code

Guillem Chillon

2025-10-23

Project 2

Import packages

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.1     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(biomaRt)
library(tximport)
library(DESeq2)
```

```
Loading required package: S4Vectors
Loading required package: stats4
Loading required package: BiocGenerics

Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:lubridate':

    intersect, setdiff, union

The following objects are masked from 'package:dplyr':

    combine, intersect, setdiff, union

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


Attaching package: 'S4Vectors'

The following objects are masked from 'package:lubridate':

    second, second<-

The following objects are masked from 'package:dplyr':

    first, rename

The following object is masked from 'package:tidyr':

    expand

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:lubridate':

    %within%

The following objects are masked from 'package:dplyr':

    collapse, desc, slice

The following object is masked from 'package:purrr':

    reduce

Loading required package: GenomicRanges
Loading required package: GenomeInfoDb
Loading required package: SummarizedExperiment
Loading required package: MatrixGenerics
Loading required package: matrixStats

Attaching package: 'matrixStats'

The following object is masked from 'package:dplyr':

    count


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: 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(vsn)
library(pheatmap)
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:Biobase':

    combine

The following object is masked from 'package:BiocGenerics':

    combine

The following object is masked from 'package:dplyr':
```

```
    combine
```

```
library(igraph)
```

```
Attaching package: 'igraph'
The following object is masked from 'package:GenomicRanges':
    union
The following object is masked from 'package:IRanges':
    union
The following object is masked from 'package:S4Vectors':
    union
The following objects are masked from 'package:BiocGenerics':
    normalize, path, union
The following objects are masked from 'package:lubridate':
    %--%, union
The following objects are masked from 'package:dplyr':
    as_data_frame, groups, union
The following objects are masked from 'package:purrr':
    compose, simplify
The following object is masked from 'package:tidyr':
    crossing
The following object is masked from 'package:tibble':
    as_data_frame
```

```
The following objects are masked from 'package:stats':

    decompose, spectrum

The following object is masked from 'package:base':

    union
```

```
library(rstatix)
```

```
Attaching package: 'rstatix'

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

    desc

The following object is masked from 'package:biomaRt':

    select

The following object is masked from 'package:stats':

    filter
```

```
library(ggpubr)
```

Import data and experimental design

Load sample data

```
samples <- read.csv("../data/exp_design.csv", header=TRUE)
# Change rownames to reflect experimental information
rownames(samples) <- factor(paste0(
  samples$Experimental_ID, sep="_",
  samples$Genotype, sep="_",
  samples$Treatment))
# Remove miCtrl treated sample from this analysis
samples <- samples[-c(9),]
```

Import abundance files

```
ensembl <- useEnsembl(biomart = "genes",
                      dataset = "mmusculus_gene_ensembl")
Tx <- getBM(attributes = c("ensembl_transcript_id",
                           "external_gene_name"),
            mart = ensembl)

colnames(Tx) <- c("tx_id", "gene_name")

files <- file.path("../data/KALLISTO",
                   paste0(samples$Sample, "_quant"), "abundance.h5")
file.exists(files)
```

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

```
# Add labels to files
IDs <- paste0(samples$Experimental_ID, sep="_",
              samples$Genotype, sep="_",
              samples$Treatment)
names(files) <- IDs

# Import Kallisto transcript counts
Txi_gene <- tximport(files,
                     type = "kallisto",
                     tx2gene = Tx,
                     txOut = FALSE,
                     ignoreTxVersion = TRUE)
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
transcripts missing from tx2gene: 1124
summarizing abundance
summarizing counts
summarizing length
```

```
names(Txi_gene)
```

```
[1] "abundance"           "counts"              "length"
[4] "countsFromAbundance"
```

