Analysis of CTG screens with the clinical drug panel

Benedikt Rauscher

7/27/2018

Contents

1 Dependencies

```
library(perm)
library(limma)
library(pheatmap)
library(PharmacoGx)
library(reshape2)
library(patchwork)
library(ggraph)
library(tidygraph)
library(ggrepel)
library(biobroom)
library(tidyverse)
```

2 Analysis

The following code describes the analysis of CTG compound screens performed with a clinically relevant panel of anticancer drugs presented in the manuscript.

2.1 Normalization of raw data

In a first step we need to normalize raw data coming from the plate reader. We screened each PDO line in two replicate. The library fits on exactly one 384 plate.

We first load the raw CTG data from an Rdata file.

```
data('ctg_data_raw', package='SCOPEAnalysis')
```

2.1.1 Plate annotation

We next load the plate annotation and merge this information with the data to know which drug at which concentration went where.

```
## read plate annotation from R data file
data('plate_anno', package='SCOPEAnalysis')

## join with ctg data
ctg_data <- ctg_data_raw %>% inner_join(plate_anno) %>%
  extract(well, c('row', 'col'), regex='([A-P])(\\d{2})', remove=F) %>%
  mutate(row=match(row, LETTERS), col=as.integer(col))
```

There was a technical issue with one of the plates. Wells have to be flagged accordingly.

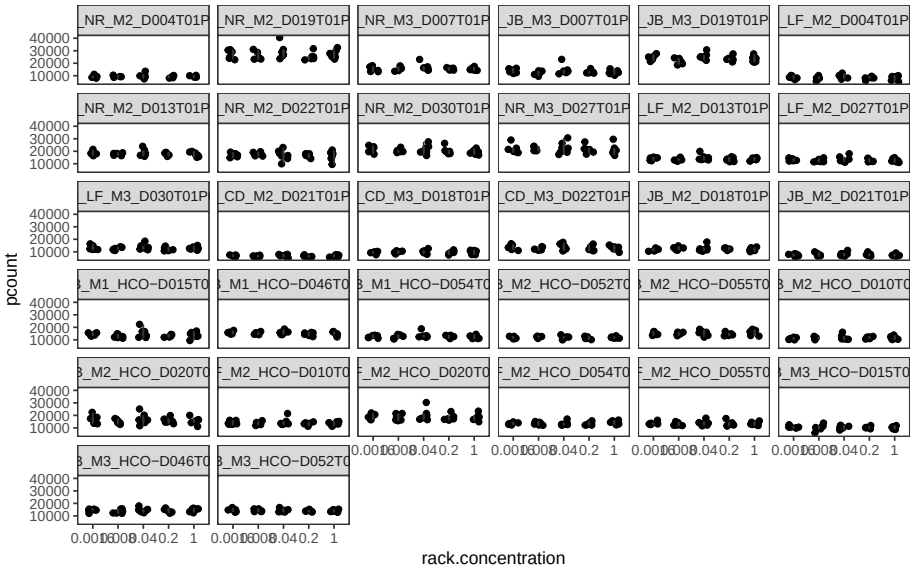
```
ctg_data <- ctg_data %>%
  mutate(pcount = ifelse(screen == '170502_NR_M2_D004T01P006L08' &
```

```
                                (row %in% seq(1, 15, by=2)) &
                                (col %in% 1:13), NA, pcount))
```

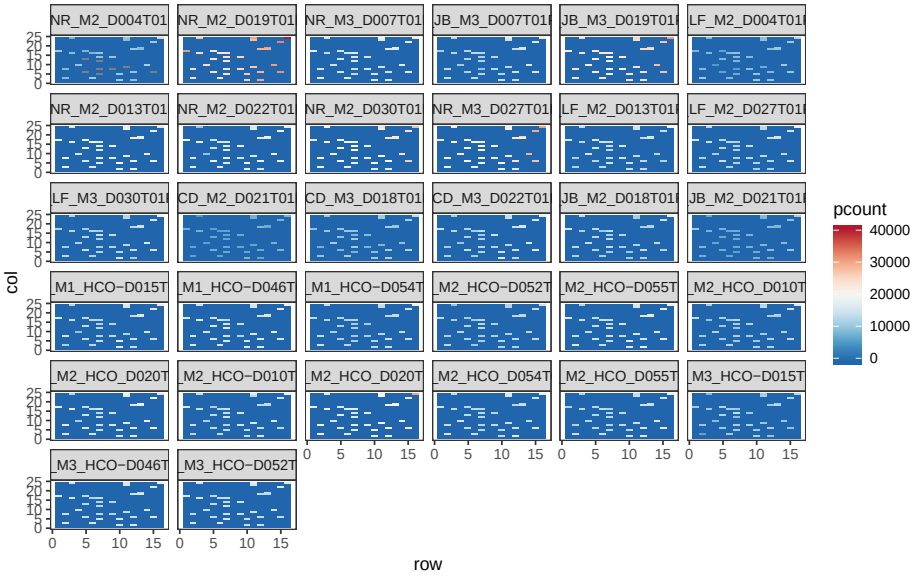
2.2 Examination of the raw data

Now we can get an idea of what the data looks like. First we look at only one plate at a time to check the photon counts of DMSO controls to see how they vary and if there might be a positional bias that needs correction.

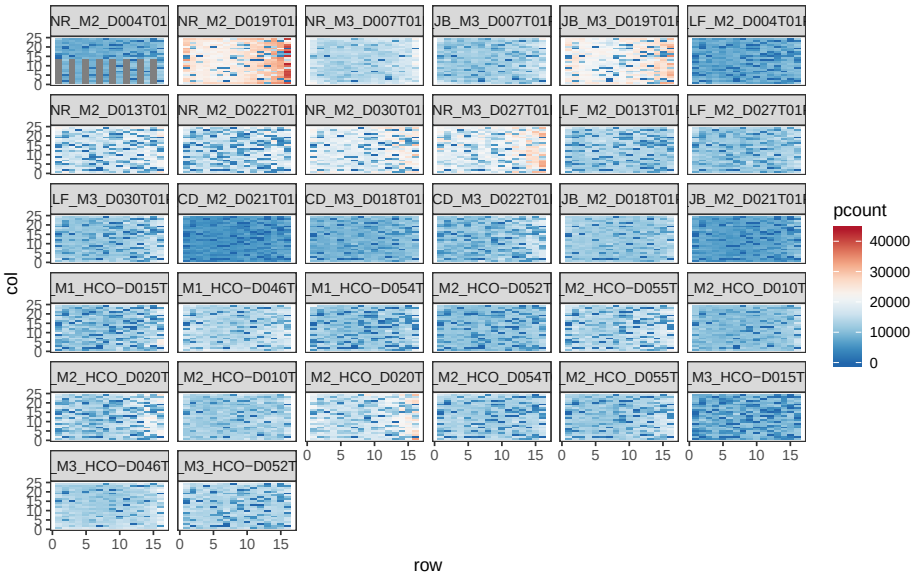
```
## colour palette for drawing
colpal <- colorRampPalette(rev(c('#b2182b','#d6604d','#f4a582',
                                 '#fddbc7','#f7f7f7','#d1e5f0',
                                 '#92c5de','#4393c3','#2166ac')))(150)

## box plot of DMSO photon counts
ctg_data %>% filter(drug == 'DMSO') %>%
  ggplot(aes(rack.concentration, pcount)) +
  geom_jitter(width=0.2) +
  facet_wrap(~screen) + geom_boxplot(alpha=0) +
  theme_bw() + theme(panel.grid=element_blank())
```




```
## heat map of DMSO photon counts, ignoring other values
ctg_data %>% mutate(pcount = ifelse(drug != 'DMSO', -1000, pcount)) %>%
  ggplot(aes(row, col, fill = pcount)) + geom_raster() +
  facet_wrap(~screen) + theme_bw() +
  theme(panel.grid=element_blank()) +
  scale_fill_gradientn(colors=colpal)
```



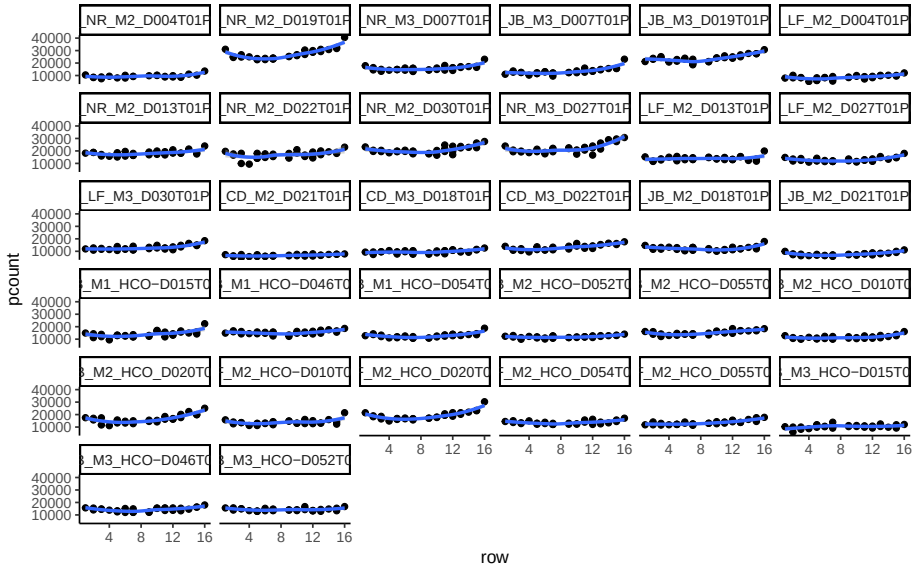

```
## heat map of all photon counts
ctg_data %>% ggplot(aes(row, col, fill = pcount)) +
  geom_raster() +
  facet_wrap(~screen) + theme_bw() +
  theme(panel.grid=element_blank()) +
  scale_fill_gradientn(colors=colpal)
```



