Bladder cohort - R Notebook

Adam Cankaya
acankaya2017@fau.edu
Based on tutorials found at https://www.costalab.org/wp-content/uploads/2020/11/R_class_D3.htm https://alexslemonade.github.io/refinebio-examples/04-advanced-topics/network-analysis_rnaseq_01_wgcna.html https://github.com/hamidghaedi/RNA-seq-differential-expression

Step 1 - Install & load packages, download data from TCGA, and prepare it for EdgeR

```
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
options(repos = r)

if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
```

```
## Bioconductor version '3.19' is out-of-date; the current release version '3.21'
##   is available with R version '4.5'; see https://bioconductor.org/install
```

```
BiocManager::install("TCGAbiolinks")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org

## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'TCGAbiolinks'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival

## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
BiocManager::install("GO.db")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org
```

```
## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'GO.db'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival

## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
BiocManager::install("preprocessCore")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org

## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'preprocessCore'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival

## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
BiocManager::install("WGCNA")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org

## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'WGCNA'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival
```

```
## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
BiocManager::install("DESeq2")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org
```

```
## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'DESeq2'
```

```
## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival
```

```
## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
BiocManager::install("org.Hs.eg.db")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: http://cran.us.r-project.org
```

```
## Bioconductor version 3.19 (BiocManager 1.30.25), R 4.4.1 (2024-06-14 ucrt)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'org.Hs.eg.db'
```

```
## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.4.1/library
##   packages:
##     boot, class, cluster, foreign, KernSmooth, lattice, MASS, Matrix, mgcv,
##     nlme, nnet, rpart, spatial, survival
```

```
## Old packages: 'cli', 'curl', 'data.table', 'fastcluster', 'future',
##   'parallelly', 'RcppArmadillo', 'recipes', 'rlang'
```

```
# knit to PDF
# install.packages("tinytex")
# tinytex::install_tinytex()

install.packages("ggforce")
```

```
## Installing package into 'C:/Users/acank/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'ggforce' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'ggforce'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\acank\AppData\Local\R\win-library\4.4\00LOCK\ggforce\libs\x64\ggforce.dll
## to C:\Users\acank\AppData\Local\R\win-library\4.4\ggforce\libs\x64\ggforce.dll:
## Permission denied

## Warning: restored 'ggforce'

##
## The downloaded binary packages are in
##  C:\Users\acank\AppData\Local\Temp\RtmpWSeH8q\downloaded_packages
```

```
install.packages('VennDiagram')
```

```
## Installing package into 'C:/Users/acank/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'VennDiagram' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\acank\AppData\Local\Temp\RtmpWSeH8q\downloaded_packages
```

```
library("TCGAbiolinks")
library("limma")
library("SummarizedExperiment")
```

```
## Loading required package: MatrixGenerics

## Loading required package: matrixStats

## Warning: package 'matrixStats' was built under R version 4.4.3

##
## 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 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("gprofiler2")
```

```
## Warning: package 'gprofiler2' was built under R version 4.4.3
```

```
library("tidyverse")
```

```
## Warning: package 'tidyverse' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'purrr' was built under R version 4.4.3

## Warning: package 'forcats' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x lubridate::%within%() masks IRanges::%within%()
## x dplyr::collapse()     masks IRanges::collapse()
## x dplyr::combine()      masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count()        masks matrixStats::count()
## x dplyr::desc()         masks IRanges::desc()
## x tidyr::expand()       masks S4Vectors::expand()
## x dplyr::filter()       masks stats::filter()
## x dplyr::first()        masks S4Vectors::first()
## x dplyr::lag()          masks stats::lag()
## x ggplot2::Position()   masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()       masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()       masks S4Vectors::rename()
## x lubridate::second()   masks S4Vectors::second()
## x lubridate::second<-() masks S4Vectors::second<-()
## x dplyr::slice()        masks IRanges::slice()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library("magrittr")
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
##
## The following object is masked from 'package:GenomicRanges':
##
##     subtract
```

```
library("WGCNA")
```

```
## Warning: package 'WGCNA' was built under R version 4.4.3
```

```
## Loading required package: dynamicTreeCut
## Loading required package: fastcluster
##
## Attaching package: 'fastcluster'
##
## The following object is masked from 'package:stats':
##
##     hclust
##
##
##
## Attaching package: 'WGCNA'
##
```

```
## The following object is masked from 'package:IRanges':
##
##     cor
##
## The following object is masked from 'package:S4Vectors':
##
##     cor
##
## The following object is masked from 'package:stats':
##
##     cor
```

```
library("ggforce")
```

```
## Warning: package 'ggforce' was built under R version 4.4.3
```

```
library("doParallel")
```

```
## Warning: package 'doParallel' was built under R version 4.4.3

## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Loading required package: iterators
## Loading required package: parallel
```

```
library("org.Hs.eg.db")
```

```
## Loading required package: AnnotationDbi
##
## Attaching package: 'AnnotationDbi'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library("AnnotationDbi")
library("edgeR")
library("glmnet")
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
```

```
##     expand, pack, unpack
##
## The following object is masked from 'package:S4Vectors':
##
##     expand
##
## Loaded glmnet 4.1-8
```

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

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

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

```
## Warning: package 'caret' was built under R version 4.4.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

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

```
## Warning: package 'gplots' was built under R version 4.4.3
```

```
##
## 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
```

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

```
## Warning: package 'survminer' was built under R version 4.4.3
```

```
## Loading required package: ggpubr
##
## Attaching package: 'survminer'
##
## The following object is masked from 'package:survival':
##
##     myeloma
```

```r
library("RColorBrewer")
library("gProfileR")
```

```
##
## Attaching package: 'gProfileR'
##
## The following objects are masked from 'package:gprofiler2':
##
##     gconvert, get_base_url, get_tls_version, get_user_agent, gorth,
##     set_base_url, set_tls_version, set_user_agent
```

```r
library("genefilter")
```

```
##
## Attaching package: 'genefilter'
##
## The following object is masked from 'package:readr':
##
##     spec
##
## The following objects are masked from 'package:MatrixGenerics':
##
##     rowSds, rowVars
##
## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars
```