```
head(Txi_gene$counts)
```

```
               A_WT_VSB   B_MUT_V1 C_MUT_VSB  D_WT_VSB E_MUT_VSB  F_WT_VSB
              3148.7212 11746.5268 11328.597 8509.3409  8229.451 7369.4944
0610030E20Rik 1051.5926  1191.1917  1373.911 1224.0489  1262.291 1121.0913
1110002E22Rik   18.0000    20.0000    21.000   12.0000    23.000   15.0000
1110004F10Rik 2586.9910  3439.9458  3790.932 2924.0000  3801.981 2866.0000
1110032F04Rik  314.0000   224.0000   334.000  365.0000   184.000  387.0000
1110038F14Rik  744.3297   869.4623  1083.034  830.9403   851.512  707.2989
               G_MUT_V1 H_MUT_VSB   K_MUT_V1  L_WT_VSB M_MUT_V1  N_MUT_V1
              6143.1015 8598.3430 8901.73774 4689.5204 5664.406 2679.2345
0610030E20Rik 1284.7320 1393.7634 1154.35843 1216.5663 1251.429 1062.1729
1110002E22Rik   14.0000    7.0000   24.22765   24.0000   19.000   12.0000
1110004F10Rik 3418.0000 3436.0000 3188.00000 2886.0000 2884.986 2835.0000
1110032F04Rik  315.0000  274.0000  239.00000  431.0000  329.000  254.0000
1110038F14Rik  897.0541  925.0348  804.51834  800.8577  794.204  718.5452
              O_MUT_VSB  P_MUT_VSB  Q_WT_VSB
              9042.4643 4206.85684 5629.5022
0610030E20Rik 1113.6483 1209.81433  985.5856
1110002E22Rik   12.0000   27.27056   14.0000
1110004F10Rik 3478.9836 3449.00000 2346.0000
1110032F04Rik  289.0000  296.00000  351.0000
1110038F14Rik  845.7179  838.11429  658.5599
```

Make DESeq2 DataSet

```
samples$Litter <- as.character(samples$Litter)
samples$RNA_batch <- as.character(samples$RNA_batch)

dds <- DESeqDataSetFromTximport(txi = Txi_gene,
                                colData = samples,
                                design = ~ Genotype + Treatment + Litter)
```

```
Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

```
using counts and average transcript lengths from tximport
```

```
dds
```

```
class: DESeqDataSet
dim: 36139 15
metadata(1): version
assays(2): counts avgTxLength
rownames(36139): '' 0610030E20Rik ... Zzef1 Zzz3
rowData names(0):
colnames(15): A_WT_VSB B_MUT_V1 ... P_MUT_VSB Q_WT_VSB
colData names(8): Sample Experimental_ID ... Treatment Sex
```

```
nrow(dds)
```

```
[1] 36139
```

```
# at least 4 samples with a count of 10 or higher
keep <- rowSums(counts(dds) >= 10)>=4
dds <- dds[keep,]
nrow(dds)
```

```
[1] 18421
```

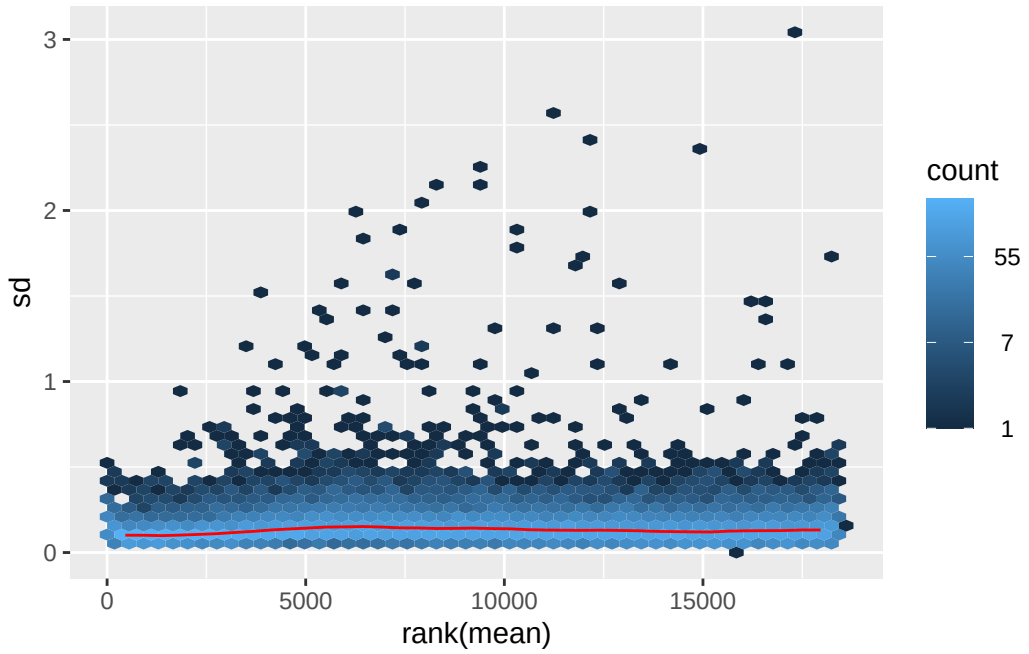
Transformation, sample clustering and visualization

```
vsd <- vst(dds, blind=TRUE)
```

```
using 'avgTxLength' from assays(dds), correcting for library size
```

```
meanSdPlot(assay(vsd))
```