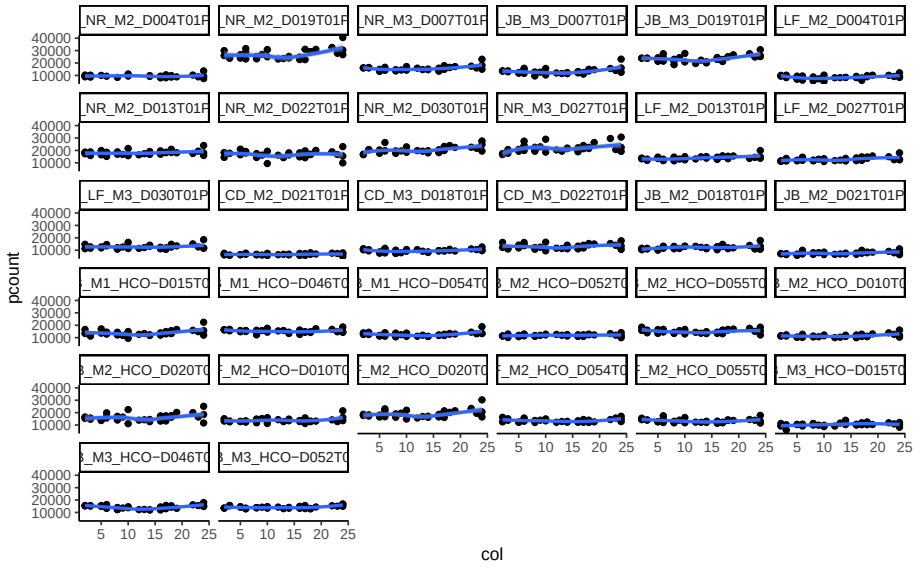
It seems that there is a spatial bias where photon counts get higher towards the bottom and the maybe edges of the plate.

```
## scatter plot with robust loess fit for rows.
ctg_data %>% filter(drug == 'DMSO') %>%
  ggplot(aes(row, pcount)) + geom_point() +
  geom_smooth() +
```

```
  facet_wrap(~screen) + theme_classic()
```




```
## scatter plot with robust loess fit for columns
ctg_data %>% filter(drug == 'DMSO') %>%
  ggplot(aes(col, pcount)) + geom_point() +
  geom_smooth() +
  facet_wrap(~screen) + theme_classic()
```



2.3 Normalization of photon counts

Looking at above plots there seem to be row and column effects that we want to correct for. To correct for spacial bias we apply a loess-normalization to the data.

```
## split data by plate and apply loess normalization
ctg_loess <- ctg_data %>% split(.$screen) %>% lapply(function(s){
  ## loess fit. include row and column effects, fit only on DMSO
  fit <- loess(pcount ~ 0 + row + col, data=filter(s, drug=='DMSO'),
               surface = 'direct')
  ## apply normalization
  s %>% mutate(norm_fac = predict(fit, data.frame(row=row, col=col)),
               pcount_norm = pcount - (norm_fac - median(norm_fac)))
}) %>% bind_rows()
```

After the loess-normalization some photon count values become negative which does not make any sense. We center the distribution of phenotypes such that the lowest photon count becomes 0.

```
ctg_loess <- ctg_loess %>%
  mutate(pcount_norm = ifelse(pcount_norm < 0, 0, pcount_norm),
         screen = gsub('HCO_', 'HCO-', screen)) %>%
  separate(screen, c('date', 'tag1', 'tag2', 'plate'), sep='_', remove=F) %>%
  mutate(date = as.Date(paste0('20', date), format='%Y%m%d'))
```

Next we normalize the photon counts to the median of the DMSO control on each plate.

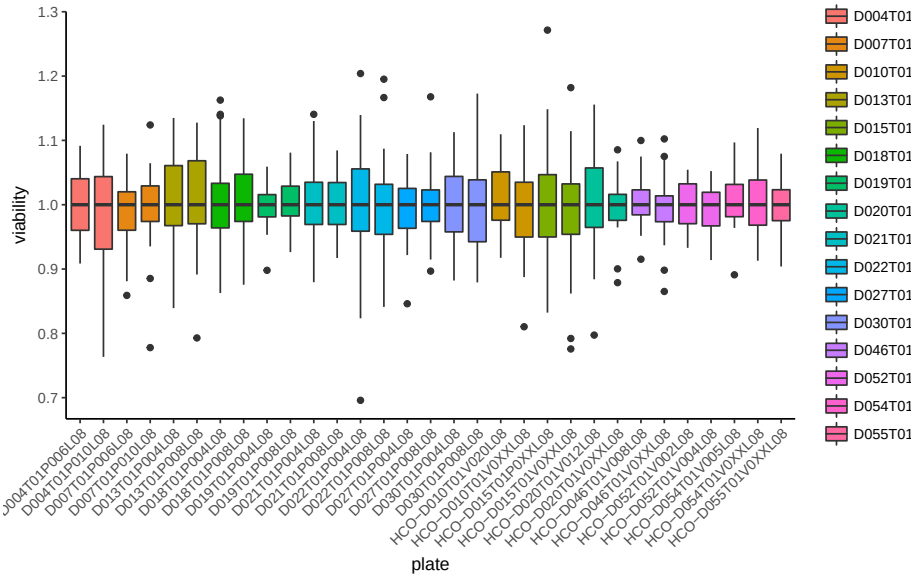
```
## median of dmso controls
ctrls_med <- ctg_loess %>% filter(drug == 'DMSO') %>%
  group_by(screen) %>%
  summarise(med_ctrl = median(pcount_norm, na.rm=T)) %>%
  ungroup()

## normalize to median of DMSO
ctg_norm <- ctg_loess %>% inner_join(ctrls_med) %>%
  group_by(screen) %>%
  mutate(viability = pcount_norm/med_ctrl) %>% ungroup() %>%
  extract(screen, 'line', regex = '.+(D0.+T01)', remove=F)
```

2.4 Quality control of normalized viability measurements

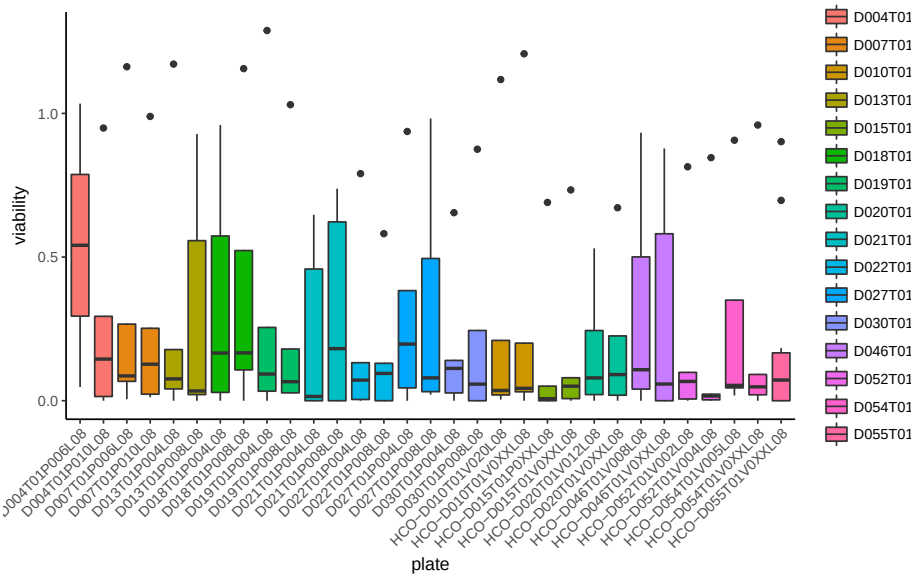
We first check if DMSO controls across all plates have a median of 1 and are comparable. We further investigate the noise that is observed for these controls. Additionally we make a plot for Bortezomib which can function as positive control.

```
ctg_norm %>% filter(drug == 'DMSO') %>%
  ggplot(aes(plate, viability, fill=line)) + geom_boxplot() +
  theme_classic() +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```

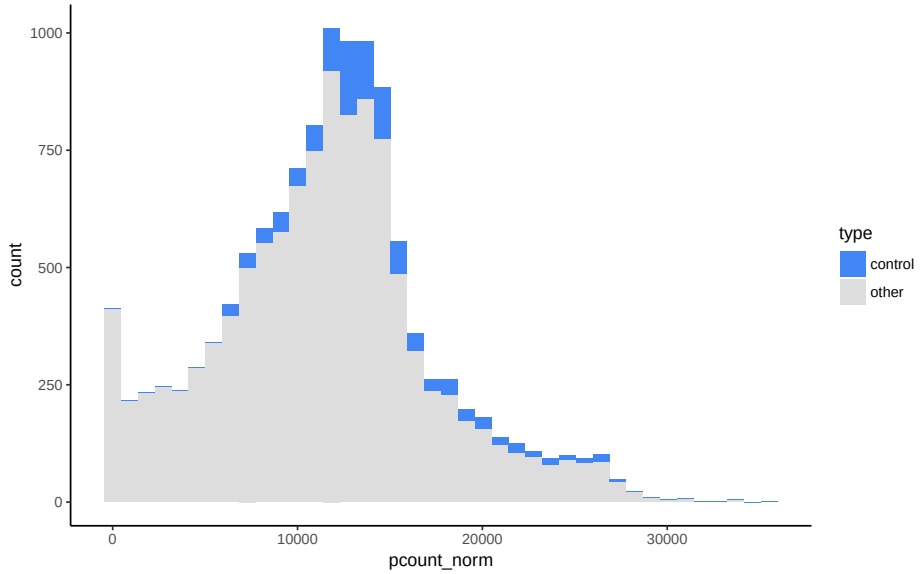
Analysis of CTG screens with the clinical drug panel




