Bladder cohort - R Notebook

First install BiocManager, edgeR, and TCGAbiolinks (uncomment code below to do this)

```
# if (!require("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#
# BiocManager::install("edgeR")
# BiocManager::install("TCGAbiolinks")
# BiocManager::install("genefilter")
```

Step 1 - Load packages, download data from TCGA, and prepare it for DEGList

```
library("TCGAbiolinks")
library("limma")
library("edgeR")
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library("factoextra")
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library("FactoMineR")
library("caret")
```

```
## Loading required package: lattice
```

```
library("SummarizedExperiment")
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
```

```
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following object is masked from 'package:limma':
##
##     plotMA

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:Matrix':
##
##     expand, unname

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomeInfoDb

## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
library("gplots")
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:IRanges':
##
##     space

## The following object is masked from 'package:S4Vectors':
##
##     space

## The following object is masked from 'package:stats':
##
##     lowess
```

```
library("survival")
```

```
##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
```

```
library("survminer")
```

```
## Loading required package: ggpubr

##
## Attaching package: 'survminer'

## The following object is masked from 'package:survival':
##
##     myeloma
```

```
library("RColorBrewer")
library("gProfileR")
library("genefilter")
```

```
##
## Attaching package: 'genefilter'
```

```
## The following objects are masked from 'package:MatrixGenerics':
##
##     rowSds, rowVars
## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars
```

```r
setwd('C:/Adam/R/')  # make sure it already exists

query_TCGA = GDCquery(
  project = "TCGA-BLCA",
  data.category = "Transcriptome Profiling", # parameter enforced by GDCquery
  data.type="Gene Expression Quantification",
  experimental.strategy = "RNA-Seq",
  workflow.type = "STAR - Counts",
  sample.type = c("Primary Tumor", "Solid Tissue Normal"))
```

```
## ------------------------------------
## o GDCquery: Searching in GDC database
## ------------------------------------
## Genome of reference: hg38
## --------------------------------------------
## oo Accessing GDC. This might take a while...
## --------------------------------------------
## ooo Project: TCGA-BLCA
## --------------------
## oo Filtering results
## --------------------
## ooo By experimental.strategy
## ooo By data.type
## ooo By workflow.type
## ooo By sample.type
## ----------------
## oo Checking data
## ----------------
## ooo Checking if there are duplicated cases
## ooo Checking if there are results for the query
## ------------------
## o Preparing output
## ------------------
```

```r
lihc_res = getResults(query_TCGA) # make results as table

GDCdownload(query = query_TCGA)
```

