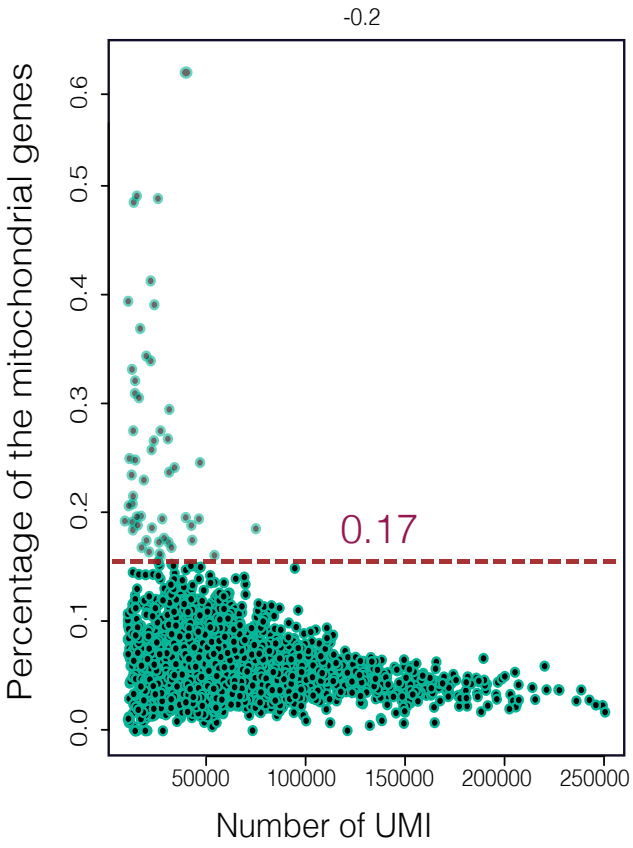
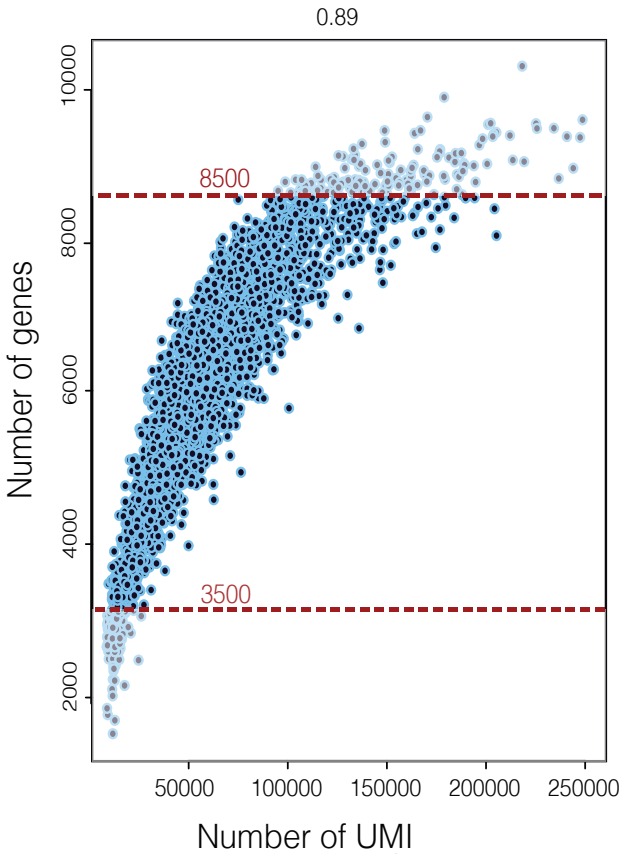
Exploring the single cell data with Seurat package

```
library(Seurat)
library(dplyr)
#loading in the UMI matrices generated by CellRanger (10xGenomics
software)
agg.reseq.data <- Read10X(data.dir = "~/Desktop/out_data/
filtered_gene_bc_matrices_2_reseq/h19/")
agg.reseq <- CreateSeuratObject(raw.data = agg.reseq.data, min.cells = 10,
min.genes = 1000)

#Naming the files according to the cell culture they were treated in.
