Lineage specific core-regulatory circuits determine gene essentiality in cancer cells

Benedikt Rauscher

2019-04-14

Abstract

Cancer cells rely on dysregulated gene expression programs to maintain their malignant phenotype. A cell's transcriptional state is controlled by a small set of interconnected transcription factors that form its core-regulatory circuit (CRC). Previous work in pediatric cancers has shown, that disruption of the CRC by genetic alterations causes tumor cells to become highly dependent on its components creating new opportunities for therapeutic intervention. However, the role of CRCs and the mechanisms by which they are controlled remain largely unknown for most tumor types. Here, we developed a method to systematically predict 'functional' CRCs and associated biological processes from context-dependent essentiality data sets. Analysis of genome-scale CRISPR-Cas9 screens in 558 cancer cell lines showed that most tumor types specifically depend on a small number of transcription factors for proliferation. We found that these transcription factors compose the CRCs in these tumor types. Moreover, they are frequently altered in patient tumor samples indicating their oncogenic potential. Finally, we show that biological processes associated with each CRC are revealed by analyzing codependency between lineage-specific essential genes. Our results demonstrate that genetic addiction to lineage-specific core transcriptional mechanisms occurs across a broad range of tumor types. We exploit this phenomenon to systematically infer CRCs from lineage specific gene essentiality. Furthermore, our findings shed light on the selective genetic vulnerabilities that arise as the consequence of transcriptional dysregulation in different tumor types and show how the plasticity of regulatory circuits might influence drug resistance and metastatic potential.

Contents

1 About

This document contains computer code to reproduce analyses and figures presented in the corresponding manuscript.

2 Dependencies

We load a number of packages which provide functionality that is required for downstream analyses.

```
library(broom)
library(reshape2)
library(mixtools)
library(pheatmap)
library(fgsea)
library(GO.db)
library(Organism.dplyr)
library(patchwork)
library(ggrepel)
library(RTCGA)
library(RTCGA.rnaseq)
library(MASS)
library(Gviz)
library(biomaRt)
library(org.Hs.eg.db)
library(mclust)
library(clusterProfiler)
library(limma)
library(Rtsne)
library(cowplot)
library(tidyverse)
```

3 Context-dependent essential genes are enriched for lineage dependency transcription factors

3.1 Data

We next load a number of datasets that downstream analyses are based on. These include formatted CERES scores for the DepMap 19Q1 dataset and a list of transcription factors in the human genome (as determined by Lambert et al., 2018).

```
## The CERES scores for DepMap 19Q1
data('depmap_ceres', package='HDCRC2019')
```

```
## list of TF
data('tf_list', package='HDCRC2019')
```

3.2 Context-specific gene dependencies

Context-dependent gene dependency can be detected by an increased dropout compared to the 'null distribution' that represents the baseline phenotype of a gene knockout. To infer the null distribution, we center CERES scores for each gene. To this end we fit a Gaussian mixture model with 2 components to the distribution of CERES scores for each gene. Assuming that phenotypes for a gene knockout are similar for the majority of cell lines, we select the component that represents most of the data as the 'null-component' and save its parameters (mean and standard deviation).

```
## mixture modelling involves some RNG
set.seed(11111)

pb <- progress_estimated(length(unique(depmap_ceres$symbol)))
mm_set <- depmap_ceres %>% mutate(s2 = symbol) %>%
  group_by(symbol) %>%
  group_map(~ {
    pb$tick()$print()
    ## fit gaussian mixture
    mm <- tryCatch(normalmixEM(.x$cscore,
                               lambda = c(0.2, 0.8),
                               mu = c(-0.5, 0)),
                   error=function(cond) return(NULL))
    if(!is.null(mm)){
      ## select largest component
      comp_count <- as_tibble(mm$posterior) %>%
        mutate(comp = ifelse(comp.1 > comp.2, 1, 2)) %>%
        count(comp)
      compl <- comp_count %>% arrange(desc(n)) %>% dplyr::slice(1) %>%
        pull(comp)
      comp_frac <- `/`(comp_count %>% filter(comp == compl) %>% pull(n),
                       sum(comp_count$n))
      ## extract paramters
      tibble(
        mu_null = mm$mu[compl],
        mu_alternative = mm$mu[-compl],
        sigma_null = mm$sigma[compl],
        sigma_alternative = mm$sigma[-compl],
        comp_size = comp_frac,
        loglik = mm$loglik,
        converge = ifelse(length(mm$all.loglik) >= 1000, F, T),
        restarts = mm$restarts
      )
    } else {
      tibble(
        mu_null = NA,
```

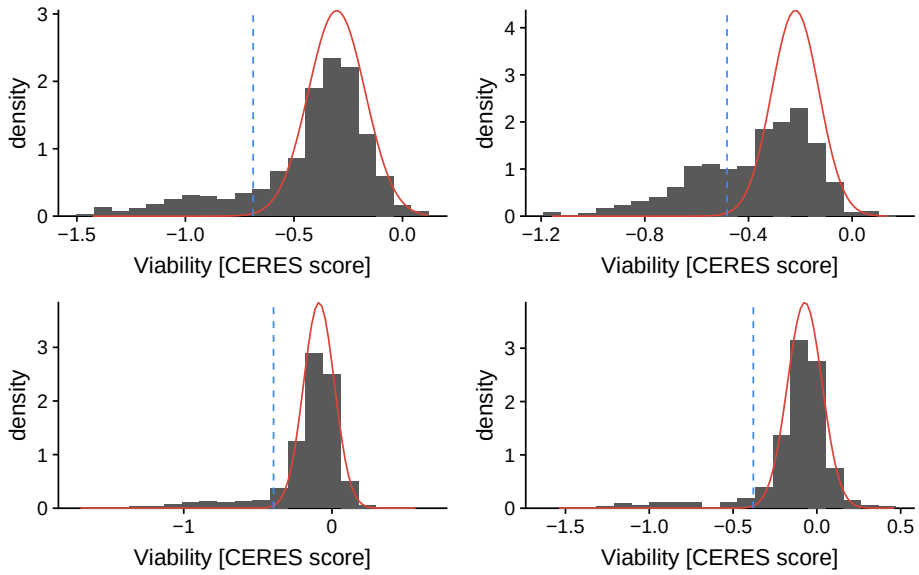
```
        sigma_null = NA
      )
    }
  }) %>% ungroup()
```

We next calculate P-values for each phenotype that represents the probability of having observed the phenotype under the inferred null distribution of knockout phenotypes for the same gene across all cell lines. We select gene dependencies as context-sepcific if FDR $<$ 20%.

```
## calculate P-values, name context-dependent genes
depmap_ceres <- depmap_ceres %>% left_join(distinct(mm_set)) %>%
  mutate(pval_cd = 2*(pnorm(cscore, mean = mu_null, sd=sigma_null)),
         FDR_cd = p.adjust(pval_cd, method='BH'),
         context_ess = ifelse(FDR_cd < 0.2, T, F))
```

We can visualize the results looking at examples of oncogenes that are expected to show contex-specific essentiality (such as KRAS, BRAF, etc.). We plot the distribution of phenotypes across all cell lines, highlight the inferred null-distribution of baseline phenotypes and highlight the 20% FDR cutoff.

```
## plot centered CERES scores indicating null distr
plot_example_nulldistr <- function(gene){
  ex_distr <- depmap_ceres %>% filter(symbol == gene)
  ex_mu <- ex_distr %>% pull(mu_null) %>% head(1)
  ex_sig <- ex_distr %>% pull(sigma_null) %>% head(1)

  ## aproximate FDR 20% cutoff
  co <- ex_distr %>% filter(FDR_cd < 0.2) %>%
    arrange(desc(pval_cd)) %>% pull(cscore) %>% head(1)

  ex_distr %>% ggplot(aes(cscore)) +
    geom_histogram(aes(y=..density..), bins=20) +
    stat_function(fun = dnorm, colour = '#db4437',
                  args=list(mean = ex_mu, sd = ex_sig)) +
    scale_y_continuous(expand=c(0,0)) +
    geom_vline(xintercept = co, linetype = 'dashed', colour='#4285f4') +
    xlab('Viability [CERES score]')
}

plot_example_nulldistr('KRAS') +
plot_example_nulldistr('PIK3CA') +
plot_example_nulldistr('BRAF') +
plot_example_nulldistr('NRAS') +
  plot_layout(c(2,2))
```

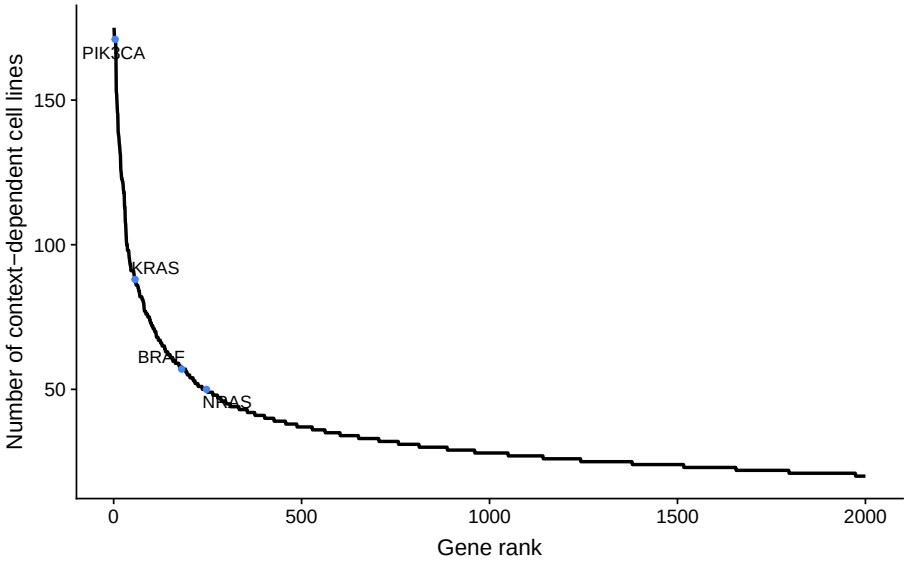


For each gene, we next count the number of cell lines for which we detect context-specific gene dependency and rank them accordingly.

```
## count number of cell lines with context-specific phenotype for each gene
context_spec_genes <- depmap_ceres %>% filter(context_ess) %>%
  count(symbol) %>% arrange(desc(n)) %>% mutate(rank = 1:n())
```

We can now draw a waterfall plot highlighting some oncogenes that are known to be context-specific essential genes.

```
## oncogenes to highlight
oncogenes <- filter(context_spec_genes,
                    symbol %in% c('BRAF', 'NRAS', 'KRAS', 'PIK3CA'))

## waterfall plot
context_spec_genes %>% dplyr::slice(1:2000) %>%
  ggplot(aes(rank, n)) + geom_line(lwd=1) +
  geom_point(data = oncogenes, aes(rank, n),
             colour = '#4285f4') +
  geom_text_repel(data = oncogenes, aes(label = symbol)) +
  xlab('Gene rank') + ylab('Number of context-dependent cell lines')
```



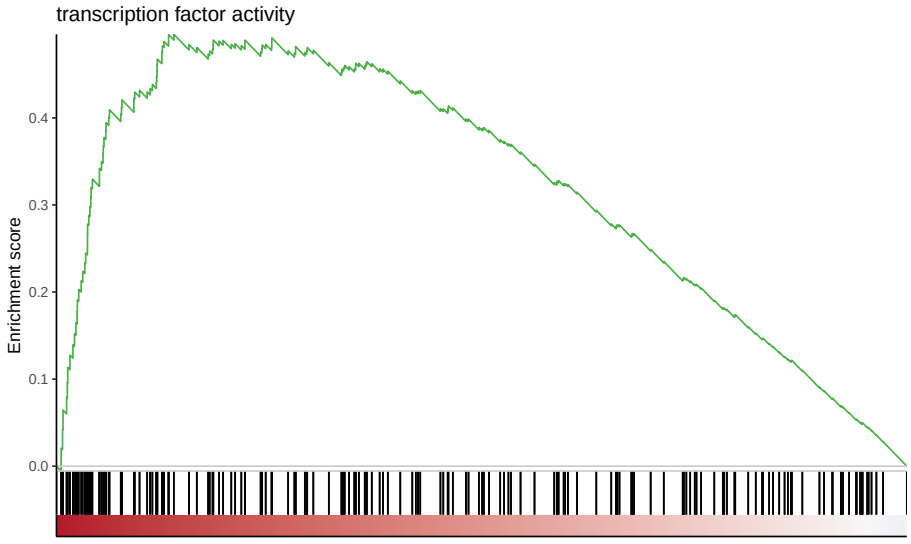
Next, we perform a pre-ranked gene set enrichment on the ranked gene list using the 'fgsea' package. This identifies enriched GO molecular function terms. We retrieve GO Terms and annotations using the GO.db and Organism.dplyr Bioconductor packages. We look at gene sets of size 15-500 using 100,000 permutations for fgsea.

```
## gene ranks
fgsea_ranks <- setNames(context_spec_genes$n, context_spec_genes$symbol)

## go molecular function
src <- src_organism('TxDb.Hsapiens.UCSC.hg38.knownGene')
goterms <- tbl(src, 'id') %>% distinct(symbol, entrez) %>%
  filter(symbol %in% context_spec_genes$symbol) %>%
  inner_join(tbl(src, 'id_go') %>% filter(ontology == 'MF')) %>%
  collect(n=Inf) %>% mutate(go_term = Term(GOTERM)[go])

## for fgsea
fgsea_go <- goterms %>% split(.$go_term) %>% map(~ unique(.x$symbol))

## run enrichment
fgsea_results = fgsea(pathways = fgsea_go,
                      stats = fgsea_ranks,
                      minSize = 15, maxSize = 500,
                      nperm = 1e5)
```

We make a barcode plot to visualize the enrichment results. Here we show that genes with transcription factor activity are enriched among the highly ranking genes. To this end we use a customized version of the barcode plot function that is implemented in the fgsea package. The 'fgsea_results' object (above) contains enrichement scores and P-values for each term.

```
## a function to draw barcode plots
custom_barcode_plot <- function(stat_vector, sig_genes, term){
  ## generate barcode plot
  bc_plot <- plotEnrichment(sig_genes, stat_vector)
```

```
  ## remove unwanted layers
  bc_plot$layers <- list()

  ## add barcode at the bottom
  lowest_pos <- min(bc_plot$data[,2])
  dash_length <- abs(purrr::reduce(range(bc_plot$data[,2]), `-`)*0.1)
  middle <- which.min(abs(sort(stat_vector, decreasing=T)))

  bc_plot_custom <- bc_plot + geom_segment(aes(x=x, xend=x), y=lowest_pos,
                           yend=lowest_pos-dash_length) +
    geom_line(colour='#4daf4a') +
    geom_hline(yintercept=lowest_pos, colour='#cccccc') +
    geom_hline(yintercept=0, colour='#cccccc') + xlab('') +
    theme_classic() +
    geom_tile(data=tibble(rank=1:length(stat_vector),
                          y=lowest_pos-(1.25*dash_length)),
              aes(x=rank, y=y, fill=rank),
                  width=1,
                  height=0.5*dash_length) +
    scale_fill_gradient2(low ='#b2182b', high='#2166ac',
                         mid='#f7f7f7', midpoint = middle) +
    scale_x_continuous(expand = c(0, 0)) +
    scale_y_continuous(expand = c(0, 0)) +
    theme(panel.grid=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x = element_blank(),
          legend.position = 'none') +
    ggtitle(term) +
    ylab('Enrichment score')

  return(bc_plot_custom)
}

signature_genes <- goterms %>%
  filter(grepl('transcription regulatory region DNA binding', go_term)) %>%
  pull(symbol) %>% unique()

custom_barcode_plot(fgsea_ranks, signature_genes,
                    'transcription factor activity')
```



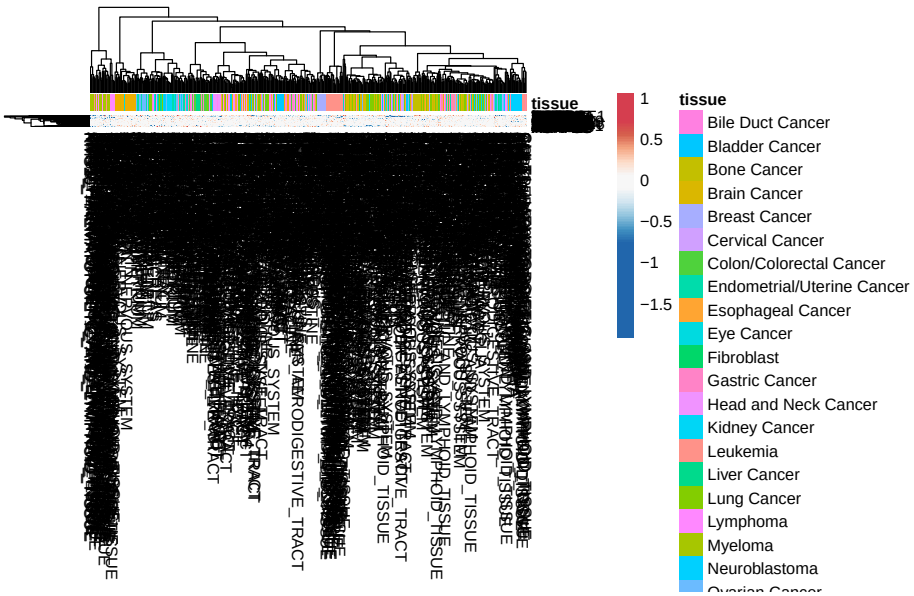
This shows that among all context-specific gene depenencies, transcription factor proteins are highly overrepresented.

3.3 Clustering context-dependent transcription factors reveals lineage specific subgroups

In order to observe how these context-dependent transcription factors group together, we plot a heatmap of their (centered) CERES scores. Specifically we select genes that are context-specific essential in more than 25 cell lines. We use centered phenotypes where the mean of the null-phenotype distribution was subtracted from each CERES score (see above). We use the ward.D2 method for clustering.

```
## colour palette is complex due to outliers
hm_cols <- rev(c(rep('#d53e4f', 3), # with tp53 i need 12
             rep('#d6604d', 1), '#f4a582', # with tp53 10
             '#fddbc7', rep('#f7f7f7', 2), '#d1e5f0',
             '#92c5de','#4393c3', rep('#2166ac', 8))) # tp53 8

## context-dependent transcription factors
context_tf <- context_spec_genes %>%
  filter(n > 25, symbol %in% tf_list) %>% pull(symbol)

## cell lines that are vulnerable to one of these TF
context_cl <- depmap_ceres %>%
  filter(symbol %in% context_tf, context_ess) %>%
  pull(cellline)

## row annotation of tissues
row_anno <- distinct(depmap_ceres, sample, tissue) %>%
  as.data.frame() %>% column_to_rownames('sample')

tf_dep_mat <- depmap_ceres %>%
  filter(symbol %in% context_tf, cellline %in% context_cl) %>%
```

```
  group_by(symbol) %>%
  mutate(cscore_ct = cscore - mu_null) %>% ungroup() %>%
  acast(symbol ~ sample, value.var = 'cscore_ct') %>%
  apply(1, function(x) x - median(x))
tf_dep_mat %>% t() %>%
  pheatmap(color = colorRampPalette(c(hm_cols))(150),
           annotation_col = row_anno,
           clustering_method = 'ward.D2')
```



3.4 Systematic identification of lineage-dependency transcription factors

3.4.1 LD-TF examples

Since the heatmap suggests that the selective dependency on transcription factor genes is related to the cancer lineage we perform an anlysis to identify all lineage-specifically essential genes and assign them to a tumor type. We infer a measure of the strength of the selective dependency, which we term LD-(lineage dependency) score and associated p-values which signal whether an LD-score significantly differs from 0.

