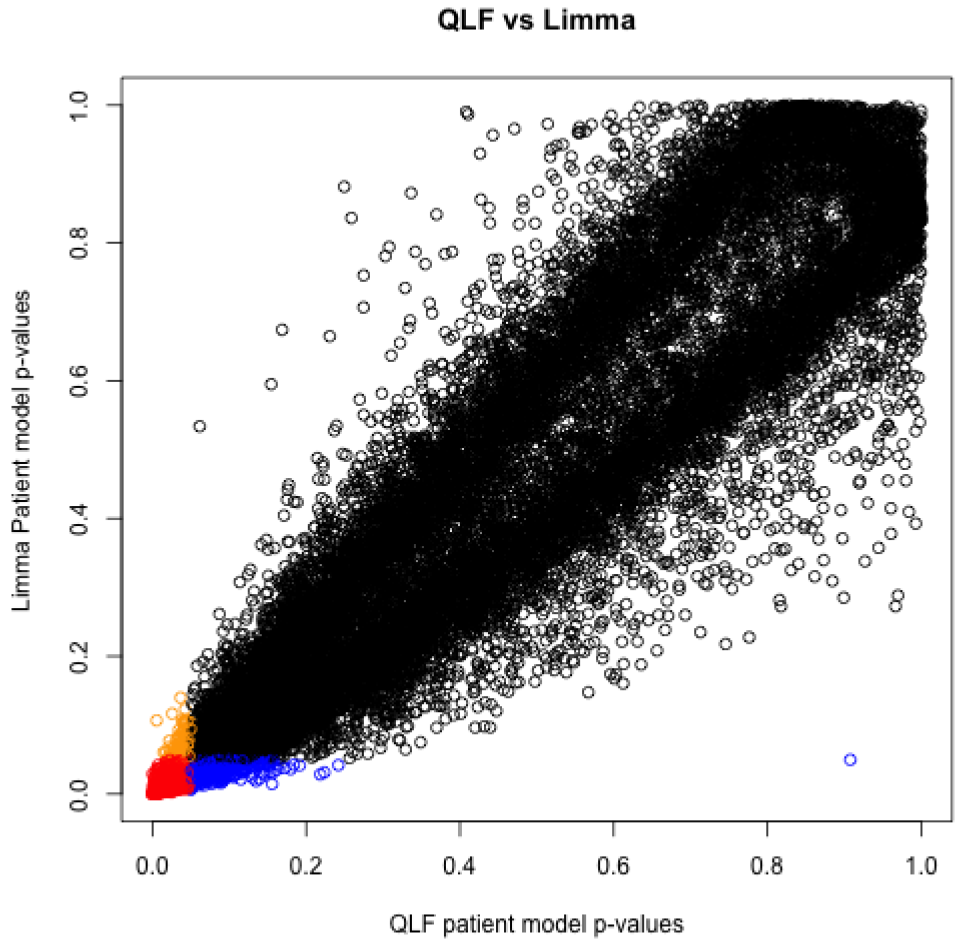
```
## [1] 0
```

Compare the results from the two different models

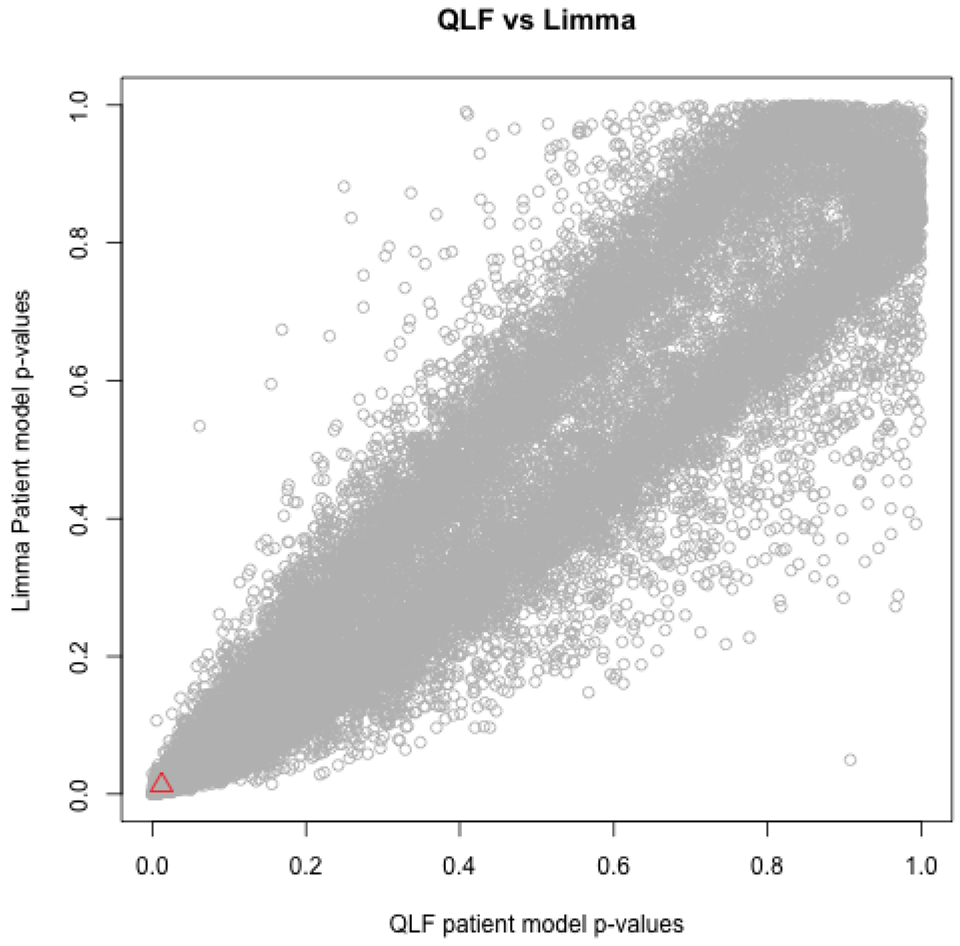
- Limma vs Quasi liklihood

```
qlf_pat_model_pvalues <- data.frame(
          ensembl_id = rownames(qlf_output_hits$table),
          qlf_patient_pvalue=qlf_output_hits$table$PValue)
limma_pat_model_pvalues <-  data.frame(
          ensembl_id = output_hits_pat$ensembl_gene_id,
          limma_patient_pvalue = output_hits_pat$P.Value)
two_models_pvalues <- merge(qlf_pat_model_pvalues,
                            limma_pat_model_pvalues,
                            by.x=1,by.y=1)

two_models_pvalues$colour <- "black"
two_models_pvalues$colour[two_models_pvalues$qlf_patient_pvalue<0.05
 <- "orange"
two_models_pvalues$colour[two_models_pvalues$limma_patient_pvalue<0.
 <- "blue"
two_models_pvalues$colour[two_models_pvalues$qlf_patient_pvalue<0.05
 & two_models_pvalues$limma_patient_pvalue<0.05] <- "red"
```

```
plot(two_models_pvalues$qlf_patient_pvalue,
     two_models_pvalues$limma_patient_pvalue,
     col = two_models_pvalues$colour,
     xlab = "QLF patient model p-values",
     ylab ="Limma Patient model p-values",
     main="QLF vs Limma")
```