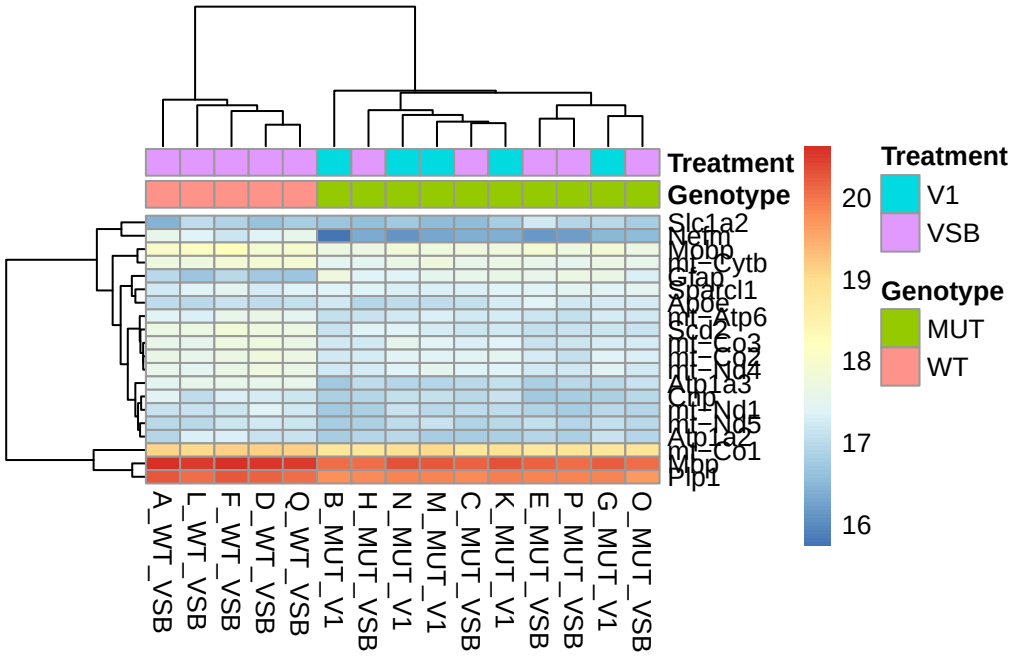
```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with `aes()`.
i See also `vignette("ggplot2-in-packages")` for more information.
i The deprecated feature was likely used in the vsn package.
  Please report the issue to the authors.
```




```
dds <- estimateSizeFactors(dds)
```

```
using 'avgTxLength' from assays(dds), correcting for library size
```

```
select <- order(rowMeans(counts(dds,normalized=TRUE)),
                decreasing=TRUE)[1:20]
df <- as.data.frame(colData(dds)[,c("Genotype","Treatment")])
pheatmap(assay(vsd)[select,], cluster_rows=TRUE, show_rownames=TRUE,
         cluster_cols=TRUE, annotation_col=df)
```




```
pcaData <- plotPCA(vsd, intgroup=c("Treatment", "Genotype", "Litter"),
                   returnData=TRUE, ntop=1000)
```