```r
library("VennDiagram")
```

```
## Warning: package 'VennDiagram' was built under R version 4.4.3
```

```
## Loading required package: grid
## Loading required package: futile.logger
```

```
## Warning: package 'futile.logger' was built under R version 4.4.3
```

```
##
## Attaching package: 'VennDiagram'
##
## The following object is masked from 'package:ggpubr':
##
##     rotate
```

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

if (file.exists("bladder_workspace.RData")) {
  load("bladder_workspace.RData") # load the saved workspace
}

# Before we perform a GDC query let's look at the TCGA-BLCA data
# As of June 2025 we should see a case count of 412
TCGAbiolinks:::getProjectSummary("TCGA-BLCA")
```

```
## $file_count
## [1] 31485
##
## $data_categories
##    file_count case_count                data_category
## 1       10293        412  Simple Nucleotide Variation
## 2        4294        412              Sequencing Reads
## 3        1760        412                   Biospecimen
## 4         994        412                      Clinical
## 5        6144        412         Copy Number Variation
## 6        1736        412      Transcriptome Profiling
## 7        1320        412               DNA Methylation
## 8         343        343           Proteome Profiling
## 9        2286        402 Somatic Structural Variation
## 10       2315        411          Structural Variation
##
## $case_count
## [1] 412
##
## $file_size
## [1] 4.113232e+14
```

```
# Download TCGA-BLCA data from GDC
# We want the complete RNA sequencing and raw gene count data
# So we run a query of the Transcriptome Profiling category and RNA-Seq experimental type
# We use the STAR - Counts workflow type because it contains the raw gene counts we need
# We ignore other sample types besides tumor and normal
# The original paper by Wang uses the HTSeq-counts workflow, but this is a legacy version
#  of the new STAR - COUNTS workflow type

# Run the query and format it as a table
# The results are a table with 431 rows (because some patients have multiple cases each)
# There are 29 columns with meta data about each case such as sample_type (tumor vs normal)
# Only run the query if we don't have results already saved
# TODO use GDCdownload and GDCprepare to save data to local file
if (!exists("lihc_res") || is.null(lihc_res) || !length(lihc_res) || is_empty(lihc_res)) {
```

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

  lihc_res = getResults(query_TCGA)

  # We can create a summary table shows there are 412 tumor and 19 normal (412+19=431)
  summary(factor(lihc_res$sample_type))

  # Go ahead and download all the data from GDC to our working directory
  GDCdownload(query = query_TCGA)

  # Now load the RNA-Seq data from the files into R workspace which we will save to file later
  tcga_data = GDCprepare(query_TCGA)
}
```

```
## ---------------------------------------
## 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
## -------------------
## Downloading data for project TCGA-BLCA
## Of the 431 files for download 431 already exist.
## All samples have been already downloaded

## |                                                  |  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
```

```
# This data object has 60660 rows and 431 columns
# This indicates there are 60660 different genes found throughout all the cases
# The object contains both clinical and expression data
dim(tcga_data)
```

```
## [1] 60660   431
```

```
# We can access the data in the object like this which verifies 412 tumor and 19 normal
table(tcga_data@colData$definition)
```

```
##
## Primary solid Tumor Solid Tissue Normal
##                 412                  19
```

```
# let's look at the first six rows (genes)
head(rowData(tcga_data))
```

```
## DataFrame with 6 rows and 10 columns
##                       source     type     score     phase            gene_id
##                     <factor> <factor> <numeric> <integer>        <character>
## ENSG00000000003.15   HAVANA     gene        NA        NA ENSG00000000003.15
## ENSG00000000005.6    HAVANA     gene        NA        NA  ENSG00000000005.6
## ENSG00000000419.13   HAVANA     gene        NA        NA ENSG00000000419.13
## ENSG00000000457.14   HAVANA     gene        NA        NA ENSG00000000457.14
## ENSG00000000460.17   HAVANA     gene        NA        NA ENSG00000000460.17
## ENSG00000000938.13   HAVANA     gene        NA        NA ENSG00000000938.13
##                         gene_type   gene_name       level     hgnc_id
##                       <character> <character> <character> <character>
## ENSG00000000003.15 protein_coding      TSPAN6           2  HGNC:11858
## ENSG00000000005.6  protein_coding        TNMD           2  HGNC:17757
## ENSG00000000419.13 protein_coding        DPM1           2   HGNC:3005
## ENSG00000000457.14 protein_coding       SCYL3           2  HGNC:19285
## ENSG00000000460.17 protein_coding    C1orf112           2  HGNC:25565
## ENSG00000000938.13 protein_coding         FGR           2   HGNC:3697
##                            havana_gene
##                            <character>
## ENSG00000000003.15 OTTHUMG00000022002.2
## ENSG00000000005.6  OTTHUMG00000022001.2
## ENSG00000000419.13 OTTHUMG00000032742.2
## ENSG00000000457.14 OTTHUMG00000035941.6
## ENSG00000000460.17 OTTHUMG00000035821.9
## ENSG00000000938.13 OTTHUMG00000003516.3
```

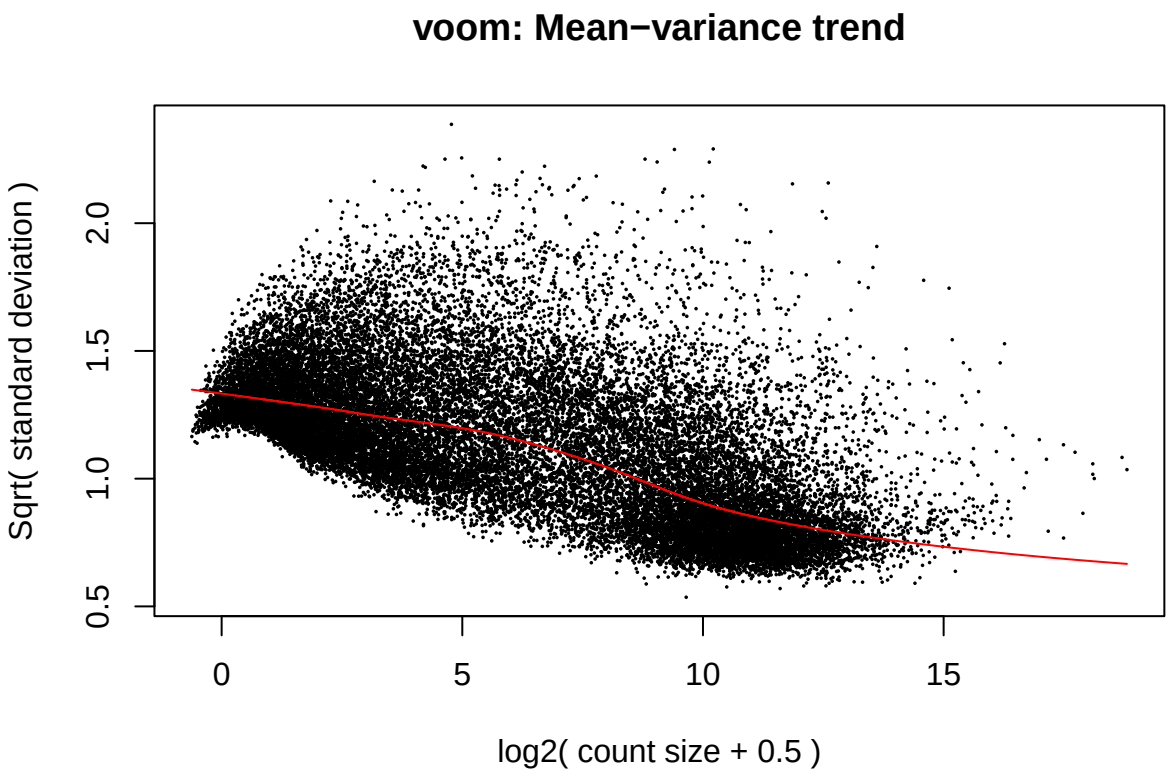
```r
# To preview the raw gene counts let's look at the expression levels of the first
#  6 genes in the first 3 cases...
rownames = values(tcga_data)$gene_name[1:6]
first6genes = head(assay(tcga_data)[,1:3])
rownames(first6genes) = rownames
colnames(first6genes) = c("Case 1", "Case 2", "Case 3")
first6genes
```