```
ctg_norm %>% filter(drug == 'Bortezomib') %>%
  ggplot(aes(plate, viability, fill=line)) + geom_boxplot() +
  theme_classic() +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```

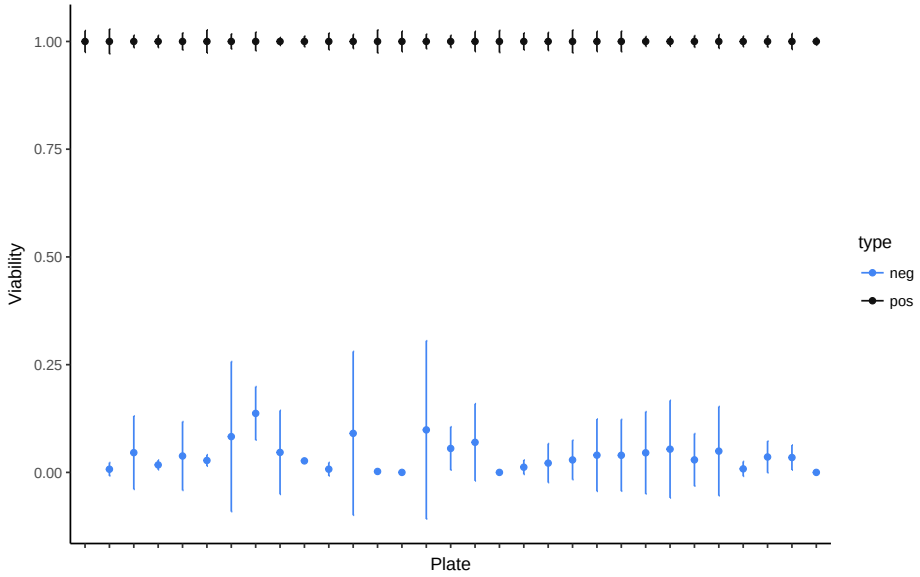



```
ctg_norm %>% mutate(type = ifelse(drug == 'DMSO', 'control', 'other')) %>%
  ggplot(aes(pcount_norm, fill=type)) + geom_histogram(bins=40) +
  theme_classic() + scale_fill_manual(values=c('#4285f4', '#dddddd'))
```



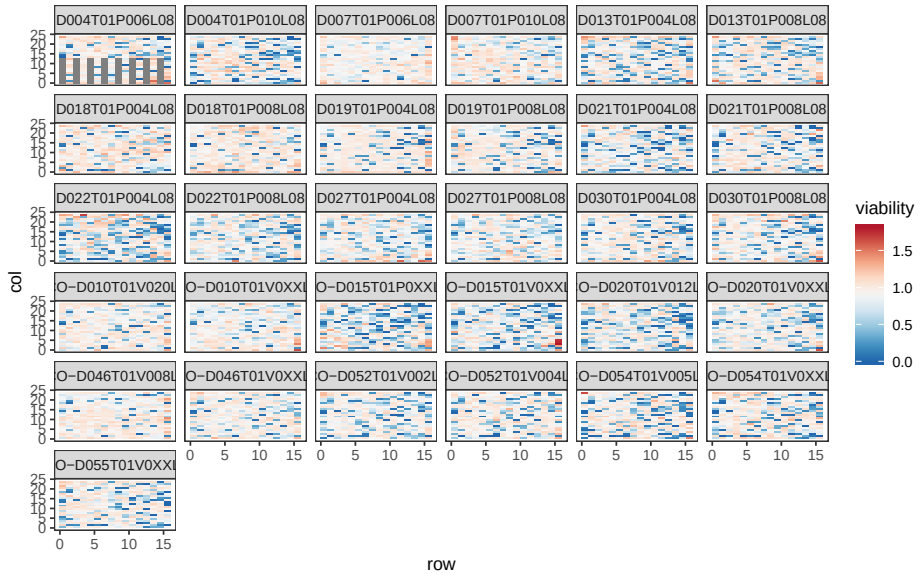
We generate a plot that shows both positive and negative controls. We show the median for each set of controls with error bars representing two standard errors.

```
ctg_norm %>%
  mutate(type = ifelse(drug == 'DMSO', 'pos',
                ifelse((drug == 'Bortezomib') &
                       (rack.concentration %in% c('0.04', '1')), 'neg', 'other'))) %>%
  filter(type != 'other', !is.na(viability)) %>%
  group_by(plate, type) %>%
  summarise(med_val = median(viability),
            se = mad(viability)/sqrt(n()),
            ymin = med_val - 2*se,
            ymax = med_val + 2*se) %>% ungroup() %>%
  ggplot(aes(plate, med_val, colour=type)) + geom_point() +
  geom_errorbar(aes(ymin = ymin, ymax = ymax), width=0) +
  theme_classic() +
  theme(axis.text.x = element_blank()) +
  ylab('Viability') +
  scale_colour_manual(values=c('#4285f4', '#111111')) +
  xlab('Plate')
```



We next plot the determined viability phenotypes for each plate in its original layout to examine whether spatial effects are gone and whether replicates of the same line expose similar response patterns.

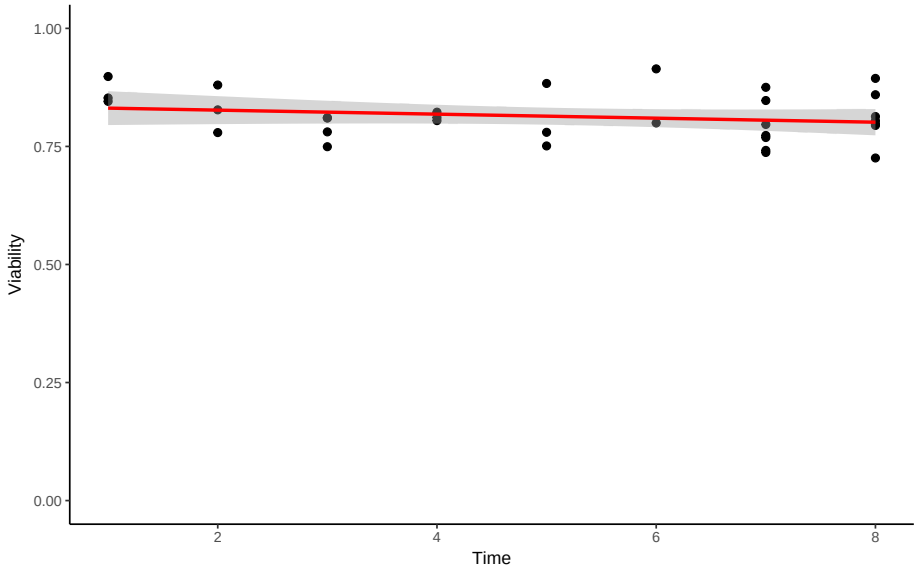
```
ctg_norm %>% ggplot(aes(row, col, fill=viability)) +
  geom_raster(hjust=0, vjust=0) +
  facet_wrap(~plate) + theme_bw() +
  theme(panel.grid=element_blank()) +
  scale_fill_gradientn(colors=colpal)
```



This looks reasonable. We next investigate if we can observe strong bias across batches (defined by screening date).