QLF vs Limma
Limma Patient model p-values
QLF patient model p-values
0.0
0.2
0.4
0.6
0.8
1.0

```r
ensembl_of_interest <- normalized_count_data$ensembl_gene_id[
  which(normalized_count_data$hgnc_symbol == "MUC16")]

two_models_pvalues$colour <- "grey"
two_models_pvalues$colour[two_models_pvalues$ensembl_id==ensembl_of_i
 <- "red"

plot(two_models_pvalues$qlf_patient_pvalue,
     two_models_pvalues$limma_patient_pvalue,
     col = two_models_pvalues$colour,
     xlab = "QLF patient model p-values",
     ylab ="Limma Patient model p-values",
     main="QLF vs Limma")

points(two_models_pvalues[
  two_models_pvalues$ensembl_id==ensembl_of_interest,2:3],
       pch=24,  col="red", cex=1.5)
```

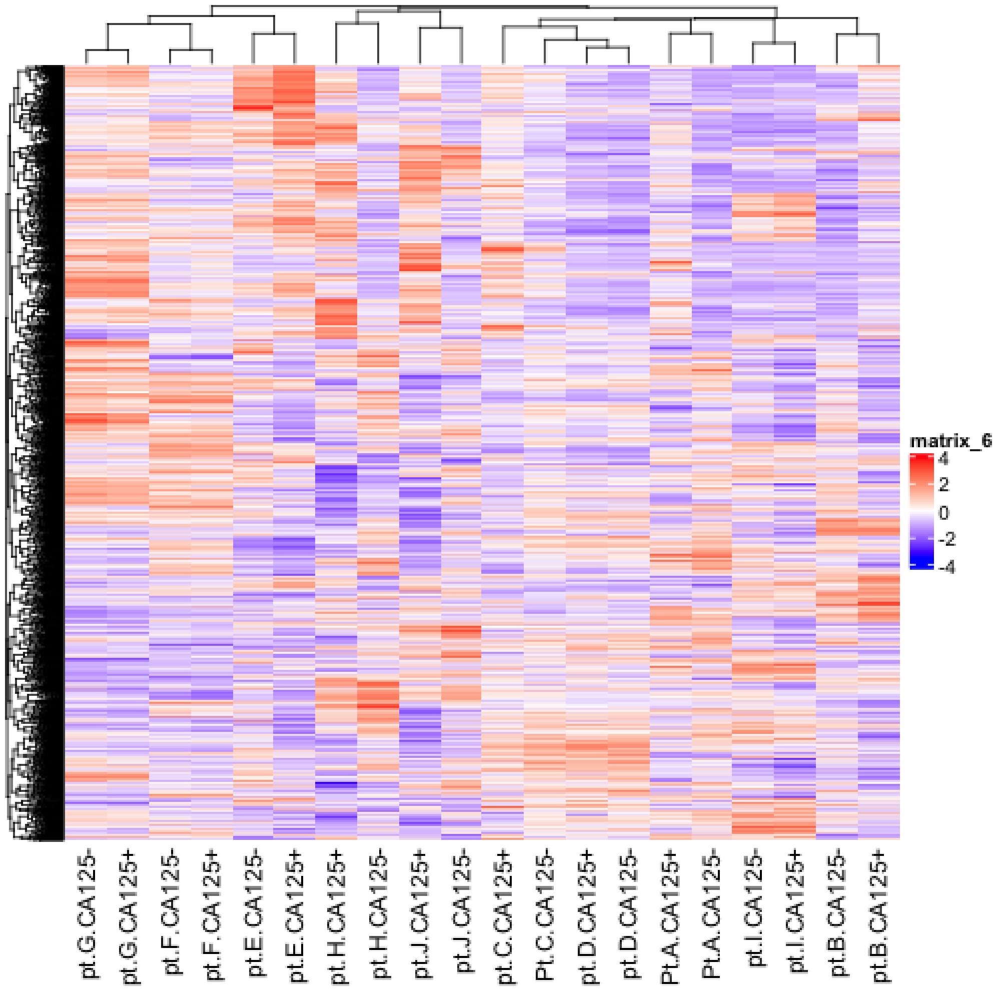

QLF vs Limma
Limma Patient model p-values
QLF patient model p-values
0.0
0.2
0.4
0.6
0.8
1.0

let's come back to the initial heatmap representation of the data

```
top_hits <- rownames(qlf_output_hits$table)
[output_hits_pat$P.Value<0.05]
heatmap_matrix_tophits <- t(
  scale(t(heatmap_matrix[which(rownames(heatmap_matrix) %in%
top_hits),])))

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                           cluster_rows = TRUE,
                           cluster_columns = TRUE,
                               show_row_dend = TRUE,
                               show_column_dend = TRUE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE,
                               )
```

Heatmap of top hits using the Quasi liklihood model (p-value < 0.05)

```
current_heatmap
```



Sort the columns by cell type.