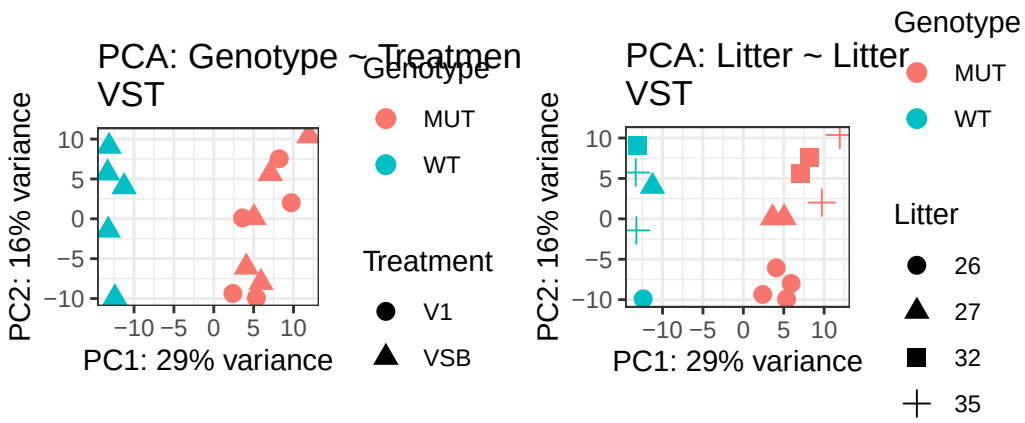
```
using ntop=1000 top features by variance
```

```
percentVar <- round(100 * attr(pcaData, "percentVar"))

p1 <- ggplot(pcaData, aes(PC1, PC2, color=Genotype, shape=Treatment)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed() +
  theme_bw() +
  ggtitle("PCA: Genotype ~ Treatment \nVST")

p2 <- ggplot(pcaData, aes(PC1, PC2, color=Genotype, shape=Litter)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed() +
  theme_bw() +
  ggtitle("PCA: Litter ~ Litter \nVST")
```

```
g <- grid.arrange(p1, p2, ncol=2)
```



Network analysis

Create adjacent matrices

Select top 200 variable genes

```
# Extract gene variance
vsd_mat <- assay(vsd)
gene_var <- apply(vsd_mat, 1, var)

# Get the names of the top 200 most variable genes
top200_genes <- names(sort(gene_var, decreasing = TRUE))[1:200]

# Subset the vsd matrix to include only the top 200 variable genes
vsd_top200 <- vsd_mat[top200_genes, ]
```

Define adaptive threshold

```
n_genes <- 200
potential_edges <- (n_genes * (n_genes - 1)) / 2
edges_keep <- round(0.05 * potential_edges) # Keep top 5% of edges
```