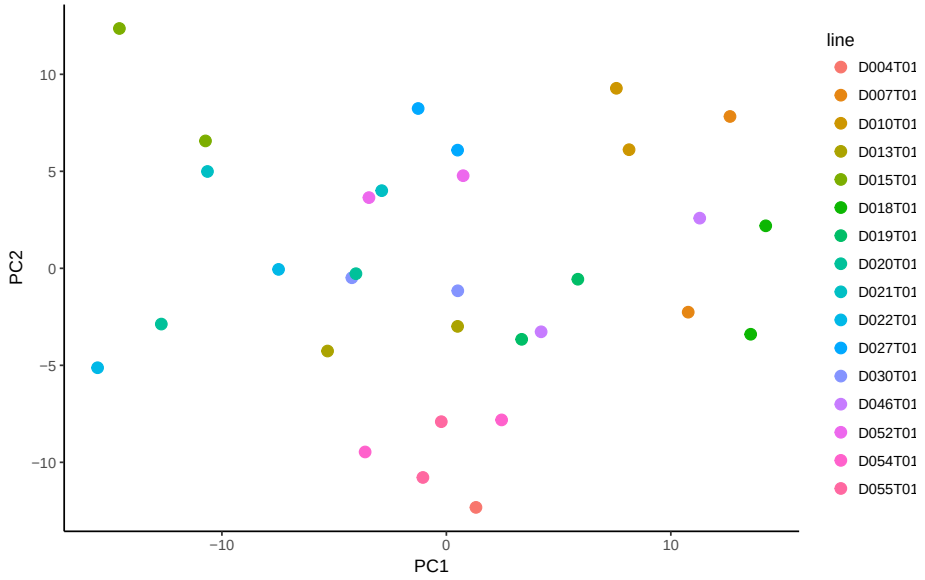
```
## we don't plot the acutal dates but a pseudo time instead
pseudo_time <- ctg_norm %>% distinct(date) %>%
```

```
  arrange(date) %>% mutate(pseudo_time = 1:n())

## only mean
ctg_norm %>% inner_join(pseudo_time) %>%
  group_by(line, plate, pseudo_time) %>%
  summarise(viability=mean(viability, na.rm=T)) %>% ungroup() %>%
  ggplot(aes(pseudo_time, viability)) +
  geom_point(size=2) +
  geom_smooth(method='lm', colour='red') +
  theme_classic() + ylim(c(0,1)) +
  xlab('Time') + ylab('Viability')
```



A principle component analysis (PCA) shows clustering of drug phenotypes.

```
pca_res <- ctg_norm %>%
  acast(well ~ screen, value.var='viability', fun.aggregate = mean) %>%
  .[,-1] %>% t() %>% .[,apply(., 2, function(x) sum(x) > 0)] %>%
  prcomp(scale=T, center=T)

## visualize
ctg_norm %>% distinct(screen, line, batch) %>%
  inner_join(pca_res$x %>% as.data.frame() %>%
               rownames_to_column('screen') %>% tbl_df %>%
               dplyr::select(screen, PC1, PC2) %>%
               mutate(screen = gsub('^X', '', gsub('\\.', '-', screen)))) %>%
  ggplot(aes(PC1, PC2)) +
  geom_point(aes(colour=line), size=3) +
  theme_classic()
```

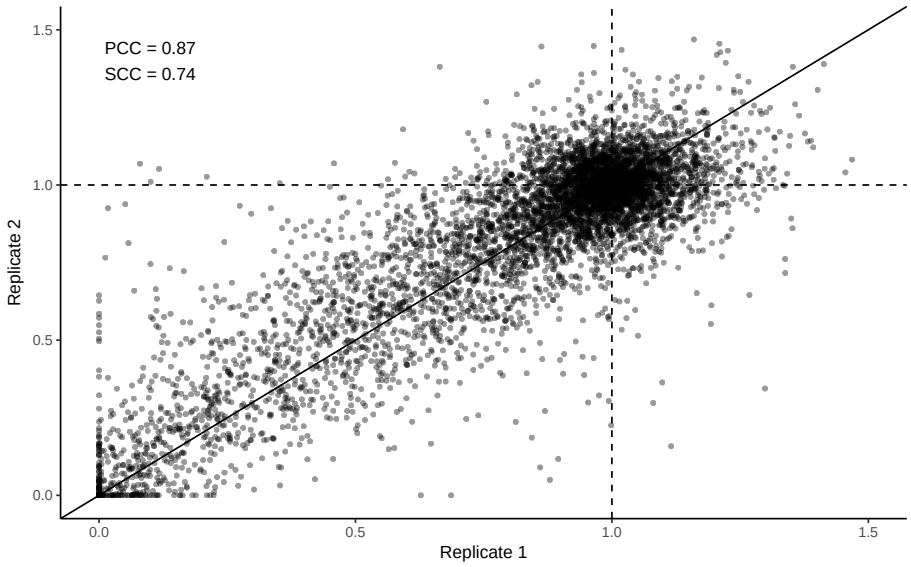


2.5 Reproducibility

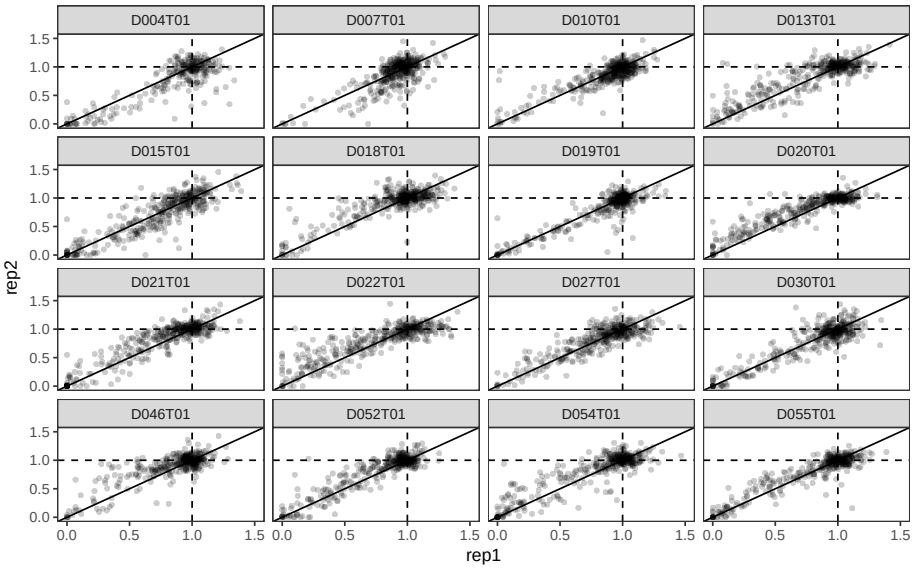
We show scatter plots that illustrate reproducibility of phenotypes across replicates.

```
## replicate annotation
reps <- ctg_norm %>%
  distinct(screen, line) %>%
  arrange(screen, line) %>%
  group_by(line) %>% mutate(rep=paste0('rep', 1:n())) %>% ungroup()

ctg_norm <- ctg_norm %>% inner_join(reps)

cor_scatter <- ctg_norm %>% distinct() %>%
  unite(barcode, line, well, sep='_') %>%
  dplyr::select(barcode, rep, viability) %>%
  spread(rep, viability) %>% drop_na()

cor_scatter %>% summarise(PCC = cor(rep1, rep2, method='pearson'),
                          SCC = cor(rep1, rep2, method='spearman'))

cor_scatter %>%
  ggplot(aes(rep1, rep2)) + geom_point(alpha=0.4, stroke=0) +
  geom_hline(yintercept = 1, linetype=2) +
  geom_vline(xintercept = 1, linetype=2) +
  geom_abline(slope = 1) + xlim(c(0,1.5)) + ylim(c(0,1.5)) +
  annotate('text', x=0.1, y=1.4, label='PCC = 0.87\nSCC = 0.74') +
  theme_classic() + xlab('Replicate 1') + ylab('Replicate 2')
```

