Differential gene expression across organoid phenotypes

Benedikt Rauscher

11/15/2020

Dependencies

```
library(SummarizedExperiment)
library(DESeq2) #optional
library(glmnet)
library(pheatmap)
library(reshape2)
library(tidyverse)
library(cowplot)
library(here)
```

Define function namespace.

```
filter <- dplyr::filter
select <- dplyr::select
rename <- dplyr::rename
slice <- dplyr::slice
count <- dplyr::count
```

Set parameters for plotting.

```
theme_set(theme_cowplot())
```

Data loading

We load the gene expression data from an Rdata object from file.

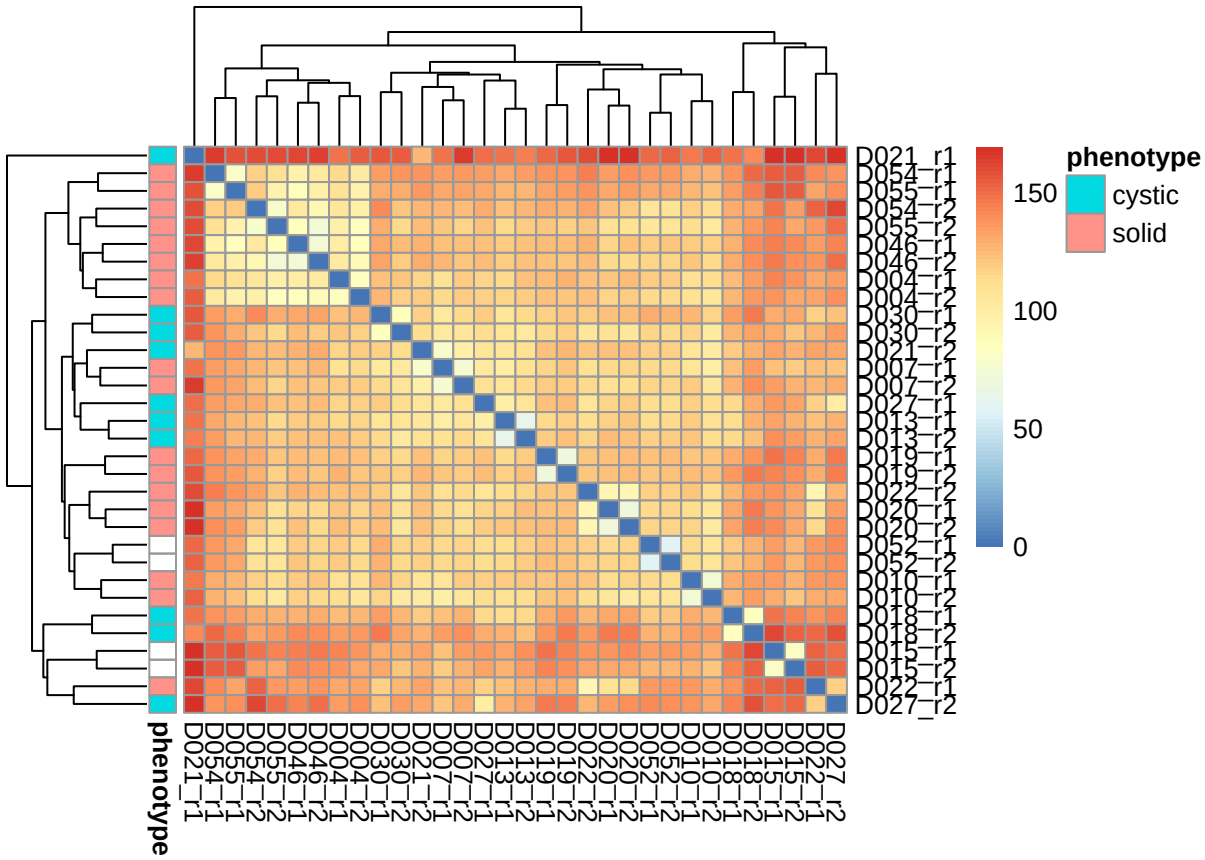
```
load(here('data/processed/expression/promise_expr.rda'))

organoid_morphology <- read_delim(here::here("references/imaging/visual_classification_organoids.csv"),
  dplyr::select(line = organoid, morphology = visual_inspection_v2)

## annotate phenotype group
solid <- c('D004', 'D007', 'D010', 'D019', 'D020',
           'D022', 'D046', 'D054', 'D055')
cystic <- c('D013', 'D018', 'D021', 'D027', 'D030')
```

```
## long data frame
promise_long <- assays(promise_expr)$expr %>%
  as_tibble(rownames = 'probe') %>%
  pivot_longer(values_to = 'expr', names_to = 'id', -probe) %>%
  left_join(as_tibble(rowData(promise_expr), rownames = 'probe')) %>%
  inner_join(as_tibble(colData(promise_expr), rownames = 'id')) %>%
  select(-chip_name)

## adding phenotype information
promise_long <- promise_long %>%
  mutate(phenotype = ifelse(line %in% solid, 'solid',
                     ifelse(line %in% cystic, 'cystic', 'other'))) %>%
  filter(phenotype != 'other')
```