```
##          Case 1 Case 2 Case 3
## TSPAN6     7473  16931  14582
## TNMD          0     92      0
## DPM1       2663   1900   2228
## SCYL3       436   1752   1158
## C1orf112    506   4506    465
## FGR         657    804    606
```

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

```r
# ********************************************************************************
# Before we can perform DEG analysis we need to normalize the data
# Let's create a limma pipeline to do this...
# The pipeline function will take in three input parameters:
#   tcga_data - the data object we created in Step 1
#   condition_variable - the variable by which we will group patients (tumor vs normal)
#   reference_group - indicates which of the condition variable values is the reference group (normal)
# The pipeline will return a list of three objects:
#   voom - the TMM normalized data returned by running voom
#   eBayes - the fitted model returned by running eBayes
#   topTable - a simple table which contains the top 1000 differentially expressed genes
#    sorted by p.value
# ********************************************************************************
limma_pipeline = function(
  tcga_data,
  condition_variable,
  reference_group=NULL) {

    # Create a design matrix
    # The factor is the category classifier for the data (tumor vs normal)
    #  limma requires it to be a factor object
    design_factor = colData(tcga_data)[, condition_variable, drop=T] # definition
    group = factor(design_factor) # Solid Normal Tissue

    # otherwise just pick the first class as the reference class
    if (!is.null(reference_group)) {
      group = relevel(group, ref=reference_group)
    }

    # make the design matrix
    design = model.matrix(~ group)

    # generate the DGEList object using the input...
```

```
    #  counts is the raw gene counts (numericla matrix - rows as genes, columns as cases)
    #  samples is the clinical data (data frame)
    #  genes is the annotation information (data frame - gene id and names)
    # the DGEList object returned is a transformed version of tcga_data
    dge = DGEList(counts=assay(tcga_data),
                  samples=colData(tcga_data),
                  genes=as.data.frame(rowData(tcga_data)))

    # filtering - by default genes with less than 10 counts per million reads are removed
    # after filtering we have 28087 genes remaining
    # no need to filter further by logfc or adjusted p-value because all
    #  entries already meet the cutoff criteria
    keep = filterByExpr(dge,design) # genes which meet are left after filtering
    dge = dge[keep,,keep.lib.sizes=FALSE] # filter the DGEList object, only keep the genes we want
    rm(keep) # remove this object from memory because we are done with it

    # TODO do we need rpkm() filtering?

    # Normalization (TMM followed by voom)
    # normalizing - minimize batch effects and variation with the TMM normalization
    # TMM - trimmed mean of M-values
    # use the voom method to convert the data to have a similar variance as arrays
    dge = calcNormFactors(dge)
    v = voom(dge, design, plot=TRUE)

    # Fit model to data given design
    #  fits a series of linear models, one to each probe
    #  then pass it to eBayes to rank the differential expression
    fit = lmFit(v, design)
    fit = eBayes(fit)

    # Save top genes
    topGenes = topTable(fit, coef=ncol(design), number=1000, sort.by="p")

    return(
      list(
        voomObj=v, # normalized data
        fit=fit, # fitted model and statistics
        topGenes=topGenes # the 1000 most differentially expressed genes
      )
    )
}

# Run the pipeline on the tcga_data from step 1 and normal tissue as the reference
#  "definition" is the column name for the tissue type (tumor vs normal)
#  "Solid Tissue Normal" is our baseline/control/reference class value
# The limma_res object returned is a list of 3 objects - voomObj, fit, topGenes
limma_res = limma_pipeline(
  tcga_data=tcga_data,
  condition_variable="definition",
  reference_group="Solid Tissue Normal"
)
```