Analysis of CTG screens with the clinical drug panel

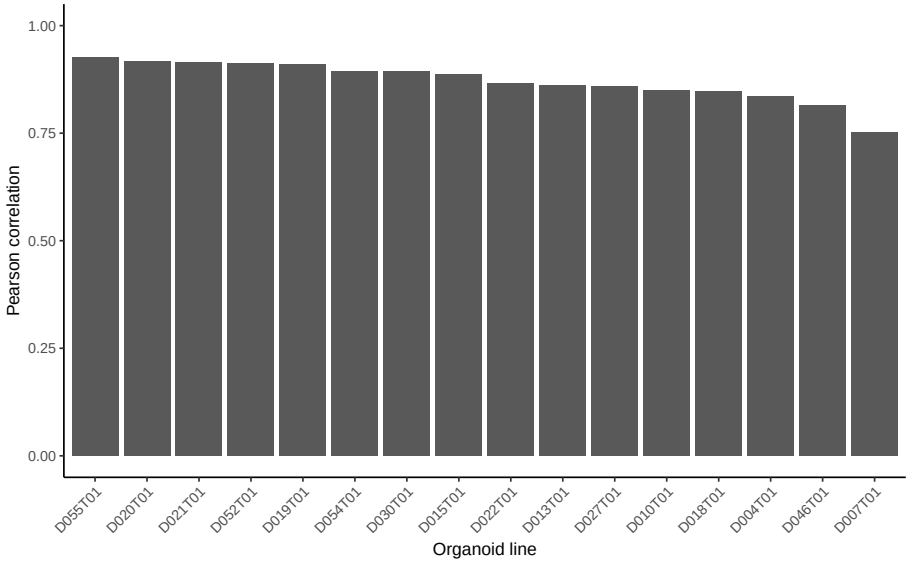



```
## for individual oragnoid lines
ctg_norm %>%
  unite(barcode, line, well, sep='_', remove=F) %>%
  dplyr::select(line, barcode, rep, viability) %>%
  spread(rep, viability) %>% drop_na() %>%
  ggplot(aes(rep1, rep2)) + geom_point(alpha=0.2, stroke=0) +
  geom_hline(yintercept = 1, linetype=2) +
  geom_vline(xintercept = 1, linetype=2) +
  geom_abline(slope = 1) + xlim(c(0,1.5)) + ylim(c(0,1.5)) +
  facet_wrap(~line) + theme_bw() + theme(panel.grid=element_blank())
```




```
## correlation coefficients
ctg_norm %>%
  unite(barcode, line, well, sep='_', remove=F) %>%
```

```
  dplyr::select(line, barcode, rep, viability) %>%
  spread(rep, viability) %>% drop_na() %>%
  group_by(line) %>% summarise(cor = cor(rep1, rep2)) %>% ungroup() %>%
  arrange(desc(cor)) %>% mutate(line = factor(line, levels=line)) %>%
  ggplot(aes(line, cor)) + geom_bar(stat='identity') +
  theme_classic() + ylim(c(0,1)) +
  ylab('Pearson correlation') + xlab('Organoid line') +
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```



3 Calculation of AUC values for drug dose response curves

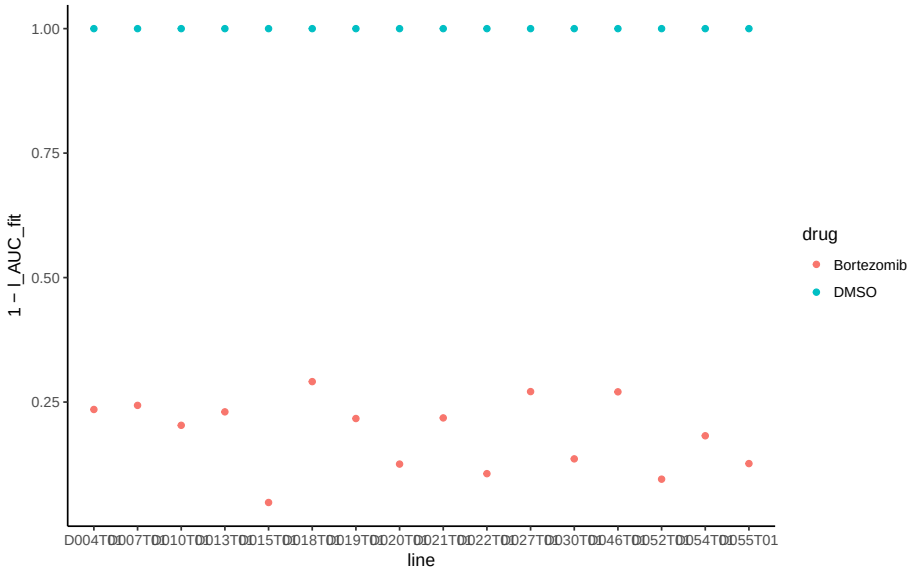
We compute AUC values using the PharmacoGX package.

```
auc_pgx <- ctg_norm %>%
  mutate(pc_norm = viability) %>%
  #filter(drug == coi) %>%
  group_by(screen, drug) %>% arrange(rack.concentration) %>%
  ## calculate auc both for fitted curve and actual values
  mutate(s_AUC_fit = computeAUC(rack.concentration, pc_norm, verbose=T,
                                viability_as_pct = F, area.type = 'Fitted'),
         s_AUC_actual = computeAUC(rack.concentration, pc_norm, verbose=T,
                                   viability_as_pct = F, area.type='Actual')) %>%
  group_by(line, drug) %>% arrange(rack.concentration) %>%
  ## calculate auc both for fitted curve and actual values
  mutate(l_AUC_fit = computeAUC(rack.concentration, pc_norm, verbose=F,
                                viability_as_pct = F, area.type = 'Fitted'),
         l_AUC_actual = computeAUC(rack.concentration, pc_norm, verbose=F,
                                   viability_as_pct = F, area.type='Actual')) %>%
  ungroup()
```

3.1 Quality control

We observe the AUCs of DMSO (positive control) and Bortezomib (negative control) across organoid lines.

```
## based on AUCs
auc_pgx %>% filter(drug %in% c('DMSO', 'Bortezomib')) %>%
  distinct(line, drug, l_AUC_fit) %>%
  ggplot(aes(line, 1-l_AUC_fit, colour=drug)) + geom_point() + theme_classic()
```



This seems to make sense. We want to further have a function that can plot a simple dose-response curve for one drug in one or more lines.

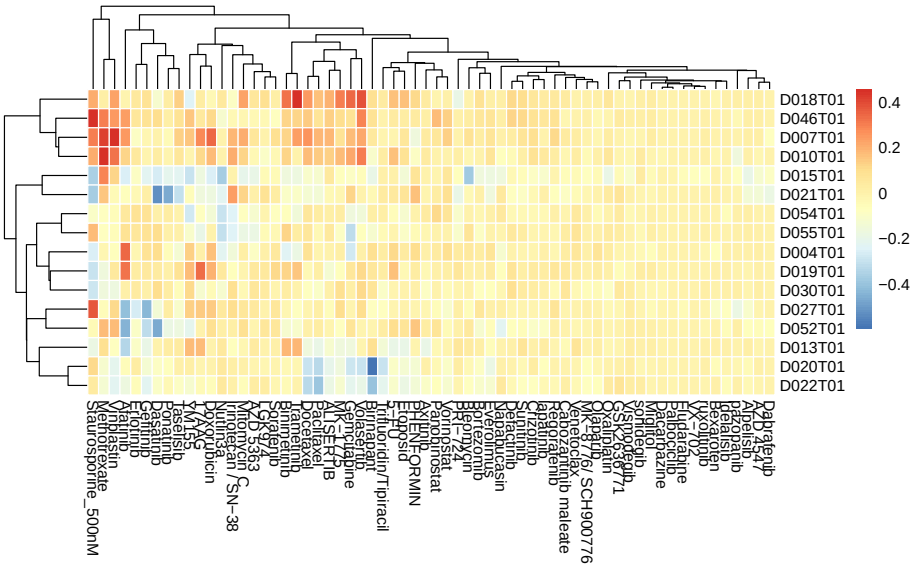
```
plot_drc <- function(dr, lines){
  ctg_norm %>% mutate(concentration = factor(rack.concentration,
                                             levels=c('0.0016', '0.008', '0.04', '0.2', '1'))) %>%
    filter(drug == dr, line %in% lines) %>%
    ggplot(aes(concentration, viability)) +
    geom_point() +
    stat_summary(aes(group=line), fun.y=mean,
                 geom="line", colour="#4285f4") +
    facet_wrap(~line) + ylim(c(0, 1.3)) +
    theme_bw() + theme(panel.grid=element_blank())
}
```

4 A heatmap of drug response at the organoid line level

4.1 Create a heat map using CTG AUCs

We center AUCs by subtracting the median. We use AUCs based on fitted curves where both replicates go into fitting each point on the DRC.

```
auc_pgx %>% distinct(drug, line, l_AUC_fit) %>%
  filter(!grepl('with|mit|DMSO', drug)) %>%
  group_by(drug) %>%
  mutate(l_AUC_fit = 1 - l_AUC_fit,
         AUC_centered = l_AUC_fit - median(l_AUC_fit)) %>%
  ungroup() %>%
  acast(line ~ drug, value.var = 'AUC_centered') %>%
  pheatmap(border_color = '#ffffff')
```



4.2 Create a heat map using LDC AUCs

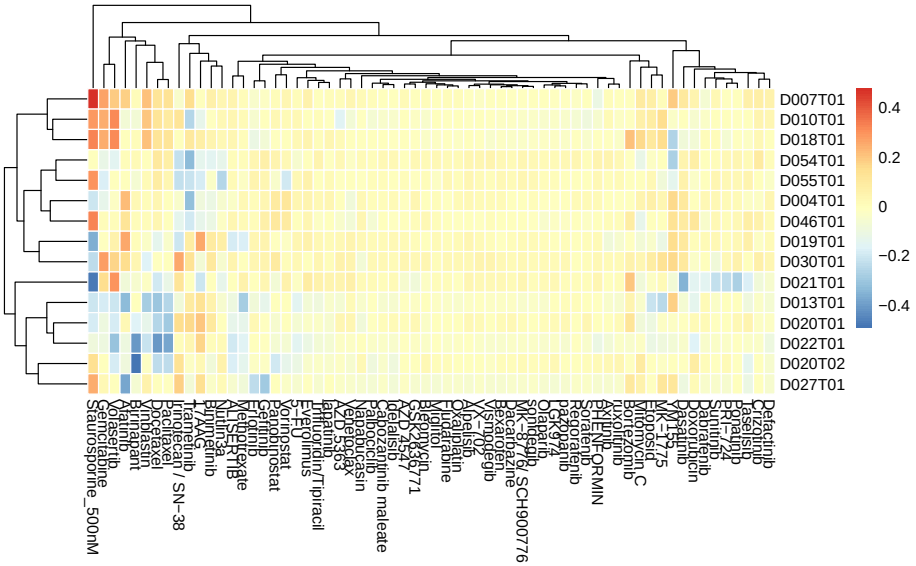
Similar to CTG analysis, we performed an analysis of viability using a life-death classifier based on the organoid images. We want to create a heatmap that shows response to drug treatment based on these results (similar to above).

We first load the LDC data from Rdata file.