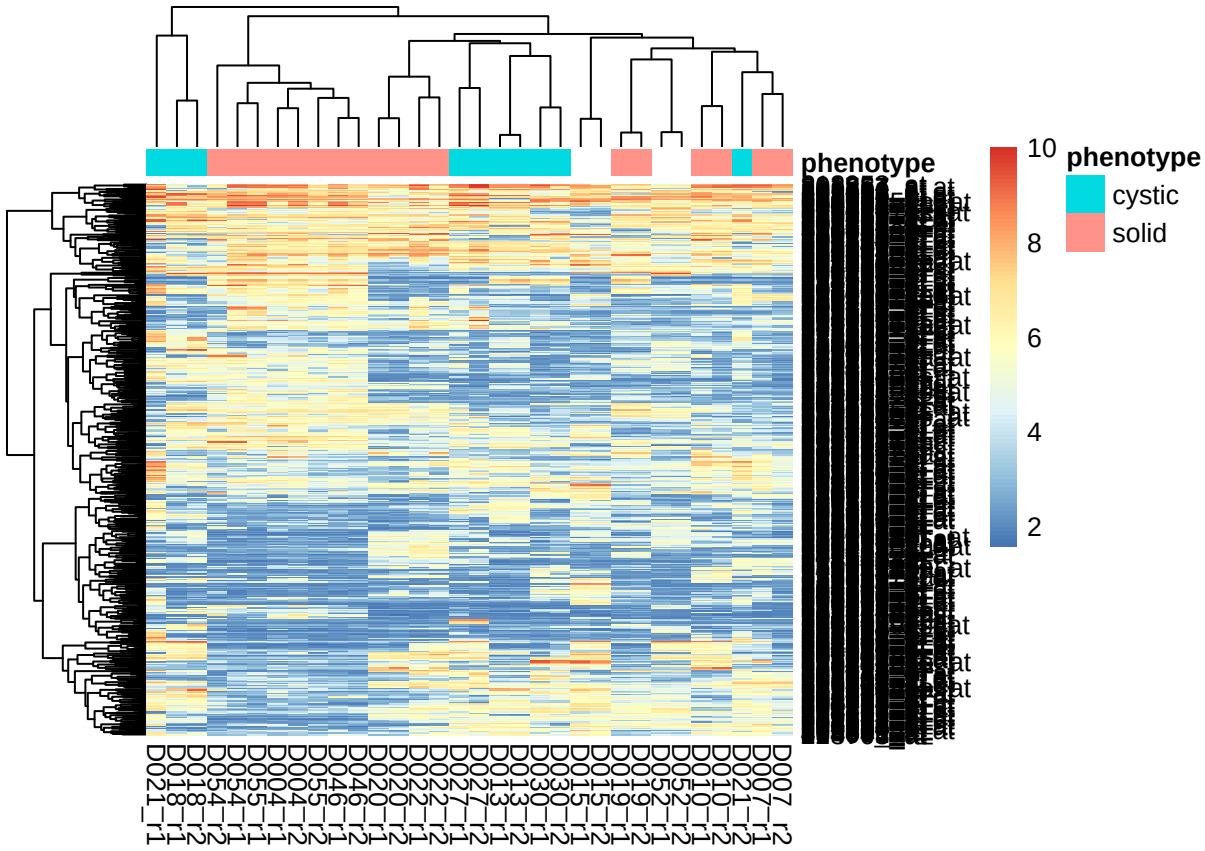
Quality control

We perform a clustering to check if the samples are grouped by line of origin.

```
## make expr matrix
expr_mat <- assays(promise_expr)$expr

## generate heatmap annotation
anno <- promise_long %>%
  distinct(line, rep, phenotype) %>%
  unite(id, line, rep) %>%
  mutate_all(as.factor) %>%
  as.data.frame() %>% column_to_rownames('id')

## visualize distance matrix
pheatmap(as.matrix(dist(t(expr_mat))),
         annotation_row = anno)
```




```
## visualize most variable probes
pheatmap(expr_mat[order(apply(expr_mat, 1, sd), decreasing = T)[1:500],],
         annotation_col = anno)
```



'D021_2017-05-19' looks like a technical outlier so we exclude D021T01 from the data. We do not see a clear separation of cystic and solid organoid lines based on the clustering.

D054T01 and D055T01 have been shown to be cross-contaminated, most likely from D046T01. We remove the samples from the dataset.