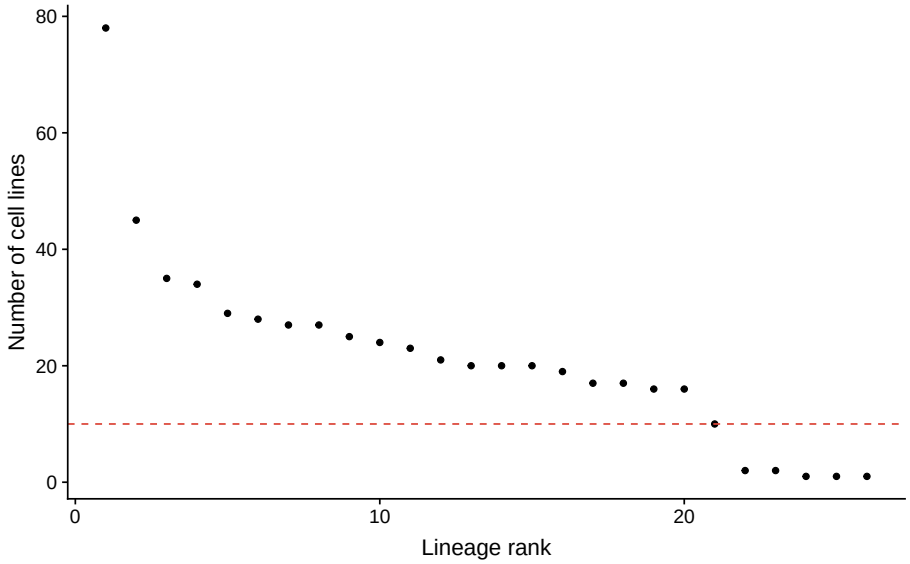
In order to visualize examples we first define a function that can plot these LD-scores. We do this only for cancer types that are sufficiently represented in the data at n $>$ 10.

```
plot_mr_score <- function(gene, ctype, plot_tissues){
  df <- depmap_ceres %>%
    filter(symbol == gene, tissue %in% plot_tissues) %>%
    mutate(cscore_sc = -1*((cscore-mu_null)/sigma_null)) %>%
    group_by(tissue) %>% mutate(mn = mean(cscore)) %>% ungroup() %>%
    arrange(mn) %>% mutate(tissue = factor(tissue, levels = unique(.$tissue))) %>%
    mutate(highlight = ifelse(tissue %in% ctype, T, F)) %>%
```

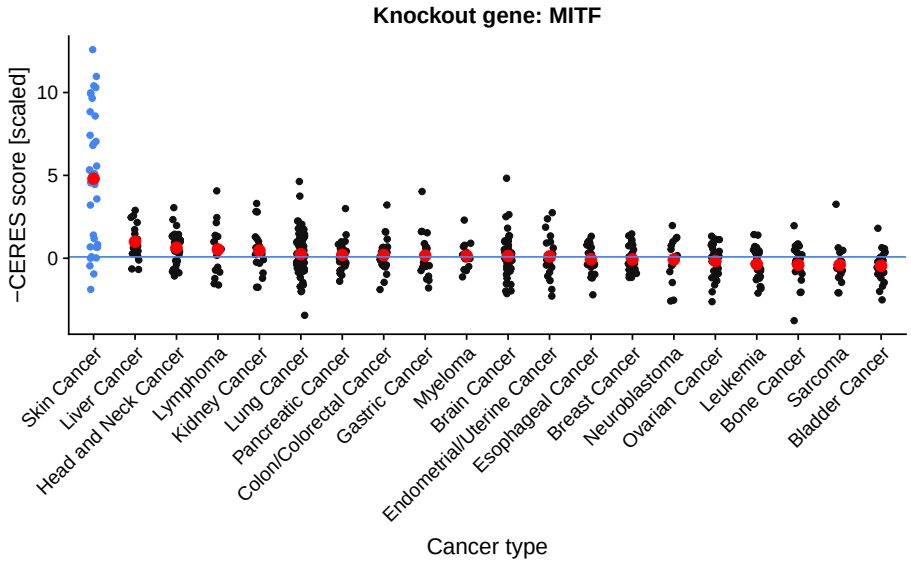
```
    filter(!is.na(tissue))

  mean_line <- df$mu_null[1]

  df %>%
    ggplot(aes(tissue, cscore_sc, colour=highlight)) +
    geom_jitter(width = 0.1) +
    stat_summary(fun.y = 'mean', fun.ymax = 'mean',
                 fun.ymin = 'mean', geom='point', size=3, colour='red') +
    geom_hline(yintercept = mean_line, colour='#4285f4') +
    theme(axis.text.x = element_text(angle=45, hjust=1),
          legend.position = 'none') +
    scale_colour_manual(values=c('#111111', '#4285f4')) +
    ylab('-CERES score [scaled]') + xlab('Cancer type') +
    ggtitle(paste('Knockout gene:', gene))
}
```

We select sufficiently represented cancer types and plot a few examples that stood out in the heatmap.

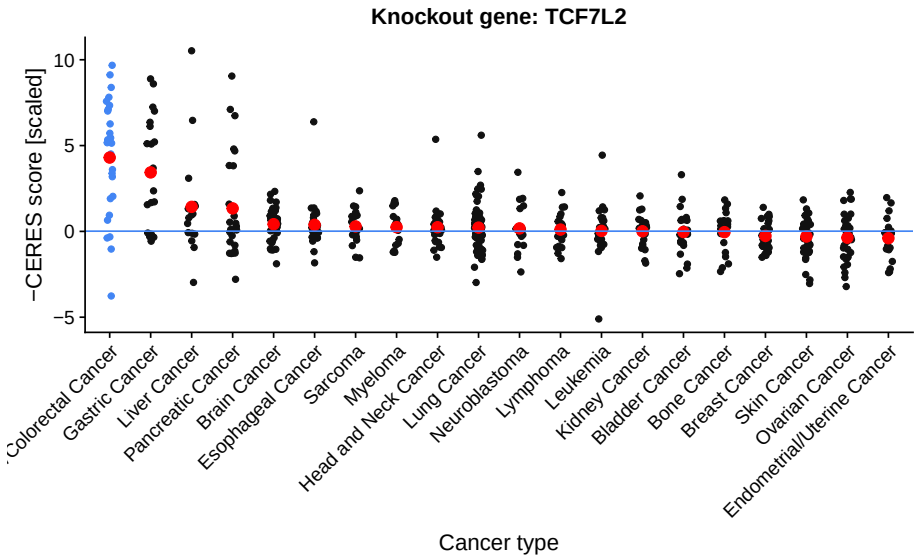
```
## select tissues represented at sufficient frequency
pt <- depmap_ceres %>% distinct(tissue, cellline) %>%
  count(tissue) %>% arrange(desc(n)) %>% filter(n > 10) %>%
  pull(tissue)

## draw a plot that indicates why 10 might be a reasonable threshold
depmap_ceres %>% distinct(tissue, cellline) %>%
  count(tissue) %>% arrange(desc(n)) %>% mutate(rank = 1:n()) %>%
  ggplot(aes(rank, n)) + geom_point() +
  geom_hline(yintercept = 10, colour = '#db4437', linetype = 'dashed') +
  ylab('Number of cell lines') + xlab('Lineage rank')
```



```
## plot LD-scores for tissues of interest
plot_mr_score('MITF', 'Skin Cancer', pt)
```




```
plot_mr_score('TCF7L2', 'Colon/Colorectal Cancer', pt)
```



3.4.2 Systematic identification of lineage dependency genes

Using a linear model based appraoch we now systematically calculate lineage dependency scores for each genes in each tumor type. Before that we annotate TP53 mutation status for each cell line so we can account for it during LD-score calculation. We do this because TP53 occur very frequently and previous studies have shown that TP53 status affects the gene essentiality profile of a cell line in a tissue-independent manner. The mutation status is inferred from mutation data in the Cancer Cell Line Encyclopedia and exclues silent mutations.

```
## load tp53 mutation status in cancer cell lines
data('tp53_status', package='HDCRC2019')
```

3.4.3 Transcription factors

We first calculate LD-scores for transcription factors (Lamber et al, 2018) in CRISPR-Cas9 screens. We exclude SOX9, which has been shown to be affected by off-target effects in the Avana library (Fortin et al., 2019) and TP53 since we account for its mutation status.

```
## subset of the data containing only TF
ceres_tf <- depmap_ceres %>%
  filter(symbol %in% tf_list, tissue %in% pt, !is.na(mu_null),
         !symbol %in% c('SOX9', 'TP53'))

## progress bar
pb <- progress_estimated(length(unique(ceres_tf$symbol)))

ldtf_crispr <- ceres_tf %>% inner_join(tp53_status) %>%
  group_by(symbol) %>%
  mutate(cscore_sc = -1*((cscore - mu_null)/sigma_null)) %>% ungroup() %>%
  nest(-symbol) %>% mutate(mr_score = purrr::map(data, ~{
    pb$tick()$print()
    lm(cscore_sc ~ 0 + tissue + TP53_status, data = .x) %>%
      tidy() %>% tbl_df()
  })) %>% unnest(mr_score) %>%
  inner_join(ceres_tf %>% group_by(symbol) %>%
               summarise(main_effect = median(cscore)) %>% ungroup()) %>%
  filter(!term %in% c('TP53_statusmut', 'TP53_statuswt')) %>%
  mutate(FDR = p.adjust(p.value, method='BH')) %>%
  arrange(p.value) %>%
  mutate(tissue = gsub('^tissue', '', term)) %>% dplyr::select(-term)
```

3.4.4 Non-transcription factors

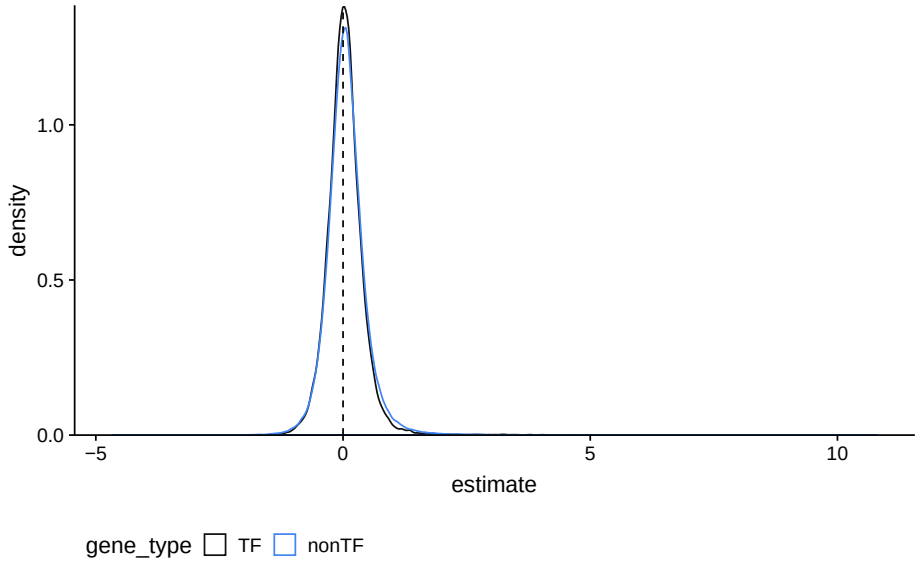
We assume that, since phenotypes of transcription factors are tissue specific, there might be tissue-specific mechanisms that regulate them. Understanding these mechanisms might be interesting for a variety of reasons. If a tissue specific mechanism exists that regulates a transcription factor showing a context-specific phenotype, other genes contributing to that mechanism might show similar phenotypes. Hence, we perform a similar analysis to the above, where we scan for tissue-essential genes that are not transcription factors. We exclude genes that are broadly non-essential across DepMap CRISPR screens (all CERES scores $>$ -0.5).

```
## only non-tf that are important at least in some lines
ceres_nontf <- depmap_ceres %>%
  filter(!symbol %in% tf_list, tissue %in% pt) %>%
  group_by(symbol) %>% filter(min(cscore) < -0.5) %>% ungroup()

## progress bar
genes <- distinct(ceres_nontf, symbol) %>% pull(symbol)
```

```
pb <- progress_estimated(length(unique(genes)))

## linear model
ldnontf_crispr <- ceres_nontf %>% inner_join(tp53_status) %>%
  group_by(symbol) %>%
  filter(!is.na(mu_null)) %>%
  mutate(cscore_sc = -1*((cscore - mu_null)/sigma_null)) %>%
  group_map(~{
    pb$tick()$print()
    lm(cscore_sc ~ 0 + tissue + TP53_status, data = .x) %>%
      tidy() %>% tbl_df()
  }) %>% ungroup() %>%
  filter(term != 'TP53_statuswt') %>%
  mutate(FDR = p.adjust(p.value, method='BH')) %>%
  mutate(tissue = gsub('^tissue', '', term)) %>%
  dplyr::select(-term) %>%
  arrange(p.value)
```

We plot the distribution of LD-scores for transcription factors and non-TF genes.

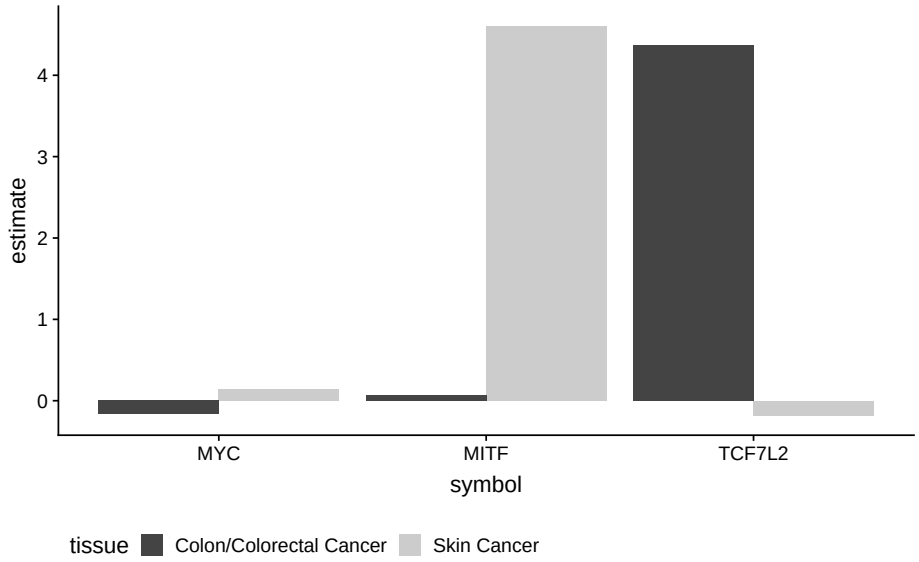
```
ldtf_crispr %>% mutate(gene_type = 'TF') %>%
  bind_rows(ldnontf_crispr %>% mutate(gene_type = 'nonTF')) %>%
  ggplot(aes(estimate, colour = gene_type)) + geom_density() +
  geom_vline(xintercept = 0, linetype = 'dashed') +
  scale_y_continuous(expand = c(0,0)) +
  scale_colour_manual(values = c('#111111', '#4285f4')) +
  theme(legend.position = 'bottom')
```



3.5 Visualizing resulting LD-TF

To visualize the resulting LD-scores we can make a bar plot highlighting a few selected LD-TF and there tissue-specific differences.

```
ldtf_crispr %>%
  filter(symbol %in% c('MITF', 'TCF7L2', 'MYC'),
         tissue %in% c('Skin Cancer', 'Colon/Colorectal Cancer')) %>%
  mutate(symbol = factor(symbol,
                         levels = c('MYC', 'MITF', 'TCF7L2'))) %>%
  ggplot(aes(symbol, estimate, fill = tissue)) +
  geom_bar(stat='identity', position = 'dodge') +
  theme(legend.position = 'bottom') +
  scale_fill_manual(values = c('#444444', '#cccccc'))
```

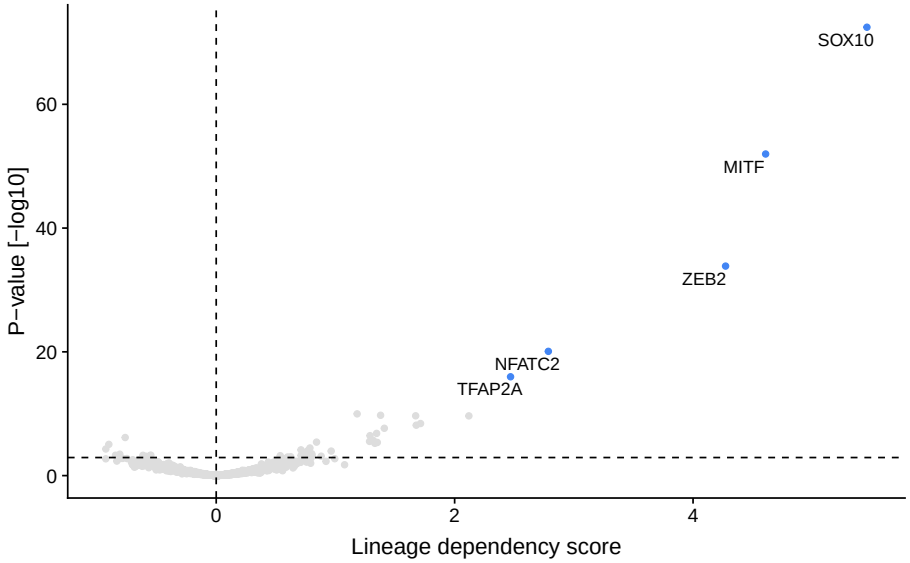


We can also visualize and highlight sets of lineage dependency transcription factors that are found by drawing volcano plots for different tissues. We define a function that can draw such volcano plots and show as enxample the LD-TFs of melanoma.

```
plot_ldtf_volcano <- function(df, ttype = 'Skin Cancer', highlight = c()){
  ## select genes to highlight
  df_highlight <- df %>% filter(symbol %in% highlight,
                                tissue == ttype)

  ## estimate 5% FDR
  fdr1_co <- df %>% filter(FDR < 0.05) %>%
    arrange(desc(FDR)) %>% pull(p.value) %>%
    head(1) %>% log10() %>% `*`(-1)

  ## make volcano
  df %>% filter(tissue == ttype, !symbol %in% highlight) %>%
    ggplot(aes(estimate, -log10(p.value))) +
    geom_point(colour = '#dddddd') +
    geom_point(data = df_highlight, aes(estimate, -log10(p.value)),
               colour = '#4285f4') +
    geom_text_repel(data = df_highlight,
                    aes(estimate, -log10(p.value), label = symbol)) +
    geom_vline(xintercept = 0, linetype = 'dashed') +
```

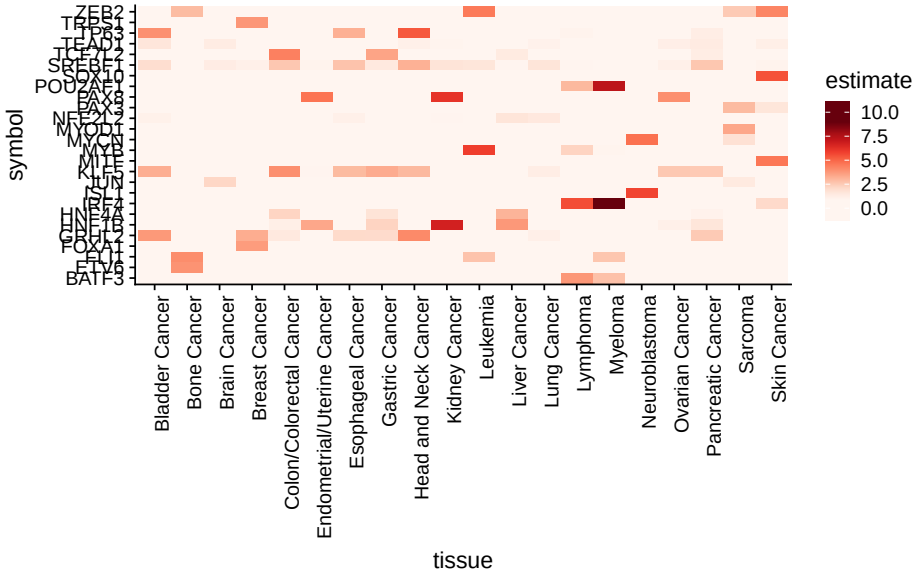
```
    geom_hline(yintercept = fdr1_co, linetype = 'dashed') +
    xlab('Lineage dependency score') + ylab('P-value [-log10]')
}
```

```
plot_ldtf_volcano(filter(ldtf_crispr, symbol != 'SOX9'),
                  'Skin Cancer',
                  c('MITF', 'SOX10', 'ZEB2', 'NFATC2', 'TFAP2A'))
```



Finally, we can make a heatmap of the top-2 LD-TF for each tissue.

```
## get the top 2 ld-tf per tissue
top2 <- ldtf_crispr %>% group_by(tissue) %>%
  top_n(2, estimate) %>% ungroup() %>% pull(symbol)

## make a tile-plot, sort by tissue
cols <- c(rep('#fff5f0', 3),'#fee0d2','#fcbba1',
          '#fc9272','#fb6a4a','#ef3b2c',
          '#cb181d','#a50f15',rep('#67000d', 3))
ldtf_crispr %>% filter(symbol %in% top2) %>%
  ggplot(aes(tissue, symbol, fill = estimate)) +
  geom_tile() +
  scale_fill_gradientn(colors=cols) +
  theme(axis.text.x = element_text(angle=90, hjust=1))
```



4 Lineage dependency transcription factors are important in both, normal and cancer tissue and can have oncogenic potential

We next ask whether the lineage dependency TF that we find in the CRISPR data can be lineage survival oncogenes (Garraway and Sellers, 2006). We start by selecting significant LD-TFs with LD-score > 1 and FDR $< 5\%$.

```
## map of TF-tissue associations
ldtf_map <- ldtf_crispr %>%
  mutate(ldtf = ifelse((FDR < 0.05) & (estimate > 1), T, F)) %>%
  distinct(symbol, tissue, ldtf) %>%
  arrange(symbol)
```

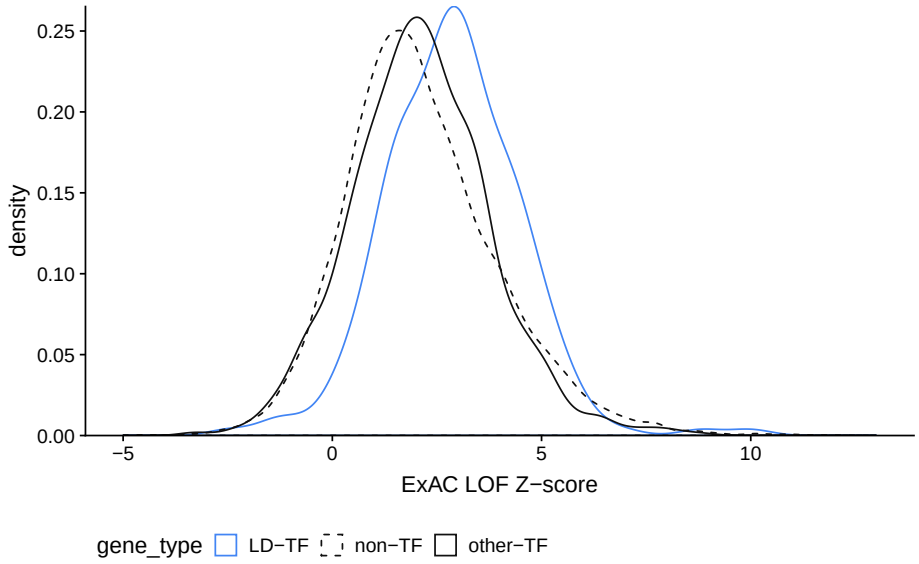
4.1 Do LD-TF play a role in normal lineage survival?