```
data('aucs_image', package='SCOPEAnalysis')
```

We center each drug by subtracting the median using the fitted AUCs at the line level. We plot the results as a heat map.

```
aucs_image %>% distinct(Line, Product.Name, l_AUC_fit) %>% tbl_df %>%
  distinct() %>%
  filter(!grepl('mit|with', Product.Name)) %>%
  group_by(Product.Name) %>%
  mutate(AUC_centered = l_AUC_fit - median(l_AUC_fit)) %>%
  # ungroup() %>%
  acast(Line ~ Product.Name, value.var = 'AUC_centered') %>%
  pheatmap(border_color = '#ffffff')
```



5 Modelling drug-gene interactions

We want to know if we can identify known drug gene interactions using the AUC data and molecular profiles based on amplicon sequencing. Hence we first load the amplicon sequencing data.

```
## load mutations
data('mutations', package='SCOPEAnalysis')

## binary mutation profile
mut_profile <- mutations %>% dplyr::select(line, SYMBOL) %>%
  mutate(mut = as.factor(1))  %>% distinct() %>%
  spread(SYMBOL, mut) %>% mutate_all(funs(ifelse(is.na(.), 0, .))) %>%
  rbind(c('D018T01', rep(0, 19)))
```

5.1 Pariwise interactions between drug response and mutations

We can scan all possible combinations of drugs and mutations to scan for pharmacogenomic interactions. Here we use a permutation test to identify genotype based differences in CTG phenotypes.

```
## annotate mutation status
drugmut <- distinct(auc_pgx, line, drug, l_AUC_fit) %>%
  left_join(mut_profile) %>% mutate_at(vars(AKT1:TP53), 'as.factor') %>%
  mutate(AUC = 1-l_AUC_fit)

## add clinical data that might influence the results
data('clindat', package='SCOPEAnalysis')

drugmut <- drugmut %>% inner_join(clindat)
```

Now we form all pairwise combinations of drugs and mutations and perform the linear modelling and LRT testing.

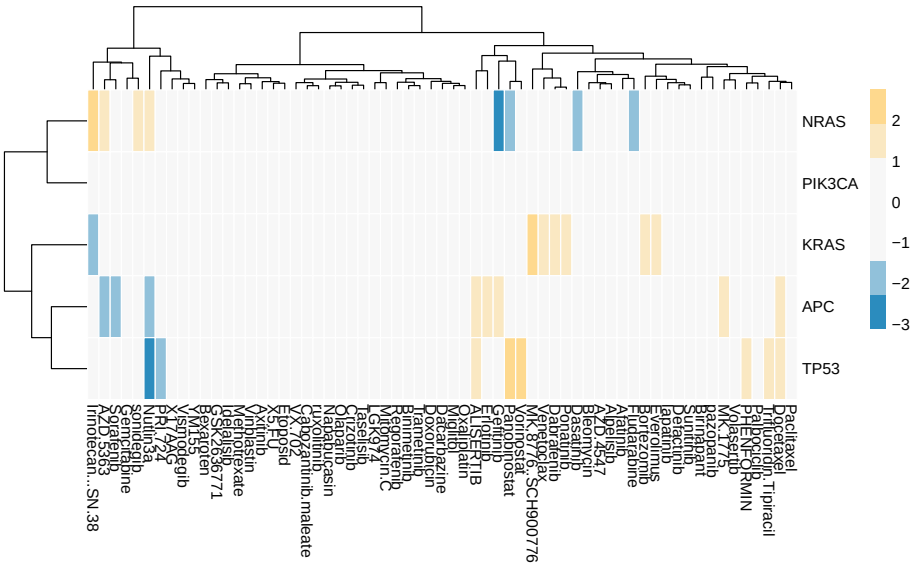
```
## form combinations
dm_combis <- expand.grid(
    unique(drugmut$drug[!drugmut$drug %in% c('Staurosporine_500nM', 'DMSO')]),
    colnames(mut_profile)[-1]
) %>% tbl_df %>% `colnames<-`(c('drug', 'genotype'))
dm_combis <- dm_combis %>% filter(!grepl('with|mit', drug))

## only genes that are mutated at least twice
dm_combis <- dm_combis %>%
  left_join(mut_profile %>% dplyr::select(-line) %>%
              summarise_all(funs(sum(as.integer(.)))) %>%
              gather(genotype, mut_count)) %>%
  filter(mut_count > 2) %>% dplyr::select(-mut_count)

## function for modelling/testing
test_druggene <- function(df){
  d <- as.character(df$drug[1])
  m <- as.character(df$genotype[1])

  ## get the right data
  data <- drugmut %>% filter(drug == d) %>%
    dplyr::select(line, AUC, matches(paste0('^', m, '$')), everything())
  colnames(data)[3] <- 'mut'


  if(sum(as.numeric(as.character(data$mut))) == 0){
    return(NA)
  } else {
    ## models
    permts <- permTS(AUC ~ as.factor(mut), data = data,
                     method="exact.mc",
                     control=permControl(nmc=10^4-1))

    return(tibble(
      drug = d,
      mutation = m,
      p.value = permts$p.value,
      dAUC = permts$estimate,
```

```
      variance = var(data$AUC)
    ))
  }
}

dgi <- dm_combis %>% rowwise() %>% do(results=test_druggene(.)) %>%
  unnest(results) %>% mutate(padj = p.adjust(p.value, method='BH')) %>%
  arrange(p.value)
```

5.1.1 Visualization of the results

We draw a D x M heat map where D are the drugs and M are mutated genes. The heat map will visualize the drug-genotype interactions by their p-value. Negative interactions will be shown in blue, positive interactions in yellow.

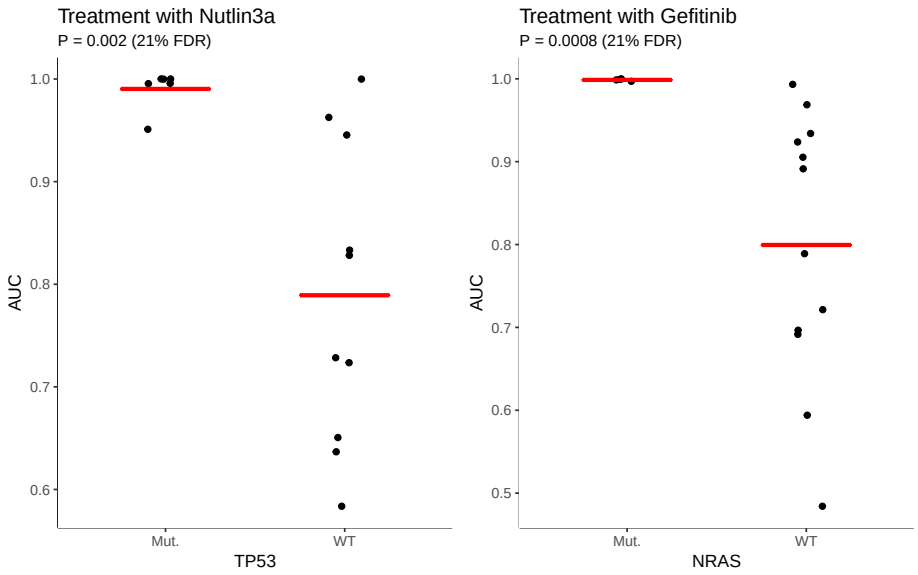
```
## blue/yellow colour-pallette
dgi_palette <- colorRampPalette(c('#2b8cbe', '#f7f7f7', '#f7f7f7', '#fed98e'))(7)

dgi %>% mutate(visval = ifelse(dAUC < 0, (-1) * (-log10(p.value)), -log10(p.value))) %>%
  dplyr::select(drug, mutation, visval) %>%
  spread(drug, visval) %>% data.frame() %>% `rownames<-`(NULL) %>%
  column_to_rownames('mutation') %>%
  pheatmap(color=dgi_palette, border_color = '#ffffff')
```



I draw a box plot of the top combinations TP53-Nutlin3a and NRAS-Gefitinib.