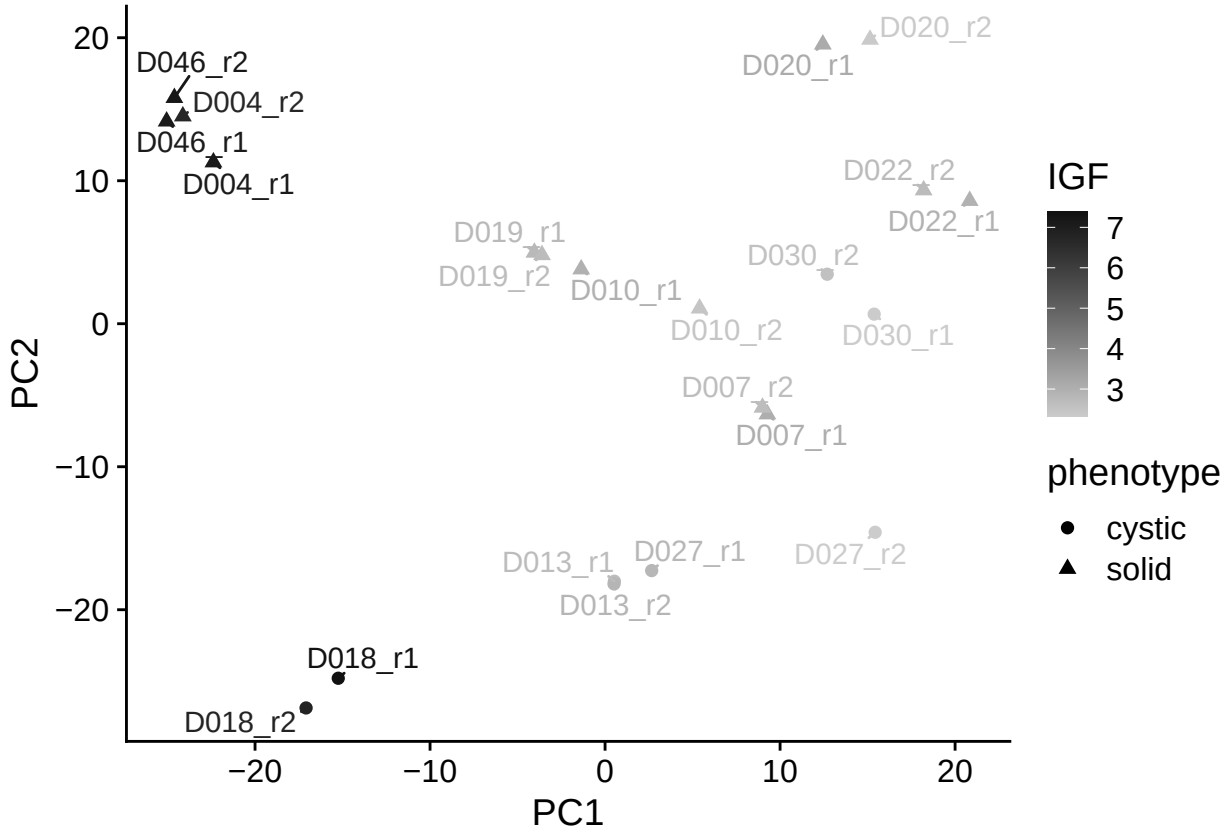
```
## exclude outlier
promise_long <- promise_long %>% filter(!line %in% c('D054', 'D055', 'D021'))
```

Unsupervised analysis

I perform a principal component analysis of individual lines at the gene level in order to develop insights into which factors drive variation in the gene expression data.

```
## select most highly expressed probe to represent each gene
select_probes <- promise_long %>% group_by(symbol, probe) %>%
  summarise(avg_probe = mean(expr)) %>% ungroup() %>%
  group_by(symbol) %>% top_n(1, avg_probe) %>% ungroup() %>% pull(probe)

## summarize replicates
gene_expr <- promise_long %>%
  # group_by(line, symbol, probe, phenotype) %>%
  # summarise(expr = mean(expr)) %>% ungroup() %>%
  filter(probe %in% select_probes)
```

```
## get to 1000 variable genes, do PCA
gene_expr_mat <- gene_expr %>% acast(symbol ~ line + rep,
                                     value.var = 'expr')
pca <- prcomp(t(gene_expr_mat[order(apply(gene_expr_mat, 1, sd), decreasing = T)[1:1000],]))

## annotate and plot pca results
pca_res <- as_tibble(pca$x, rownames = 'id') %>% select(id, PC1:PC5) %>%
  inner_join(distinct(gene_expr %>% distinct(id, phenotype))) %>%
  inner_join(gene_expr %>% filter(symbol == 'MIR483') %>%
               distinct(id, IGF = expr))

pca_res %>% ggplot(aes(PC1, PC2, color = IGF, shape = phenotype)) +
  geom_point(size = 2) +
  ggrepel::geom_text_repel(aes(label = id), min.segment.length = 0) +
  scale_color_gradient(low = '#cccccc', high = '#111111')
```