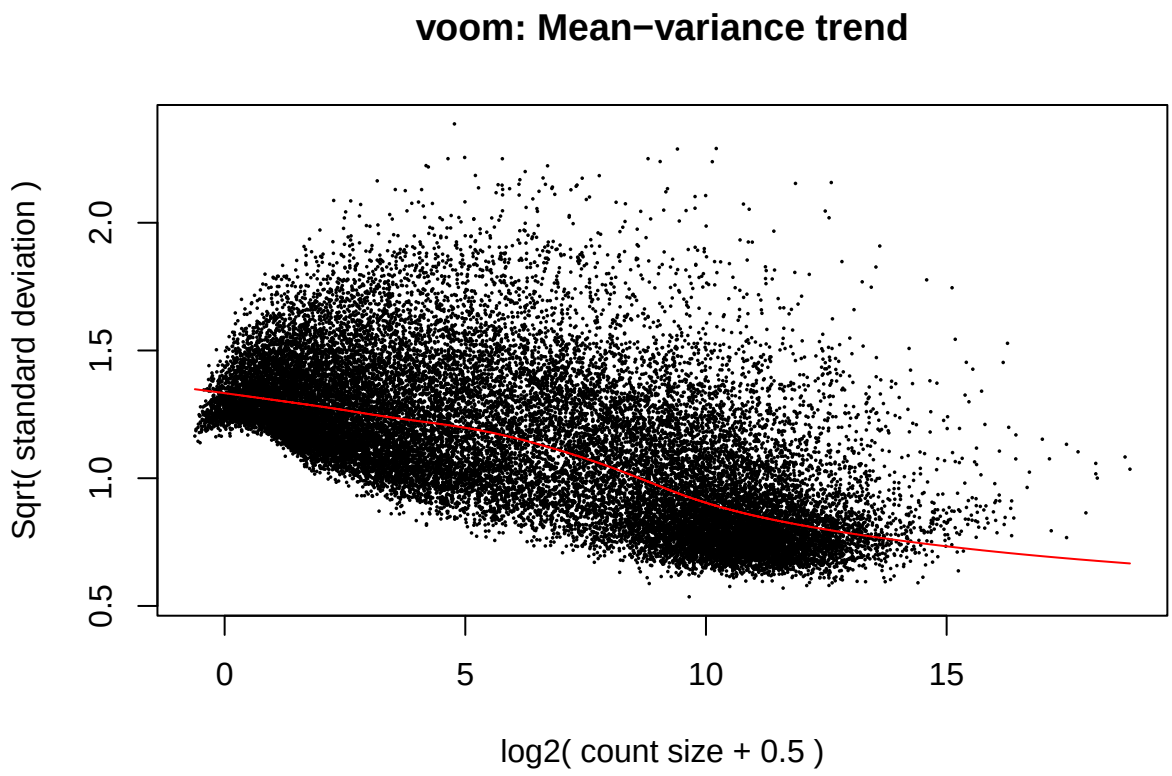
```
## Downloading data for project TCGA-BLCA
## Of the 431 files for download 431 already exist.
## All samples have been already downloaded
tcga_data = GDCprepare(query_TCGA)
```

```
## |                                                  |  0%                      |
## Starting to add information to samples
##  => Add clinical information to samples
##  => Adding TCGA molecular information from marker papers
##  => Information will have prefix 'paper_'
## blca subtype information from:doi:10.1016/j.cell.2017.09.007
## Available assays in SummarizedExperiment :
##   => unstranded
##   => stranded_first
##   => stranded_second
##   => tpm_unstrand
##   => fpkm_unstrand
##   => fpkm_uq_unstrand
```

Step 2 - Generate DEGList, filter low counts, and normalize data

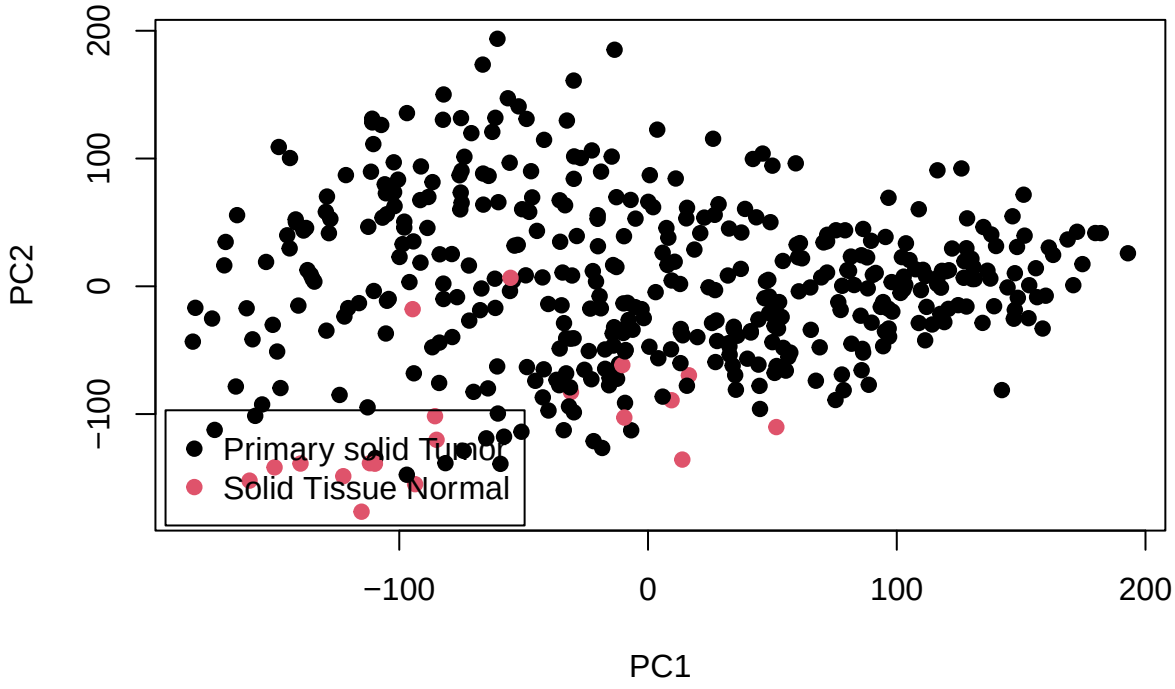
```
limma_pipeline = function(
  tcga_data,
  condition_variable,
  reference_group=NULL){

  design_factor = colData(tcga_data)[, condition_variable, drop=T]

  group = factor(design_factor)
  if(!is.null(reference_group)){group = relevel(group, ref=reference_group)}

  design = model.matrix(~ group)

  dge = DGEList(counts=assay(tcga_data),
                samples=colData(tcga_data),
                genes=as.data.frame(rowData(tcga_data)))

  # filtering
  keep = filterByExpr(dge,design)
  dge = dge[keep,,keep.lib.sizes=FALSE]
  rm(keep)

  # Normalization (TMM followed by voom)
  dge = calcNormFactors(dge)
  v = voom(dge, design, plot=TRUE)

  # Fit model to data given design
```

```
  fit = lmFit(v, design)
  fit = eBayes(fit)

  # Show top genes
  topGenes = topTable(fit, coef=ncol(design), number=100, sort.by="p")

  return(
    list(
      voomObj=v, # normalized data
      fit=fit, # fitted model and statistics
      topGenes=topGenes # the 100 most differentially expressed genes
    )
  )
}

limma_res = limma_pipeline(
  tcga_data=tcga_data,
  condition_variable="definition",
  reference_group="Solid Tissue Normal"
)
```