4.1.1 ExAC database

The ExAC database contains data about exome sequencing studies in the population. Since we are interested in non-cancer individuals here, we are going to look at the dataset that excludes samples from the TCGA. ExAC contains loss-of-function Z-scores that indicate whether or not a gene is more or less likely to retain damaging mutations in individuals. A lower likelihood indicates that there is negative selection pressure on these mutations which would further indicate that the genes play an important role.

```
## load Exac scores for relevant genes (in Avana library)
data('exac_scores', package='HDCRC2019')
```

```
## select significant LD-TF (1% FDR, -0.15 LD-score)
ldtf_sig <- ldtf_crispr %>%
  filter(FDR < 0.05, estimate > 1) %>% pull(symbol)

## is gene ld-tf, other tf or non-tf?
exac_ldtf <- exac %>% dplyr::select(gene, lof_z, pLI) %>%
  mutate(gene_type = ifelse(gene %in% ldtf_sig, 'LD-TF',
                     ifelse((gene %in% tf_list) & !(gene %in% ldtf_sig), 'other-TF',
                            'non-TF'))) %>%
  gather(score_type, value, lof_z, pLI)

## plot LOF z-scores
exac_ldtf %>% filter(score_type == 'lof_z') %>%
  ggplot(aes(value, colour=gene_type, linetype = gene_type)) +
  geom_density() +
  scale_colour_manual(values = c('#4285f4', '#111111', '#111111')) +
  scale_linetype_manual(values = c('solid', 'dashed', 'solid')) +
  scale_y_continuous(expand = c(0, 0)) +
  xlim(c(-5, 13)) + xlab('ExAC LOF Z-score') +
  theme(legend.position = 'bottom')
```

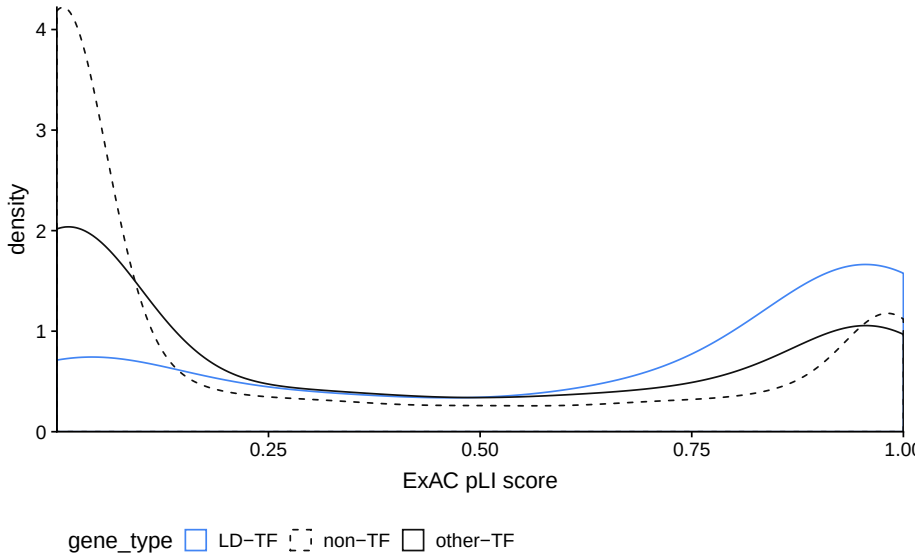



```
## test for statistical significance (LD-TF vs other TF)
exac_ldtf %>% filter(gene_type %in% c('other-TF', 'LD-TF'),
                     score_type == 'lof_z') %>%
  t.test(value ~ gene_type, data = ., var.equal=T)
```

We make a similar plot for the pLI probabilities that is shown in the supplement.

```
exac_ldtf %>% filter(score_type == 'pLI') %>%
  ggplot(aes(value, colour=gene_type, linetype = gene_type)) +
  geom_density() +
  scale_colour_manual(values = c('#4285f4', '#111111', '#111111')) +
```

```
  scale_linetype_manual(values = c('solid', 'dashed', 'solid')) +
  scale_y_continuous(expand = c(0, 0)) +
  scale_x_continuous(expand = c(0, 0)) +
  xlab('ExAC pLI score') +
  theme(legend.position = 'bottom')
```



5 LD-TFs compose cancer lineage specific core-regulatory circuits

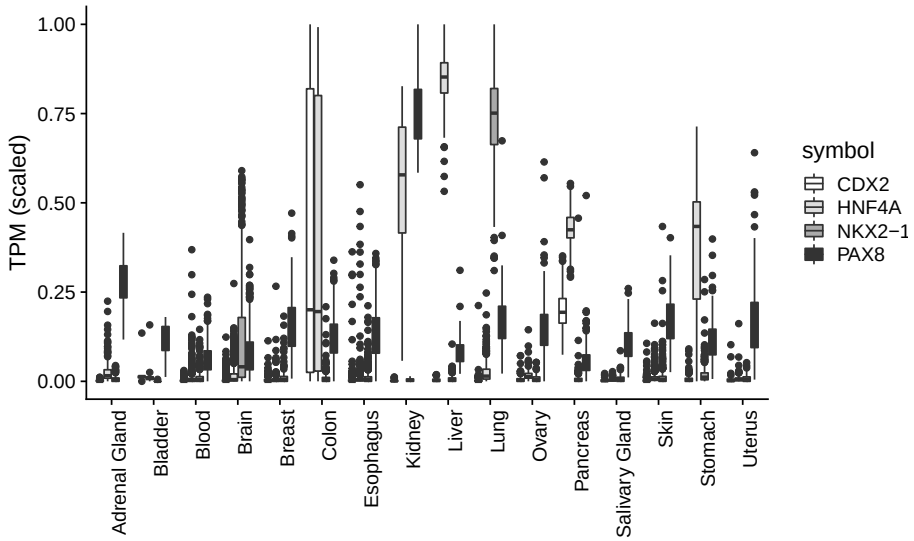
5.1 GTEx

We intend to show that LD TF are expressed in corresponding normal tissues. To this end we make use of the GTEx data resource that contains a large amount of non-cancer RNA-sequencing data. We load preoprocessed GTEx TPM values for tissues that could be matched to DepMap cancer lineages (see Supplementary information for details). We then scale the expression values for each gene to range [0,1] where 0 represents minimal and 1 maximal gene expression for that gene.

```
## load TPM values for LDTF genes
data('gtex_tpm', package='HDCRC2019')

## scale gene expression from 0 (not expr.) to 1 (fully expressed)
scaled_gtex <- gtex_tpm %>% group_by(symbol) %>%
  mutate(log_tpm = log(tpm + 1),
         tpm_scaled = scales::rescale(log_tpm, to = c(0,1))) %>%
  ungroup()
```

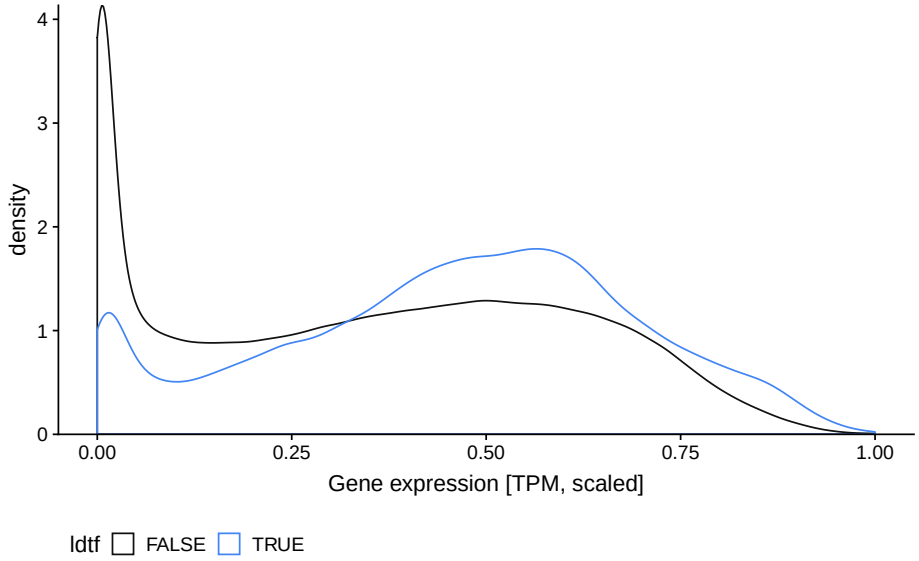
We can plot scaled expression values for a number of selected example LD-TF to observe whether LD-TF genes might in fact be expressed more highly in associated genes.

```
plot_gtex_tissue_median <- function(g){
  scaled_gtex %>% filter(symbol %in% g) %>%
    group_by(symbol, gtex) %>% mutate(med_tpm = median(tpm_scaled)) %>% ungroup() %>%
    ggplot(aes(gtex, tpm_scaled, fill = symbol)) +
    geom_boxplot() +
    scale_fill_manual(values = c('#ffffff', '#dddddd', '#aaaaaa', '#333333')) +
    theme(axis.text.x = element_text(angle=90, hjust=1)) +
    xlab('') + ylab('TPM (scaled)')
}

## Different genes that I know something about
plot_gtex_tissue_median(c('CDX2', 'NKX2-1', 'PAX8', 'HNF4A'))
```



We can now compare the gene expression distributions of LD-TFs in associated compared to non-associated tissues. Non-associated here means that cell lines of a tumor type did not show lineage-specific dependency on a TF (LD-score < 1 or FDR $> 5\%$).

```
## plot distributions
scaled_gtex %>% filter(symbol %in% ldtf_sig) %>%
  left_join(ldtf_map %>% dplyr::select(tumor_type = tissue, everything())) %>%
  ggplot(aes(tpm_scaled, colour = ldtf)) +
  geom_density() +
  scale_colour_manual(values = c('#111111', '#4285f4')) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('Gene expression [TPM, scaled]') +
  theme(legend.position = 'bottom')
```




```
## test for significance
scaled_gtex %>% filter(symbol %in% ldtf_sig) %>%
  left_join(ldtf_map %>% dplyr::select(tumor_type = tissue, everything())) %>%
  wilcox.test(tpm_scaled ~ ldtf, data = .)
```

5.2 Are LD-TF overexpressed in associated cancers?

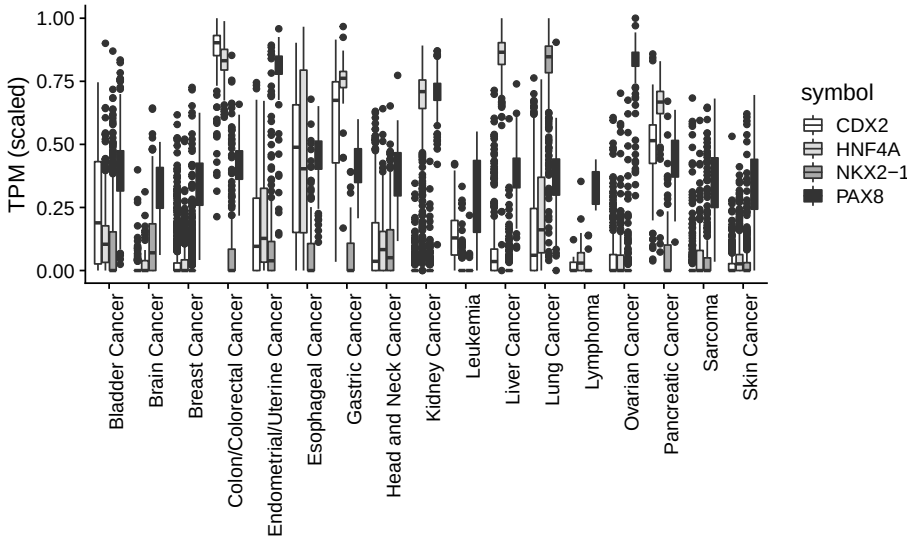
5.2.1 TCGA

We assume that if LD-TF are specifically essential in certain tissues their expression might also be restricted to these tissues. To test this we load RNA-seq data from the TCGA via the RTCGA and RTCGA.rnaseq Bioconductor packages. We then compare gene expression of significant LD-TFs between associated (tumors depend on the TF) and non-associated tumors. TCGA projects were linked to the cancer types in the DepMap dataset (see Supplementary information for details). Similar to above we scale expression values for each gene to range [0,1].

```
## load preprocessed TCGA data from file
data('tcga_data', package = 'HDCRC2019')

## scale data
tcga_ldtf_com <- tcga_data %>%
  filter(stype == 'tumor') %>%
  group_by(symbol) %>%
  mutate(log_fpkm = log(fpkm + 1),
         fpkm_scaled = scales::rescale(log_fpkm, to = c(0,1))) %>%
  ungroup() %>% inner_join(ldtf_map)
```

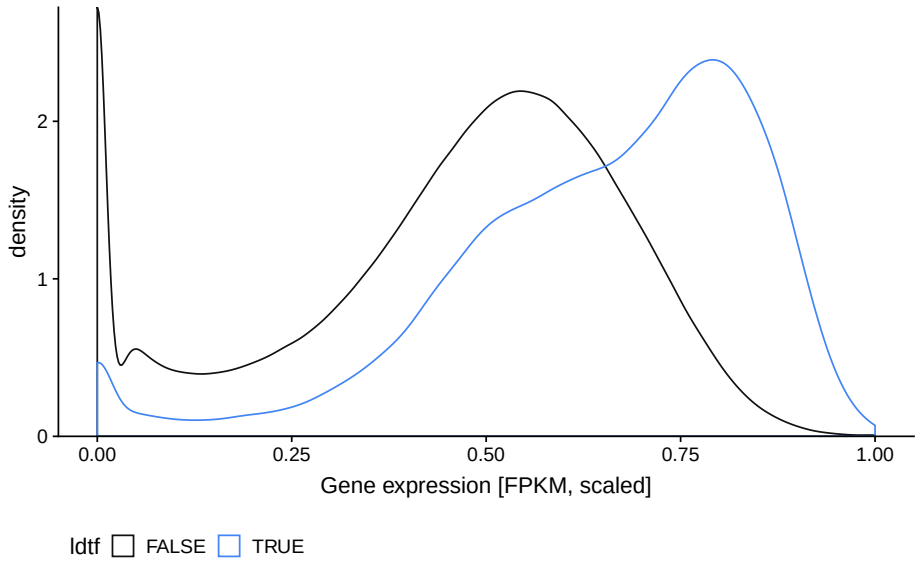
Again we can plot expression values for a number of selected TFs (same as above) to observe their expression across tissues.

```
## function to plot TF expression in TCGA tumor tissues
plot_tcga_expr <- function(g){
  tcga_ldtf_com %>% filter(symbol %in% g) %>%
    ggplot(aes(tissue, fpkm_scaled, fill=symbol)) +
    geom_boxplot() +
    scale_fill_manual(values = c('#ffffff', '#dddddd', '#aaaaaa', '#333333')) +
    theme(axis.text.x = element_text(angle=90, hjust=1)) +
    xlab('') + ylab('TPM (scaled)')
}

plot_tcga_expr(c('CDX2', 'HNF4A', 'NKX2-1', 'PAX8'))
```



As above we can now plot the scaled gene expression distributions for LD-TFs in their associated cancer types as compared to the other cancer types.

```
tcga_ldtf_com %>% filter(symbol %in% ldtf_sig) %>%
  ggplot(aes(fpkm_scaled, colour = ldtf)) +
  geom_density() +
  scale_colour_manual(values = c('#111111', '#4285f4')) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('Gene expression [FPKM, scaled]') +
  theme(legend.position = 'bottom')
```




```
## wilcox test to quantify significance
wilcox.test(fpkm_scaled ~ ldtf, data = filter(tcga_ldtf_com, symbol %in% ldtf_sig))
```

We next aim to examine if LD-TFs are overexpressed in tumors compared to healthy samples in the tumor lineages that depend on their activity. Healthy samples are only available in TCGA for 13 of the tissues so we can only look at these. We iterate over each LD-TF cancer lineage combination (e.g., MITF in melanoma, TCF7L2 in colorectal cancer, ...) and test whether the gene is differentially expressed between healthy and tumor samples using a t.test. We control the false discovery rate at 5% using the Benjamini Hochberg method and select genes as overexpressed if the difference between expression means is $<$-0.1 and as downregulated if diff. means is $>$0.1.

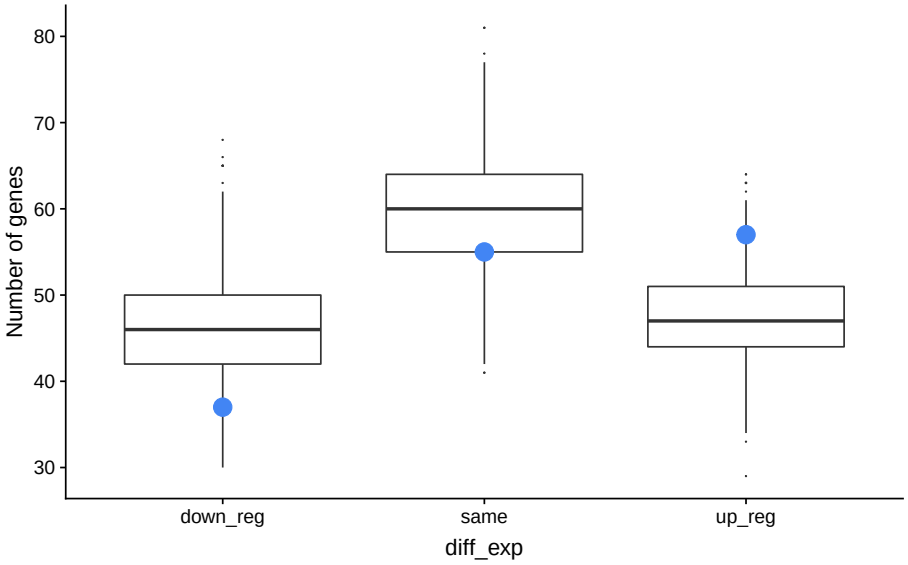
```
## list of tumors where healthy samples are available
healthy_sample_avail <- tcga_data %>%
  filter(stype == 'normal') %>% distinct(tissue) %>%
  pull(tissue)

## for each LD-TF test whether diff. expressed to normal samples
ldtf_diffexp_healthy <- tcga_data %>%
  filter(tissue %in% healthy_sample_avail) %>%
  group_by(symbol) %>%
  mutate(log_fpkm = log(fpkm + 1)) %>%
  ungroup() %>% inner_join(ldtf_map) %>%
  filter(ldtf) %>% nest(-symbol, -tissue) %>%
  mutate(test = purrr::map(data, ~ broom::tidy(t.test(log_fpkm ~ stype, data = .x)))) %>%
  unnest(test) %>% mutate(FDR  = p.adjust(p.value, method= 'BH')) %>%
  mutate(diff_exp = ifelse((estimate < -0.1) & (FDR < 0.2), 'up_reg',
                    ifelse((estimate > 0.1) & (FDR < 0.2), 'down_reg', 'same')))

ldtf_diffexp_summary <- ldtf_diffexp_healthy %>% count(diff_exp)
```

We find that out of xyz combinations we tested, upregulation was detected in 49 cases, downregulation was detected in 35 cases and no significant change was found for 75 of the cases. How does this compare to a random-sample of tissue-TF associations? We selected random samples of xzy transcription factors and perform a similar analysis. We repeat this 1,000 times to infer null distributions for up-, down- and unchanged expression between healthy and tumor samples.

```
df_for_sampling <- tcga_data %>%
  filter(tissue %in% healthy_sample_avail) %>%
  group_by(symbol) %>% mutate(log_fpkm = log(fpkm + 1)) %>% ungroup() %>%
  nest(-c(tissue, symbol))

## how many genes do I need to sample?
n_genes <- tcga_data %>% distinct(tissue) %>%
  filter(tissue %in% healthy_sample_avail) %>%
  inner_join(ldtf_map) %>% filter(ldtf) %>%
  distinct(symbol, tissue) %>% nrow()

set.seed(1234)
pb <- progress_estimated(1e3)
random_samples_summary <- map_df(1:1e3, ~{
  pb$tick()$print()
  df_for_sampling %>% sample_n(n_genes) %>%
    mutate(test = purrr::map(data, ~ broom::tidy(t.test(log_fpkm ~ stype, data = .x)))) %>%
    unnest(test) %>% mutate(FDR  = p.adjust(p.value, method= 'BH')) %>%
    mutate(diff_exp = ifelse((estimate < -0.1) & (FDR < 0.2), 'up_reg',
                      ifelse((estimate > 0.1) & (FDR < 0.2), 'down_reg', 'same'))) %>%
    count(diff_exp) %>% mutate(iteration = .x)
})
```