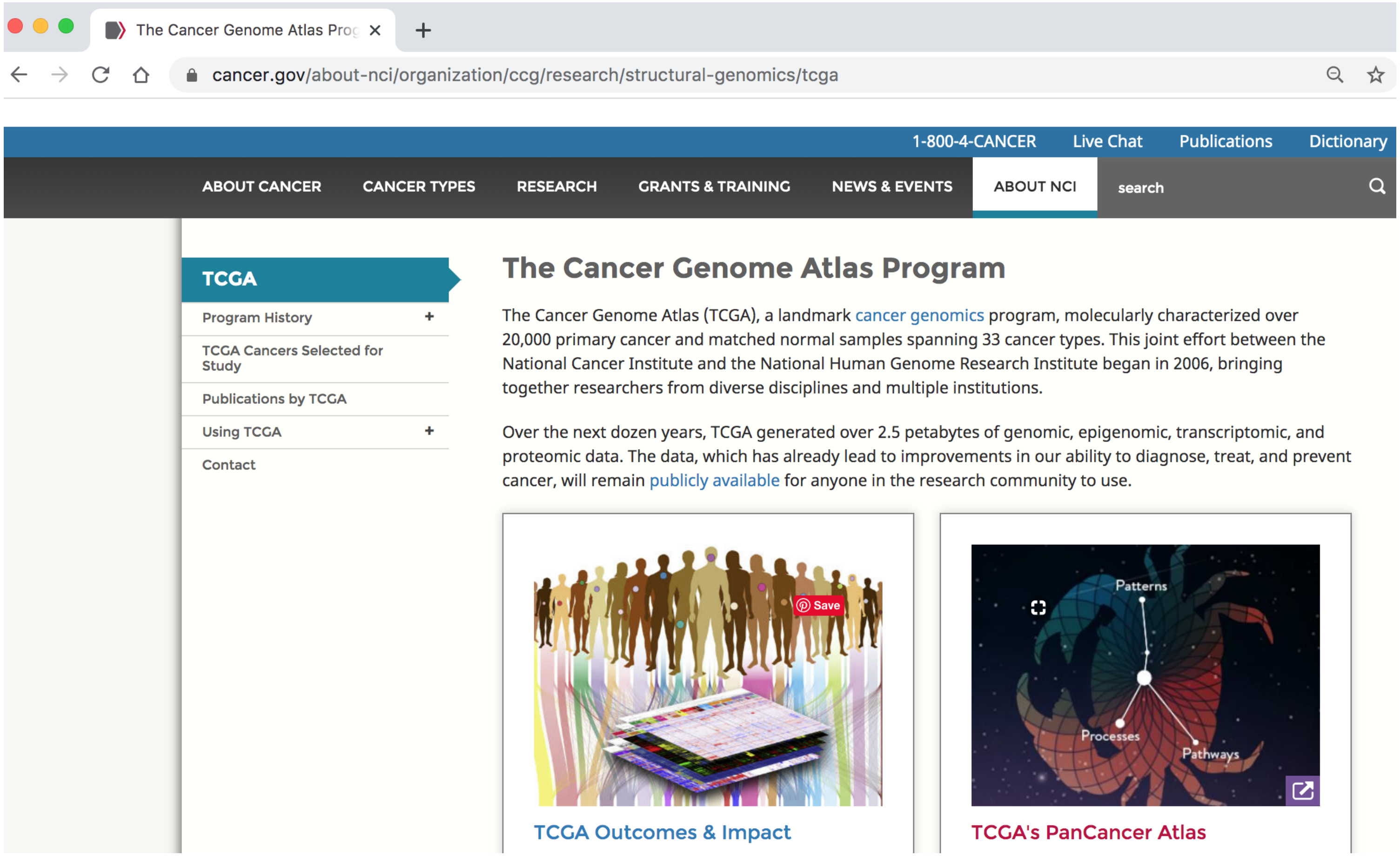
```
top_hits <- rownames(qlf_output_hits$table)
[output_hits_pat$P.Value<0.05]
heatmap_matrix_tophits <- t(
  scale(t(heatmap_matrix[which(rownames(heatmap_matrix) %in%
top_hits),])))

heatmap_matrix_tophits<- heatmap_matrix_tophits[,
       c(grep(colnames(heatmap_matrix_tophits),pattern = "\\+"),
grep(colnames(heatmap_matrix_tophits),pattern = "\\-"))]

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                           cluster_rows = TRUE,
                           cluster_columns = FALSE,
                               show_row_dend = TRUE,
                               show_column_dend = FALSE,
                               col=heatmap_col,
                               show_column_names = TRUE,
                               show_row_names = FALSE,
                               show_heatmap_legend = TRUE,
                               )
```

Heatmap of top hits using the Quasi liklihood model (p-value < 0.05)

- sort columns according to cell type

```
current_heatmap
```

The Cancer Genome Atlas (TCGA)



The Cancer Genome Atlas Prog ×

cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

1-800-4-CANCER | Live Chat | Publications | Dictionary

ABOUT CANCER | CANCER TYPES | RESEARCH | GRANTS & TRAINING | NEWS & EVENTS | ABOUT NCI | search

TCGA

- Program History +
- TCGA Cancers Selected for Study
- Publications by TCGA
- Using TCGA +
- Contact

The Cancer Genome Atlas Program

The Cancer Genome Atlas (TCGA), a landmark cancer genomics program, molecularly characterized over 20,000 primary cancer and matched normal samples spanning 33 cancer types. This joint effort between the National Cancer Institute and the National Human Genome Research Institute began in 2006, bringing together researchers from diverse disciplines and multiple institutions.

Over the next dozen years, TCGA generated over 2.5 petabytes of genomic, epigenomic, transcriptomic, and proteomic data. The data, which has already lead to improvements in our ability to diagnose, treat, and prevent cancer, will remain publicly available for anyone in the research community to use.

TCGA Outcomes & Impact

TCGA's PanCancer Atlas

Get TCGA OV data

```r
library(TCGAbiolinks)
library("SummarizedExperiment")
```

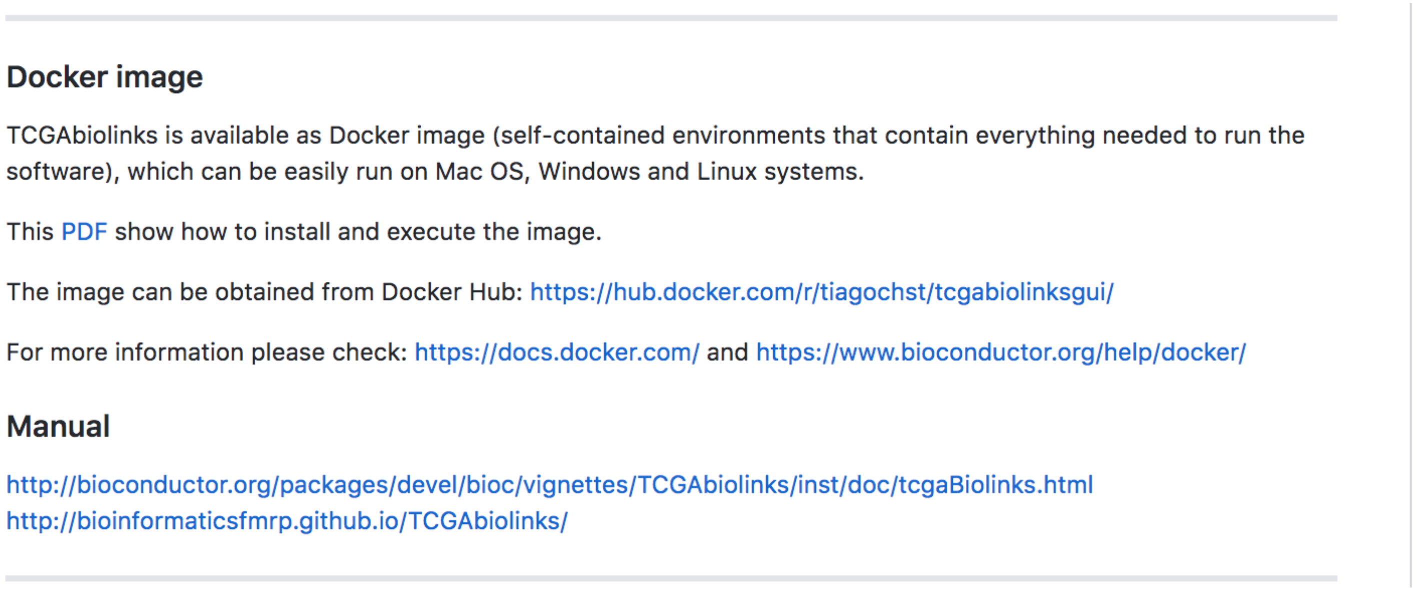
- Get the counts data

```r
query_counts <- GDCquery(project = "TCGA-OV",
                         data.category = "Transcriptome Profiling",
                         data.type = "Gene Expression Quantification",
                         experimental.strategy = "RNA-Seq",
                         workflow.type = "HTSeq - Counts" ,
                         legacy = FALSE)
```

```
## --------------------------------------

## o GDCquery: Searching in GDC database

## --------------------------------------

## Genome of reference: hg38

## --------------------------------------------

## oo Accessing GDC. This might take a while...
```