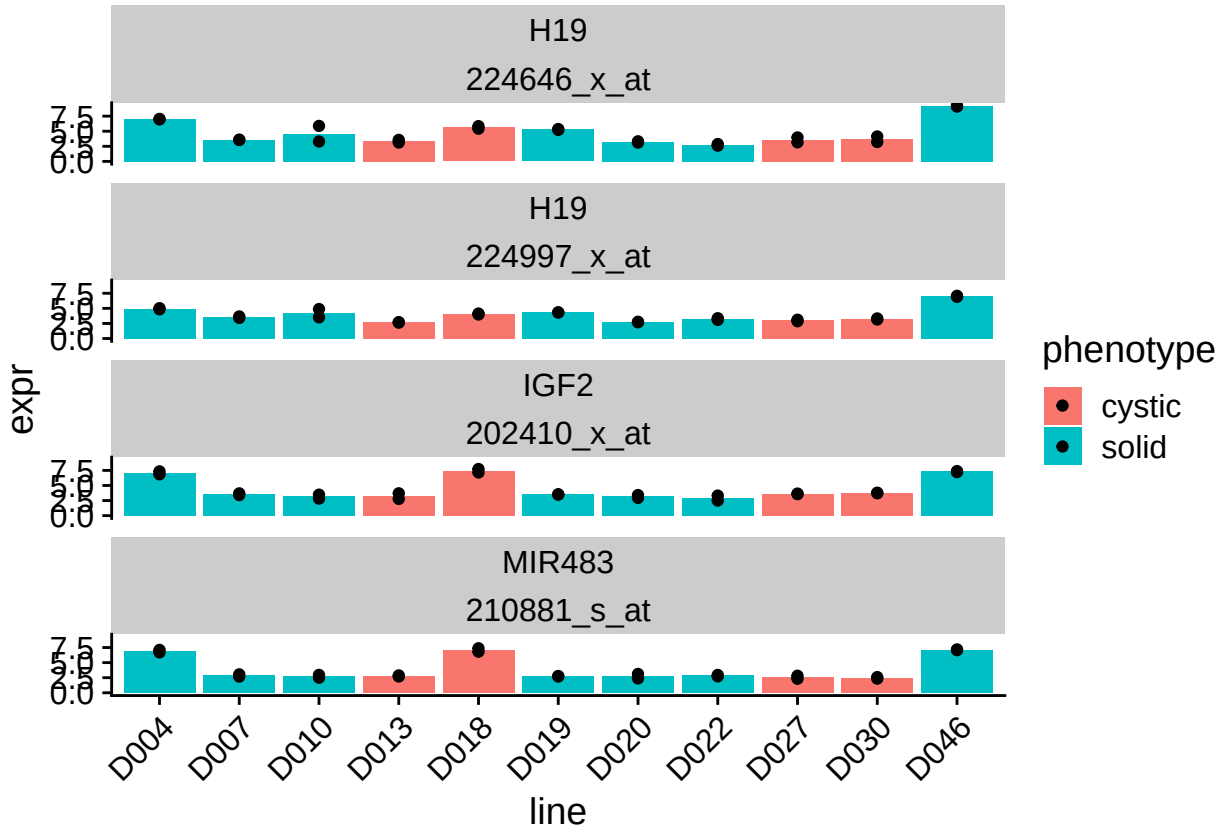



```
top_genes = select_probes %>% length()
top_genes = top_genes * 0.05
top_genes
```