Define bootstrap iteration

```
set.seed(123)
bootstrap_q <- function(sample_names, vsd, n_edges) {
  # Resample 5 samples with replacement
  samples_boot <- sample(sample_names, 5, replace = TRUE)

  # Create correlation matrix
  cor_mat <- cor(t(vsd[, samples_boot]),
                 method = "spearman",
                 use = "pairwise.complete.obs")
  # Remove NA values
  cor_mat[is.na(cor_mat)] <- 0
  # Get unique corr values
  cor_values <- abs(cor_mat[upper.tri(cor_mat)])
  # Sort them in decreasing order
  cor_values_sorted <- sort(cor_values, decreasing = TRUE)
  # Apply threshold
  threshold <- cor_values_sorted[n_edges]
  adj_mat <- (abs(cor_mat) >= threshold) * 1
  # Remove self-loops
  diag(adj_mat) <- 0

  # Build igraph object
  g <- graph_from_adjacency_matrix(adj_mat, mode = "undirected", diag = FALSE)
  V(g)$name <- rownames(adj_mat)

  # Perform modularity maximization
  if (gsize(g) > 0) {
    mod_result <- cluster_louvain(g)
    return(modularity(mod_result))
  } else {
    mod_result <- NA
  }
}
```

```
n_bootstrap <- 300

samples <- samples %>% mutate(condition = paste0(Genotype, "_", Treatment))
sample_list <- list(
  WT = rownames(samples %>% filter(condition == "WT_VSB")),
  VSB = rownames(samples %>% filter(condition == "MUT_VSB")),
```

```
  Tx = rownames(samples %>% filter(condition == "MUT_V1"))
)

q_scores <- suppressWarnings(lapply(sample_list, function(sample_names) {
  # Use replicate to perform bootstrap iterations
  replicate(n_bootstrap, {
    bootstrap_q(sample_names, vsd_top200, edges_keep)
  })
}))
```

```
lapply(q_scores, summary)
```

```
$WT
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.6043  0.6587  0.6633  0.8804  0.8988

$VSB
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.6390  0.6599  0.6767  0.8836  0.8966

$Tx
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.6069  0.6515  0.6523  0.8666  0.8884
```

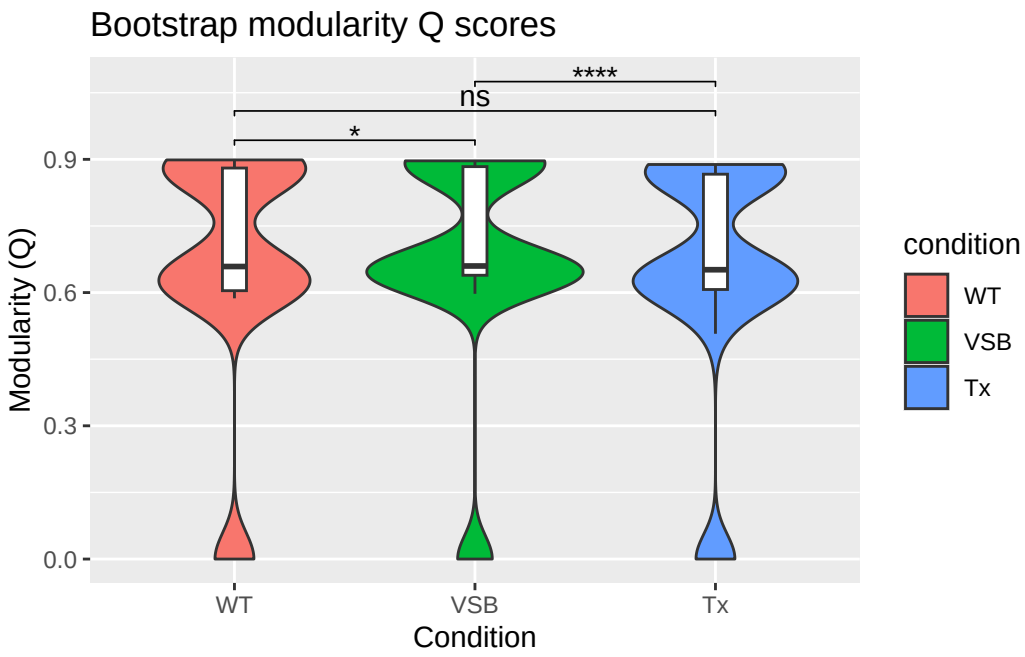
```
# Check statistical significant
df <- as.data.frame(q_scores) %>% pivot_longer(
  cols=everything(),
  names_to = "condition",
  values_to = "q_score"
) %>%
  mutate(condition = factor(condition, levels=c("WT", "VSB", "Tx")))

wilcox <- df %>%
  wilcox_test(q_score ~ condition, p.adjust.method="bonferroni")  %>%
  add_xy_position(x = "condition", step.increase = 4)
wilcox
```

```
# A tibble: 3 x 13
  .y.     group1 group2    n1    n2 statistic          p      p.adj p.adj.signif
  <chr>   <chr>  <chr>  <int> <int>     <dbl>      <dbl>      <dbl> <chr>
1 q_score WT     VSB      300   300    39381  0.008      0.024      *
```

```
2 q_score WT     Tx       300   300    49264. 0.044      0.133      ns
3 q_score VSB    Tx       300   300    55319  0.00000116 0.00000348 ****
# i 4 more variables: y.position <dbl>, groups <named list>, xmin <dbl>,
#   xmax <dbl>
```

```r
p <- ggplot(df, aes(x=condition, y=q_score, fill=condition)) +
  geom_violin() +
  geom_boxplot(width=0.1, fill="white", outlier.shape = NA) +
  stat_pvalue_manual(
    wilcox, label = "p.adj.signif", tip.length = 0.01
  ) +
  labs(
    title = "Bootstrap modularity Q scores",
    x = "Condition",
    y = "Modularity (Q)"
  )
p
```