TCGA Biolinks

TCGABiolink - docker image!



Docker image

TCGAbiolinks is available as Docker image (self-contained environments that contain everything needed to run the software), which can be easily run on Mac OS, Windows and Linux systems.

This PDF show how to install and execute the image.

The image can be obtained from Docker Hub: https://hub.docker.com/r/tiagochst/tcgabiolinksgui/

For more information please check: https://docs.docker.com/ and https://www.bioconductor.org/help/docker/

Manual

http://bioconductor.org/packages/devel/bioc/vignettes/TCGAbiolinks/inst/doc/tcgaBiolinks.html
http://bioinformaticsfmrp.github.io/TCGAbiolinks/

Ovarian Cancer - TCGA data

Of note, Output from the GDCprepare:

- GDCquery: Searching in GDC database
  - Genome of reference: hg38
- Accessing GDC. This might take a while...
- Project: TCGA-OV
- Filtering results
  - By experimental.strategy
  - By data.type
  - By workflow.type
- Checking data
  - Check if there are duplicated cases
  - Check if there results for the query
- Preparing output
  - Downloading data for project TCGA-OV
  - GDCdownload will download "'379'" files. A total of 97.709866 MB
  - Downloading as: Mon_Feb__3_20_24_38_2020.tar.gz

Of note, Output from the GDCprepare: Cont'd

Starting to add information to samples

```
  Add clinical information to samples
```

Add FFPE information. More information at:

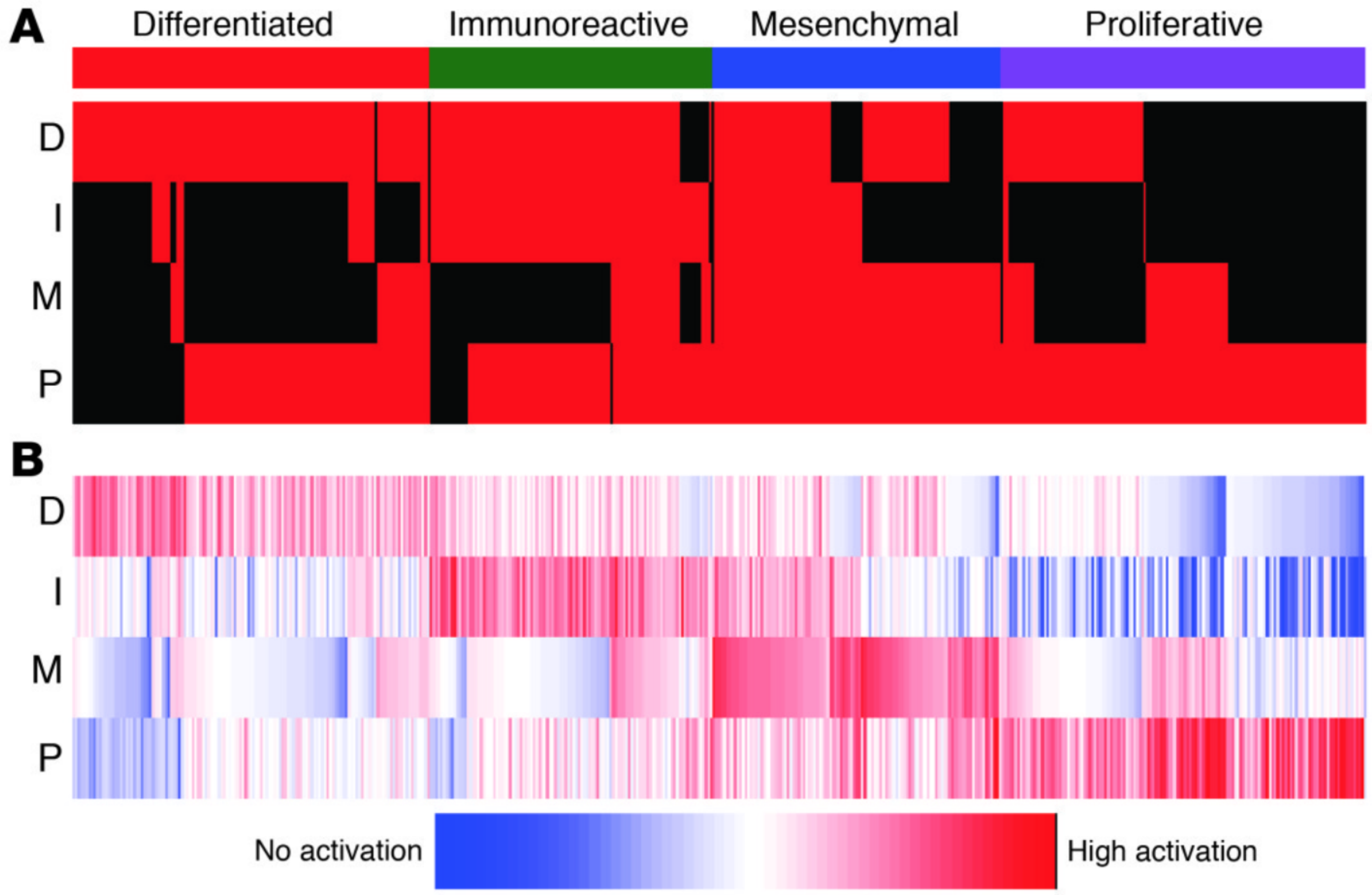
```
  https://cancergenome.nih.gov/cancersselected/biospeccriteria

http://gdac.broadinstitute.org/runs/sampleReports/latest/FPPP_FFPE_Cases.html
  Adding subtype information to samples
```

Accessing www.ensembl.org to get gene information

```
  Downloading genome information (try:0) Using: Human genes
(GRCh38.p13)
```

From the 60483 genes we couldn't map 3984"

Experimental Design?