We can now compare our LD-TF observations against the inferred null distributions.

```
## summarise the results
random_samples_summary %>% ggplot(aes(diff_exp, n)) +
  geom_boxplot(outlier.size = 0) +
  geom_point(data = ldtf_diffexp_summary,
             aes(diff_exp, n), colour = '#4285f4', size = 5) +
  ylab('Number of genes')
```



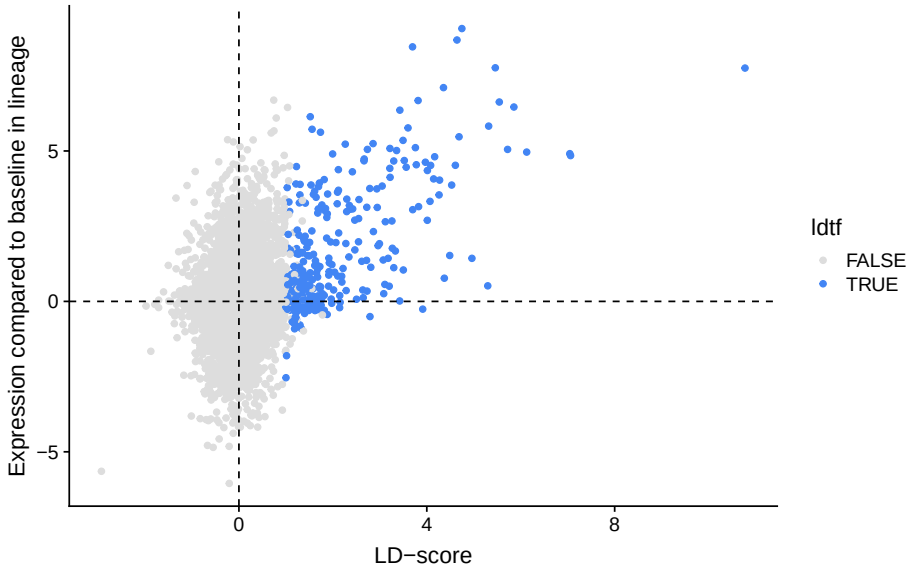

```
## p-values for each category
params <- random_samples_summary %>%
  group_by(diff_exp) %>%
  summarise(mu = mean(n), sigma = sd(n))

2*pnorm(ldtf_diffexp_summary %>% pull(n) %>% .[1],
      mean = params %>% pull(mu) %>% .[1],
      sd = params %>% pull(sigma) %>% .[1])
2*pnorm(ldtf_diffexp_summary %>% pull(n) %>% .[2],
      mean = params %>% pull(mu) %>% .[2],
      sd = params %>% pull(sigma) %>% .[2])
2*(1-pnorm(ldtf_diffexp_summary %>% pull(n) %>% .[3],
      mean = params %>% pull(mu) %>% .[3],
      sd = params %>% pull(sigma) %>% .[3]))
```

5.3 Lineage-selective expression does not predict dependency

Since lineage dependency genes are in general up-regulated in their associated tissues, we asked the opposite - are genes with lineage-restricted gene expression always essential? This is not the case. We can visualize this observation by plottin LD-scores and gene expression values in cancer cell lines, highlighting combinations of significant lineage dependency.

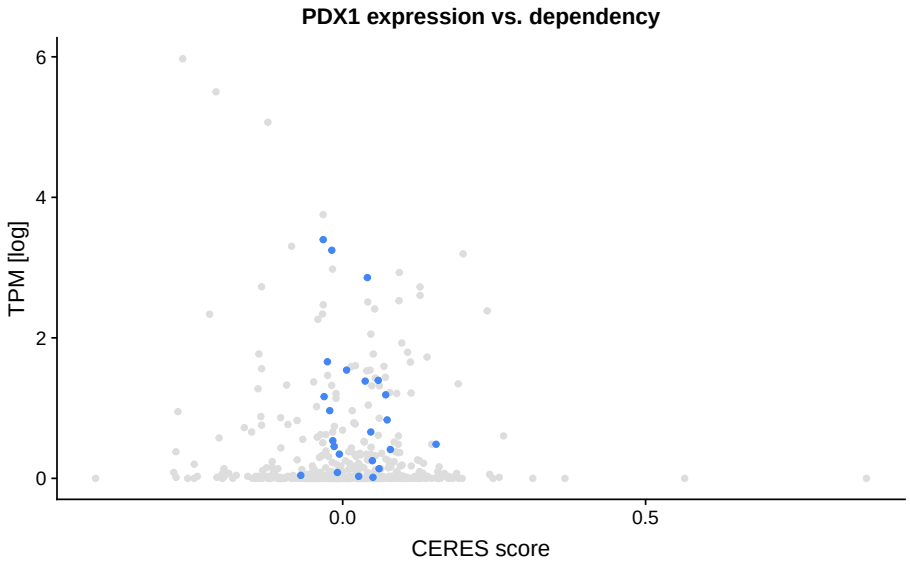
```
## load CCLE gene expression data
data('cl_expr_tf', package='HDCRC2019')

## make a plot
ldtf_crispr %>%
  mutate(ldtf = ifelse((FDR < 0.05) & (estimate > 1), T, F)) %>%
  dplyr::select(symbol, ld_score=estimate, tissue, ldtf) %>%
  inner_join(cl_expr_tf) %>%
  group_by(symbol) %>% mutate(tpm_base = tpm - median(tpm)) %>% ungroup() %>%
```

```
  group_by(symbol, tissue, ldtf) %>%
  summarise(ld_score = ld_score[1], tpm = median(tpm_base)) %>%
  ungroup() %>%
  ggplot(aes(ld_score, tpm, colour = ldtf)) +
  geom_point() +
  geom_vline(xintercept = 0, linetype = 'dashed') +
  geom_hline(yintercept = 0, linetype = 'dashed') +
  scale_colour_manual(values = c('#dddddd', '#4285f4')) +
  xlab('LD-score') + ylab('Expression compared to baseline in lineage')
```



We can also look at individual examples, such as PDX1. This TF is known to be highly selectively expressed in pancreatic cancer. It does, however, not show a phenotype in the DepMap CRISPR-Cas9 screens (all CERES scores are ~0).

```
df <- depmap_ceres %>% filter(symbol == 'PDX1') %>%
  inner_join(cl_expr_tf %>% filter(symbol == 'PDX1') %>%
               dplyr::select(cellline, tpm)) %>%
  mutate(lin = ifelse(tissue == 'Pancreatic Cancer', T, F))

ggplot() +
  geom_point(data = subset(df, !lin), aes(cscore, tpm),
             colour = '#dddddd') +
  geom_point(data = subset(df, lin), aes(cscore, tpm),
             colour = '#4285f4') +
  xlab('CERES score') + ylab('TPM [log]') +
  ggtitle('PDX1 expression vs. dependency')
```

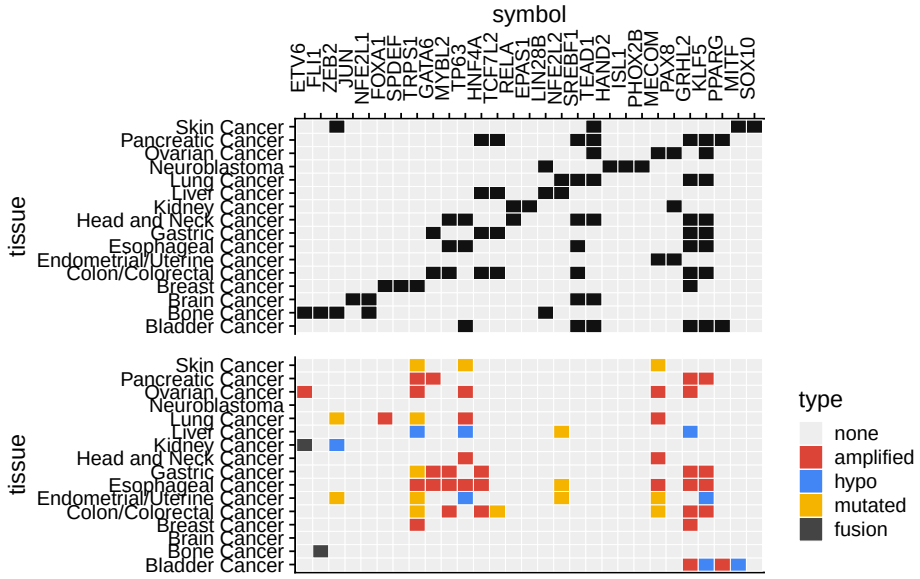


5.4 Are LD-TF altered in cancer samples

To answer the question whether or whether not LD-TF are genes with oncogenic potential we investigate whether variants in these genes are found in tumors; at least in a subset of the LD-TF associated tumor samples. We load data from the COSMIC database addressing four different types of potential molecular variants - mutations, copy number changes, fusions or hypomethylation. We consider a gene as frequently altered if alterations in that gene occur in at least 5% of all tumor samples corresponding to a cancer type. COSMIC tumor type annotations were linked to DepMap cell line tissues as described in the Supplementary information file.

```
## load variant annotation (based on COSMIC)
data('variant_heatmap_data', package = 'HDCRC2019')

## top LD-TF per tissue
top_ldtf <- ldtf_crispr %>% filter(FDR < 0.05, estimate > 1) %>%
  filter(tissue %in% variant_heatmap_data$tissue) %>%
  group_by(tissue) %>%
  arrange(p.value) %>% dplyr::slice(1:3) %>% ungroup()

symbol_levels <- top_ldtf %>% group_by(symbol) %>%
  arrange(estimate) %>% dplyr::slice(1) %>% ungroup() %>%
  arrange(tissue) %>% pull(symbol)

## LD-TF annotation
ldtf_annotation <- ldtf_map %>% spread(tissue, ldtf) %>%
  filter(symbol %in% top_ldtf)

## variant heatmap
var_hm <- variant_heatmap_data %>%
  filter(symbol %in% top_ldtf$symbol, symbol != 'TP53') %>%
  mutate(symbol = factor(symbol, levels = symbol_levels)) %>%
```

```
  ggplot(aes(symbol, tissue, fill = type)) + geom_tile(colour = '#ffffff') +
  scale_fill_manual(values = c('#eeeeee', '#db4437', '#4285f4', '#f4b400', '#444444')) +
  theme(#axis.text.x = element_text(angle=45, hjust=1),
        axis.text.x = element_blank(), axis.title.x = element_blank(),
        axis.ticks.x = element_blank())

## LDTF annotation heat map
ldtf_hm <- ldtf_map %>%
  filter(symbol %in% top_ldtf$symbol, symbol != 'TP53',
         symbol %in% variant_heatmap_data$symbol,
         tissue %in% variant_heatmap_data$tissue) %>%
  arrange(desc(tissue)) %>%
  mutate(tissue = factor(tissue, levels = rev(unique(tissue)))) %>%
  mutate(symbol = factor(symbol, levels = symbol_levels)) %>%
  ggplot(aes(symbol, tissue, fill = ldtf)) +
  geom_tile(colour = '#ffffff') +
  scale_fill_manual(values = c('#eeeeee', '#111111')) +
  scale_x_discrete(position = 'top') +
  theme(axis.text.x = element_text(angle=90),
        legend.position = 'none')

ldtf_hm + var_hm + plot_layout(ncol=1, heights = c(1,1))
```



Using Fisher's exact test we determine whether significant LD-TF (FDR $< 5\%$, LD-score > 1) are altered more often than expected in their associated cancer lienages.

```
ldtf_map %>% filter(tissue %in% variant_heatmap_data$tissue) %>%
  inner_join(variant_heatmap_data) %>%
  mutate(isaltered = ifelse(type == 'none', F, T)) %>%
  dplyr::select(ldtf, isaltered) %>%
  table() %>% fisher.test()
```

5.5 Quality control: expression in cell lines

It is possible that there might be some false positives among the list of LD-TF - perhaps due to unexpected CRISPR off-targets. We check for each gene in the LD-TF list whether that gene is actually expressed in the cell lines that depend on it.

```
##load average gene expr. data for each gene in each lineage
data('cellline_expr_median', package='HDCRC2019')
```

We add these data to the LD-TF list and check whether there are lineage specific phenotypes that are not backed up by expression data.

```
ldtf_crispr <- ldtf_crispr %>% left_join(cellline_expr_median)
ldnontf_crispr <- ldnontf_crispr %>% left_join(cellline_expr_median)
```

There are a few examples where this is indeed the case. We will exclude these from downstream analyses. In addition we annotate ExAC LOF Z scores for each LD-TF.

```
ldtf_crispr <- ldtf_crispr %>%
    left_join(exac_ldtf %>% filter(score_type == 'lof_z') %>%
                  dplyr::select(symbol = gene, exac_lof_z = value))
```

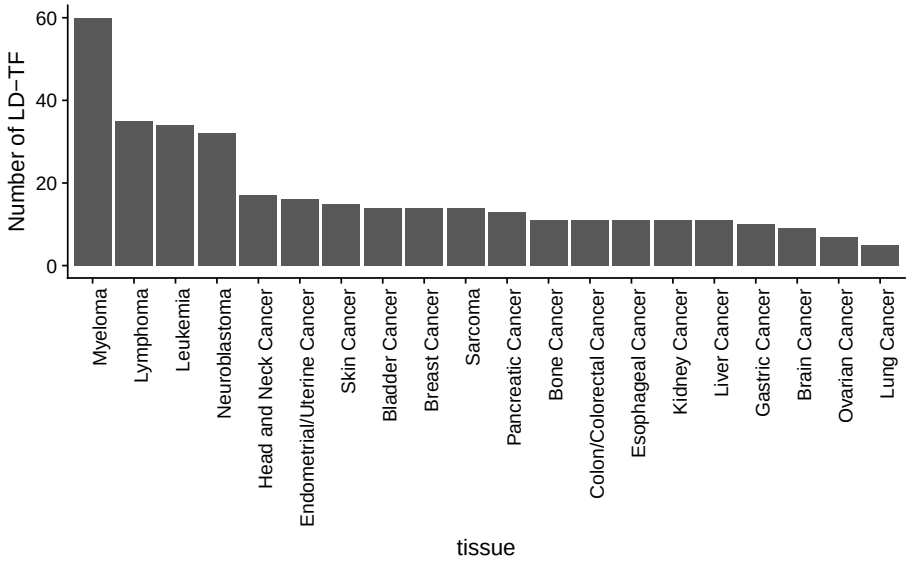
6 LD-TFs compose cancer lineage specific core-regulatory circuits

We filter LD-TF that are not expressed in their respective tissues and not required in the population (ExAC).

```
## filter LD-TF
ldtf_fil <- ldtf_crispr %>%
  filter(estimate > 1, FDR < 0.05,
         exac_lof_z > 1 | is.na(exac_lof_z),
         med_expr > 0.01 | is.na(med_expr))
```

We can generate a plot that shows the number of significant LD-TF after filtering for each tumor lineage.

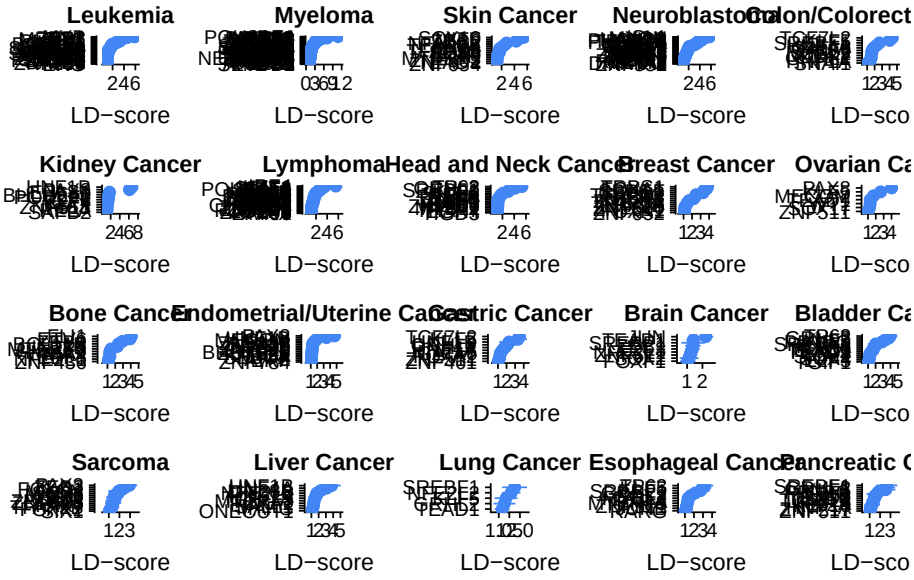
```
ldtf_fil %>% count(tissue) %>%
  arrange(desc(n)) %>%
  mutate(tissue = factor(tissue, levels = tissue)) %>%
  ggplot(aes(tissue, n)) +
  geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle=90, hjust=1)) +
  ylab('Number of LD-TF')
```



6.1 Putative core-regulatory circuits

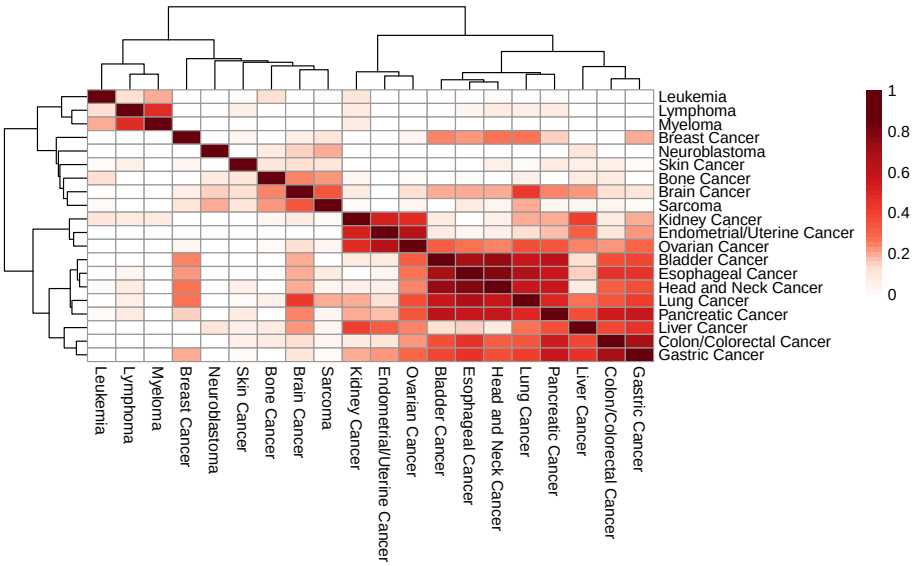
We assemble filtered LD-TF genes into putative core-regulatory circuits for each of the cancer lineages.

```
## error bars are 2 sem.
crc_plots <- ldtf_fil %>% nest(-tissue) %>%
  mutate(p = map2(data, tissue, ~{
    .x %>% mutate(ymin = estimate - (2*std.error),
                  ymax = estimate + (2*std.error)) %>%
      arrange(estimate) %>%
      mutate(symbol = factor(symbol, levels = symbol)) %>%
      ggplot(aes(symbol, estimate)) +
      geom_point(colour = '#4285f4', size=3) +
      geom_linerange(aes(ymin = ymin, ymax = ymax), colour = '#4285f4') +
      coord_flip() + ggtitle(.y) + ylab('LD-score') + xlab('')
}))

## draw plots
reduce(crc_plots$p, `+`)
```



We cluster cancer lineages by LD-scores of all CRC TFs. This reveals clusters that indicate how common developmental origin drives the composition of LD-TFs in these tissues.

```
## color key for heatmap
cols <- c(rep('#ffffff', 2),'#fee0d2',
          '#fb6a4a','#ef3b2c','#cb181d',
          '#a50f15', rep('#67000d', 2))
colkey <- colorRampPalette(cols)(50)

## cluster by Pearson correlation
ldtf_crispr %>% filter(symbol %in% ldtf_fil$symbol) %>%
  acast(symbol ~ tissue, value.var = 'estimate') %>%
  cor() %>%
  pheatmap::pheatmap(color = colkey, clustering_method = 'ward.D2')
```



6.2 Transcription factor target gene relationships

6.2.1 ChiP-seq data

In order to assume that TFs are part of a core-regulatory circuit (CRC) we want to show that they can regulate each other. For that purpose we can use ChiP-seq data obtained from Harmonizome. We downloaded preprocessed data of TF-target gene relationships across several sources. For all TFs belonging to CRC that we predict we pull out these regulatory relationships as evidence that the TFs can regulate each other.

```
data('tft_combi', package = 'HDCRC2019')
```