Create modularity maps

Define function to build network and modules

```
build_network_and_modules <- function(sample_names, vsd_data, n_edges) {
  # Create correlation matrix
  cor_mat <- cor(t(vsd_data[, sample_names]),
                 method = "spearman",
                 use = "pairwise.complete.obs")
  cor_mat[is.na(cor_mat)] <- 0 # Handle NAs

  # Apply adaptive threshold
  cor_values <- abs(cor_mat[upper.tri(cor_mat)])
  cor_values_sorted <- sort(cor_values, decreasing = TRUE)
  threshold <- cor_values_sorted[n_edges]

  adj_mat <- (abs(cor_mat) >= threshold) * 1
  diag(adj_mat) <- 0

  # Build igraph object
  g <- graph_from_adjacency_matrix(adj_mat, mode = "undirected", diag = FALSE)
  V(g)$name <- rownames(adj_mat)

  # Perform modularity maximization
  if (gsize(g) > 0) {
    modules <- cluster_louvain(g)
  } else {
    modules <- list(membership = rep(1, vcount(g)), modularity = NA)
    class(modules) <- "communities"
  } # Return both parts
  return(list(graph = g, modules = modules))
}
```

Build igraph objects and modularity results

```
original_network_results <- suppressWarnings(lapply(sample_list, function(names) {
  build_network_and_modules(names, vsd_top200, edges_keep) # edges_keep defined earlier
}))
igraph_objects <- lapply(original_network_results, function(res) res$graph)
modularity_results <- lapply(original_network_results, function(res) res$modules)
```

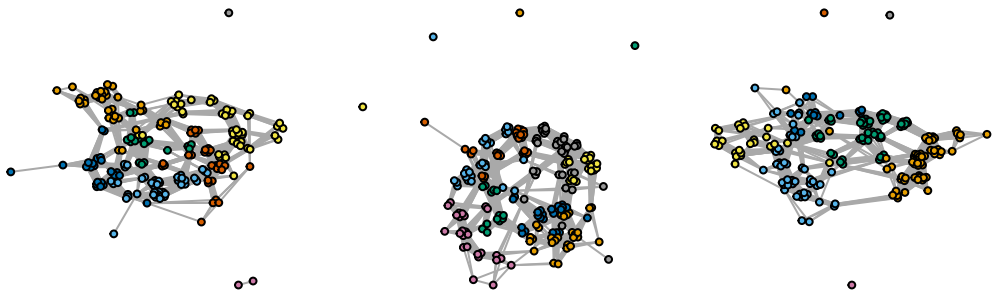
Plot modularity clusters

```
conditions=c("WT", "VSB", "Tx")
par(mfrow=c(1,3), mar=c(1,1,2,1))
for (i in 1:length(conditions)) {
  cond <- conditions[i]
  g <- igraph_objects[[cond]]
  modules <- modularity_results[[cond]]

  V(g)$color <- membership(modules)
  l <- layout_with_fr(g)

  plot(
    g,
    layout = l,
    vertex.size = 5,
    vertex.label = NA,
    main = paste0(cond, " Modularity Clusters")
  )
}
```