Each cell id has a number that corresponds to the order the samples were
loaded into the sequencer.
agg.reseq@meta.data[grepl("-1$", rownames(agg.reseq@meta.data)), "sample"]
<- "Naive"
agg.reseq@meta.data[grepl("-2$", rownames(agg.reseq@meta.data)), "sample"]
<- "Cisplatin"
agg.reseq@meta.data[grepl("-3$", rownames(agg.reseq@meta.data)), "sample"]
<- "JQ1andCisplatin"
agg.reseq@meta.data[grepl("-4$", rownames(agg.reseq@meta.data)), "sample"]
<- "Resistant"

###Quality control analysis ###
#finding mitochondrial genes
mito.genes <- grep(pattern = "^MT-", x = rownames(x = agg.reseq@data), value
= TRUE)
#find the ratio of mt genes to the rest of the genes from raw data
percent.mito <- Matrix::colSums(agg.reseq@raw.data[mito.genes, ]) /
Matrix::colSums(agg.reseq@raw.data)
#Adding the colomn with the percentages to the meta object
agg.reseq <- AddMetaData(object = agg.reseq, metadata = percent.mito,
col.name = "percent.mito")
#making single cell plots to visualize amount of mitochondrial genes in the
samples per total UMI (unique molecular identifiers)

GenePlot(object = agg.reseq, gene1 = "nUMI", gene2 = "percent.mito")
GenePlot(object = agg.reseq, gene1 = "nUMI", gene2 = "nGene")


#analysing and removing the outlier cells with too high/low mitochondrial
genome content because they are potential cell duplets.
#I will also filter out cells with the too high number of genes. The
thresholds were chosen according to the graphs below.

agg.reseq <- FilterCells(object = agg.reseq, subset.names = c("nGene",
 "percent.mito"), low.thresholds = c(3500,0.0), high.thresholds = c(8500,
 0.17))
```



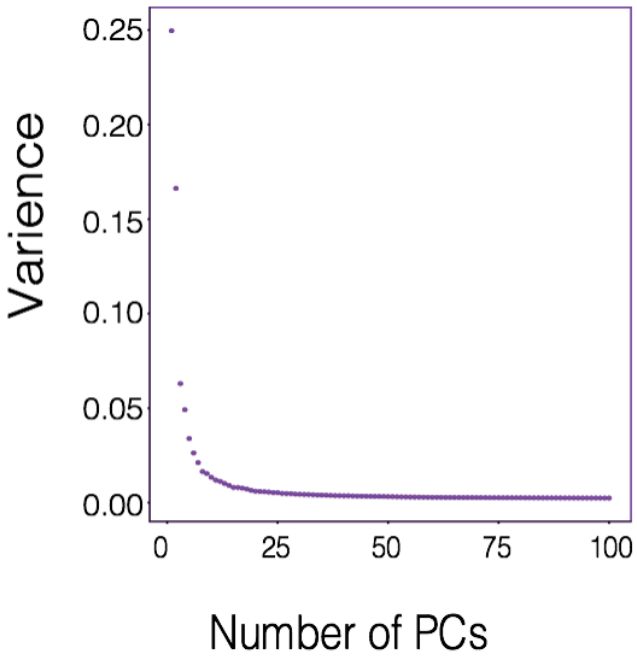



```
# About 150 low-quality cell were removed. There was no bias to cells from a
particular type of samples. Therefore, about the same ratios to the total from
each group got removed.