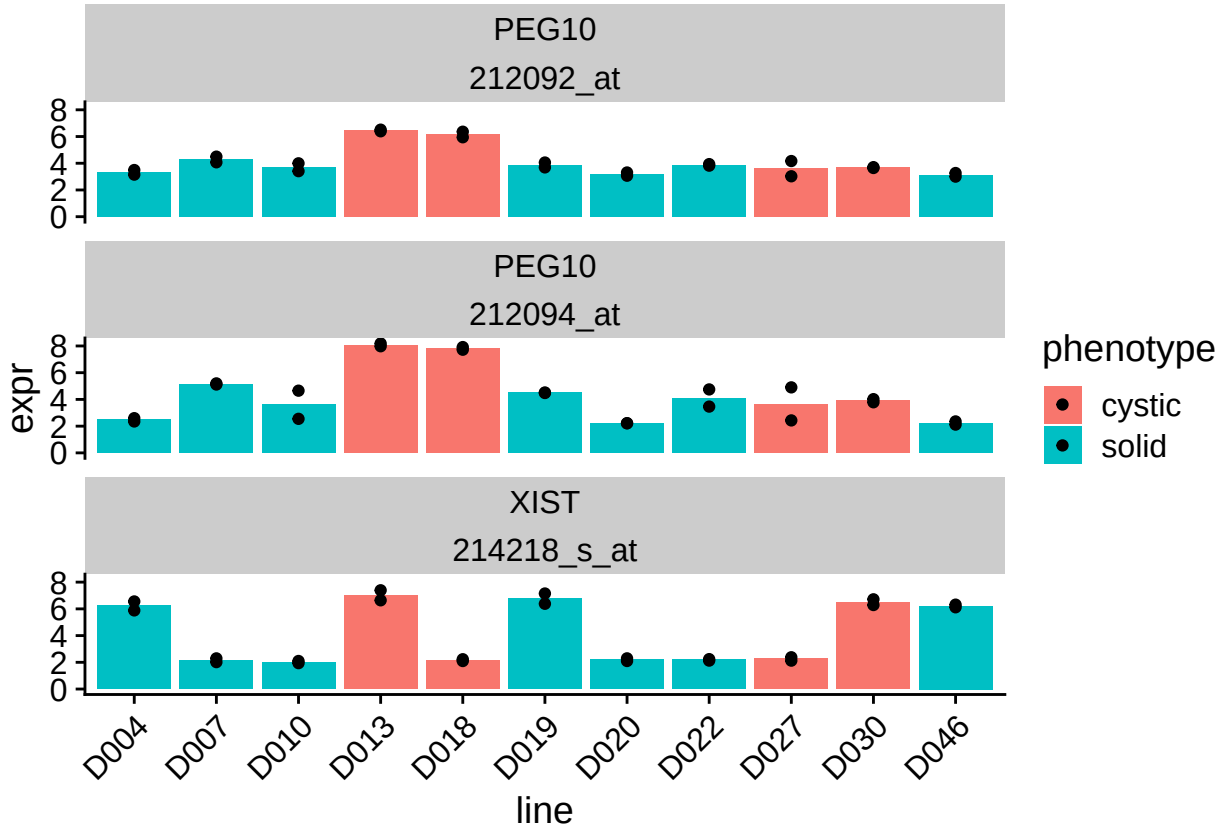
```
## [1] 922.85
```

I plot H19/IGF2 imprinting genes across lines.

```
promise_long %>% mutate(symbol = ifelse(probe == '202410_x_at', 'IGF2', symbol)) %>%
  filter(symbol %in% c('H19', 'MIR483', 'IGF2')) %>%
  ggplot(aes(line, expr, fill = phenotype)) +
  stat_summary(fun = 'mean', geom = 'bar') +
  geom_point() +
  facet_wrap(~ symbol + probe, ncol = 1) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
promise_long %>% filter(symbol %in% c('PEG10') | (probe == '214218_s_at')) %>%
  ggplot(aes(line, expr, fill = phenotype)) +
  stat_summary(fun = 'mean', geom = 'bar') +
  geom_point() +
  facet_wrap(~ symbol + probe, ncol = 1) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Expression markers

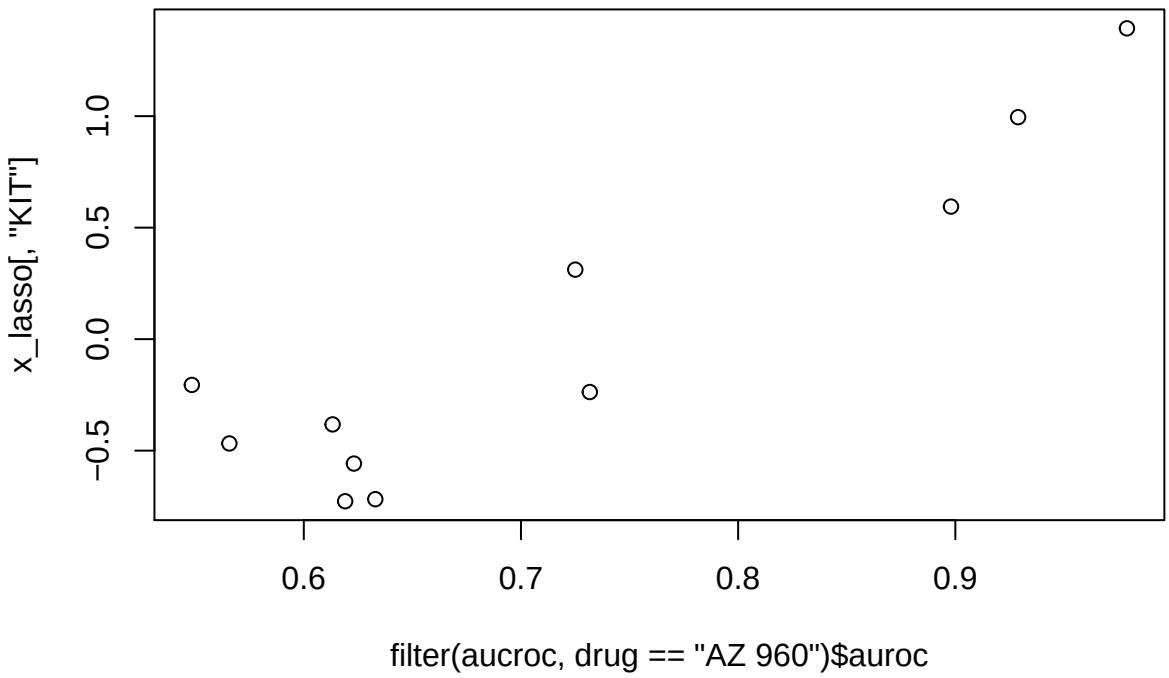
Do any of the gene expression probes predict whether an organoid line will respond to a drug treatment? I fit a penalized regression model with the AUROC curve from the SVM analysis as dependent and the gene expression measurements as independent variables. I use a LASSO appraoch to cut down on the number of coefficients since my sample sizes are very low.