Step 3 - Visualize

```r
plot_PCA = function(voomObj, condition_variable){
  group = factor(voomObj$targets[, condition_variable])
  pca = prcomp(t(voomObj$E))
  # Take PC1 and PC2 for the plot
  plot(pca$x[,1:2],col=group, pch=19)
  # include a legend for points
  legend("bottomleft", inset=.01, levels(group), pch=19, col=1:length(levels(group)))
  return(pca)
}
res_pca = plot_PCA(limma_res$voomObj, "definition")
```



Step 4 - Classification model training, testing, and evaluation

```r
# Transpose and make it into a matrix object
d_mat = as.matrix(t(limma_res$voomObj$E))

# As before, we want this to be a factor
d_resp = as.factor(limma_res$voomObj$targets$definition)

# Divide data into training and testing set

# Set (random-number-generator) seed so that results are consistent between runs
set.seed(42)
train_ids = createDataPartition(d_resp, p=0.75, list=FALSE)
```

```
x_train = d_mat[train_ids, ]
x_test  = d_mat[-train_ids, ]

y_train = d_resp[train_ids]
y_test  = d_resp[-train_ids]

# Train model on training dataset using cross-validation
res = cv.glmnet(
  x = x_train,
  y = y_train,
  alpha = 0.5,
  family = "binomial")

# Test/Make prediction on test dataset
y_pred = predict(res, newx=x_test, type="class", s="lambda.min")

confusion_matrix = table(y_pred, y_test)

# Evaluation statistics
print(confusion_matrix)
```

```
##                       y_test
## y_pred                 Primary solid Tumor Solid Tissue Normal
##   Primary solid Tumor                 103                   1
##   Solid Tissue Normal                   0                   3
```

```
print(paste0("Sensitivity: ",sensitivity(confusion_matrix)))
```

```
## [1] "Sensitivity: 1"
```

```
print(paste0("Specificity: ",specificity(confusion_matrix)))
```

```
## [1] "Specificity: 0.75"
```

```
print(paste0("Precision: ",precision(confusion_matrix)))
```

```
## [1] "Precision: 0.990384615384615"
```

```
# Getting genes that contribute for the prediction
res_coef = coef(res, s="lambda.min") # the "coef" function returns a sparse matrix

# get coefficients with non-zero values
res_coef = res_coef[res_coef[,1] != 0,]
dim(res_coef)
```

```
## NULL
```

```
head(res_coef)
```

```
##        (Intercept) ENSG00000034971.17 ENSG00000078804.13  ENSG00000081181.8
##        -6.79958906         0.02423500         0.14199986         0.03321564
## ENSG00000086991.13 ENSG00000101057.16
##        -0.04076774        -0.02742725
```

```
# remove first coefficient as this is the intercept, a variable of the model itself
res_coef = res_coef[-1]
```

```
relevant_genes = names(res_coef) # get names of the (non-zero) variables.
length(relevant_genes) # number of selected genes
```

```
## [1] 83
```

```
head(relevant_genes) # few select genes
```

```
## [1] "ENSG00000034971.17" "ENSG00000078804.13" "ENSG00000081181.8"
## [4] "ENSG00000086991.13" "ENSG00000101057.16" "ENSG00000102683.8"
```

```
head(limma_res$voomObj$genes)
```

```
##                     source type score phase            gene_id      gene_type
## ENSG00000000003.15 HAVANA gene    NA    NA ENSG00000000003.15 protein_coding
## ENSG00000000005.6  HAVANA gene    NA    NA  ENSG00000000005.6 protein_coding
## ENSG00000000419.13 HAVANA gene    NA    NA ENSG00000000419.13 protein_coding
## ENSG00000000457.14 HAVANA gene    NA    NA ENSG00000000457.14 protein_coding
## ENSG00000000460.17 HAVANA gene    NA    NA ENSG00000000460.17 protein_coding
## ENSG00000000938.13 HAVANA gene    NA    NA ENSG00000000938.13 protein_coding
##                     gene_name level    hgnc_id          havana_gene
## ENSG00000000003.15    TSPAN6     2 HGNC:11858 OTTHUMG00000022002.2
## ENSG00000000005.6       TNMD     2 HGNC:17757 OTTHUMG00000022001.2
## ENSG00000000419.13      DPM1     2  HGNC:3005 OTTHUMG00000032742.2
## ENSG00000000457.14     SCYL3     2 HGNC:19285 OTTHUMG00000035941.6
## ENSG00000000460.17  C1orf112     2 HGNC:25565 OTTHUMG00000035821.9
## ENSG00000000938.13       FGR     2  HGNC:3697 OTTHUMG00000003516.3
```

```
relevant_gene_names = limma_res$voomObj$genes[relevant_genes,"external_gene_name"]

head(relevant_gene_names) # few select genes (with readable names now)
```

```
## NULL
```

```
print(intersect(limma_res$topGenes$ensembl_gene_id, relevant_genes))
```

```
## NULL
```

Step 5 - Hierarchical clustering

```
 # define the color palette for the plot
hmcol = colorRampPalette(rev(brewer.pal(9, "RdBu")))(256)

# perform complete linkage clustering
clust = function(x) hclust(x, method="complete")
# use the inverse of correlation as distance.
dist = function(x) as.dist((1-cor(t(x)))/2)

# Show green color for genes that also show up in DE analysis
colorLimmaGenes = ifelse(
  # Given a vector of boolean values
  (relevant_genes %in% limma_res$topGenes$ensembl_gene_id),
  "green", # if true, return green for that value
  "white" # if false, return white for that value
)

# As you've seen a good looking heatmap involves a lot of parameters
```

```r
gene_heatmap = heatmap.2(
  t(d_mat[,relevant_genes]),
  scale="row",           # scale the values for each gene (row)
  density.info="none",   # turns off density plot inside color legend
  trace="none",          # turns off trace lines inside the heat map
  col=hmcol,             # define the color map
  labRow=relevant_gene_names, # use gene names instead of ensembl annotation
  RowSideColors=colorLimmaGenes,
  labCol=FALSE,          # Not showing column labels
  ColSideColors=as.character(as.numeric(d_resp)), # Show colors for each response class
  dendrogram="both",     # Show dendrograms for both axis
  hclust = clust,        # Define hierarchical clustering method
  distfun = dist,        # Using correlation coefficient for distance function
  cexRow=.6,             # Resize row labels
  margins=c(1,5)         # Define margin spaces
)
```