```

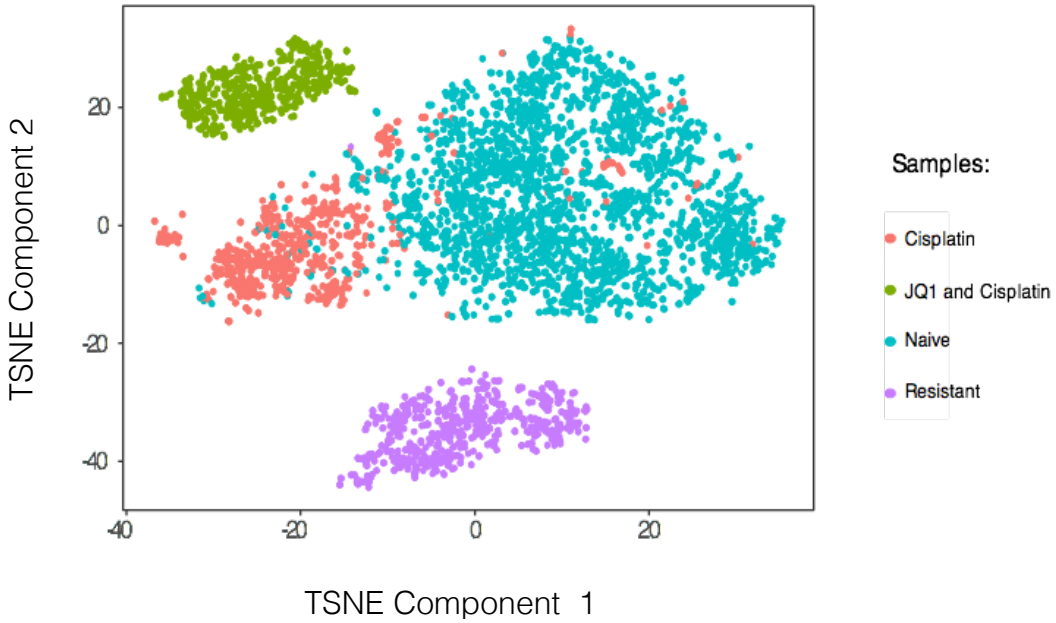
Building single celll trajectories with the Monocle Package [Trapnell Lab] :

```
library(monocle)
# Importing the file from previous Seurat analysis (CellDataSet- CDS).
Trajectory_analysis <- importCDS(agg.reseq)
#Calculating size factors to normalize for differences in mRNA across cells,
 and dispersion to perform differential expression analysis later.
Trajectory_analysis <- estimateSizeFactors(Trajectory_analysis)
Trajectory_analysis <- estimateDispersions(Trajectory_analysis)
#121 outlier cells were removed at this step
#Filtering out cells with low mean expression values (either dead cells or
 empty wells)
Trajectory_analysis <- detectGenes(Trajectory_analysis, min_expr = 0.1)
#at least 10 genes should be detected per each cell - lower number would mean a t