```
int_nutlin <- drugmut %>% filter(drug == 'Nutlin3a') %>%
  mutate(TP53 = ifelse(TP53 == '0', 'WT', 'Mut.')) %>%
  ggplot(aes(TP53, AUC)) + geom_jitter(width=0.1) +
  stat_summary(fun.y='mean', fun.ymax = 'mean',
               fun.ymin = 'mean', geom='crossbar', colour='red',
               width=0.5) +
```

```
  theme_classic() +
  labs(title='Treatment with Nutlin3a',
       subtitle = 'P = 0.002 (21% FDR)')

int_gefitinib <- drugmut %>% filter(drug == 'Gefitinib') %>%
  mutate(NRAS = ifelse(NRAS == '0', 'WT', 'Mut.')) %>%
  ggplot(aes(NRAS, AUC)) + geom_jitter(width=0.1) +
  stat_summary(fun.y='mean', fun.ymax = 'mean',
               fun.ymin = 'mean', geom='crossbar', colour='red',
               width=0.5) +
  theme_classic() +
  labs(title='Treatment with Gefitinib',
       subtitle = 'P = 0.0008 (21% FDR)')

int_nutlin + int_gefitinib
```



6 Correlations between drug response and gene expression

We first prepare the AUC drug response data by centering the AUCS of each drug.

```
aucs <- auc_pgx %>% group_by(drug, line, rep) %>%
  summarise(AUC = 1-mean(l_AUC_fit)) %>% ungroup() %>%
  group_by(drug) %>% mutate(AUC_centered = AUC - median(AUC)) %>% ungroup()
```

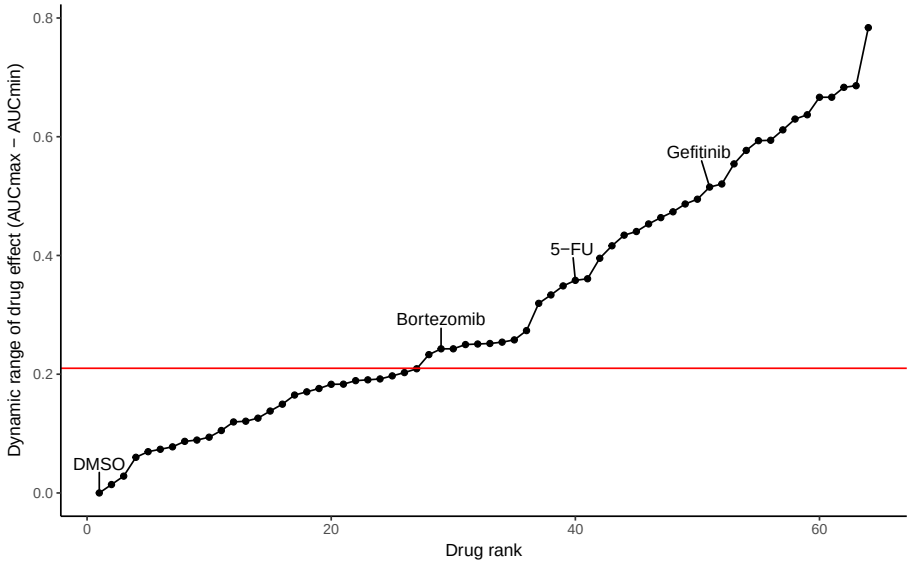
Next we load the expression data. These data re processed and normalized as described in the manuscript.

```
data('promise_expr', package='SCOPEAnalysis')
```

For each drug we want to now know if the response can be predicted by gene expression.

First I want to exclude drugs with a small dynamic phenotype range as interactions identified for these drugs are likely not biologically meaningful.

```
## dynamic phenotype range for each drug
drugs_dr <- aucs %>% group_by(drug, line) %>%
  filter(!grepl('with|mit|Staurosporine_500nM', drug)) %>%
  summarise(AUC = mean(AUC)) %>% ungroup() %>%
  group_by(drug) %>%
  mutate(dynamic_range = range(AUC) %>% reduce(`-`) %>% abs()) %>%
  ungroup() %>% distinct(drug, dynamic_range)

## visualize
drugs_dr %>% arrange(dynamic_range) %>%
  mutate(rank = 1:n(),
         label = ifelse(drug %in% c('DMSO', 'Gefitinib', '5-FU', 'Bortezomib'), drug, '')) %>%
  ggplot(aes(rank, dynamic_range)) +
  geom_point() + geom_line() +
  geom_text_repel(aes(label=label), nudge_y = 0.05) +
  geom_hline(yintercept = 0.21, colour = 'red') +
  theme_classic() +
  ylab('Dynamic range of drug effect (AUCmax - AUCmin)') +
  xlab('Drug rank')
```




```
## select drugs to include in analysis
drugs_include <- drugs_dr %>% filter(dynamic_range > 0.21) %>% .$drug
```

We next perform tests using limma's moderated t-test.

```
## expr matrix
expr_mat <- promise_expr %>%
  reshape2::acast(probe ~ id, value.var = 'expr', fun.aggregate =mean)
```

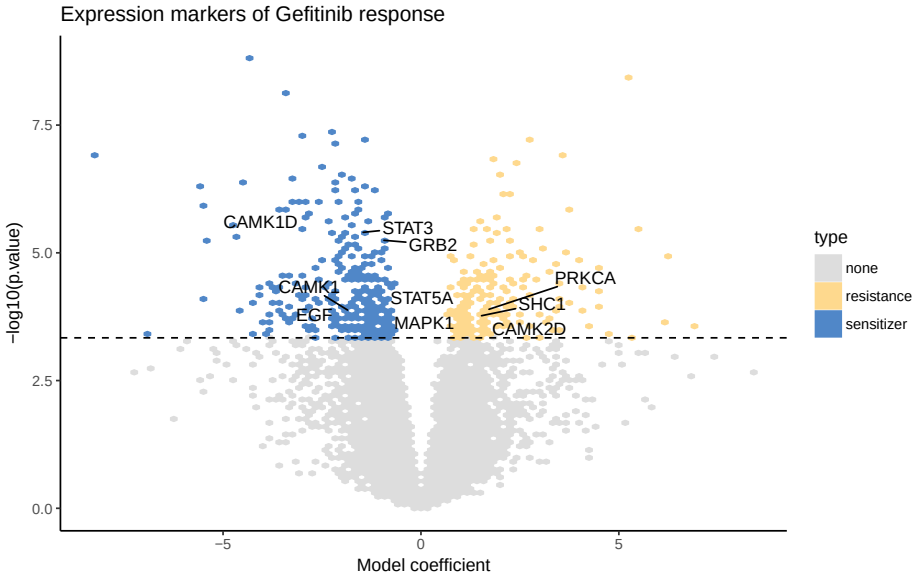
```
## drug response
pb <- progress_estimated(length(drugs_include))
expr_interactions <- aucs %>% filter(drug %in% drugs_include) %>%
  group_by(drug, line) %>%
  summarise(AUC = mean(AUC)) %>% ungroup() %>%
  arrange(line) %>% nest(-drug) %>%
  mutate(res = map(data, ~{
    pb$tick()$print()
    if(!identical(.x$line, colnames(expr_mat))){
      mm <- model.matrix(~AUC, data = bind_rows(.x, .x) %>% arrange(line))
      tidy(eBayes(lmFit(expr_mat, mm), robust=T))
    } else {
      stop('Sample frame does not fit to expression matrix')
    }
  })) %>% unnest(res) %>% arrange(p.value) %>%
  dplyr::select(probe=gene, everything()) %>%
  inner_join(promise_expr %>% distinct(probe, symbol)) %>%
  mutate(fdr=p.adjust(p.value, method='BH'))
```

6.1 Volcano plots of MDM2 and EGFR inhibitors

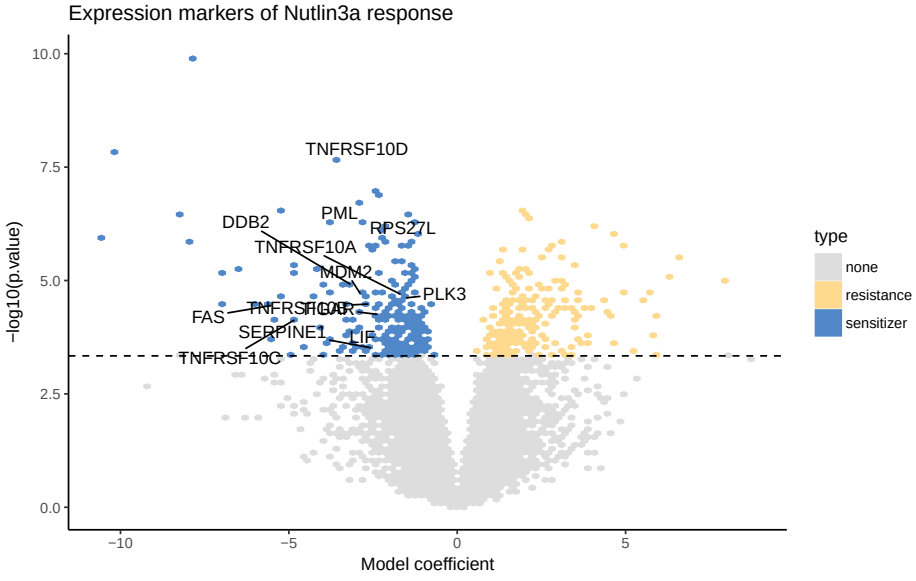
We visualize the results for two selected drugs - Gefitinib (EGFR inhibitor) and Nutlin-3a (MDM2 inhibitor). We highlight genes previously reported to be involved in EGFR/MDM2 inhibtion pathways.