```
# TODO why are we doing this?
# clinical data
clinical <- data.frame(tcga_data@colData)
# replace spaces with "_" in levels of definition column
clinical$definition <-  gsub(" ", "_", clinical$definition)
# make the definition column a factor
clinical$definition <- as.factor(clinical$definition)
# relevel to ensure tumors are compared to normal tissue.
levels(clinical$definition)
```

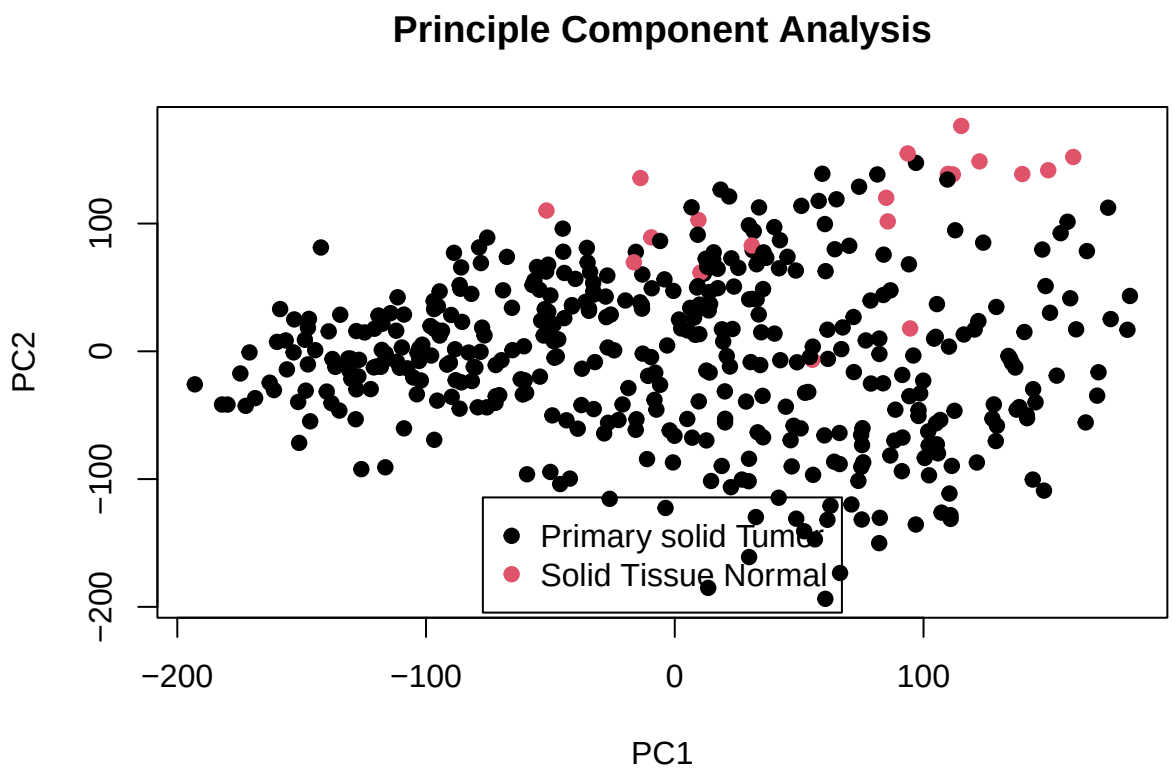
```
## [1] "Primary_solid_Tumor" "Solid_Tissue_Normal"
```

```
clinical$definition <- relevel(clinical$definition, ref = "Solid_Tissue_Normal")
```

Step 3 - Visualize DEGs with a scatter plot, a heatmap, and a volcano plot

```
# ******************************************************************************
# generate a scatter plot to show a separation of tumor vs normal points
# ******************************************************************************
plot_PCA = function(voomObj, condition_variable) {
  # create a factor
  group = factor(voomObj$targets[, condition_variable])
  # perform a principal component analysis
```

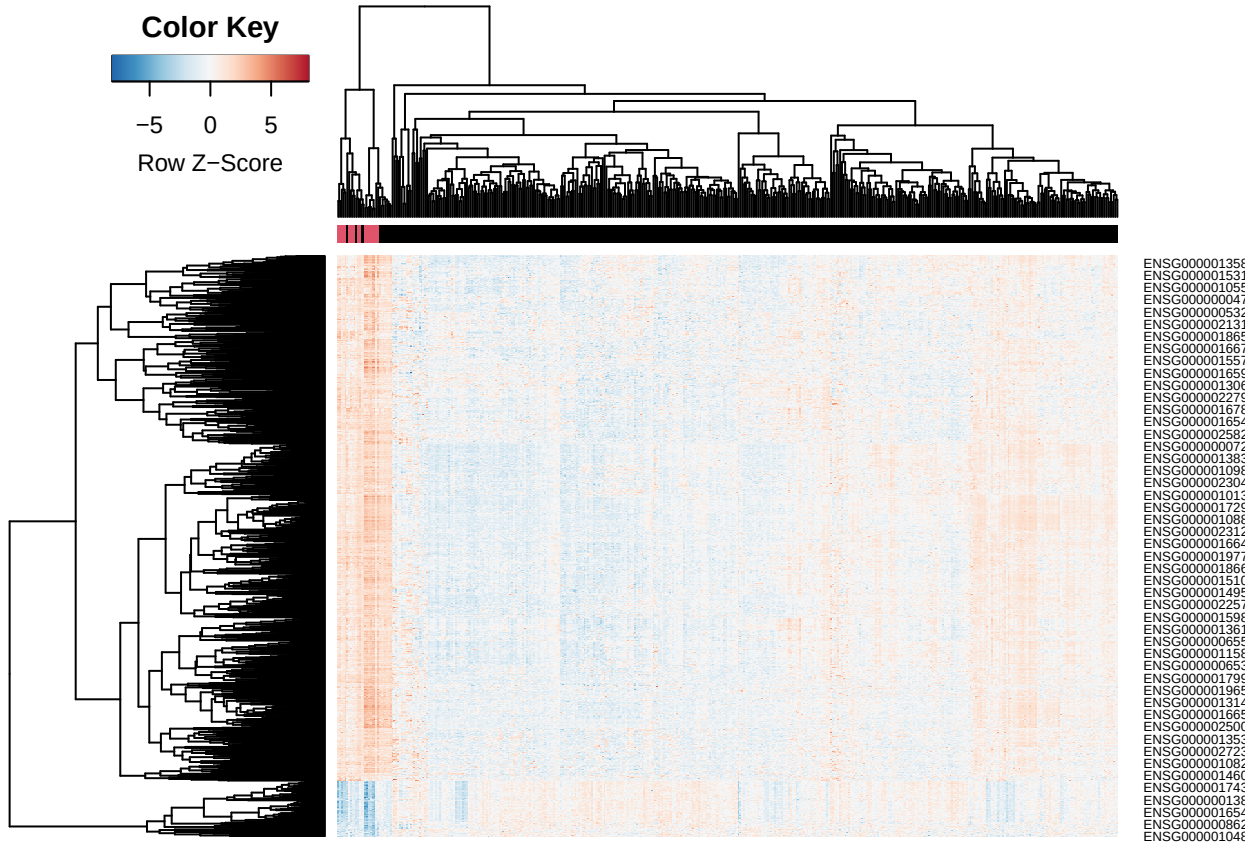
```
  pca = prcomp(t(voomObj$E))
  # Take PC1 and PC2 for the plot
  plot(pca$x[,1:2],col=group, pch=19)
  # include a legend for points
  legend("bottom", inset=.01, levels(group), pch=19, col=1:length(levels(group)))
  title("Principle Component Analysis")
  return(pca)
}