expressed_genes <- row.names(subset(fData(Trajectory_analysis),
 num_cells_expressed >= 10))
Trajectory_analysis <- Trajectory_analysis[expressed_genes, ]
```

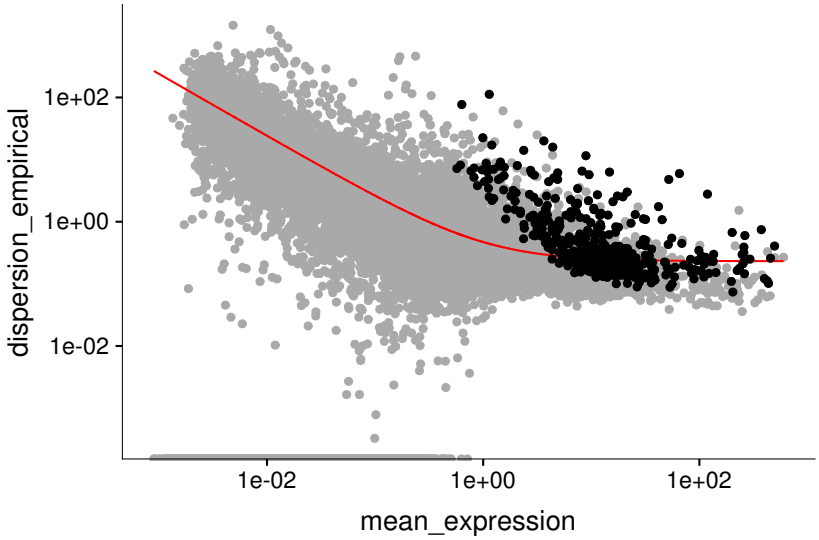
```
fData(Trajectory_analysis)$use_for_ordering <- fData(Trajectory_analysis)
 $num_cells_expressed > 0.05 * ncol(Trajectory_analysis)

#this is how to decide amount of PCs I should apply:
plot_pc_variance_explained(Trajectory_analysis, return_all = FALSE)
```



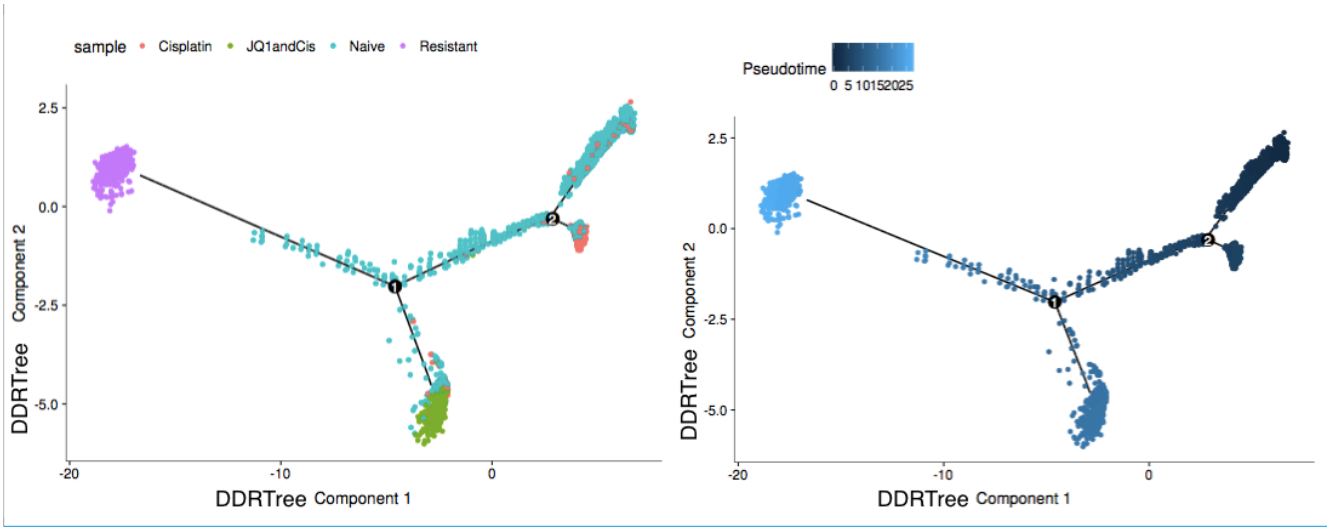

```
#reduce dimensions according to previous plot
Trajectory_analysis <- reduceDimension(Trajectory_analysis,
                                       max_components = 2,
                                       num_dim = 10,
                                       reduction_method = 'tSNE',
                                       verbose = TRUE)
#now I cluster the cells.