6.2.2 Null distribution for TF-target-relationships

Depending on the size of putative CRC we draw random sets of transcription factors from the ChIP-seq data and calculate the number of regulatory relationships between these TFs. We repeat this 1,000 times for each CRC size to create null distributions that we can then compare the numbers of regulatory relationships between CRC members with each other.

```
## random seed for reproducibility
set.seed(1234)

## infer null distributions for different TF numbers
null_distr <- ldtf_fil %>% count(tissue) %>% distinct(n) %>%
  mutate(stats = map(n, function(n_tf){
    n_iter <- 1000

    pb <- progress_estimated(n_iter)
    null_dist <- map(1:n_iter, function(i){
      pb$tick()$print()
      tf_sample <- tf_list[tf_list %in% c(tft_combi$tf,
                                          tft_combi$target)] %>%
        sample(n_tf)

      ## summarise across sources and calculate number of edges
      tft_combi %>% filter(tf %in% tf_sample,
                           target %in% tf_sample) %>%
      group_by(tf, target) %>%
      summarise(value = sum(as.numeric(value))) %>% ungroup() %>%
      nrow()
    }) %>% unlist()
  })) %>%
  mutate(test = map(stats, ~ tibble(sd = sd(.x), mean = mean(.x)))) %>%
  unnest(test)
```

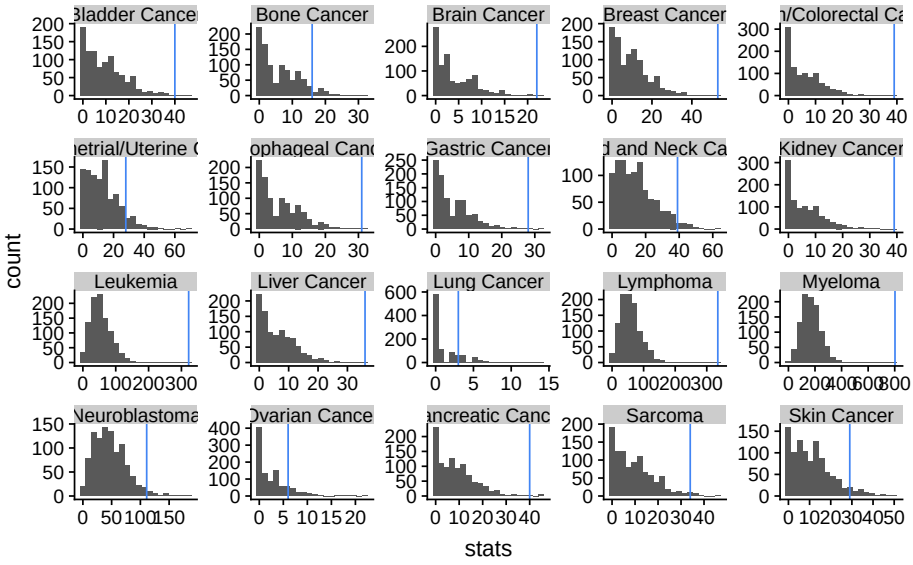
Using a Kolmogorov-Smirnov test we test whether the number of TF-target gene relationships observed between TFs that compose a predicted CRC is higher than expected by chance.

```
## sample TF and calculate number of edges in resulting network
connectivity_test <- ldtf_fil %>% nest(-tissue) %>%
```

```
  mutate(connectivity = map(data, ~{
    n_edges <- tft_combi %>% filter(tf %in% .x$symbol,
                                    target %in% .x$symbol) %>%
      group_by(tf, target) %>%
      summarise(value = sum(as.numeric(value))) %>% ungroup() %>%
      nrow()

    ## params of null distr
    dnull <- null_distr %>% filter(n == nrow(.x)) %>%
      unnest(stats) %>% pull(stats)
    ## fit poisson distr.
    nb_fit <- fitdistr(dnull, 'negative binomial')

    ## calculate p-value
    pval <- ks.test(n_edges, 'pnbinom',
                    mu=nb_fit$estimate['mu'],
                    size=nb_fit$estimate['mu'])$p.value

    return(tibble(
      n_edges = n_edges,
      p_value = pval
    ))
  })) %>% unnest(connectivity)
```

We can now visualize the amount of TF-target gene relationships reported for each CRC compared to their null distributions. To this end we plot a histogram of the sampled values and indicate the number of observed TF-target relationships as a vertical line.

```
tf_conn_plots <- connectivity_test %>% mutate(n = map_int(data, nrow)) %>%
  left_join(null_distr) %>%
  unnest(stats) %>% nest(-tissue) %>%
  mutate(p = map(data, ~{
    ggplot(.x, aes(stats)) + geom_histogram(bins=20) +
      geom_vline(aes(xintercept = n_edges[1]), colour = '#4285f4')
    }))

## plot them to canvas
connectivity_test %>% mutate(n = map_int(data, nrow)) %>%
  left_join(null_distr) %>%
  unnest(stats) %>%
  ggplot(aes(stats)) + geom_histogram(bins=20) +
  geom_vline(aes(xintercept = n_edges), colour = '#4285f4') +
  facet_wrap(~tissue, scales='free')
```



6.3 Super enhancer elements

We assume that in ChIP-seq studies, super enhancer elements associated with LD-TFs are more frequently identified compared to other TFs. To test this we dnwloaded SE data for all samples (both cell line and real tissue) that we could match to the tumor lineages in the DepMap data set from SEdb (Jiang et al., 2018). We now load these data and loop through all cancer lineages and their CRCs to test whether SE elements are more frequently associated with these TFs compared to other TFs/genes.

```
## read SEdb data
data('sedb_data', package = 'HDCRC2019')
```

Now we can test for each tissue whether LD-TFs have more associated super enhancers than other TF. We also generate boxplots to visualize these results.

```
## for each tissue test LD-TF vs non LD-TF SE count
sedb_pvals <- sedb_data %>%
  filter(map_int(se_data, nrow) != 0) %>%
  mutate(se_stats = map2(tissue, se_data, ~{
  ## get tf
  ld_tf <- ldtf_fil %>% filter(tissue == .x) %>% pull(symbol)
  ## how many samples do I have
  n_samples <- nrow(.y %>% distinct(sample_ID))

  ## filter data and annotate ld-tf
  df <- tibble(tf = tf_list[tf_list %in% ldtf_crispr$symbol]) %>%
    left_join(.y %>% distinct(tf = `closest active gene`, se_id = `SE ID`) %>%
                count(tf)) %>% mutate(n = ifelse(is.na(n), 0, n)) %>%
    mutate(ldtf = ifelse(tf %in% ld_tf, T, F))

  ## make a boxplot
  bp <- df %>% ggplot(aes(ldtf, n)) + geom_boxplot() +
```

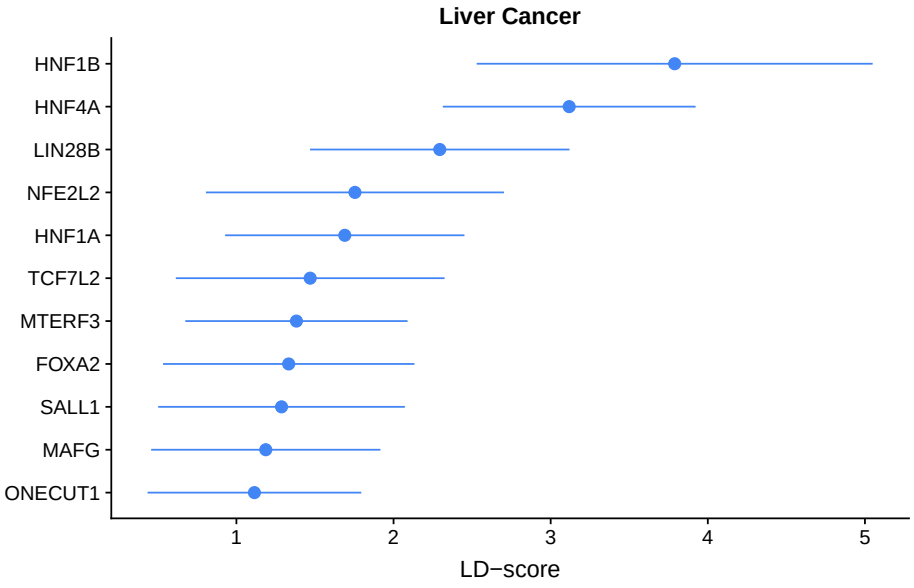
```
      xlab('LD-TF') + ylab('Number of SE elements') +
      ggtitle(paste(.x, ',', n_samples, 'samples'))

  ## test difference SE count
  pval <- wilcox.test(n ~ ldtf, data = df)$p.value

  return(list(bp = bp, pval = pval, n_samples = n_samples))
}))
```

6.4 Validation: discovery of known CRC

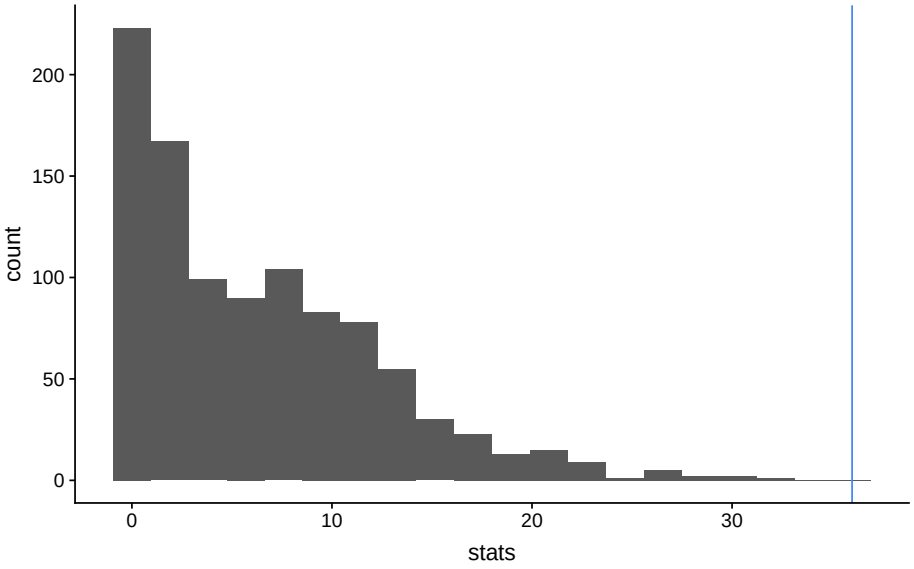
6.4.1 Hepatocellular carcinoma

The Liver CRC has been extensively studied in the past (Odom et al., 2004; Odom et al., 2006). We generate plots for CRC genes, transcription factor-target gene relationships and super enhancer elements in liver cancer to see whether there is overlap.

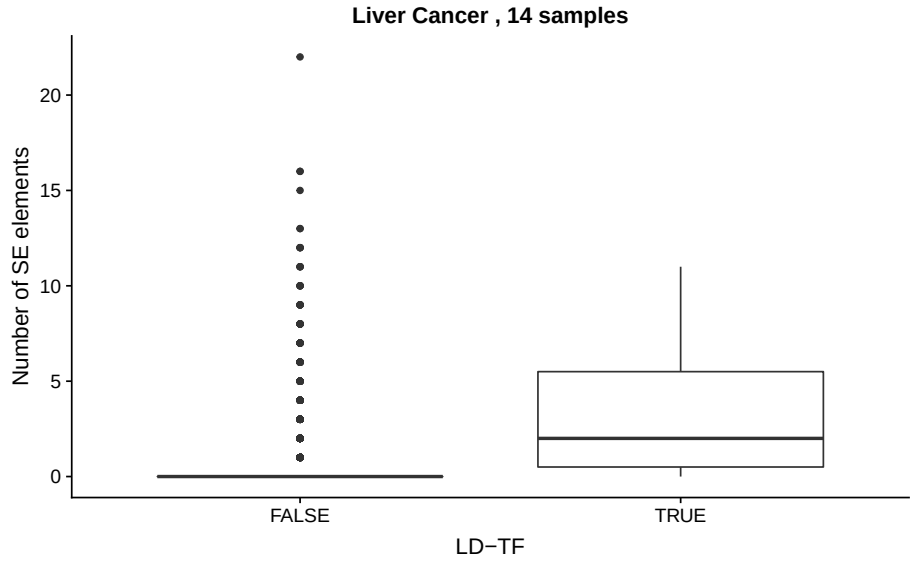
```
## LD-TF plot
crc_plots %>% filter(tissue == 'Liver Cancer') %>% pull(p)
```




```
## number of tf-target genes
tf_conn_plots %>% filter(tissue == 'Liver Cancer') %>% pull(p)
```




```
connectivity_test %>% filter(tissue == "Liver Cancer") %>% pull(p_value)

## Super enhancer elements
sedb_pvals %>% filter(tissue == 'Liver Cancer') %>% .$se_stats
```



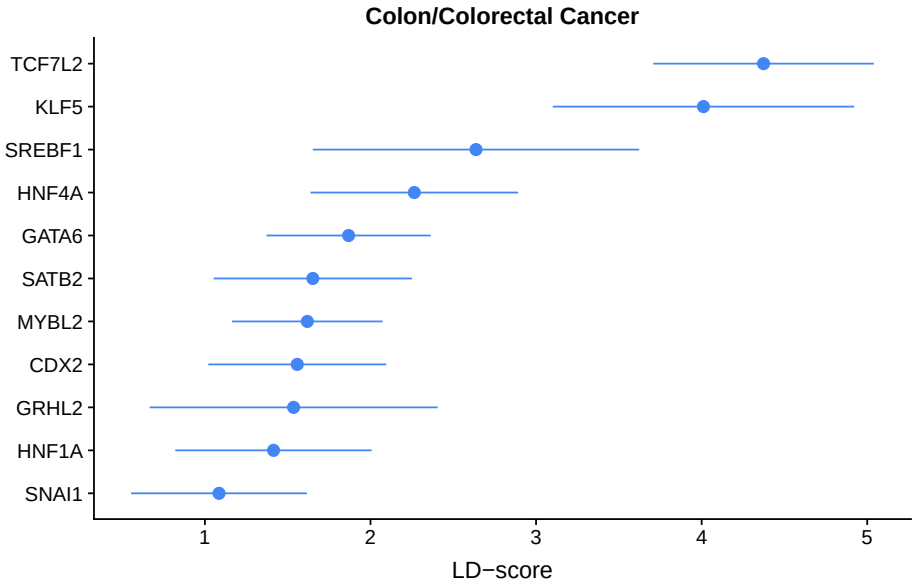
6.5 The CRC of colorectal cancer cells

We next have a look at the CRC of colorectal cancer cells that as such has not previously been described.

6.5.1 LD-TF scores for putative colorectal CRC TF

We have generated all required plots above, now we simply extract the relevant ones (for colorectal cancer) and plot them here.

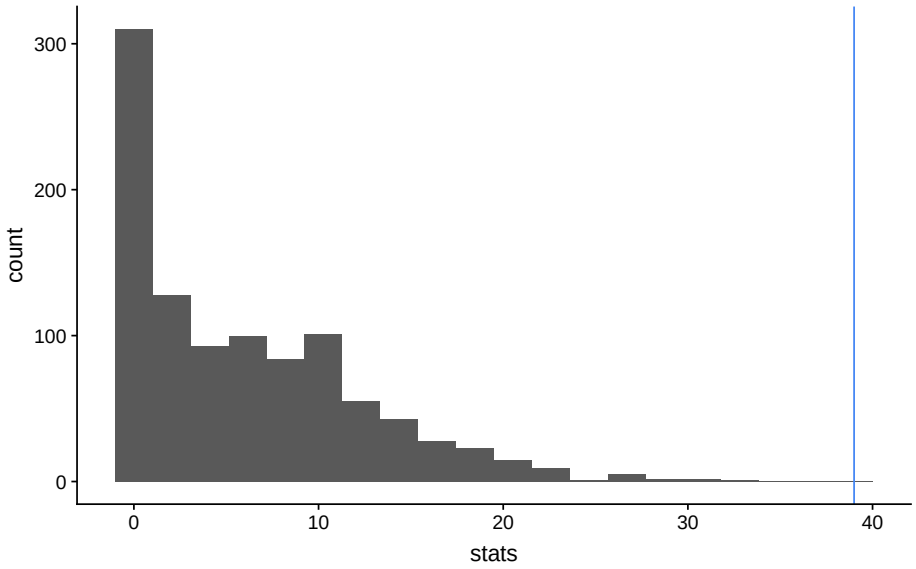
```
crc_plots %>% filter(tissue == 'Colon/Colorectal Cancer') %>% pull(p)
```



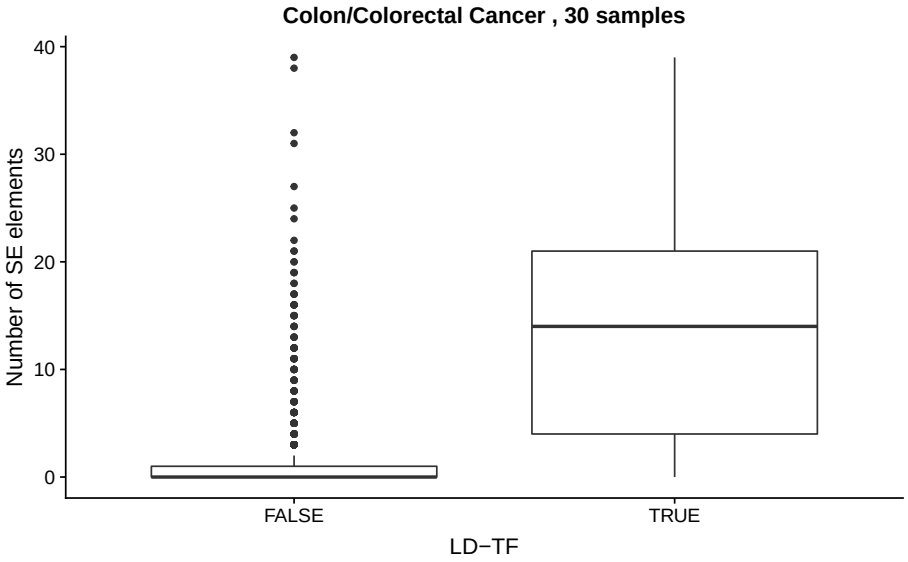
6.5.2 TF-target gene relationships and SE elements

I plot TF-target gene info and super enhancer elements as above.

```
## number of tf-target genes
tf_conn_plots %>% filter(tissue == 'Colon/Colorectal Cancer') %>% pull(p)
```



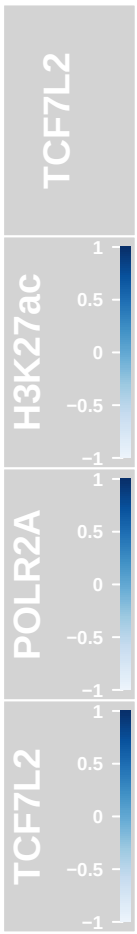
```
connectivity_test %>% filter(tissue == 'Colon/Colorectal Cancer') %>% pull(p_value)

## Super enhancer elements
sedb_pvals %>% filter(tissue == 'Colon/Colorectal Cancer') %>% .$se_stats
```

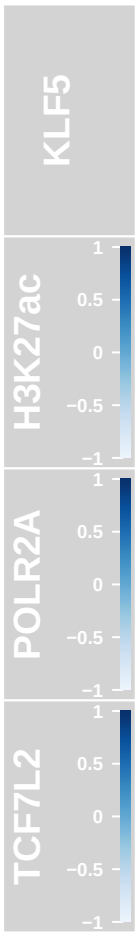


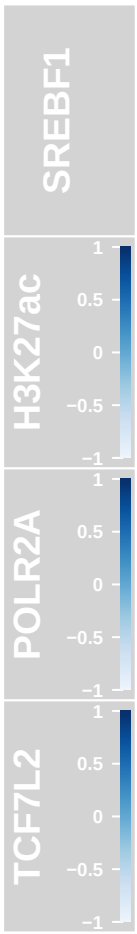
6.5.3 HCT116 ChIP-seq data

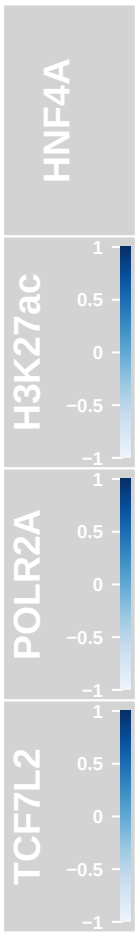
For the HCT116 colorectal cancer cell line there are a number of ChIP-seq experiments available from ENCODE. These include Enhancer-profiling by H3K37ac, transcriptional activity by POLR2A and, importantly, binding positions for the key (top-scoring) colorectal CRC TF TCF7L2. We can make plots visualizing that TCF7L2 binds together with POLR2A to enhancer regions at the TSS of other CRC genes.