# call the plot function with the voom object and the definition column
res_pca = plot_PCA(limma_res$voomObj, "definition")
```




```
# *** create a volcano plot ***
x = limma_res$topGenes$logFC
y = limma_res$topGenes$adj.P.Val
TCGAVisualize_volcano(
  x,
  y,
  xlab = "logFC",
  title = "Volcano plot of top 1000 genes",
  filename = "volcano_top1000.pdf"
)
```

```
## Saving file as: volcano_top1000.pdf
```

```r
# *** create a heatmap (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)

d_mat = as.matrix(t(limma_res$voomObj$E))
d_resp = as.factor(limma_res$voomObj$targets$definition)

gene_heatmap = heatmap.2(
  t(d_mat[,limma_res$topGenes$gene_id]),
  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
  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
)
```



Step 4 - Run WGCNA to find gene modules

```
# ********************************************************
# prints a plot for a given gene module_name
# ********************************************************
plot_module <- function(module_name) {
  print(
    ggplot(
      module_df,
      aes(
        x = definition,
        y = module_name,
        color = definition
      )
    ) +
    # a boxplot with outlier points hidden (they will be in the sina plot)
    geom_boxplot(width = 0.2, outlier.shape = NA) +
    # A sina plot to show all of the individual data points
    ggforce::geom_sina(maxwidth = 0.3) +
    theme_classic()
  )
}

# ********************************************************
```

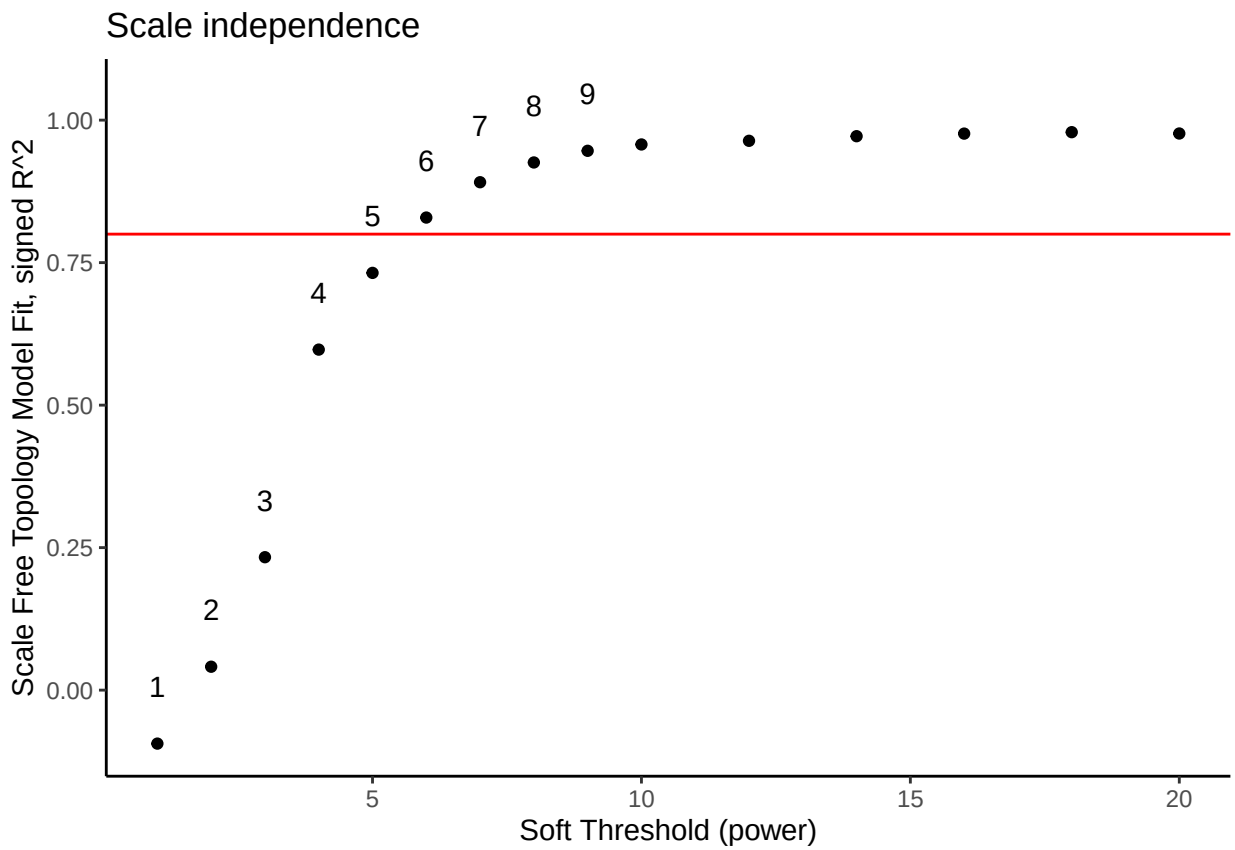
```r
# Returns a string of genes belonging to a given module
# ****************************************************
get_module_genes <- function(module_name) {
  gene_module_key <- tibble::enframe(bwnet$colors, name = "gene", value = "module") %>%
    # Let's add the `ME` part so its more clear what these numbers are and it matches elsewhere
    dplyr::mutate(module = paste0("ME", module))
  gene_module_key %>%
    dplyr::filter(module == module_name)