```
## load auroc from file
data('aucroc', package = 'SCOPEAnalysis')

## exclude lknes 21, 54, 55
aucroc <- aucroc %>% filter(!line %in% c('D055T01', 'D021T01', 'D020T02', 'D021T01'))

## matrix with independnt variables (gene expression), merge reps
x_lasso <- promise_long %>%
  group_by(line, symbol, probe, phenotype) %>%
  summarise(expr = mean(expr)) %>% ungroup() %>%
  filter(probe %in% select_probes) %>%
  mutate(line = paste0(line, 'T01')) %>%
  acast(line ~ symbol, value.var = 'expr')
## center by subtracting means and retain genes with high var.
x_lasso <- apply(x_lasso, 2, function(x) x - mean(x))
x_lasso <- x_lasso[,order(apply(x_lasso, 2, var), decreasing=T)[1:1000]]
```

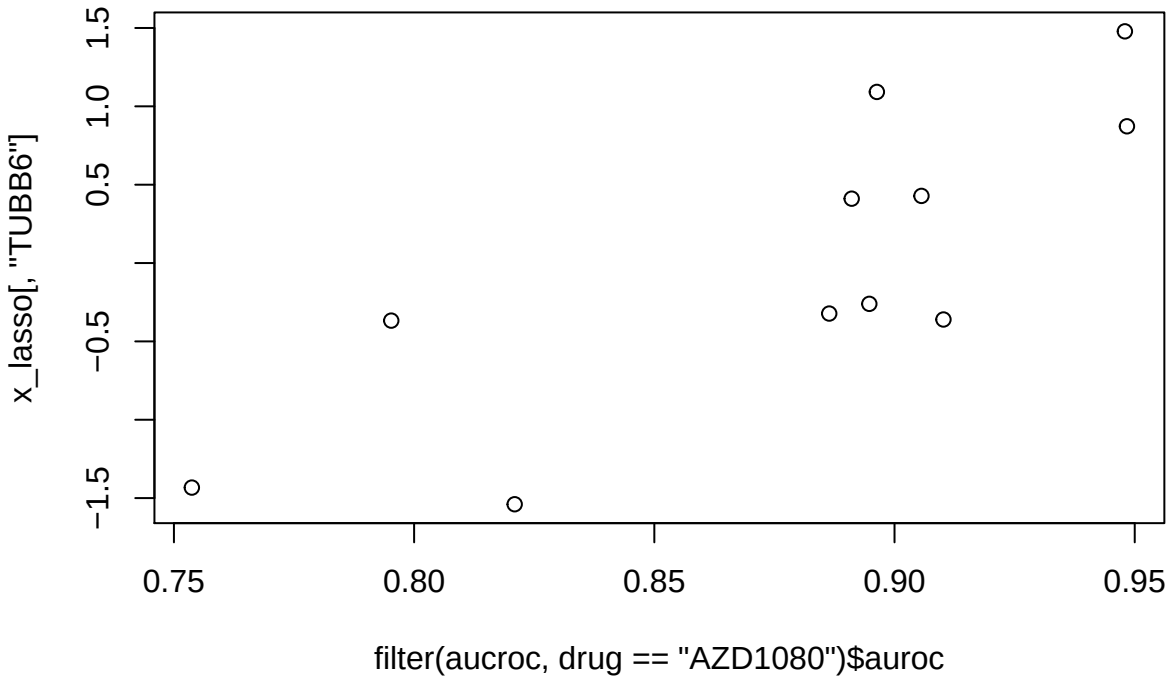
```
drug_expr <- aucroc %>% group_by(drug) %>%
  filter(!any(is.na(auroc))) %>%
  group_modify(~{
    ## fit model
    stopifnot(identical(rownames(x_lasso), .x$line))
    fit <- cv.glmnet(
      x = x_lasso,
      y = .x$auroc,
      family = 'gaussian',
      # alpha = 0.5,
      grouped = F
    )
    ## extract non-zero coefficients
    as_tibble(as.matrix(coef(fit, s = fit$lambda.min)),
              rownames = 'symbol') %>%
      `colnames<-`(c('symbol', 'coef')) %>%
      filter(coef > 0)
  }) %>% ungroup()