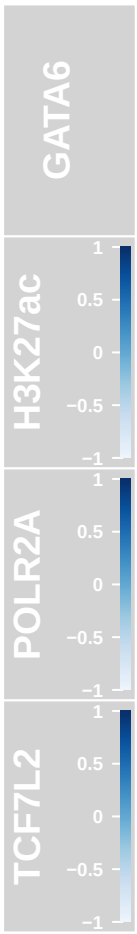
```
## load the processed chip peak data downloaded from encode
data('crc_chip_tracks', package = 'HDCRC2019')

tr_plots <- crc_chip_tracks %>%
  mutate(p = pmap(list(genome_track, data_tracks, start, end, chr),
                  function(gt, dt, st, en, ch){
    plotTracks(c(gt, unlist(dt)),
                  from = as.integer(st), to = as.integer(en),
                  chromosome = ch,
                  sizes=c(1,rep(1, length(dt))), type = 'heatmap')
  }))
```

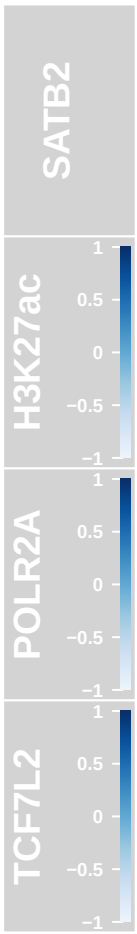

TCF7L2
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1

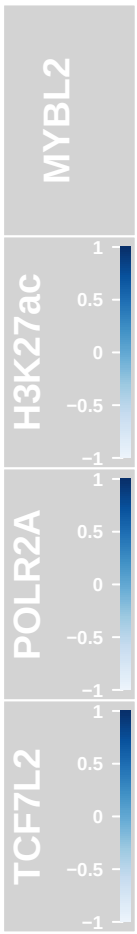

KLF5
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1

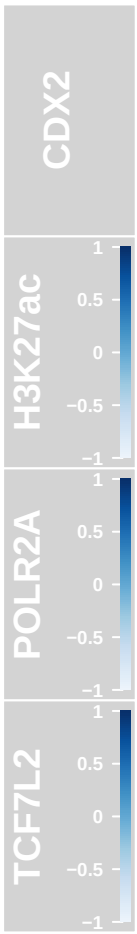

SREBF1
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1

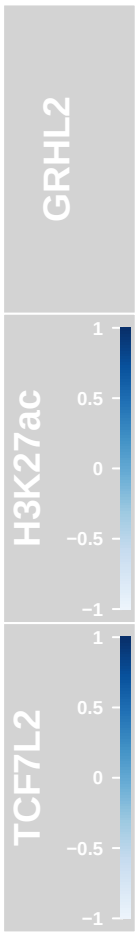

HNF4A
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1

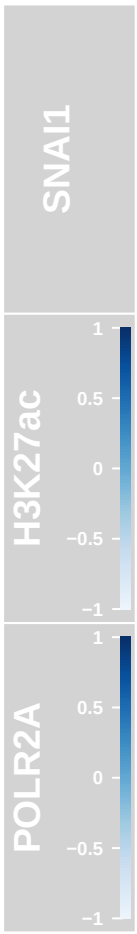

GATA6
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1


SATB2
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1


MYBL2
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1


CDX2
H3K27ac
1
0.5
0
−0.5
−1
POLR2A
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1


GRHL2
H3K27ac
1
0.5
0
−0.5
−1
TCF7L2
1
0.5
0
−0.5
−1



7 Codependency predicts mechanisms leading to CRC deregulation in cancer

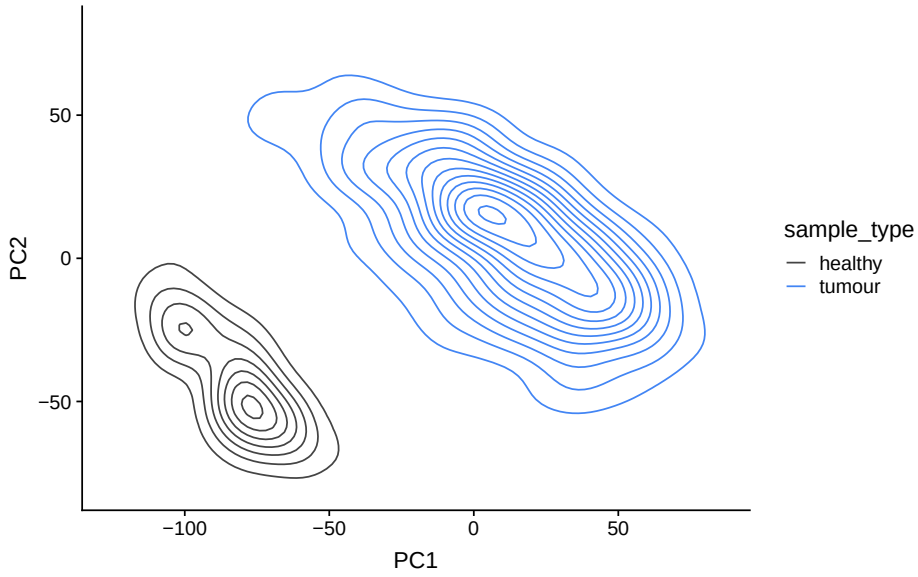
We first filter the lists of lineage dependency genes (both TF and non-TF), excluding non-essential TF (ExAC) and genes that are not expressed in their corresponding tissues.

```
## filter LD-TF
ldtf_fil <- ldtf_crispr %>%
  filter(estimate > 1, FDR < 0.05,
         exac_lof_z > 1 | is.na(exac_lof_z),
         med_expr > 0.01 | is.na(med_expr))

## filter LD non-TF
ldnontf_fil <- ldnontf_crispr %>%
  filter(FDR < 0.05, estimate > 1,
         med_expr > 0.01 | is.na(med_expr))
```

7.1 From healthy to cancer transcriptomic states

We can use TCGA data to show that tumor and healthy transcriptomes appear unimodal in PCA analysis on expression data. This is similar to the graph displayed in Califano and Alvarez, 2017. We use colorectal cancer as an example.

```
## load preprocessed TCGA expression data in colorectal cancers
data('colo_pt_expr', package = 'HDCRC2019')

## pca
colo_pca <- colo_pt_expr %>% acast(sample_nr ~ symbol, value.var = 'fpkm') %>% prcomp()

## plot as contour plot
as_tibble(colo_pca$x, rownames='sample_nr') %>%
  dplyr::select(sample_nr, PC1, PC2) %>%
  inner_join(distinct(colo_pt_expr, sample_nr, sample_type)) %>%
  filter(sample_type %in% c('healthy', 'tumour')) %>%
  ggplot(aes(PC1, PC2, colour = sample_type)) + stat_density_2d() +
  scale_colour_manual(values = c('#444444', '#4285f4')) +
  xlim(c(-125, 85)) + ylim(c(-80, 80))
```

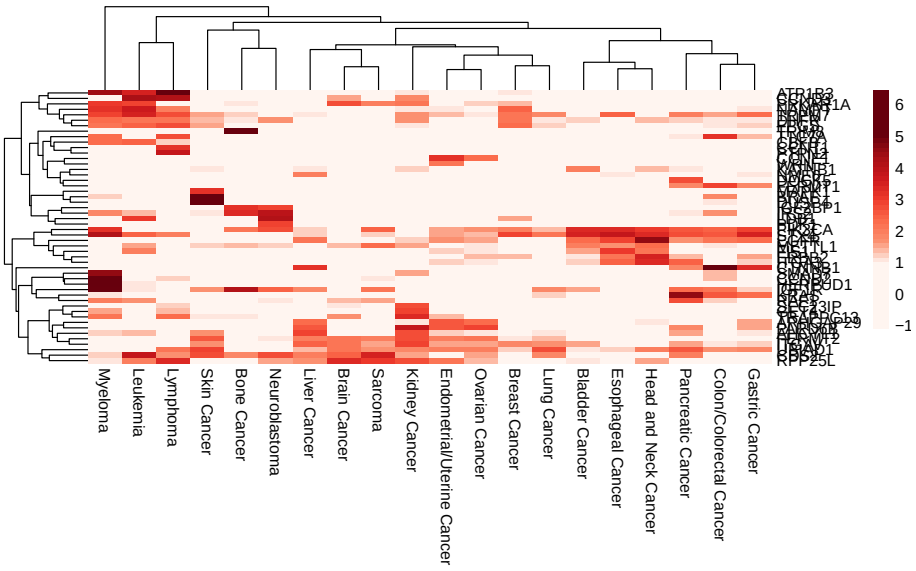


7.2 Lineage-dependency non-TF genes

To visualize lineage specific dependencies that are not transcription factors, we can plot a heatmap showing the 3 strongest non-TF dependencies for each tumor type.

```
## list of strong non-TF dependencies
notf_deps <- ldnontf_fil %>% group_by(tissue) %>%
  top_n(3, -p.value) %>% ungroup() %>% pull(symbol)

## visualize as heat map
cols <- c(rep('#fff5f0', 4),'#fc9272',
```

```
              '#fb6a4a','#ef3b2c','#cb181d','#a50f15',rep('#67000d', 3))
ldnontf_crispr %>% filter(symbol %in% notf_deps) %>%
  acast(symbol ~ tissue, value.var = 'estimate') %>%
  pheatmap(border_color = NA,
           color = colorRampPalette(cols)(50))
```



7.3 A coessentiality matrix of LD genes

In order to group LD-TF genes in an unbiased way we generate a codependency matrix (where codependency is defined as the Pearson correlation coefficient of gene dependency profiles across all 558 cell lines) including all genes that are specifically essential in some cancer type (as determined by the lists above).

```
## co-essentiality matrix
coess_mat <- depmap_ceres %>%
  filter(symbol %in% c(ldtf_fil$symbol, ldnontf_fil$symbol),
         tissue %in% ldtf_fil$tissue) %>%
  acast(cellline ~ symbol, value.var = 'cscore') %>% cor()
```

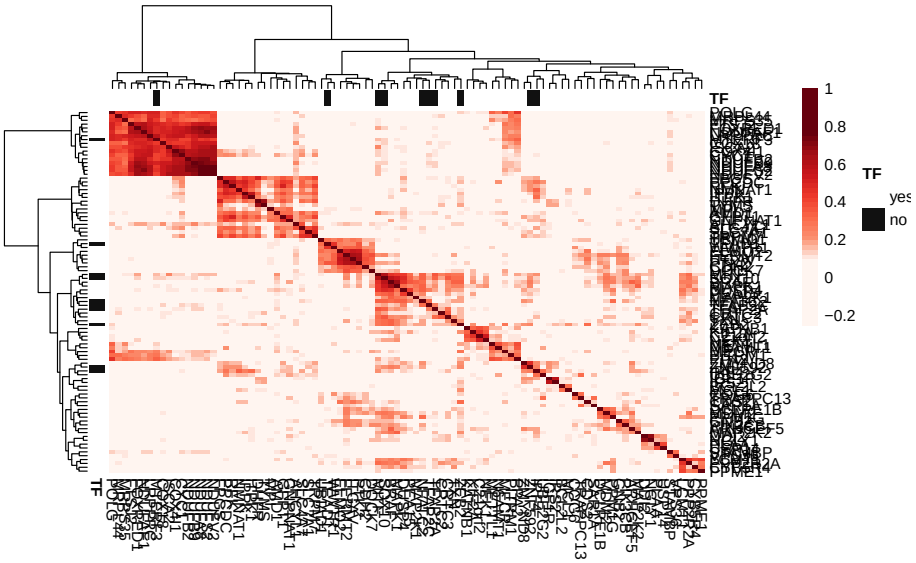
7.4 Example core regulatory circuits

We can now use these data to predict biological processes involved in regulating the linage-specific core-regulatory circuits.

7.4.1 Melanoma

We start with a CRC whose components and regulation have been studied extensively in the past to examine whether the predicted associations recover known biology.

```
## melanoma LD-TF
melanoma_tf <- ldtf_fil %>% filter(tissue == 'Skin Cancer') %>% pull(symbol)
melanoma_nontf <- ldnontf_fil %>% filter(tissue == 'Skin Cancer') %>% pull(symbol)
mgenes <- unique(c(melanoma_tf, melanoma_nontf))

## annotate whether TF or not
annotation <- data.frame(symbol = mgenes) %>%
  mutate(TF = ifelse(symbol %in% melanoma_tf, 'no', 'yes')) %>%
  column_to_rownames('symbol')

## colors for heatmap
cols <- c(rep('#fff5f0', 4),'#fc9272',
            '#fb6a4a','#ef3b2c','#cb181d','#a50f15',rep('#67000d', 3))
ann_colors <- list(TF = c(yes='#ffffff', no='#111111'))

## melanoma gene codependency matrix
mela_matrix <- coess_mat[rownames(coess_mat) %in% mgenes,
                         colnames(coess_mat) %in% mgenes]

## remove genes that don't codepend with anything
codep_genes <- mela_matrix %>%
  apply(1, function(x) ifelse(x == 1, 0, x)) %>%
  apply(1, 'max') %>% .[. > 0.3]
mela_matrix <- mela_matrix[rownames(mela_matrix) %in% names(codep_genes),
                           colnames(mela_matrix) %in% names(codep_genes)]

pheatmap(mela_matrix,
         annotation_row = annotation,
         annotation_col = annotation,
         annotation_colors = ann_colors,
         color = colorRampPalette(cols)(50),
         clustering_method = 'ward.D2',
         border_color = NA)
```



7.5 CRC associated biological processes

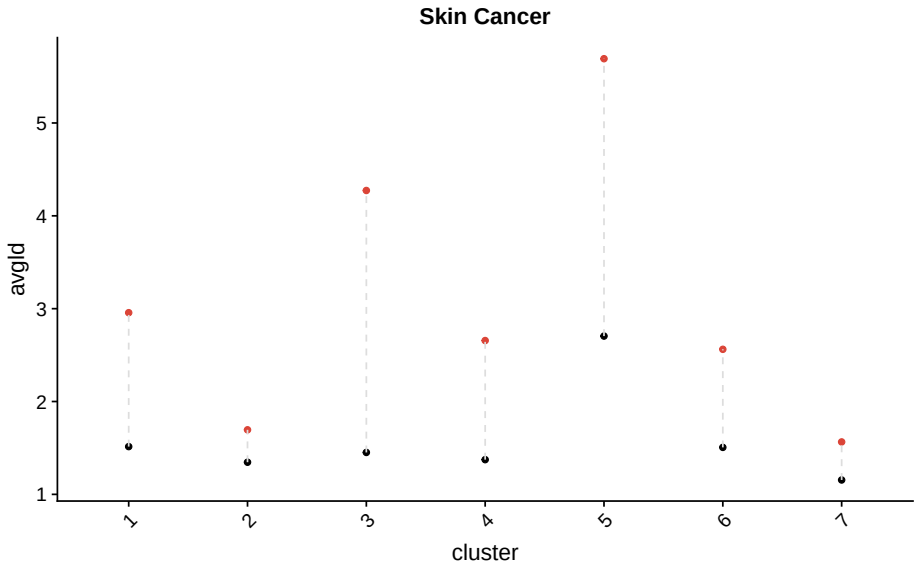
The results above that CRC-related biological processes can indeed be inferred from codependency. We aim to exploit this and infer biological processes associated with each CRC in each tumor type. To this end we group the lineage dependency genes for each cancer lineage into 2-10 clusters. We use a model based clusterin approach (Mclust, based on Gaussian mixture models) where the ideal number of clusters can be selected automatically based on the Bayesian Information Criterion (BIC). We then iterate over each of the inferred clusters and perform a gene set overrpresentation analysis on Gene ontology biological process terms. We use the functions 'goana' and 'topGO' implemented in the limma package to extract GO terms with the strongest enrichment for each cluster.

```
## a list of all avana library genes
avana_genes <- unique(depmap_ceres$symbol)
## get entrez gene IDs
avana_genes <- bitr(avana_genes,
                    fromType = 'SYMBOL',
                    toType = 'ENTREZID',
                    OrgDb = org.Hs.eg.db)

## progress bar
pb <- progress_estimated(length(unique(ldtf_fil$tissue)))

ld_genes <- ldtf_fil %>% bind_rows(ldnontf_fil)
enr_results <- ld_genes %>%
  group_by(tissue) %>%
  group_map(~{
    ## co-dependency matrix
    codep <- coess_mat[rownames(coess_mat) %in% .x$symbol,
                       colnames(coess_mat) %in% .x$symbol]
    ## cluster using model based clustering
    clust_fit <- Mclust(codep, G=2:10)
    clusters <- tibble(symbol = names(clust_fit$classification),
                   cluster = as.factor(clust_fit$classification))

    ## run GO enrichment
    enr <- clusters %>% mutate(tissue = .x$tissue[1]) %>%
      group_by(cluster) %>%
      group_map(~{
        ## convert cluster genes to entrez id
        gene <- avana_genes %>% filter(SYMBOL %in% .x$symbol) %>%
          pull(ENTREZID)

        ## run go over-representation analysis
        go_out <- goana(gene,
                        geneid = depmap_entrez$ENTREZID,
                        FDR = 0.2)
        go_out <- topGO(go_out, ontology = 'BP')

        ## return results
        as_tibble(go_out, rownames='go_id') %>%
```

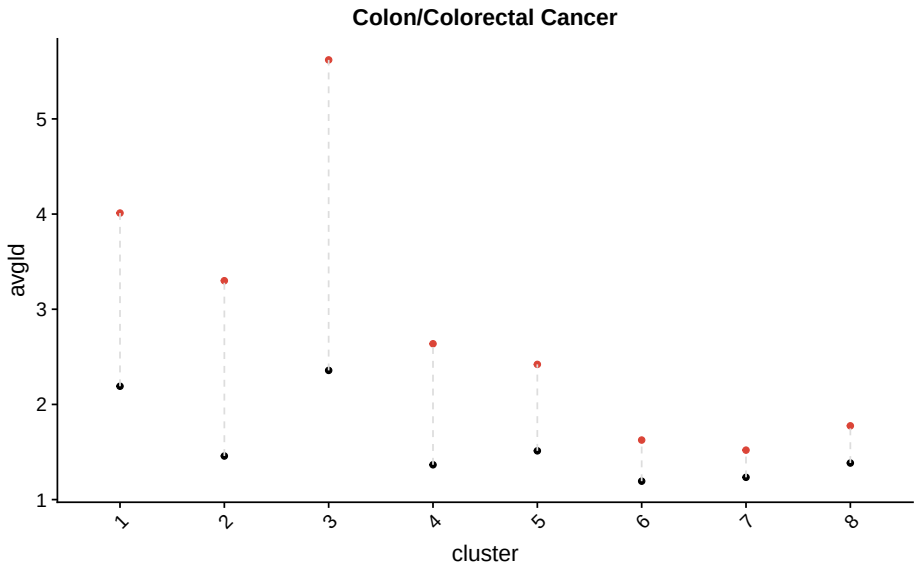
```
            mutate(genes = paste(.x$symbol, collapse=', '))
      })

    ## tick progress bar
    pb$tick()$print()

    ## return results
    return(enr)
  }) %>% ungroup()

## calculating LD-score statistics for each cluster
cluster_stats <- enr_results %>%
  distinct(tissue, cluster, genes) %>%
  mutate(avgld = map2(tissue, genes, ~{
    ld_genes %>% filter(symbol %in% unlist(strsplit(.y, ', ')),
                        tissue == .x) %>%
      summarise(avgld = mean(estimate), maxld = max(estimate))
  })) %>% unnest(avgld)

## combine everything
cluster_results <- enr_results %>% inner_join(cluster_stats)
```

We can plot the most significant term for each cluster in each tissue with their respective average and maximal LD-scores.