Load in the predefined classes as described in the Verhaak et al paper.

```
classDefinitions_RNASeq <- read.table(
  file.path("data",
"Supplementary_Table11_RNASeq_classdefinitions.txt"),
  header = TRUE, sep = "\t", quote="\"", stringsAsFactors = FALSE)

tcga.read.counts <- assay(OVRnaseqSE_counts)
colnames(tcga.read.counts) <- gsub(colnames(tcga.read.counts),
                                   pattern = "-",replacement =
"\\.")
```

How many of the samples in the Verhaak paper are in out TCGA data

```
length(which(classDefinitions_RNASeq$patient %in%
colnames(tcga.read.counts)))
```

```
## [1] 257
```

Add the missing samples to the class definitions table

```r
missing_patients <- colnames(tcga.read.counts)[
  which(!colnames(tcga.read.counts) %in%
classDefinitions_RNASeq$patient)]

missing_subtypes <- data.frame(barcode =
substring(missing_patients,1,12),
                               patient =missing_patients ,
                               SUBTYPE = "Not_defined")
classDefinitions_RNASeq <- rbind(classDefinitions_RNASeq,
                                missing_subtypes )
classDefinitions_RNASeq <- classDefinitions_RNASeq[
  which(classDefinitions_RNASeq$patient %in%
colnames(tcga.read.counts)),]

classDefinitions_RNASeq <-
classDefinitions_RNASeq[order(classDefinitions_RNASeq$patient),]
tcga.read.counts <-
tcga.read.counts[,order(colnames(tcga.read.counts))]
```

filter the data

```
cpms <- cpm(tcga.read.counts)
keep <- rowSums(cpms > 1) >= 50
counts <- tcga.read.counts[keep,]
```

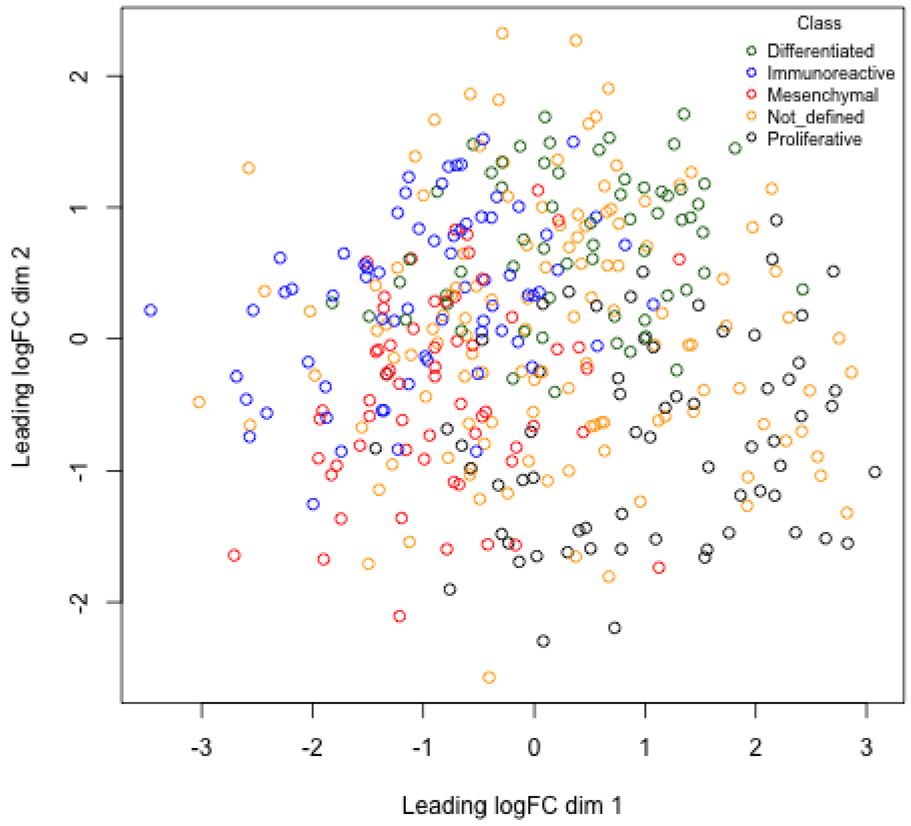
Normalize the data

```
# create data structure to hold counts and subtype information for
each sample.
d <- DGEList(counts=counts, group=classDefinitions_RNASeq$SUBTYPE)

#Normalize the data
d <- calcNormFactors(d)
```

Plot MDS plot of samples

```r
mds_filename <- file.path("data", "mdsplot_allsamples.png")
png(filename = mds_filename)
mds_output <- plotMDS(d, labels=NULL, pch = 1,
col= c("darkgreen","blue","red", "orange","black")
[factor(classDefinitions_RNASeq$SUBTYPE)])


legend("topright",
       legend=levels(factor(classDefinitions_RNASeq$SUBTYPE)),
       pch=c(1), col= c("darkgreen","blue","red",
"orange","black"),
       title="Class",
       bty = 'n', cex = 0.75)

dev.off()
```

```
## quartz_off_screen
##                 2
```


Class
Differentiated
Immunoreactive
Mesenchymal
Not_defined
Proliferative
Leading logFC dim 1
Leading logFC dim 2
-3
-2
-1
0
1
2
3

At this stage I am going to reduce the samples to only the ones we have subtype information for.

```
keep_samples <- classDefinitions_RNASeq$patient[
  classDefinitions_RNASeq$SUBTYPE != "Not_defined"]

classDefinitions_RNASeq <- classDefinitions_RNASeq[
classDefinitions_RNASeq$SUBTYPE != "Not_defined",]
counts <- counts[,keep_samples]
```