## check best coefficients
drug_expr <- drug_expr %>% filter(symbol != '(Intercept)') %>%
  arrange(desc(coef))
```

Check if there are correlations in the gene expression data for some of the top coefficients.

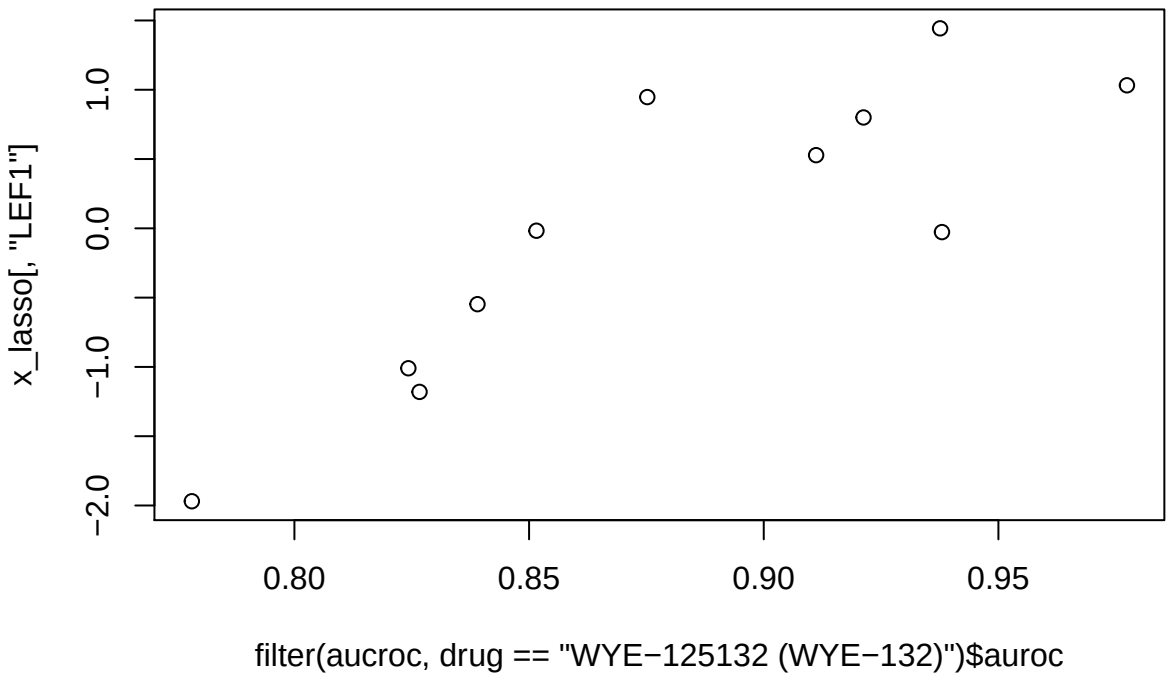
```
plot(filter(aucroc, drug == 'AZ 960')$auroc, x_lasso[,'KIT'])
```



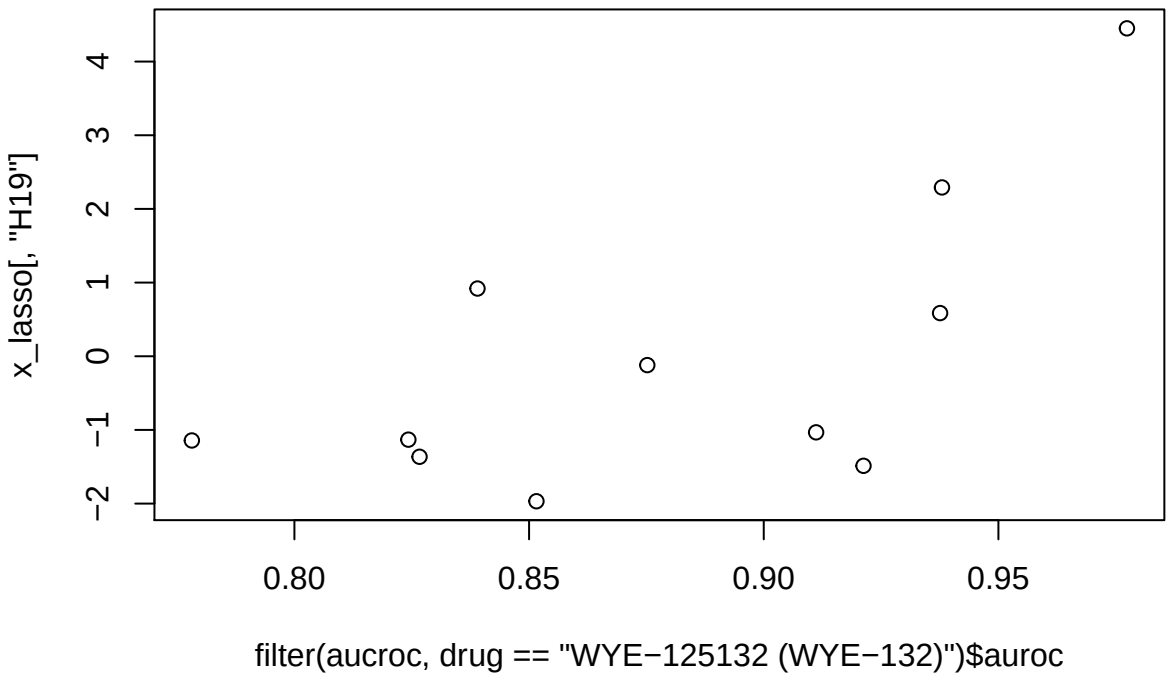

```
plot(filter(aucroc, drug == 'AZD1080')$auroc, x_lasso[,'TUBB6'])
```




```
plot(filter(aucroc, drug == 'WYE-125132 (WYE-132)')$auroc, x_lasso[,'LEF1'])
```




```
plot(filter(aucroc, drug == 'WYE-125132 (WYE-132)')$auroc, x_lasso[,'H19'])
```



I load drug target annotations from file. Do the same drugs have the same targets?