```
drug_expr_volcano <- function(d, highlight){
  df <- expr_interactions %>% filter(drug == d) %>%
    group_by(drug, symbol) %>% top_n(1, desc(p.value)) %>%
    ungroup()

  ## fdr 0.05
  co <- df %>% filter(fdr < 0.05) %>% arrange(desc(fdr)) %>%
    .$p.value %>% .[1] %>% log10() %>% `*`(-1)

  df %>% mutate(label = ifelse(symbol %in% highlight, symbol, ''),
         type = ifelse((estimate < 0) & (fdr < 0.05), 'sensitizer',
                ifelse((estimate > 0) & (fdr < 0.05), 'resistance', 'none'))) %>%
  ggplot(aes(estimate, -log10(p.value))) +
  geom_hex(aes(fill = type, colour=type), bins=100) + theme_classic() +
  scale_fill_manual(values = c('#dddddd', '#FED98E', '#5087C8')) +
  scale_colour_manual(values = c('#dddddd', '#FED98E', '#5087C8')) +
  geom_text_repel(aes(label = label)) +
  geom_hline(yintercept = co, linetype = 'dashed') +
  ggtitle(paste('Expression markers of', d, 'response')) +
  xlab('Model coefficient')
}

## EGFR Inhibitor Pathway; q-value = 0.002; pathway source = PharmGKB; tool = ConsensusPathDB
gefitinib_hits <- c('EGF', 'CAMK2D', 'CAMK1D', 'SHC1', 'STAT3',
```

```
                    'PRKCA', 'GRB2', 'CAMK1', 'MAPK1', 'STAT5A')
drug_expr_volcano('Gefitinib', gefitinib_hits)
```




```
# TP53 Regulates Transcription of Death Receptors and Ligands;
## pathway source = Reactome; q-value = 0.004; tool = ConsensusPathDB
nutlin_hits <- c('LIF', 'PML', 'FAS', 'RPS27L', 'DDB2', 'SERPINE1',
                 'TIGAR', 'TNFRSF10A', 'TNFRSF10B', 'TNFRSF10C',
                 'TNFRSF10D', 'PLK3', 'MDM2')
drug_expr_volcano('Nutlin3a', nutlin_hits)
```



7 Drug-genotype interactions in feature space

Some drugs do not influence organoid viability but they induce other morphological phenotypes. These morpholoigcal phenotypes are described by a set of features that are calculated from the organoid images. These features can be summarized by an area under the precision-recall curve metric that describes for each drug whether it induces a phenotypic response that makes the treated organoids look 'significantly' different to the untreated ones. We now would like to ask whether there are mutations that influence this behaviour. For example, we could ask: does Nutlin3a treatement induce different morpholigical phenotypes depending on the TP53 mutation status of the organoid line?

First we load pre-computed area under the curve values from a file.

```
data('aucroc', package = 'SCOPEAnalysis')
```

Now we pair these drug responses with genes mutated at least 3 times and check for significant differences using a permutation test.

```
drug_morph_interaction <- mut_profile %>%
  gather(mutation, status, -line) %>%
  group_by(mutation) %>% filter(sum(as.integer(status)) >=3) %>%
  nest(-mutation) %>%
  mutate(tres = map(data, ~{
    .x %>% inner_join(aucroc) %>% nest(-drug) %>%
      mutate(res = map(data,
                       ~ broom::tidy(t.test(auroc ~ status, data = .x)))) %>%
      unnest(res)
  })) %>% unnest(tres)

drug_morph_interaction <- drug_morph_interaction %>%
  arrange(p.value) %>% mutate(FDR = p.adjust(p.value, method='BH'))
```

We extend the network of drugs connected by SVM hyperplane angles with the drug-mutation interactions predicted above excluding mutations of the NRAS oncogene.

```
## load feature angles which are the edges of the network
data('feature_angles', package='SCOPEAnalysis')

## add drug-genotype interactions
feature_angles <- drug_morph_interaction %>%
  dplyr::select(source=mutation, target=drug, p.value) %>%
  bind_rows(feature_angles)

## select drugs with at least one interaction
good_drugs <- feature_angles %>% filter((angle < 45) | (p.value < 0.05)) %>%
  distinct(source, target)
good_drugs <- unique(c(good_drugs$source, good_drugs$target))

## select mutations/drugs to be labeled
for_labeling <- c('APC', 'PIK3CA', 'KRAS', 'TP53',
                  'Tyrphostin AG 879', 'K02288',
                  'WIKI4', 'AT7867')
```

```
## make a tidy graph
graph <- feature_angles %>%
  filter((angle < 45) | (p.value < 0.05),
         source != target,
         source != 'NRAS') %>%
  as_tbl_graph() %>%
  activate(nodes) %>%
  mutate(type = ifelse(name %in% drug_morph_interaction$mutation, 'mutation', 'drug')) %>%
  activate(nodes) %>%
  filter(name %in% good_drugs) %>%
  mutate(label = ifelse(name %in% for_labeling, name, ''),
         index = 1:n(),
         has_edge = ifelse((index %in% .E()$from) |
                           (index %in% .E()$to), T, F),
         size = ifelse(type == 'mutation', 2, 1))

## draw the graph
graph %>% ggraph(layout = 'fr') +
  geom_edge_fan(colour = '#dddddd') +
  geom_node_point(aes(shape = type, colour = type, size = size)) +
  geom_node_text(aes(label = label), nudge_y = 0.2) +
  theme_graph() +
  scale_colour_manual(values = c('#aaaaaa', '#4285f4'))
```

8 Session info

```
sessionInfo()
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] hexbin_1.27.2      bindrcpp_0.2.2     forcats_0.3.0
##  [4] stringr_1.3.0      dplyr_0.7.4        purrr_0.2.4
##  [7] readr_1.1.1        tidyr_0.8.0        tibble_1.4.2
## [10] tidyverse_1.2.1    biobroom_1.10.1    broom_0.4.4
## [13] ggrepel_0.7.0      tidygraph_1.1.0    ggraph_1.0.1
```

```
## [16] ggplot2_2.2.1.9000 patchwork_0.0.1    reshape2_1.4.3
## [19] PharmacoGx_1.8.3   pheatmap_1.0.8     limma_3.34.9
## [22] perm_1.0-0.0       BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##  [1] fgsea_1.4.1         colorspace_1.3-2    rprojroot_1.3-2
##  [4] lsa_0.73.1          rstudioapi_0.7      SnowballC_0.5.1
##  [7] lubridate_1.7.4     xml2_1.2.0          mnormt_1.5-5
## [10] knitr_1.20          jsonlite_1.5        magicaxis_2.0.3
## [13] cluster_2.0.7-1     ggforce_0.1.1       mapproj_1.2.6
## [16] compiler_3.4.1      httr_1.3.1          backports_1.1.2
## [19] assertthat_0.2.0    lazyeval_0.2.1      cli_1.0.0
## [22] tweenr_0.1.5        htmltools_0.3.6     tools_3.4.1
## [25] igraph_1.2.1        gtable_0.2.0        glue_1.2.0
## [28] RANN_2.5.1          maps_3.3.0          fastmatch_1.1-0
## [31] Rcpp_0.12.16        slam_0.1-42         Biobase_2.38.0
## [34] cellranger_1.1.0    gdata_2.18.0        nlme_3.1-137
## [37] udunits2_0.13       psych_1.8.4         xfun_0.1
## [40] rvest_0.3.2         gtools_3.5.0        statmod_1.4.30
## [43] MASS_7.3-50         scales_0.5.0.9000   hms_0.4.2
## [46] relations_0.6-8     parallel_3.4.1      RColorBrewer_1.1-2
## [49] sets_1.0-18         yaml_2.1.19         gridExtra_2.3
## [52] downloader_0.4      stringi_1.2.2       NISTunits_1.0.1
## [55] plotrix_3.7         caTools_1.17.1      BiocGenerics_0.24.0
## [58] BiocParallel_1.12.0 rlang_0.2.0.9001    pkgconfig_2.0.1
## [61] bitops_1.0-6        pracma_2.1.4        evaluate_0.10.1
## [64] lattice_0.20-35     bindr_0.1.1         labeling_0.3
## [67] tidyselect_0.2.4    plyr_1.8.4          magrittr_1.5
## [70] bookdown_0.7        R6_2.2.2            gplots_3.0.1
## [73] sm_2.2-5.5          pillar_1.2.2        haven_1.1.1
## [76] foreign_0.8-70      withr_2.1.2         units_0.5-1
## [79] modelr_0.1.1        crayon_1.3.4        KernSmooth_2.23-15
## [82] utf8_1.1.3          rmarkdown_1.9       viridis_0.5.1
## [85] grid_3.4.1          readxl_1.1.0        data.table_1.10.4-3
## [88] marray_1.56.0       piano_1.18.1        digest_0.6.15
## [91] munsell_0.4.3       celestial_1.4.1     viridisLite_0.3.0
## [94] tcltk_3.4.1
```