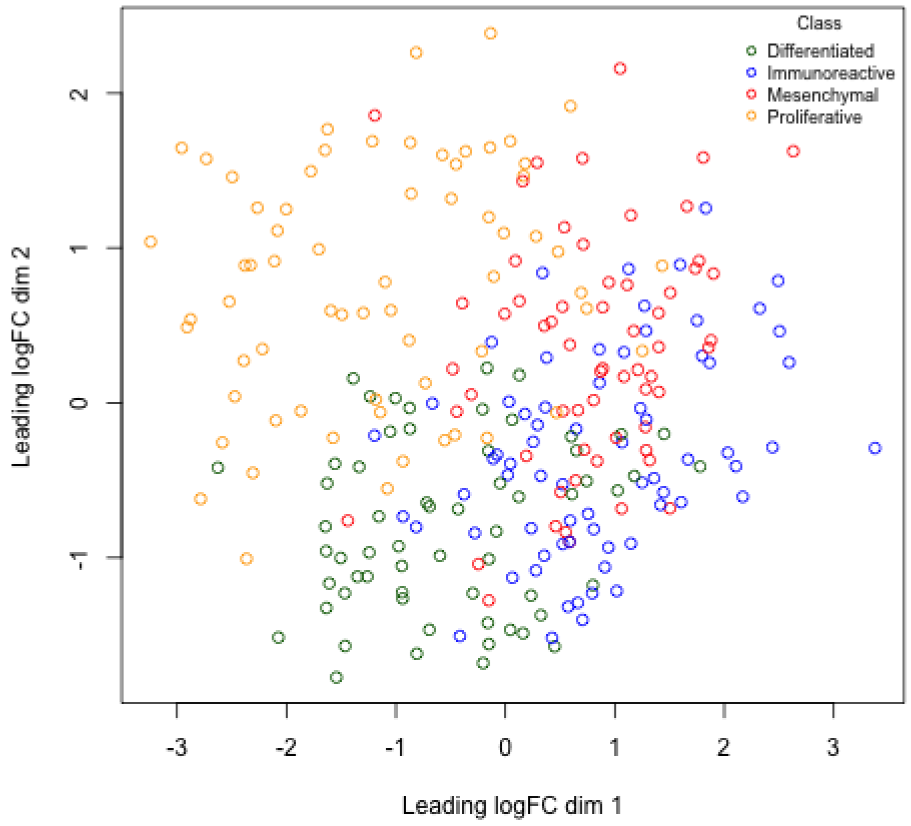
Normalize the data

```
# create data structure to hold counts and subtype information for
each sample.
d <- DGEList(counts=counts, group=classDefinitions_RNASeq$SUBTYPE)

#Normalize the data
d <- calcNormFactors(d)
```

Plot MDS plot of samples

```r
mds_filename <- file.path("data",
"mdsplot_allsamples_minus_undef.png")
png(filename = mds_filename)
mds_output <- plotMDS(d, labels=NULL, pch = 1,
col= c("darkgreen","blue","red", "orange","black")
[factor(classDefinitions_RNASeq$SUBTYPE)])


legend("topright",
       legend=levels(factor(classDefinitions_RNASeq$SUBTYPE)),
       pch=c(1), col= c("darkgreen","blue","red", "orange"),
       title="Class",
       bty = 'n', cex = 0.75)

dev.off()
```

```
## quartz_off_screen
##                 2
```


Class
Differentiated
Immunoreactive
Mesenchymal
Proliferative
Leading logFC dim 1
Leading logFC dim 2
-3
-2
-1
0
1
2
3
-1
0
1
2

Define the model matrix

```
classes <- classDefinitions_RNASeq$SUBTYPE
model_design_tcga <- model.matrix(~ 0 + classes)
model_design_tcga[1:5,1:4]
```

```
##   classesDifferentiated classesImmunoreactive classesMesenchymal
## 1                     0                     1                  0
## 2                     1                     0                  0
## 3                     0                     0                  0
## 4                     0                     1                  0
## 5                     0                     0                  0
##   classesProliferative
## 1                    0
## 2                    0
## 3                    1
## 4                    0
## 5                    1
```

Calculate dispersion

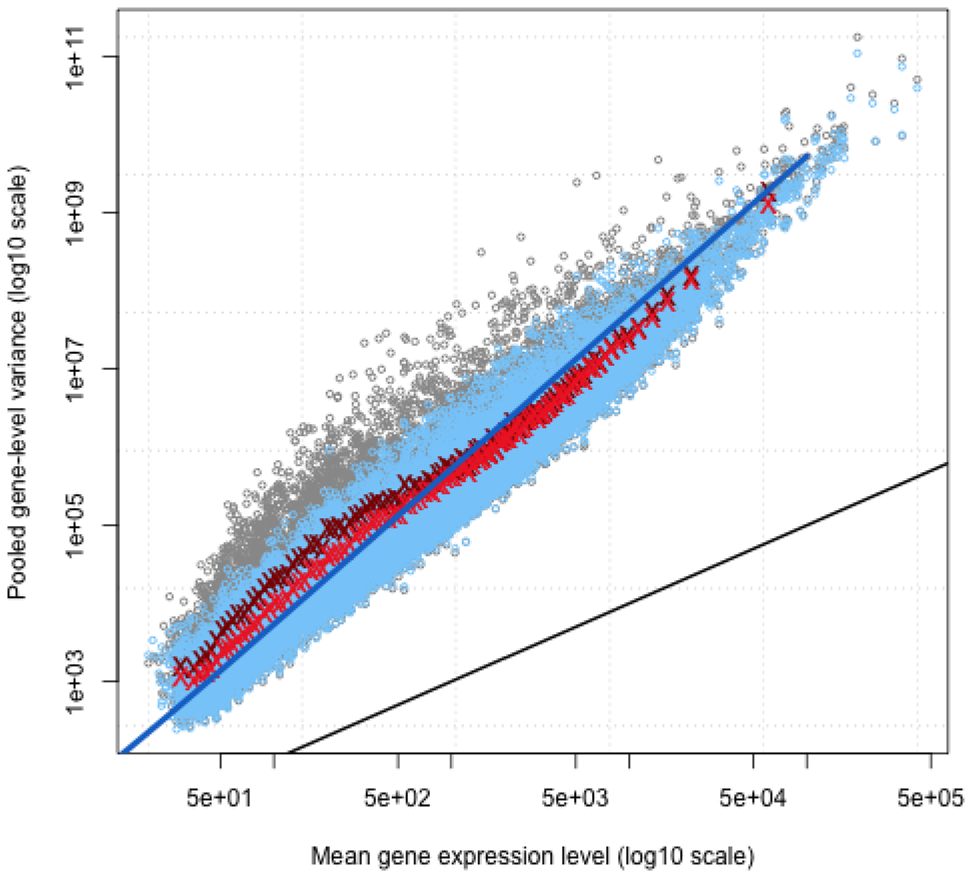
```
d <- estimateDisp(d,model_design_tcga)
```