  # *** For easy reading let's convert from ENSEMBLE to gene symbols ***
  module_genes_ensembl <- gene_module_key[gene_module_key$module == module_name,]
  # convert to numeric to trim off the decimal place which represents the ensembl version
  module_genes_ensembl_numeric <- sub("\\..*", "", module_genes_ensembl$gene)
  # convert from ensembl to regular gene symbols and names
  module_genes <- select(org.Hs.eg.db, keys = module_genes_ensembl_numeric,
                         keytype = 'ENSEMBL', columns = c('SYMBOL', "GENENAME"))
  # remove NA's
  module_genes <- na.omit(module_genes)
  print(head(module_genes))

  # save as string for sending to AI
  module_genes_string <- paste(module_genes$SYMBOL, collapse=' ')

  return(module_genes_string)
}

# Calculate adjacency threshold parameter value using the normalized data from limma
if (!exists("sft") || is.null(sft) || !length(sft) || is_empty(sft)) {
  sft <- pickSoftThreshold(d_mat,
    dataIsExpr = TRUE,
    corFnc = cor,
    networkType = "signed"
  )
}
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
##    Power SFT.R.sq slope truncated.R.sq  mean.k. median.k.  max.k.
## 1      1   0.0938 33.70          0.908 14100.00  1.41e+04 14400.0
## 2      2   0.0411 -8.78          0.948  7240.00  7.23e+03  7760.0
## 3      3   0.2330 -9.81          0.872  3800.00  3.77e+03  4470.0
## 4      4   0.5970 -9.73          0.923  2050.00  2.01e+03  2770.0
## 5      5   0.7320 -7.26          0.946  1130.00  1.09e+03  1810.0
## 6      6   0.8290 -5.80          0.968   634.00  6.04e+02  1250.0
## 7      7   0.8910 -4.72          0.981   366.00  3.40e+02   893.0
## 8      8   0.9260 -3.99          0.987   216.00  1.95e+02   661.0
## 9      9   0.9460 -3.53          0.991   131.00  1.13e+02   512.0
## 10    10   0.9570 -3.18          0.993    81.30  6.69e+01   408.0
## 11    12   0.9640 -2.71          0.992    33.90  2.43e+01   275.0
## 12    14   0.9720 -2.38          0.996    15.60  9.36e+00   197.0
## 13    16   0.9760 -2.14          0.998     7.94  3.78e+00   148.0
## 14    18   0.9790 -1.96          0.998     4.40  1.60e+00   115.0
## 15    20   0.9770 -1.82          0.995     2.63  7.13e-01    92.3
```

```
# Calculate the signed R^2 (measure of model fit)
sft_df <- data.frame(sft$fitIndices) %>%
  dplyr::mutate(model_fit = -sign(slope) * SFT.R.sq)

# plot model fitting by the power soft threshold
ggplot(sft_df, aes(x = Power, y = model_fit, label = Power)) +
# Plot the points
geom_point() +
# Put the Power labels  above the data points
geom_text(nudge_y = 0.1) +
# Plot what WGCNA recommends as an R^2 cutoff
geom_hline(yintercept = 0.80, col = "red") +
# Make sure we can still see low levels
ylim(c(min(sft_df$model_fit), 1.05)) +
# Adjust axis
xlab("Soft Threshold (power)") +
ylab("Scale Free Topology Model Fit, signed R^2") +
ggtitle("Scale independence") +
# Add some aesthetics
theme_classic()
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (‘geom_text()‘).
```



```r
# *** the plot shows a peak positive inflection around a soft threshold power of 12 so we use that ***

# run WGCNA to find gene co-expression modules

# use saved network if available
if (file.exists("2025_05_22_wgcna_results.RDS")) {
    bwnet <- readRDS("2025_05_22_wgcna_results.RDS")
} else {
    bwnet <- blockwiseModules(d_mat,
      maxBlockSize = 25000, # What size chunks (how many genes) the calculations should be run in
      TOMType = "signed", # topological overlap matrix
      power = 12, # soft threshold for network construction
      numericLabels = TRUE, # Let's use numbers instead of colors for module labels
      randomSeed = 1234, # there's some randomness associated with this calculation so we should set a
    )
    # save result for loading later
    readr::write_rds(bwnet, "2025_05_22_wgcna_results.RDS")
}

module_eigengenes <- bwnet$MEs
head(module_eigengenes)
```

```
##                                     ME11         ME21         ME13
## TCGA-BT-A3PJ-01A-21R-A220-07 -0.011440674 -0.049665580 -0.009713237
## TCGA-DK-A6B2-01A-11R-A30C-07 -0.031753982  0.003534689  0.020596060
## TCGA-GV-A3QK-01B-11R-A23N-07 -0.042278721 -0.015474648  0.046910172
## TCGA-DK-A2I6-01A-12R-A18C-07  0.034479668  0.006641445  0.055280849
## TCGA-C4-A0F7-01A-11R-A084-07 -0.009387537 -0.055937035 -0.017267250
## TCGA-G2-A2ES-01A-11R-A180-07  0.018282326  0.009181909  0.097323235
##                                      ME9         ME16        ME18         ME22
## TCGA-BT-A3PJ-01A-21R-A220-07 -0.025966659 -0.021328874 -0.03472114 -0.052813847
## TCGA-DK-A6B2-01A-11R-A30C-07 -0.031244280 -0.002176018  0.04623553 -0.003745483
## TCGA-GV-A3QK-01B-11R-A23N-07  0.003953182  0.033716911 -0.07713281  0.080085267
## TCGA-DK-A2I6-01A-12R-A18C-07  0.047522290  0.002286637  0.01502626  0.042147895
## TCGA-C4-A0F7-01A-11R-A084-07  0.016025853 -0.050918974  0.04994291  0.035442753
## TCGA-G2-A2ES-01A-11R-A180-07  0.041432623  0.040348122 -0.02640860  0.014823970
##                                      ME6        ME10         ME19         ME12
## TCGA-BT-A3PJ-01A-21R-A220-07 -0.017366281 -0.08601696 -0.042668283  0.074402961
## TCGA-DK-A6B2-01A-11R-A30C-07  0.025735041  0.04130087  0.045765492  0.027345502
## TCGA-GV-A3QK-01B-11R-A23N-07  0.031260138  0.06837768  0.041842719  0.026646059
## TCGA-DK-A2I6-01A-12R-A18C-07  0.005856003 -0.06196811 -0.037403200 -0.008746892
## TCGA-C4-A0F7-01A-11R-A084-07 -0.009137548 -0.04960274 -0.008967307 -0.045363357
## TCGA-G2-A2ES-01A-11R-A180-07  0.013329655 -0.07072350 -0.088550079  0.064555365
##                                     ME15         ME1         ME4         ME14
## TCGA-BT-A3PJ-01A-21R-A220-07  0.038519871  0.02268054 -0.01670367 -0.04279248
## TCGA-DK-A6B2-01A-11R-A30C-07  0.041072941  0.04024125  0.06457350  0.02766255
## TCGA-GV-A3QK-01B-11R-A23N-07  0.005355924  0.01039371  0.01449953  0.01365486
## TCGA-DK-A2I6-01A-12R-A18C-07 -0.047622141 -0.03819859 -0.05227461 -0.04715729
## TCGA-C4-A0F7-01A-11R-A084-07 -0.038038136 -0.04360380 -0.06258483 -0.04722944
## TCGA-G2-A2ES-01A-11R-A180-07  0.078048559  0.02139515 -0.03238700 -0.03041298
##                                      ME7           ME2          ME5
## TCGA-BT-A3PJ-01A-21R-A220-07 -0.030202611  0.0100301590 -0.053553730
## TCGA-DK-A6B2-01A-11R-A30C-07  0.031428099  0.0387348571  0.008667189
```