Trajectory_analysis <- clusterCells(Trajectory_analysis,color
 ="sample",num_clusters = 3, verbose = FALSE)
plot_cell_clusters(Trajectory_analysis,color="sample")
```



```
clustering_DEG_genes <- differentialGeneTest(Trajectory_analysis,
                        fullModelFormulaStr = '~sample',
                        cores = 8) #well, my processor has only 2 cores but I
was an optimist:D

#finding genes with lowest q -> (false positive discovery rate):
clustering_DEG_genes %>% arrange(qval) %>% head()
#Choosing only top 400 most interesting genes
my_ordering_genes2 <- row.names(clustering_DEG_genes2)
[order(clustering_DEG_genes$qval)][1:400]
Trajectory_analysis<- setOrderingFilter(Trajectory_analysis, ordering_genes
= my_ordering_genes2)
plot_ordering_genes(Trajectory_analysis)
```

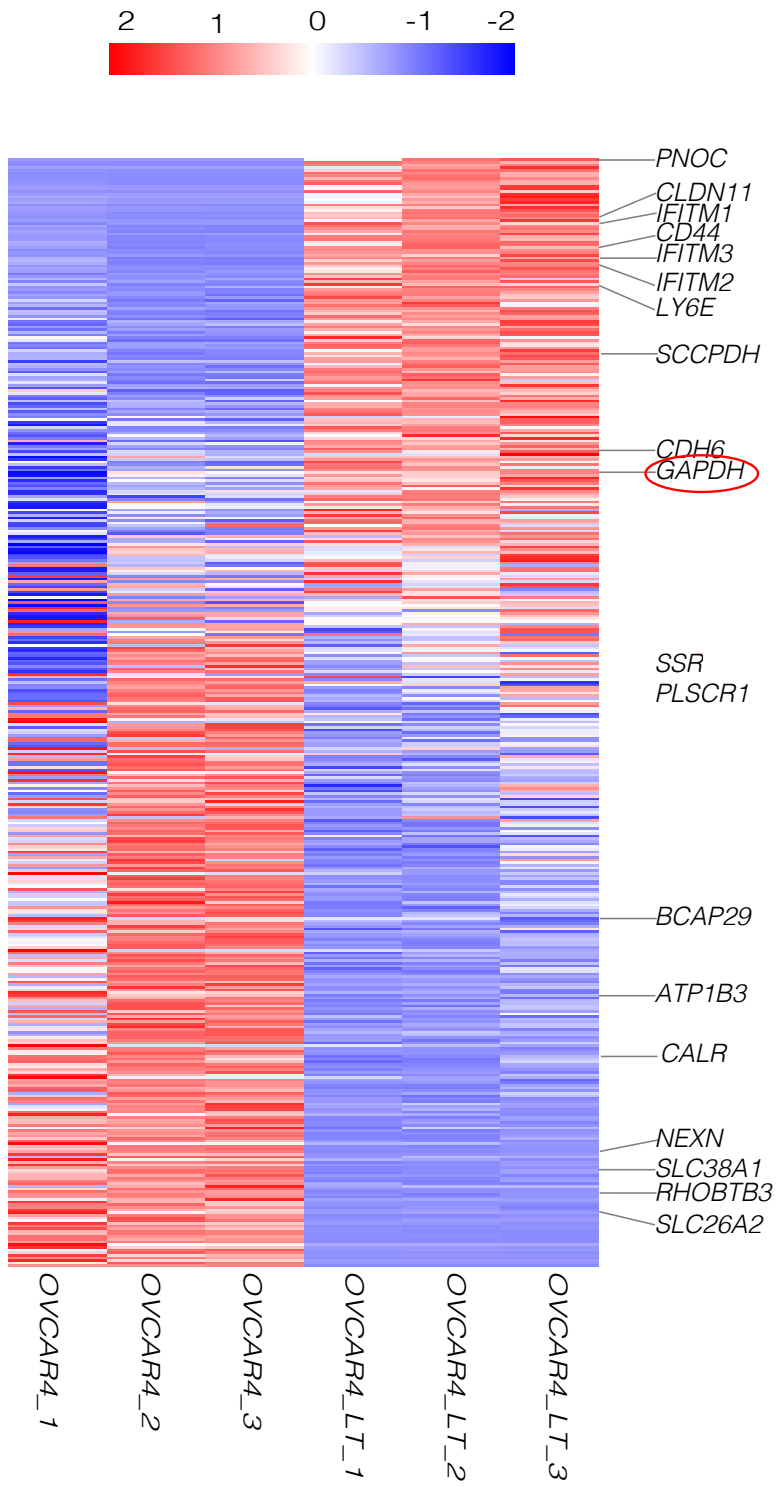



```
#after getting reassured by the distribution on the plot I can do the trajectories

Trajectory_analysis <- reduceDimension(Trajectory_analysis, max_components =
2, method = 'DDRTree') #DDRTree may have only 2 components; for 3d plots
UMAP should be used. 'Pseudotime' is a synonym to biological differentiation time
Trajectory_analysis <- orderCells(Trajectory_analysis)
plot_cell_trajectory(Trajectory_analysis, color_by = "sample")
plot_cell_trajectory(Trajectory_analysis, color_by = "Pseudotime")
```



Defining the reference genes for the OVCAV4 cell line.