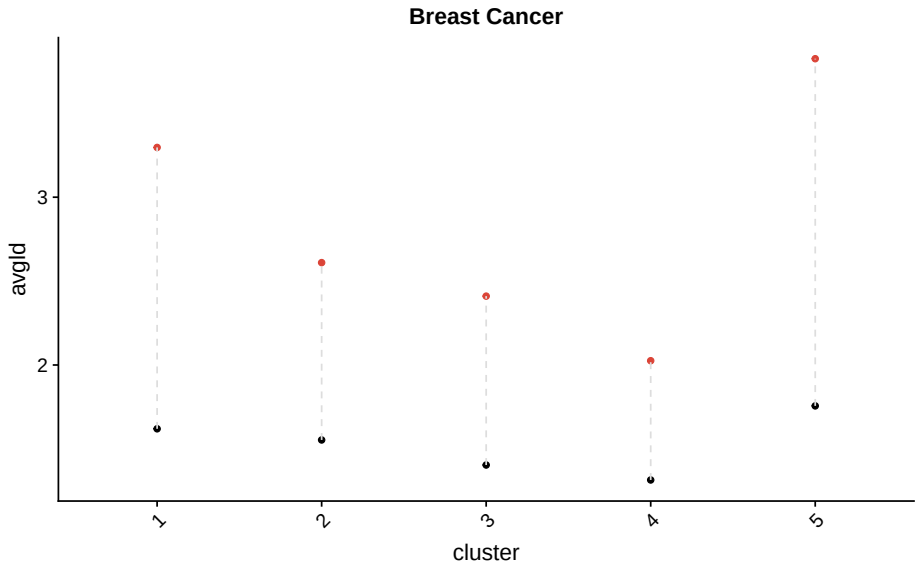
```
enr_p <- cluster_results %>%
  group_by(tissue, cluster) %>% top_n(1, -P.DE) %>% ungroup() %>%
  nest(-tissue) %>%
  mutate(p = purrr::map(data, ~{
    ggplot(.x, aes(x = cluster, label = Term, info = genes)) +
      geom_point(aes(y = avgld)) +
      geom_point(aes(y = maxld), colour = '#db4437') +
      geom_linerange(aes(ymin = avgld, ymax = maxld),
                     colour = '#dddddd', linetype='dashed') +
      theme(axis.text.x = element_text(angle=45, hjust=1))
  }))

## melanoma
enr_p$p[[20]] + ggtitle(enr_p$tissue[20])
```




```
## colorectal cancer
enr_p$p[[5]] + ggtitle(enr_p$tissue[5])
```




```
## breast cancer
enr_p$p[[4]] + ggtitle(enr_p$tissue[4])
```



8 Metastatic cancer cells might alter their core-regulatory circuit to adapt to their new niche

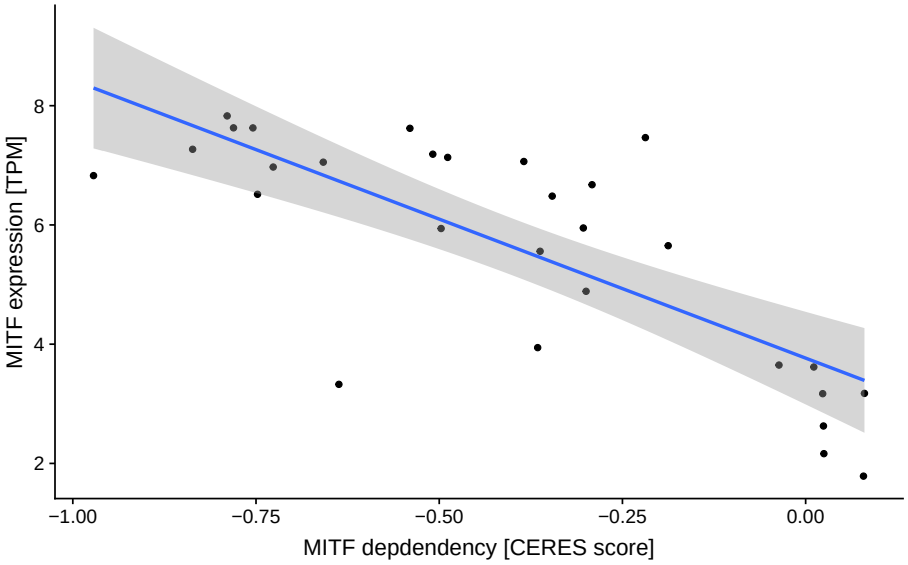
We first have to load melanoma patient tumor expression data (TCGA) and cancer cell line expression data (CCLE).

```
## melanoma patient tumor expr
data('melanoma_pt_expr', package = 'HDCRC2019')
## cell line expr
data('CCLE_18q4_TPM_long', package = 'HDCRC2019')
```

8.1 MITF dependency correlates with MITF expression

We make a scatter plot showing that MITF expression correlates with MITF dependency.

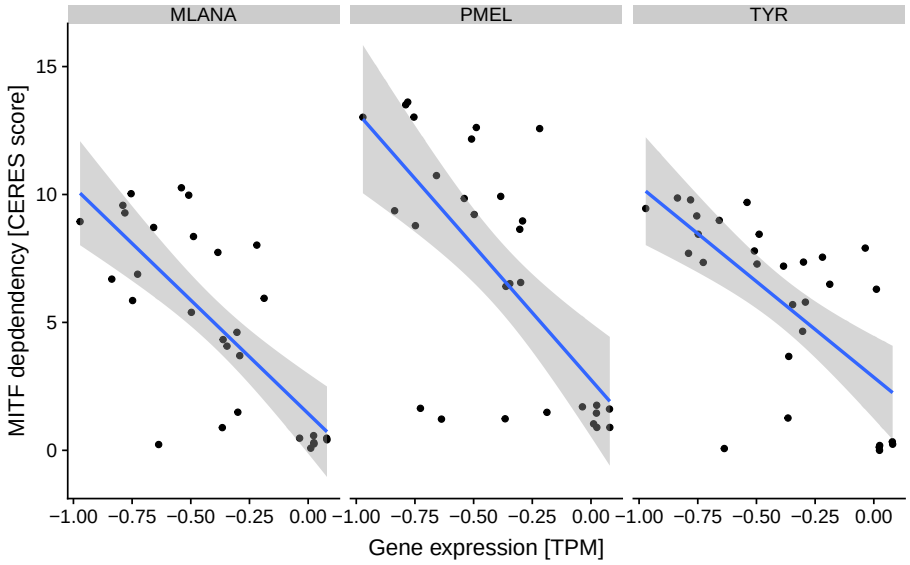
```
## make a df that contains MITF dep and expr for melanoma lines
mitf_dep_expr <- depmap_ceres %>%
  filter(tissue == 'Skin Cancer', symbol == 'MITF') %>%
  dplyr::select(symbol, cscore, cellline) %>%
  inner_join(ccle_expr %>% filter(gene == 'MITF') %>%
               dplyr::select(cellline, tpm))

## draw a scatter plot
mitf_dep_expr %>% ggplot(aes(cscore, tpm)) + geom_point() +
  geom_smooth(method='lm') +
  xlab('MITF depdendency [CERES score]') +
  ylab('MITF expression [TPM]')
```




```
## correlation coefficients
mitf_cor <- mitf_dep_expr %>%
  summarise(PCC = cor(cscore, tpm, method='pearson'),
            SCC = cor(cscore, tpm, method='spearman'))
```

We can make a similar scatter plot for a number of melanoma differentiation markers.

```
## some differentiation markers
melanoma_diff_markers <- c('MLANA', 'PMEL', 'TYR')

## scatter plots
diff_markers <- depmap_ceres %>%
  filter(symbol == 'MITF', tissue == 'Skin Cancer') %>%
  dplyr::select(cellline, cscore) %>%
  inner_join(ccle_expr %>% filter(gene %in% melanoma_diff_markers) %>%
               dplyr::select(gene, cellline, tpm))

diff_markers %>% ggplot(aes(cscore, tpm)) + geom_point() +
  geom_smooth(method='lm') + facet_wrap(~ gene, scales = 'free_x', nrow=1) +
  xlab('Gene expression [TPM]') + ylab('MITF depdendency [CERES score]')
```



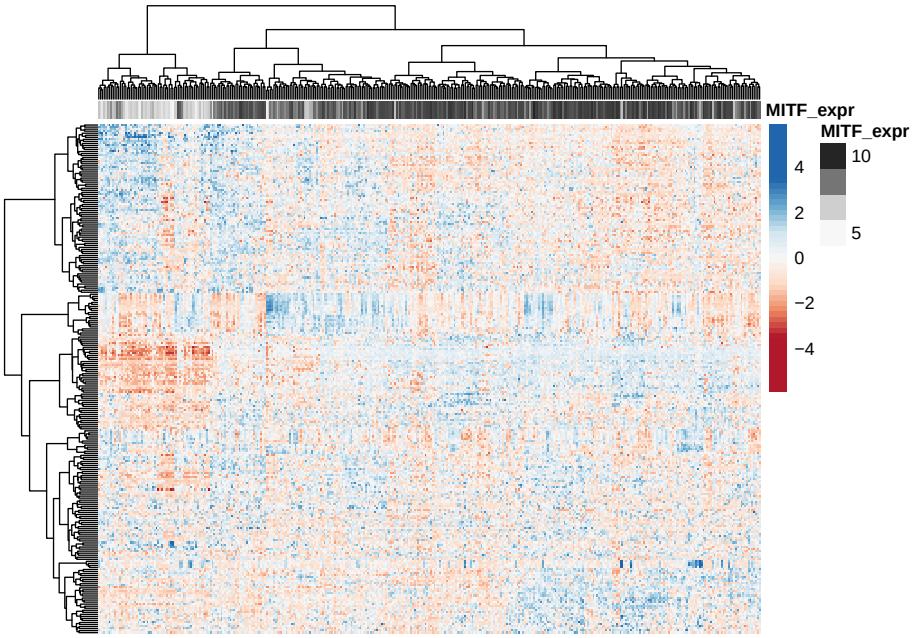

```
## correlation coefficients
diff_markers %>% group_by(gene) %>%
  summarise(PCC = cor(cscore, tpm, method='pearson'),
            SCC = cor(cscore, tpm, method='spearman'))
```

8.2 MITF expression in primary tumors

A similar melanoma subtype can be found by looking at patient tumor sample gene expression data. Specifically, there is a subset of samples where MITF expression is lost/decreased. This subtype has been previously linked to loss of differentiation and increased aggressiveness/metastatic potential. We plot a heatmap of TCGA primary tumor expression data that visualizes this subtype.

```
## expression matrix without non-expressed genes
pt_expr_mat <- melanoma_pt_expr %>% group_by(symbol) %>%
  mutate(med_expr = median(fpkm)) %>% ungroup() %>%
  filter(med_expr > 1.5) %>%
  acast(symbol ~ sample_nr, value.var = 'fpkm')

## top 500 most variably expressed genes
pt_expr_mat <- pt_expr_mat[order(apply(pt_expr_mat, 1, sd), decreasing=T)[1:250],]

## sample annotation with MITF expresssion
hm_anno <- melanoma_pt_expr %>% filter(symbol == 'MITF') %>%
  dplyr::select(-symbol, -sample_type, MITF_expr = fpkm, sample_nr) %>%
  as.data.frame() %>% column_to_rownames('sample_nr')
b2w_cols <- c('#f7f7f7','#d9d9d9','#bdbdbd','#525252','#252525')
annotation_colors <- list(
  MITF_expr = colorRampPalette(b2w_cols)(50)
)
```

```
## draw heatmap
hm_col_diverging <- c(rep('#b2182b', 3),'#ef8a62','#fddbc7',
                      '#f7f7f7','#d1e5f0','#67a9cf',rep('#2166ac', 3))
pheatmap(pt_expr_mat, scale = 'row',
         show_rownames = F, show_colnames=F,
         annotation_col = hm_anno,
         clustering_method = 'ward.D2',
         color = colorRampPalette(hm_col_diverging)(50),
         annotation_colors = annotation_colors)
```



8.2.1 Overexpressed genes in MITFlow samples

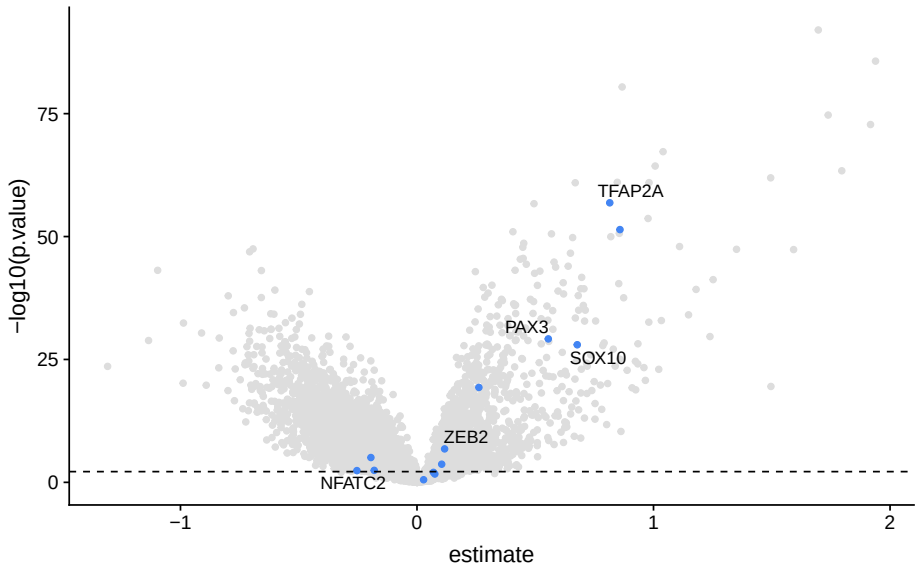
To understand what changes when tumors stop expressing MITF we determine all genes whose expression is significantly anti-correlated to MITF. We can then look at these genes and the biology that they might represent.

```
mitf_coreg <- melanoma_pt_expr %>%
  inner_join(melanoma_pt_expr %>% filter(symbol == 'MITF') %>%
               dplyr::select(sample_nr, MITF_expr = fpkm))

## progress bar
pb <- progress_estimated(length(unique(mitf_coreg$symbol)))

## statistical test
mitf_coreg_res <- mitf_coreg %>% group_by(symbol) %>%
  group_map(~ {
      pb$tick()$print()
      broom::tidy(lm(fpkm ~ MITF_expr, data = .x))
    }) %>% ungroup() %>%
```

```
  mutate(FDR = p.adjust(p.value, method='BH'))
mitf_coreg_res <- mitf_coreg_res %>%
  filter(term != '(Intercept)', symbol != 'MITF') %>%
  arrange(p.value)

## for each gene annotate main KO effect in cell lines
mitf_coreg_res <- mitf_coreg_res %>%
  left_join(depmap_ceres %>% filter(tissue == 'Skin Cancer') %>%
              group_by(symbol) %>%
              summarise(main_effect = median(cscore)) %>% ungroup())
```

We make a volcano plot highlighting differentialy expressed genes. Specifically, I highlight genes that are part of the predicted melanoma core-regulatory circuit.

```
## downregulated genes
crc <- ldtf_crispr %>%
  filter(estimate > 1, FDR < 0.05,
         med_expr > 0.01,
         tissue == 'Skin Cancer') %>%
  pull(symbol)

## cutoff (1% FDR)
co <- mitf_coreg_res %>% filter(FDR < 0.01) %>%
  arrange(desc(p.value)) %>% pull(p.value) %>%
  head(1) %>% log10() %>% `*`(-1)

## genes to highligh/label
label <- c('MITF', 'SOX10', 'ZEB2', 'PAX3', 'NFATC2', 'TFAP2A')
## subsample non-significant dots to make less heavy scatter plot
df <- mitf_coreg_res %>% filter(FDR > 0.01) %>%
  sample_n(1000) %>%
  bind_rows(mitf_coreg_res %>% filter(FDR < 0.01))

## draw volcano
ggplot() +
  geom_point(data = df,
                 aes(estimate, -log10(p.value)),
           colour = '#dddddd', fill = '#dddddd') +
  geom_point(data = filter(mitf_coreg_res, symbol %in% crc),
             aes(estimate, -log10(p.value)), colour = '#4285f4') +
  geom_text_repel(data = filter(mitf_coreg_res,
                                symbol %in% label),
                  aes(estimate, -log10(p.value), label = symbol)) +
  geom_hline(yintercept = co, linetype = 'dashed')
```



8.3 t-SNE analysis

We might hypothesize that invading melanoma cells can alter their core-regulatory circuit to adapt to their new environment. To test this we perform a t-SNE analysis to cluster tumor samples by LD-TF expression. We can then observe where MITF-low and MITF-high samples locate within the plot.

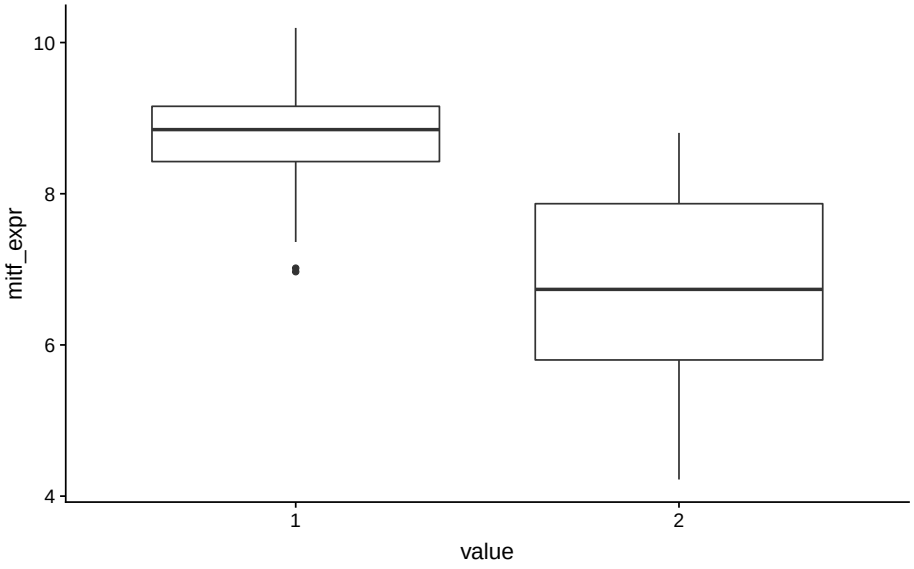
8.3.1 Primary tumors

To this end we first perform this analysis on primary tumor data which should better represent the true nature of the tumor biology. Also many more samples are available here (TCGA). We highlight samples that express low amounts of MITF (as indicited by the low-MITF cluster in the above heatmap). We first select the samples that represent this subtype.

```
## select samples in cluster with low MITF
low_mitf_samples <- pt_expr_mat %>% t() %>%
  dist() %>% hclust() %>% cutree(k=2) %>%
  melt() %>% as_tibble(rownames='sample') %>%
  inner_join(melanoma_pt_expr %>% filter(symbol == 'MITF') %>%
               dplyr::select(sample = sample_nr, mitf_expr = fpkm)) %>%
  mutate(value = as.character(value))
```

We draw a boxplot of MITF expression just to make sure the correct samples were selcted.

```
## can we see the difference in MITF?
low_mitf_samples %>% ggplot(aes(value, mitf_expr)) + geom_boxplot()
```

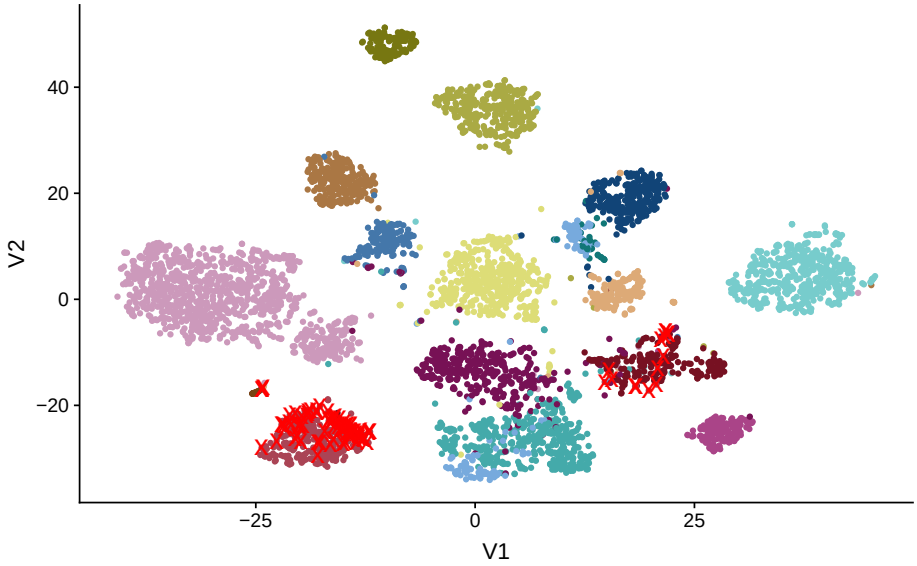



```
## select low MITF melanoma samples
low_mitf_samples <- low_mitf_samples %>% filter(value == '2') %>%
  pull(sample)
```

Next, we perform a t-SNE analysis clustering samples by their expression of significant LD-TF. Before performing the t-SNE we Z-transform the log-scaled FPKM expression values. In the plot we highlight the 'low MITF' tumors, comparing them to 'high MITF' melanoma tumors.

```
## select significant LD-TF
ldtf_sig <- ldtf_crispr %>% filter(FDR < 0.05, estimate > 1) %>%
  pull(symbol) %>% unique()

## load patient tumor pan cancer expression data for LD-TF
data('pt_pancancer_expr', package = 'HDCRC2019')

## perform t-SNE (try default params)
set.seed(1234)
tsne <- Rtsne(t(pt_pancancer_expr))

## merge tSNEs with sample info
tsne_plot_data <- as_tibble(tsne$Y) %>%
  mutate(case_id = colnames(pt_pancancer_expr)) %>%
  inner_join(distinct(tcga_data, case_id, tissue, stype))

## colour palette
tol18rainbow <- c('#771155', '#AA4488', '#CC99BB',
                  '#114477', '#4477AA', '#77AADD',
                  '#117777', '#44AAAA', '#77CCCC',
                  '#777711', '#AAAA44', '#DDDD77',
                  '#774411', '#AA7744', '#DDAA77',
                  '#771122', '#AA4455', '#DD7788')

## draw dot plot, highlighting low-MITF
```

```
tsne_p <- tsne_plot_data %>%
  ggplot(aes(V1, V2, colour = tissue, shape = stype)) + geom_point() +
  geom_point(data = tsne_plot_data %>% filter(case_id %in% low_mitf_samples),
             aes(V1, V2), colour = 'red', size = 5, stroke=2, shape = 'x') +
  theme(legend.position = 'none') +
  scale_colour_manual(values = tol18rainbow)

tsne_p
```

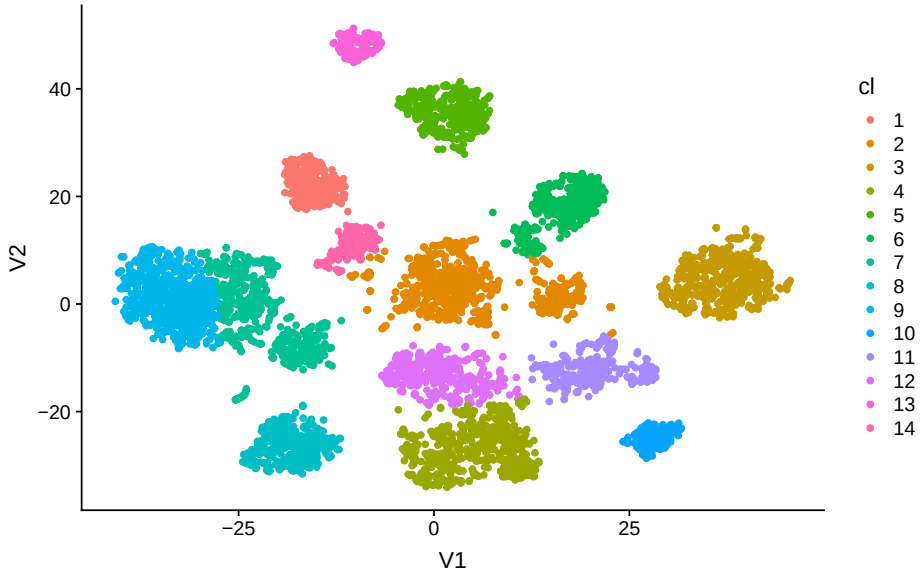


We can clearly see that LD-TF expression is sufficient to separate the different cancer types from each other. We can also see that melanoma tumors that express low levels of MITF start to jump to cluster with a different tumor type (sarcoma). This could indicate that invading melanoma cells can alter their core-regulatory circuit to blend in with their new tissue niche.