```
## TCGA-GV-A3QK-01B-11R-A23N-07 -0.001612316 -0.0561233543 -0.044821335
## TCGA-DK-A2I6-01A-12R-A18C-07 -0.020621159 -0.0022594217 -0.031449081
## TCGA-C4-A0F7-01A-11R-A084-07  0.030553097  0.0007483783  0.017040502
## TCGA-G2-A2ES-01A-11R-A180-07  0.001025383  0.0181782521 -0.024282909
##                                        ME8         ME24         ME23
## TCGA-BT-A3PJ-01A-21R-A220-07  0.039327095  0.12549069 -0.004232003
## TCGA-DK-A6B2-01A-11R-A30C-07  0.027934115 -0.03302481 -0.012833777
## TCGA-GV-A3QK-01B-11R-A23N-07  0.015173456 -0.02794133 -0.064942690
## TCGA-DK-A2I6-01A-12R-A18C-07 -0.001590417  0.06840482  0.105777195
## TCGA-C4-A0F7-01A-11R-A084-07  0.070058871  0.11352577  0.016040705
## TCGA-G2-A2ES-01A-11R-A180-07  0.032392851  0.01399121  0.030003253
##                                        ME3         ME17          ME20
## TCGA-BT-A3PJ-01A-21R-A220-07  0.0564560971  0.04819074  0.013920857
## TCGA-DK-A6B2-01A-11R-A30C-07 -0.0000154232  0.02738674  0.013404980
## TCGA-GV-A3QK-01B-11R-A23N-07 -0.0134256143 -0.01712478 -0.021627450
## TCGA-DK-A2I6-01A-12R-A18C-07  0.0570827313  0.04672159  0.079341712
## TCGA-C4-A0F7-01A-11R-A084-07  0.0645554479  0.01940087 -0.008427715
## TCGA-G2-A2ES-01A-11R-A180-07  0.0400912983  0.06178334  0.035062937
##                                        ME0
## TCGA-BT-A3PJ-01A-21R-A220-07 -0.075624470
## TCGA-DK-A6B2-01A-11R-A30C-07  0.001950957
## TCGA-GV-A3QK-01B-11R-A23N-07  0.070222805
## TCGA-DK-A2I6-01A-12R-A18C-07 -0.053005640
## TCGA-C4-A0F7-01A-11R-A084-07 -0.042344440
## TCGA-G2-A2ES-01A-11R-A180-07 -0.067726764
```

```
# make sure samples are in same order
all.equal(clinical$barcode, rownames(module_eigengenes))
```

```
## [1] TRUE
```

```
# Create the design matrix from the definition variable (tumor vs normal)
des_mat <- model.matrix(~clinical$definition)

# Run linear model on each module.
# limma wants our tests to be per row and lmFit() needs a transposed version of matrix
fit <- limma::lmFit(t(module_eigengenes), design = des_mat)

# Apply empirical Bayes to smooth standard errors
fit <- limma::eBayes(fit)

# Apply multiple testing correction and obtain stats in a data frame
stats_df <- limma::topTable(fit, number = ncol(module_eigengenes)) %>%
  tibble::rownames_to_column("module")
```

```
## Removing intercept from test coefficients
```

```
# look at the most significant modules
head(stats_df)
```

```
##   module       logFC       AveExpr         t      P.Value    adj.P.Val
## 1    ME3  0.11242029 -2.052186e-18 10.161025 3.806372e-24 9.515931e-23
```

```
## 2    ME5 -0.10773122  3.772697e-19 -9.733512 2.683090e-22 3.353863e-21
## 3   ME23  0.07613728 -3.850050e-18  6.864358 7.048104e-12 5.873420e-11
## 4    ME7 -0.07407338 -7.490051e-19 -6.677524 2.550320e-11 1.593950e-10
## 5    ME6  0.06141998  1.222557e-18  5.533382 3.214925e-08 1.607462e-07
## 6    ME2 -0.05717064 -1.308243e-18 -5.149607 2.656628e-07 1.106928e-06
##           B
## 1 43.846168
## 2 39.650816
## 3 16.099022
## 4 14.844821
## 5  7.913862
## 6  5.878218
```

```
# put the modules into a data frame
module_df <- module_eigengenes %>%
  tibble::rownames_to_column("barcode") %>%
# Here we are performing an inner join with a subset of metadata
dplyr::inner_join(clinical %>%
  dplyr::select(barcode, definition),
  by = c("barcode" = "barcode")
)