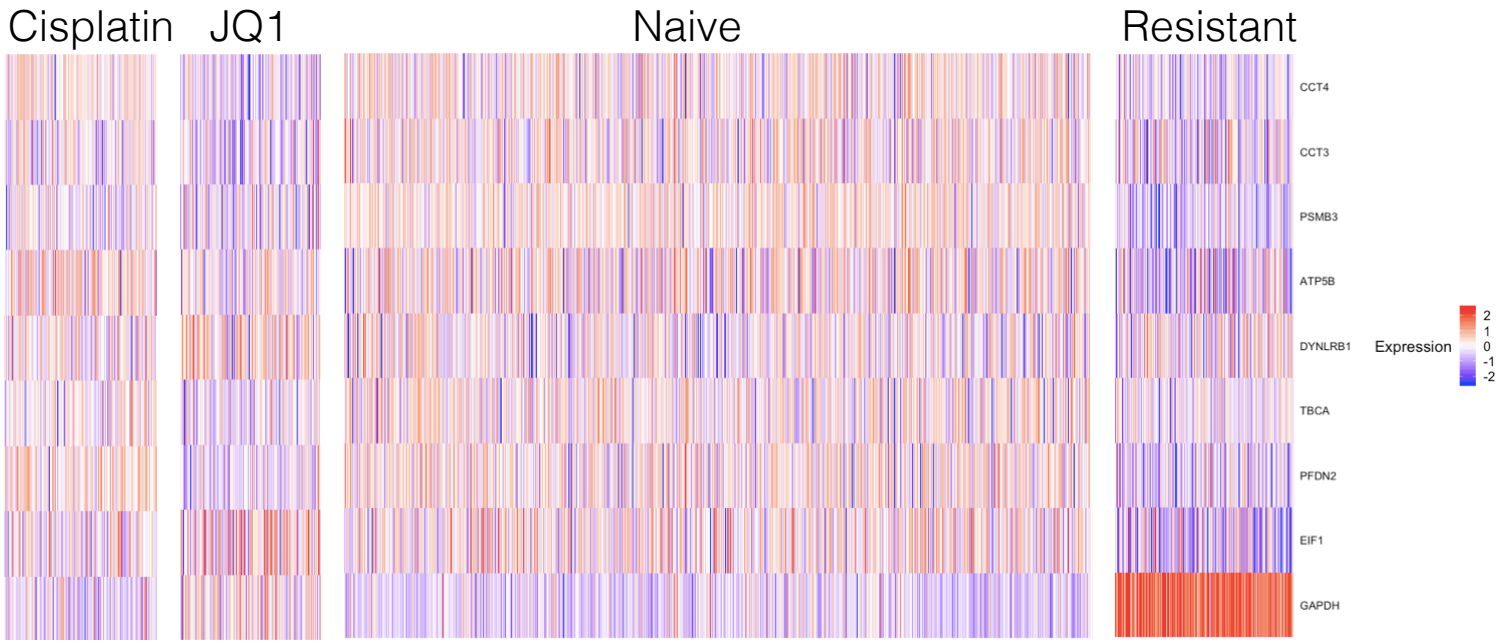
Later I would use PANTHER to classify the genes for their cellular location. I was specifically interested in the genes that were found on the plasma membrane. As I was plotting the genes on my heatmap I have noticed presence of the GAPDH and several other typical housekeeping genes among the differentially expressed genes. For GAPDH this could be explained by Warburg effect. The problem is that since those genes are frequently used as references for qPCR analysis it could make the results biased against the experimenter.