Graphing the BCV

- tagwise = genewise, each dot represents the BCV for a gene

```
plotBCV(d,col.tagwise = "black",col.common = "red")
```

```
plotMeanVar(d, show.raw.vars = TRUE, show.tagwise.vars=TRUE,
            show.ave.raw.vars = TRUE,
            NBline=TRUE,
            show.binned.common.disp.vars = TRUE)
```



Calculate differential expression

- We have a lot of different option to look at here - Immuno subtype vs mesenchymal

```
contrast_mesenvsimmuno <- makeContrasts(
                  mesenvsimmuno ="classesMesenchymal-
classesImmunoreactive",
                  levels=model_design_tcga)

contrast_mesenvsimmuno
```

```
##                          Contrasts
## Levels                     mesenvsimmuno
##   classesDifferentiated               0
##   classesImmunoreactive              -1
##   classesMesenchymal                  1
##   classesProliferative                0
```

```
fit_qlf_tcga <-glmQLFit (d,model_design_tcga)

qlf.immuno_vs_mesechymal <- glmQLFTest(fit_qlf_tcga,

contrast=contrast_mesenvsimmuno)

tt_mesenvsimmuno <- topTags(qlf.immuno_vs_mesechymal,n=nrow(d))
```

top hits

```r
#merge in the gene names first
ovRNASeq_gene_countsdata <- rowData(OVRnaseqSE_counts)

data_display <- merge(ovRNASeq_gene_countsdata[,1:2],
                      topTags(qlf.immuno_vs_mesechymal),
                      by.x=1, by.y = 0)

kable(data_display, type="html")
```

ensembl_gene_id	external_gene_name	logFC	logCPM	F	PValue	FDR
ENSG00000084636	COL16A1	1.830897	5.407136	120.0345	0	0
ENSG00000103196	CRISPLD2	2.182952	5.473646	108.7335	0	0
ENSG00000106624	AEBP1	2.086450	8.644920	115.0045	0	0
ENSG00000113140	SPARC	1.694264	10.538137	108.9378	0	0
ENSG00000133466	C1QTNF6	1.572266	4.956225	107.9188	0	0
ENSG00000136859	ANGPTL2	1.539721	5.191807	122.2606	0	0
ENSG00000166147	FBN1	2.116491	6.015916	121.9697	0	0
ENSG00000169604	ANTXR1	1.664972	7.412542	135.3873	0	0
ENSG00000174498	IGDCC3	5.822141	2.501262	123.3080	0	0
ENSG00000182492	BGN	1.790193	8.989584	107.8897	0	0

```
kable(data_display, type="html",digits = 32)
```

ensembl_gene_id	external_gene_name	logFC	logCPM	F	PValue	FD
ENSG00000084636	COL16A1	1.830897	5.407136	120.0345	3.545814e-23	1.359891e 19
ENSG00000103196	CRISPLD2	2.182952	5.473646	108.7335	1.832311e-21	4.392049e 18
ENSG00000106624	AEBP1	2.086450	8.644920	115.0045	2.021539e-22	6.460840e 19
ENSG00000113140	SPARC	1.694264	10.538137	108.9378	1.704300e-21	4.392049e 18
ENSG00000133466	C1QTNF6	1.572266	4.956225	107.9188	2.446857e-21	4.491265e 18
ENSG00000136859	ANGPTL2	1.539721	5.191807	122.2606	1.653666e-23	8.756399e 20
ENSG00000166147	FBN1	2.116491	6.015916	121.9697	1.826533e-23	8.756399e 20
ENSG00000169604	ANTXR1	1.664972	7.412542	135.3873	2.016424e-25	3.866694e 2
ENSG00000174498	IGDCC3	5.822141	2.501261	123.3080	1.156833e-23	8.756399e 20

How many genes have p-values less than 0.05

```
length(which(tt_mesenvsimmuno$table$PValue<0.05))
```

```
## [1] 7521
```

How many genes pass correction?

```
length(which(tt_mesenvsimmuno$table$FDR < 0.05))
```

```
## [1] 5559
```

Try a different comparison:

- compare immuno to the rest of the samples (not undefined though)

```
contrast_immuno <- makeContrasts(
  immunovsrest ="classesImmunoreactive-(classesMesenchymal +
  classesProliferative +classesDifferentiated)/3",
  levels=model_design_tcga)

qlf.immuno_vs_all <- glmQLFTest(fit_qlf_tcga,
                                       contrast=contrast_immuno)

tt_immunovsall <- topTags(qlf.immuno_vs_all,n=nrow(d))
```

How many genes have p-values less than 0.05

```
length(which(tt_immunovsall$table$PValue<0.05))
```

```
## [1] 8178
```

How many genes pass correction?

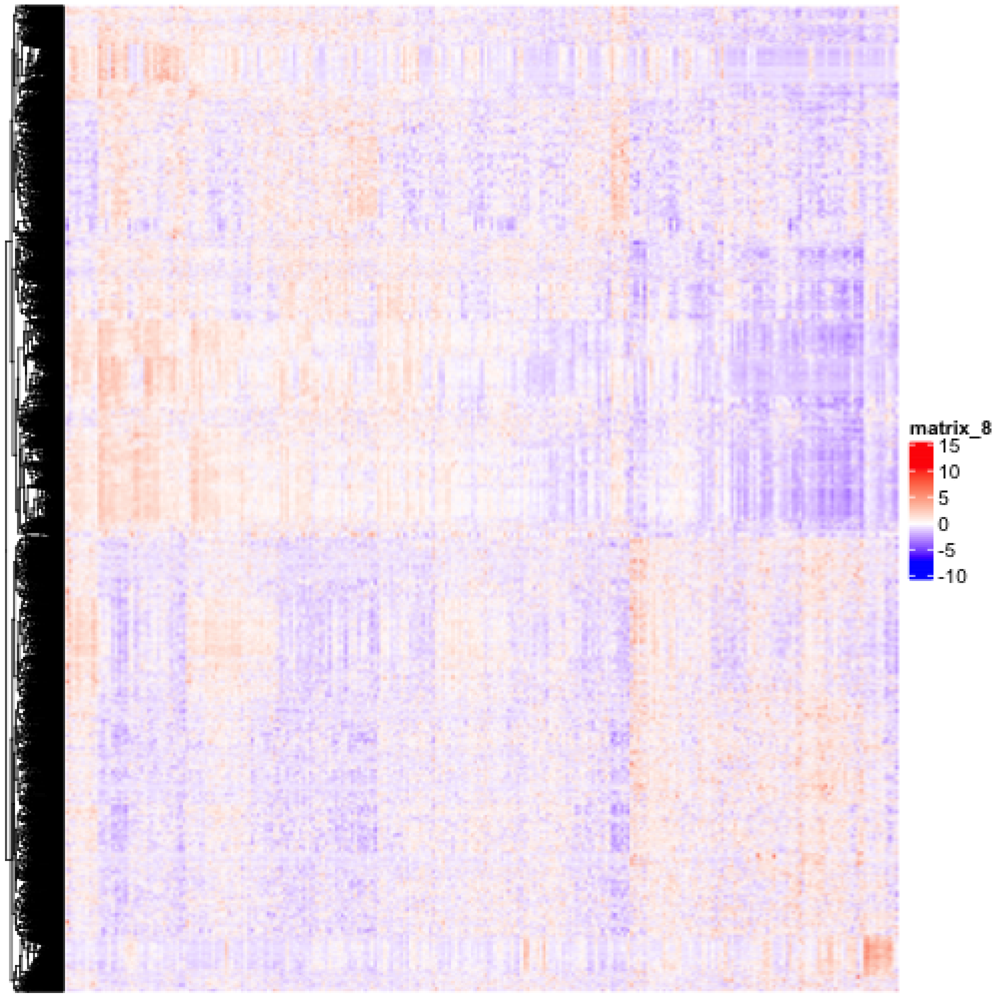
```
length(which(tt_immunovsall$table$FDR < 0.05))
```

```
## [1] 6258
```