WT Modularity Clusters **VSB Modularity Clusters** **Tx Modularity Clusters**



```
par(mfrow=c(1,1))
```

Save session info

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS 26.0.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] ggpubr_0.6.1                rstatix_0.7.2
 [3] igraph_2.1.4                gridExtra_2.3
 [5] pheatmap_1.0.13             vsn_3.72.0
 [7] DESeq2_1.44.0               SummarizedExperiment_1.34.0
 [9] Biobase_2.64.0              MatrixGenerics_1.16.0
[11] matrixStats_1.5.0           GenomicRanges_1.56.2
[13] GenomeInfoDb_1.40.1         IRanges_2.38.1
[15] S4Vectors_0.42.1            BiocGenerics_0.50.0
[17] tximport_1.32.0             biomaRt_2.60.1
[19] lubridate_1.9.4             forcats_1.0.1
[21] stringr_1.5.2               dplyr_1.1.4
[23] purrr_1.1.0                 readr_2.1.5
[25] tidyr_1.3.1                 tibble_3.3.0
[27] ggplot2_4.0.0               tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] DBI_1.2.3               httr2_1.2.1             rlang_1.1.6
 [4] magrittr_2.0.4          compiler_4.4.2          RSQLite_2.4.3
 [7] png_0.1-8               vctrs_0.6.5             pkgconfig_2.0.3
[10] crayon_1.5.3            fastmap_1.2.0           backports_1.5.0
```

```
[13] dbplyr_2.5.1           XVector_0.44.0          labeling_0.4.3
[16] utf8_1.2.6             rmarkdown_2.29          tzdb_0.5.0
[19] UCSC.utils_1.0.0       preprocessCore_1.66.0   tinytex_0.57
[22] bit_4.6.0              xfun_0.53               zlibbioc_1.50.0
[25] cachem_1.1.0           jsonlite_2.0.0          progress_1.2.3
[28] blob_1.2.4             rhdf5filters_1.16.0     DelayedArray_0.30.1
[31] Rhdf5lib_1.26.0        BiocParallel_1.38.0     broom_1.0.10
[34] parallel_4.4.2         prettyunits_1.2.0       R6_2.6.1
[37] stringi_1.8.7          RColorBrewer_1.1-3      limma_3.60.6
[40] car_3.1-3              Rcpp_1.1.0              knitr_1.50
[43] Matrix_1.7-4           timechange_0.3.0        tidyselect_1.2.1
[46] rstudioapi_0.17.1      abind_1.4-8             yaml_2.3.10
[49] codetools_0.2-20       affy_1.82.0             curl_7.0.0
[52] lattice_0.22-7         withr_3.0.2             KEGGREST_1.44.1
[55] S7_0.2.0               evaluate_1.0.5          BiocFileCache_2.12.0
[58] xml2_1.4.0             Biostrings_2.72.1       pillar_1.11.1
[61] affyio_1.74.0          BiocManager_1.30.26     filelock_1.0.3
[64] carData_3.0-5          generics_0.1.4          hms_1.1.3
[67] scales_1.4.0           glue_1.8.0              tools_4.4.2
[70] hexbin_1.28.5          ggsignif_0.6.4          locfit_1.5-9.12
[73] rhdf5_2.48.0           grid_4.4.2              AnnotationDbi_1.66.0
[76] colorspace_2.1-2       GenomeInfoDbData_1.2.12 Formula_1.2-5
[79] cli_3.6.5              rappdirs_0.3.3          S4Arrays_1.4.1
[82] gtable_0.3.6           digest_0.6.37           SparseArray_1.4.8
[85] farver_2.1.2           memoise_2.0.1           htmltools_0.5.8.1
[88] lifecycle_1.0.4        httr_1.4.7              statmod_1.5.0
[91] bit64_4.6.0-1
```