```
## load drug annotation
drug_anno <- readxl::read_excel('data/Compound_Annotation_Libraries_New.xlsx') %>% distinct(drug = 'Com

## annotate model coefficients
drug_expr <- drug_expr %>% left_join(drug_anno)
```

Generate network diagram.

```
library(tidygraph)
library(ggraph)

## correlation between gene expr as links
genes <- drug_expr %>% pull(symbol)
gene_cor <- cor(x_lasso[,colnames(x_lasso) %in% genes])
diag(gene_cor) <- NA
gene_cor <- gene_cor %>% as_tibble(rownames = 'source') %>%
  pivot_longer(names_to = 'target', values_to = 'val', -source) %>%
  filter(val > 0.85) %>%
  mutate(edge_type = 'gene_cor')

## lasso coefficients
drug_assoc <- drug_expr %>%
  select(source = drug, target = symbol, val = coef) %>%
  mutate(edge_type = 'drug_assoc')
```

```
## combine and make graph
g <- gene_cor %>% bind_rows(drug_assoc) %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  arrange(name) %>%
  mutate(type = ifelse(name %in% genes, 'gene', 'drug'))

## drug target annotation for network
targ_anno <- drug_anno %>% rename(name = drug) %>%
  bind_rows(distinct(tibble(name = genes, target = 'none'))) %>%
  filter(!is.na(name), name %in% as_tibble(g)$name) %>%
  arrange(name)

g <- g %>% activate(nodes) %>% mutate(target = targ_anno$target)

## visualize
g %>% ggraph(layout = 'kk') +
  geom_edge_link(aes(color = edge_type)) +
  geom_node_point(size = 2, aes(color = target)) +
  geom_node_text(aes(label = name), colour = 'black', vjust = 0.4)
```

Session info

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.2 LTS
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.8.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=C
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] here_0.1                cowplot_1.0.0
##  [3] forcats_0.5.0           stringr_1.4.0
##  [5] dplyr_1.0.0             purrr_0.3.4
##  [7] readr_1.3.1             tidyr_1.1.0
##  [9] tibble_3.0.1            ggplot2_3.3.1
```

```
## [11] tidyverse_1.3.0             reshape2_1.4.4
## [13] pheatmap_1.0.12             glmnet_4.0
## [15] Matrix_1.2-18               DESeq2_1.30.1
## [17] SummarizedExperiment_1.20.0 Biobase_2.50.0
## [19] GenomicRanges_1.42.0        GenomeInfoDb_1.26.7
## [21] IRanges_2.24.1              S4Vectors_0.28.1
## [23] BiocGenerics_0.36.1         MatrixGenerics_1.2.1
## [25] matrixStats_0.59.0
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-147           bitops_1.0-6           fs_1.4.1
##  [4] lubridate_1.7.8        bit64_0.9-7            RColorBrewer_1.1-2
##  [7] httr_1.4.1             rprojroot_1.3-2        tools_4.0.0
## [10] backports_1.1.7        R6_2.4.1               DBI_1.1.0
## [13] colorspace_1.4-1       withr_2.2.0            tidyselect_1.1.0
## [16] bit_1.1-15.2           compiler_4.0.0         cli_2.0.2
## [19] rvest_0.3.5            xml2_1.3.2             DelayedArray_0.16.3
## [22] labeling_0.3           scales_1.1.1           genefilter_1.72.1
## [25] digest_0.6.25          rmarkdown_2.2          XVector_0.30.0
## [28] pkgconfig_2.0.3        htmltools_0.4.0        dbplyr_1.4.4
## [31] rlang_0.4.6            readxl_1.3.1           rstudioapi_0.11
## [34] RSQLite_2.2.0          farver_2.0.3           shape_1.4.4
## [37] generics_0.0.2         jsonlite_1.6.1         BiocParallel_1.24.1
## [40] RCurl_1.98-1.2         magrittr_1.5           GenomeInfoDbData_1.2.4
## [43] fansi_0.4.1            Rcpp_1.0.4.6           munsell_0.5.0
## [46] lifecycle_0.2.0        stringi_1.4.6          yaml_2.2.1
## [49] zlibbioc_1.36.0        plyr_1.8.6             grid_4.0.0
## [52] blob_1.2.1             ggrepel_0.8.2          crayon_1.3.4
## [55] lattice_0.20-41        haven_2.3.1            splines_4.0.0
## [58] annotate_1.68.0        hms_0.5.3              locfit_1.5-9.4
## [61] knitr_1.28             pillar_1.4.4           geneplotter_1.68.0
## [64] codetools_0.2-16       reprex_0.3.0           XML_3.99-0.3
## [67] glue_1.4.1             evaluate_0.14          modelr_0.1.8
## [70] vctrs_0.3.1            foreach_1.5.0          cellranger_1.1.0
## [73] gtable_0.3.0           assertthat_0.2.1       xfun_0.14
## [76] xtable_1.8-4           broom_0.5.6            survival_3.1-12
## [79] iterators_1.0.12       AnnotationDbi_1.52.0   memoise_1.1.0
## [82] ellipsis_0.3.1
```