Visualize this data set

```
# get the normalized counts
tcga_normalized_counts <- log2(cpm(d) +1)

#create the scaled heatmap object
heatmap_matrix <- tcga_normalized_counts

top_hits <- rownames(tt_immunovsall)[which(tt_immunovsall$table$FDR
< 0.001)]
heatmap_matrix_tophits <- t(
  scale(t(heatmap_matrix[which(rownames(heatmap_matrix) %in%
top_hits),])))

if(min(heatmap_matrix_tophits) == 0){
    heatmap_col = colorRamp2(c( 0, max(heatmap_matrix_tophits)),
                             c( "white", "red"))
  } else {
    heatmap_col = colorRamp2(c(min(heatmap_matrix_tophits), 0,
max(heatmap_matrix_tophits)), c("blue", "white", "red"))
  }

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                  cluster_rows = TRUE,   show_row_dend = TRUE,
                  cluster_columns = TRUE,show_column_dend = FALSE,
                  col=heatmap_col,show_column_names = FALSE,
                  show_row_names = FALSE,show_heatmap_legend =
TRUE)
```

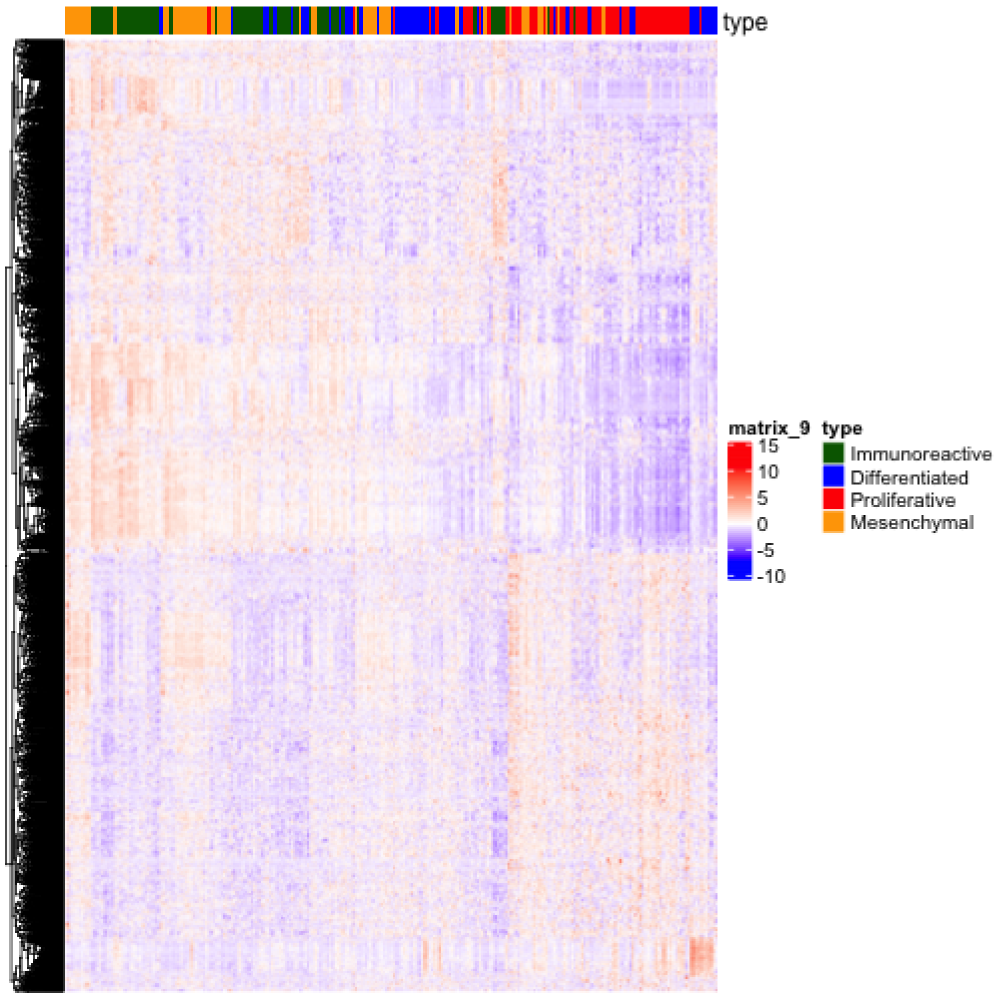
Immuno vs Rest heatmap



Annotating the heatmap could really help here.

```
ha_colours <- c("darkgreen","blue","red", "orange")
names(ha_colours) <- unique(classDefinitions_RNASeq$SUBTYPE)

ha <- HeatmapAnnotation(df=data.frame(
  type = classDefinitions_RNASeq$SUBTYPE),
  col =  list(type = ha_colours))

current_heatmap <- Heatmap(as.matrix(heatmap_matrix_tophits),
                  cluster_rows = TRUE,  show_row_dend = TRUE,
                  cluster_columns = TRUE,show_column_dend = FALSE,
                  col=heatmap_col,show_column_names = FALSE,
                  show_row_names = FALSE,show_heatmap_legend =
TRUE,
                  top_annotation = ha)
```

Immuno vs Rest heatmap - annotated



Homework for next week

Next week we will be looking at "What do we do with all these hits?"

- Find an annotation data set (excluding GO and Reactome) for human genes.
- Any data set that adds functional, process or location data to a set of genes.
- Record in your journal and add it to the list of annotation sources on the Student Wiki:
  - When was it published? Was is published?
  - How is it released? What identifiers does it use?
  - What sort of information does it offer us?