# *** The most significant module with positive log fc is ME3 (more associated with tumors)***
ME3_genes <- get_module_genes("ME3")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
##           ENSEMBL SYMBOL
## 1 ENSG00000000460  FIRRM
## 2 ENSG00000006634   DBF4
## 3 ENSG00000007968   E2F2
## 4 ENSG00000010292 NCAPD2
## 5 ENSG00000011426   ANLN
## 6 ENSG00000012048  BRCA1
##                                                GENENAME
## 1 FIGNL1 interacting regulator of recombination and mitosis
## 2                      DBF4-CDC7 kinase regulatory subunit
## 3                                E2F transcription factor 2
## 4                  non-SMC condensin I complex subunit D2
## 5                          anillin, actin binding protein
## 6                             BRCA1 DNA repair associated
```

```
plot_module("ME3")
```


```
# *** The most significant module with negative log fc is ME5 (more associated with normal tissue)***
ME5_genes <- get_module_genes("ME5")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
##           ENSEMBL  SYMBOL                                     GENENAME
## 1 ENSG00000004776   HSPB6 heat shock protein family B (small) member 6
## 2 ENSG00000018236   CNTN1                                   contactin 1
## 3 ENSG00000018625  ATP1A2   ATPase Na+/K+ transporting subunit alpha 2
## 4 ENSG00000022267    FHL1              four and a half LIM domains 1
## 5 ENSG00000025423 HSD17B6      hydroxysteroid 17-beta dehydrogenase 6
## 6 ENSG00000049540     ELN                                   elastin
```

```
plot_module("ME5")
```


```
# Calculate the adjacency matrix
# adj_mat <- adjacency(d_mat, power=12, corFnc="bicor", type="signed",  corOptions = "use = 'p'")

# Calculate the topological overlap matrix (TOM)
# TOM <- TOMsimilarityFromExpr(d_mat, power = 12)
```

Step 5 - Generate venn diagrams and other visuals

```
module_genes <- tibble::enframe(bwnet$colors, name = "gene", value = "module") %>%
  dplyr::mutate(module = paste0("ME", module))

# *** Show venn diagram overlap between module genes and DEGs (positive log fc) ***

# TODO make this work with a character map of all the module names
# top_5_positive_modules <- head(arrange(stats_df, desc(logFC)), 5)
# target <- strsplit(top_5_positive_modules$module, " ") %>% paste(collapse = "|")
# the top five positive logfc modules are ME3, ME23, ME6, ME21, ME11
# top_5_negative_modules <- head(arrange(stats_df, logFC), 5)
# target <- strsplit(top_5_negative_modules$module, " ") %>% paste(collapse = "|")
# the top five negative logfc modules are ME5, ME7, ME2, ME14, ME1

positive_module_genes_ensembl <- module_genes %>%
  dplyr::filter(module == "ME3" | module == "ME23" | module == "ME6"
```

```
                | module == "ME21" | module == "ME11")
# convert to numeric to trim off the decimal place which represents the ensembl version
positive_module_genes_ensembl_numeric <- sub("\\..*", "", positive_module_genes_ensembl$gene)
# convert from ensembl to regular gene symbols and names
positive_module_genes_ensembl <- select(org.Hs.eg.db, keys = positive_module_genes_ensembl_numeric,
                      keytype = 'ENSEMBL', columns = c('SYMBOL', "GENENAME"))
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# remove NA's
positive_module_genes_ensembl <- na.omit(positive_module_genes_ensembl)

positive_degs <- na.omit(limma_res$topGenes[limma_res$topGenes$logFC > 0,]$gene_name)

venn.diagram(
  main = "Positive logFC (high exp in tumor tissue) - Module genes vs DEGs",
  x = list(positive_degs, positive_module_genes_ensembl$SYMBOL),
  category.names = c("DEGs", "Module genes (ME3,ME23,ME6,ME21,ME11)"),
  cat.pos = c(225,180),
  filename = 'Positive_LogFC-Module_genes_vs_DEGs.png',
  output=TRUE,
  col=c("darkorchid2", "cadetblue1"),
  fil=c("darkorchid2", "cadetblue1")
)
```

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

```
# *** Show overlap between module genes and DEGs (negative log fc) ***
module_genes <- tibble::enframe(bwnet$colors, name = "gene", value = "module") %>%
  dplyr::mutate(module = paste0("ME", module))
negative_module_genes_ensembl <- module_genes %>%
  dplyr::filter(module == "ME5" | module == "ME7" | module == "ME2"
                | module == "ME14" | module == "ME1")
# convert to numeric to trim off the decimal place which represents the ensembl version
negative_module_genes_ensembl_numeric <- sub("\\..*", "", negative_module_genes_ensembl$gene)
# convert from ensembl to regular gene symbols and names
negative_module_genes_ensembl <- select(org.Hs.eg.db, keys = negative_module_genes_ensembl_numeric,
                      keytype = 'ENSEMBL', columns = c('SYMBOL', "GENENAME"))
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# remove NA's
negative_module_genes_ensembl <- na.omit(negative_module_genes_ensembl)

negative_degs <- na.omit(limma_res$topGenes[limma_res$topGenes$logFC < 0,]$gene_name)

venn.diagram(
  main = "Negative logFC (high exp in normal tissue) - Module genes vs DEGs",
  x = list(negative_module_genes_ensembl$SYMBOL, negative_degs),
  category.names = c("Module genes (ME5,ME7,ME2,ME14,ME1)" , "DEGs"),
  cat.pos = c(0,45),
```

```
  filename = 'Negative_LogFC-Module_genes_vs_DEGs.png',
  output=TRUE,
  col=c("darkorchid2", "cadetblue1"),
  fil=c("darkorchid2", "cadetblue1")
)
```

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

```
# *** Create gene clustering tree (dendrogram) of TCGA modules ***
```

Step 6 - a) experiment with parameters such as blocksize, soft threshold, more filtering…look at original paper's values b) examine all significant modules c) add timer code and speed enhancements (saving results to hd, multi-threading, AI?, cloud?, doParallel?) d) AI enhancements (Posit?, ask AI to speed up my code?) e) library(doParallel) registerDoParallel(cores=4) f) allowWGCNAThreads(8) g) other TODOs

```
  # Save the workspace for future loading
  save.image(file = "bladder_workspace.RData")
```