8.3.2 Cluster analysis of melanoma and sarcoma clusters

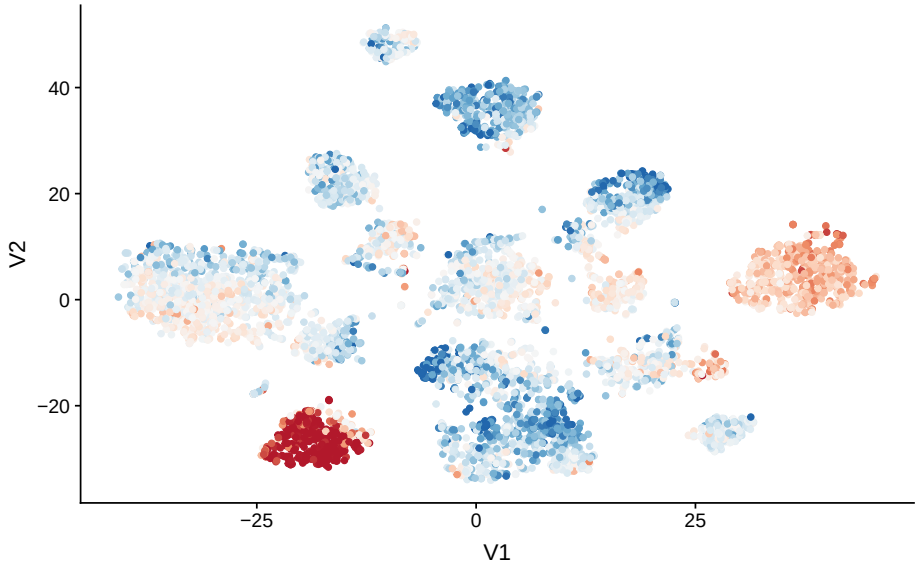
We see that in the patient tumor tSNE, MITF-low samples tend to cluster with sarcoma samples. We infer LD-TF whose expression drives each cluster to gain some insight into what might be going on here. To this end we cluster the data into 14 groups using model based clustering. We then use a linear model to derrive expression markers for each of those groups.

```
## selelct tSNEs 1&2 and make a matrix
cl <- tsne_plot_data %>% .[,1:3] %>% as.data.frame() %>% column_to_rownames('case_id')
## perform model based clustering
cl <- Mclust(cl, G= 14)$classification
cl <- tibble(case_id = names(cl), cl = as.factor(cl))

## visualize resulting clusters
tsne_plot_data %>% inner_join(cl) %>%
  ggplot(aes(V1, V2, colour = cl)) + geom_point()
```

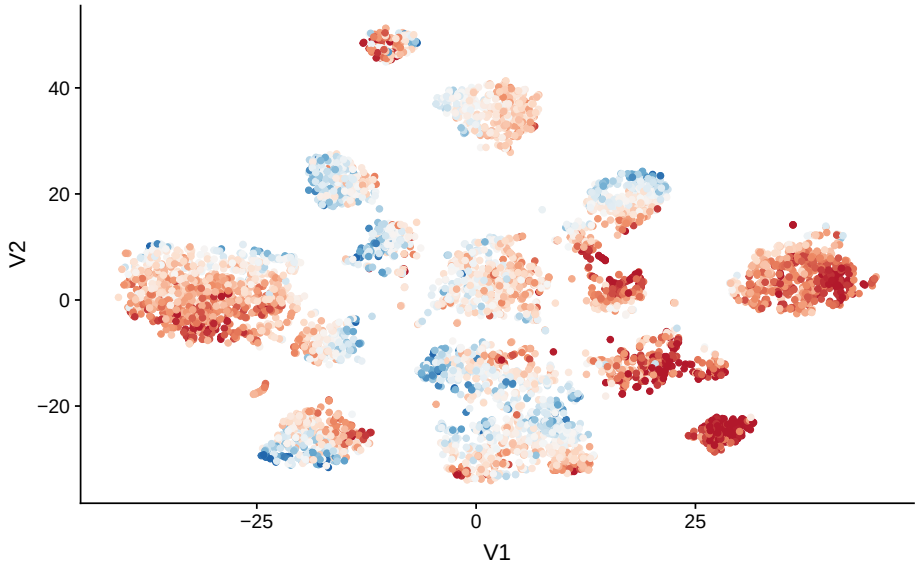



```
## markers for each cluster (LD-TF only)
cl_markers <- tcga_data %>% filter(symbol %in% ldtf_sig) %>%
  inner_join(cl) %>% mutate(log_fpkm = log(fpkm + 1)) %>%
  filter(stype == 'tumor') %>%
  group_by(symbol) %>%
  mutate(fpkm_ct = log_fpkm - median(log_fpkm)) %>%
  group_map(~ broom::tidy(lm(fpkm_ct ~ 0 + cl, data = .x))) %>%
  ungroup() %>%
  arrange(p.value) %>% mutate(FDR = p.adjust(p.value, method='BH'))

## add tissue info
cl_markers <- cl_markers %>% dplyr::select(cl=term, everything()) %>%
  inner_join(cl %>% inner_join(tsne_plot_data %>% dplyr::select(case_id, tissue)) %>%
               count(cl, tissue) %>%
               group_by(cl) %>% top_n(1, n) %>% ungroup() %>%
               dplyr::select(-n) %>% mutate(cl = paste0('cl', cl)))

## best marker for each cluster
best_marker <- cl_markers %>% group_by(cl) %>% top_n(1, estimate) %>% ungroup()

## this plot can highlight expression of marker genes in the tSNE
highlight_marker_in_tsne <- function(marker_gene){
  ## merge tSNEs with sample info
  pdata <- as_tibble(tsne$Y) %>%
    mutate(case_id = colnames(pt_pancancer_expr)) %>%
    left_join(tcga_data %>% filter(symbol == marker_gene) %>%
                mutate(SIX1 = log(fpkm + 1)) %>%
                dplyr::select(case_id, SIX1))

  ## draw plot
  pdata %>%
    ggplot(aes(V1, V2, colour = SIX1)) + geom_point() +
```

```
        theme(legend.position = 'none') +
        scale_colour_gradientn(colors = colorRampPalette(rev(hm_col_diverging))(50))
    }

    highlight_marker_in_tsne('MITF')
```

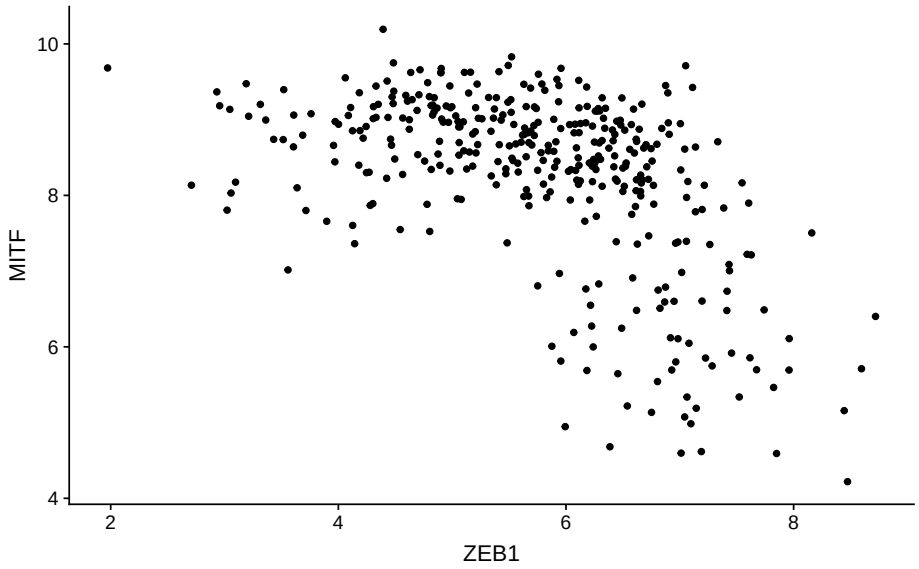



```
    highlight_marker_in_tsne('ZEB1')
```




```
    ## expression of MITF compared to ZEB1
    tcga_data %>% filter(symbol %in% c('MITF', 'ZEB1', 'ZEB2'),
                         tissue == 'Skin Cancer') %>%
      mutate(log_fpkm = log(fpkm + 1)) %>%
      dplyr::select(case_id, symbol, log_fpkm) %>%
      spread(symbol, log_fpkm) %>%
```

```
  ggplot(aes(ZEB1, MITF)) +
  geom_point()
```



9 MITF anticorrelated depdendencies

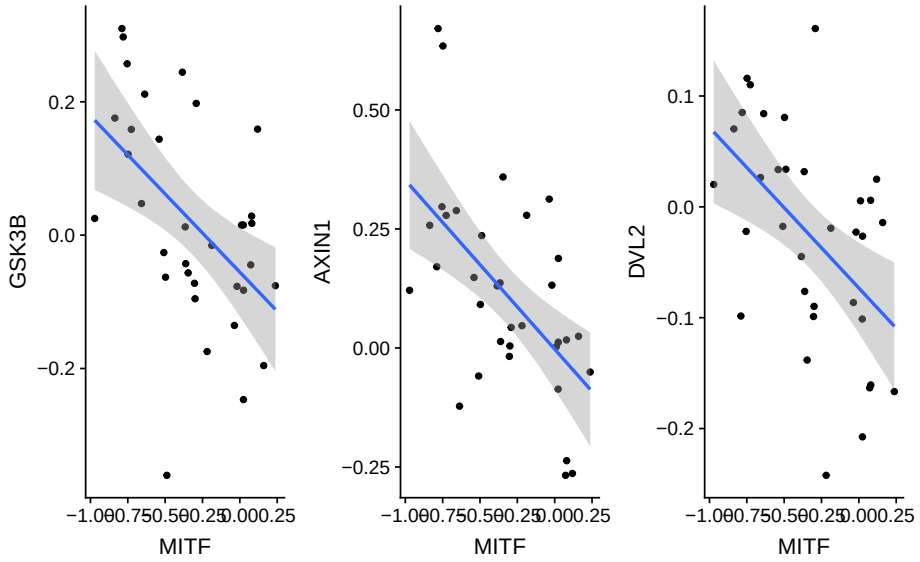
We might hypothesize that cell lines that do not express MITF and hence lose MITF dependency alter their core-regulatory circuit which might come with new depdendencies. Therefore, we perform an analysis that systematically identifies MITF 'inverse-essential' genes.

```
## negative correlations with MITF dep in melanoma
mitf_codep <- depmap_ceres %>% filter(tissue == 'Skin Cancer') %>%
  dplyr::select(symbol, cscore, cellline) %>%
  inner_join(depmap_ceres %>% filter(tissue == 'Skin Cancer',
                                     symbol == 'MITF') %>%
               dplyr::select(cellline, MITF_dep = cscore))

## run linear model as above
mitf_codep_res <- mitf_codep %>% group_by(symbol) %>%
  group_map(~ broom::tidy(lm(cscore ~ MITF_dep, data = .x))) %>%
  mutate(FDR = p.adjust(p.value, method='BH'))
mitf_codep_res <- mitf_codep_res %>% filter(term == 'MITF_dep') %>%
  arrange(p.value) %>% filter(symbol != 'MITF')
```

It is noticable that the most 'inverse-dependent' genes include several members of the Wnt-signaling destruction complex. We can generate scatter plots to highlight these cases. It is known that MITF activity is in part regulated by Wnt-signaling. It could therefore be the cases that negative regulators of the Wnt pathway play an important role in keeping MITF levels low.

```
## plot MITF coessentiality
plot_mitf_coess <- function(gene){
```

```
    depmap_ceres %>% filter(symbol %in% c('MITF', gene),
                            tissue == 'Skin Cancer') %>%
      dplyr::select(symbol, cscore, cellline) %>%
      spread(symbol, cscore) %>%
      ggplot(aes_string('MITF', gene)) + geom_point() +
      geom_smooth(method='lm')
  }

  p_gsk3b <- plot_mitf_coess('GSK3B')
  p_axin1 <- plot_mitf_coess('AXIN1')
  p_dvl2 <- plot_mitf_coess('DVL2')

  ## draw to canvas
  p_gsk3b + p_axin1 + p_dvl2 + plot_layout(c(3,1))
```




```
  ## correlation coefficients
  depmap_ceres %>%
    filter(symbol %in% c('MITF', 'GSK3B', 'AXIN1', 'DVL2'),
           tissue == 'Skin Cancer') %>%
    dplyr::select(symbol, cscore, cellline) %>%
    spread(symbol, cscore) %>%
    gather(gene, cscore, -cellline, -MITF) %>%
    group_by(gene) %>%
    summarise(PCC = cor(MITF, cscore, method='pearson'),
              SCC = cor(MITF, cscore, method='spearman')) %>%
    ungroup()
```

10 Session info

```
sessionInfo()
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.14.3
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] forcats_0.4.0             stringr_1.4.0
##  [3] purrr_0.3.2               readr_1.3.1
##  [5] tidyr_0.8.3               tibble_2.1.1
##  [7] tidyverse_1.2.1           cowplot_0.9.4
##  [9] Rtsne_0.15                limma_3.34.9
## [11] clusterProfiler_3.6.0     DOSE_3.4.0
## [13] mclust_5.4.3              org.Hs.eg.db_3.5.0
## [15] biomaRt_2.34.2            Gviz_1.22.3
## [17] GenomicRanges_1.30.3      GenomeInfoDb_1.14.0
## [19] MASS_7.3-51.1             RTCGA.rnaseq_20151101.8.0
## [21] RTCGA_1.8.0               ggrepel_0.8.0
## [23] ggplot2_3.1.0             patchwork_0.0.1
## [25] Organism.dplyr_1.6.2      AnnotationFilter_1.2.0
## [27] dplyr_0.8.0.1             GO.db_3.5.0
## [29] AnnotationDbi_1.40.0      IRanges_2.12.0
## [31] S4Vectors_0.16.0          Biobase_2.38.0
## [33] BiocGenerics_0.24.0       fgsea_1.4.1
## [35] Rcpp_1.0.1                pheatmap_1.0.12
## [37] mixtools_1.1.0            reshape2_1.4.3
## [39] broom_0.5.1               BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.1
##   [2] backports_1.1.3
##   [3] Hmisc_4.2-0
##   [4] AnnotationHub_2.10.1
##   [5] fastmatch_1.1-0
##   [6] BiocFileCache_1.2.3
##   [7] igraph_1.2.4
##   [8] plyr_1.8.4
##   [9] lazyeval_0.2.2
```

```
##  [10] splines_3.4.1
##  [11] BiocParallel_1.12.0
##  [12] digest_0.6.18
##  [13] GOSemSim_2.4.1
##  [14] BiocInstaller_1.28.0
##  [15] ensembldb_2.2.2
##  [16] htmltools_0.3.6
##  [17] viridis_0.5.1
##  [18] fansi_0.4.0
##  [19] magrittr_1.5
##  [20] checkmate_1.9.1
##  [21] memoise_1.1.0
##  [22] BSgenome_1.46.0
##  [23] cluster_2.0.7-1
##  [24] Biostrings_2.46.0
##  [25] modelr_0.1.4
##  [26] matrixStats_0.54.0
##  [27] prettyunits_1.0.2
##  [28] colorspace_1.4-1
##  [29] blob_1.1.1
##  [30] rvest_0.3.2
##  [31] rappdirs_0.3.1
##  [32] haven_2.1.0
##  [33] xfun_0.5
##  [34] jsonlite_1.6
##  [35] crayon_1.3.4
##  [36] RCurl_1.95-4.12
##  [37] survival_2.43-3
##  [38] VariantAnnotation_1.24.5
##  [39] zoo_1.8-5
##  [40] glue_1.3.1
##  [41] survminer_0.4.3
##  [42] gtable_0.2.0
##  [43] zlibbioc_1.24.0
##  [44] XVector_0.18.0
##  [45] TxDb.Hsapiens.UCSC.hg38.knownGene_3.4.0
##  [46] DelayedArray_0.4.1
##  [47] scales_1.0.0
##  [48] DBI_1.0.0
##  [49] ggthemes_4.1.0
##  [50] viridisLite_0.3.0
##  [51] xtable_1.8-3
##  [52] progress_1.2.0
##  [53] cmprsk_2.2-7
##  [54] htmlTable_1.13.1
##  [55] foreign_0.8-71
##  [56] bit_1.1-14
##  [57] km.ci_0.5-2
##  [58] Formula_1.2-3
##  [59] htmlwidgets_1.3
##  [60] httr_1.4.0
```

```
##  [61] RColorBrewer_1.1-2
##  [62] acepack_1.4.1
##  [63] pkgconfig_2.0.2
##  [64] XML_3.98-1.19
##  [65] nnet_7.3-12
##  [66] dbplyr_1.3.0
##  [67] utf8_1.1.4
##  [68] labeling_0.3
##  [69] tidyselect_0.2.5
##  [70] rlang_0.3.2
##  [71] later_0.8.0
##  [72] cellranger_1.1.0
##  [73] munsell_0.5.0
##  [74] tools_3.4.1
##  [75] cli_1.1.0
##  [76] generics_0.0.2
##  [77] RSQLite_2.1.1
##  [78] evaluate_0.13
##  [79] yaml_2.2.0
##  [80] knitr_1.22
##  [81] bit64_0.9-7
##  [82] survMisc_0.5.5
##  [83] nlme_3.1-137
##  [84] mime_0.6
##  [85] DO.db_2.9
##  [86] xml2_1.2.0
##  [87] compiler_3.4.1
##  [88] rstudioapi_0.10
##  [89] curl_3.3
##  [90] interactiveDisplayBase_1.16.0
##  [91] stringi_1.4.3
##  [92] GenomicFeatures_1.30.3
##  [93] lattice_0.20-38
##  [94] ProtGenerics_1.10.0
##  [95] Matrix_1.2-15
##  [96] KMsurv_0.1-5
##  [97] pillar_1.3.1
##  [98] data.table_1.12.0
##  [99] bitops_1.0-6
## [100] qvalue_2.10.0
## [101] httpuv_1.5.0
## [102] rtracklayer_1.38.3
## [103] R6_2.4.0
## [104] latticeExtra_0.6-28
## [105] bookdown_0.9
## [106] RMySQL_0.10.17
## [107] promises_1.0.1
## [108] gridExtra_2.3
## [109] dichromat_2.0-0
## [110] assertthat_0.2.1
## [111] SummarizedExperiment_1.8.1
```

```
## [112] withr_2.1.2.9000
## [113] GenomicAlignments_1.14.2
## [114] Rsamtools_1.30.0
## [115] GenomeInfoDbData_1.0.0
## [116] hms_0.4.2
## [117] rpart_4.1-13
## [118] rvcheck_0.1.3
## [119] rmarkdown_1.12
## [120] segmented_0.5-3.0
## [121] ggpubr_0.2
## [122] biovizBase_1.26.0
## [123] lubridate_1.7.4
## [124] shiny_1.2.0
## [125] base64enc_0.1-3
```