My new goal was to find reliable genes for the OVCAR4 cell line since it is one of the cell lines Adli lab frequently used. For this reason I have downloaded the TKO database of the core fitness genes and analyzed their expression in the data. (https://www.sciencedirect.com/science/article/pii/S0092867415014956)

```
#only 828 genes from the TKO core database were chosen (using only genes
expressed in all cell lines):some of the genes (n=5) were not found in the scRNA-
seq data data
#selective criteria for the 828 genes:
TKOgenes <- read_excel("~/loadTKOgenes.xlsx")
length(row.names(TKOgenes))
[1] 828
formatedGenes<-as.data.frame(TKOgenes)
#Using the Seurat function want to get the average the expression within the
samples.
formatedGenes_scaled<-AverageExpression(agg.reseq1,use.scale = TRUE, genes.use =
formatedGenes,add.ident = "sample")
formatedGenes_not_scaled <-AverageExpression(agg.reseq1, genes.use =
formatedGenes,add.ident = "sample")


#I selected the genes that were expressed equally in most samples (with z-score
between +/- 0.5) in the scaled data.
a<-row.names(formatedGenes_scaled[apply(abs(formatedGenes_scaled),1,max)<0.5, ])

#Then I chose the genes that are constitutively expressed in all samples by
checking the mean expression values. I choose this optimal threshold after
several attempts to apply the following functions.It seems puzzling to find a
trade-off between high expression and equal distribution of those genes both
within and between the clusters.
b<-row.names(formatedGenes_not_scaled[apply(abs(formatedGenes_not_scaled),
1,max)>3, ])

#genes that are selected for both criteria will overlap:
#library(GeneOverlap)
go.obj<-newGeneOverlap(a,b)
go.obj <-testGeneOverlap(go.obj)
go.obj@intersection

# Eventually Out of 828 genes in TKO database, I saw 8 genes that could be the
best fits for the reference genes.
intersections<-AverageExpression(agg.reseq1,use.scale = TRUE, genes.use
=c(go.obj@intersection,"GAPDH"),add.ident = "sample")

# Now I would want to see if the genes are expressed equally not just by several
ones in the sample. The average expression is something I could get just with the
bulk RNA sequencing so I will make a heatmap that will visualize the expression
in each cell.

DoHeatmap(agg.reseq1, data.use = NULL, use.scaled = TRUE, cells.use =
NULL,genes.use = row.names(intersections), disp.min = -2.5, disp.max = 2.5,
group.by = "sample",
      group.order = NULL, draw.line = TRUE, col.low = "blue",
      col.mid = "white", col.high = "red", slim.col.label = FALSE,
      remove.key = FALSE, rotate.key = FALSE, title = NULL, cex.col = 10,
      cex.row = 10, group.label.loc = "bottom", group.label.rot = FALSE,
      group.cex = 15, group.spacing = 0.15, assay.type = "RNA",
```

```
      do.plot = TRUE)
```




```
#Here is also a ridge plot that shows their relative expressions in our data and
you can also see the expression of GAPDH gene for the comparison.

RidgePlot(agg.reseq,features.plot = row.names(intersections), return.plotlist =
TRUE, group.by = "sample")
```