Multi-omics integration identifies a selective vulnerability of colorectal cancer subtypes to YM155

Benedikt Rauscher

2021-02-25

Abstract

Tumor heterogeneity is a major challenge to the treatment of colorectal cancer (CRC). Recently, a transcriptome-based classification was developed, segregating CRC into four consensus molecular subtypes (CMS) with distinct biological and clinical characteristics. Here, we applied the CMS classification on CRC cell lines to identify novel subtype-specific drug vulnerabilities. We combined publicly available transcriptome data from multiple resources to assign 159 CRC cell lines to CMS. By integrating results from large scale drug screens, we discovered that CMS1 cancer is highly vulnerable to the survivin suppressor YM155. We confirmed our results using an independent panel of CRC cell lines and demonstrate a 100-fold higher sensitivity of CMS1 lines. This vulnerability was specific to YM155 and not observed for commonly used chemotherapeutic agents. In CMS1 cancer, low concentrations of YM155 induced apoptosis and expression signatures associated with NFkappaB and ER stress mediated apoptosis signaling. Using a genome-wide CRISPR/Cas9 screen, we further discovered a novel role of genes involved in LDL-receptor recycling as modulators of YM155 response in CMS1 CRC. Our work shows that combining drug response data with CMS classification in cell lines can reveal specific vulnerabilities and propose YM155 as novel CMS1 specific drug.

Contents

1 About

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

2 Dependencies

We load a number of packages whose functions are needed throughout the analysis.

```
library(tidyverse)
library(affy)
library(GEOquery)
library(preprocessCore)
library(pheatmap)
library(openxlsx)
library(reshape2)
library(ggsignif)
library(CMSclassifier)
library(CMScaller)
library(ggrepel)
library(perm)
library(patchwork)
library(readxl)
library(lumi)
library(limma)
library(fgsea)
library(GO.db)
library(Organism.dplyr)
library(edgeR)
library(patchwork)
```

3 Meta data

In the process of the analysis we will require meta data such as, for example, gene ID maps. In the following we load these meta data, which are provided within this package. We load a map that links gene IDs to microarray probe IDs for a number of different microarray platform used throughout the analysis. We further load a list of genes based on which classification of molecular subtypes is performed.

```
## gene probe ID map
data('probe_map', package='CMSYM1552018')
## reference genes for CMS classification
data('entrez_ref', package='CMSYM1552018')
## list of protein coding genes
data('pc', package='CMSYM1552018')
```

4 Preprocessing microarray data

In the next steps we load and process microarray data from different colorectal cancer cell lines for subtype classification.

4.1 Adai cell line

Adai cell line is data set featured in Oncomine (Rhodes et al., n.d.). Expression profiels for several colorectal cancer cell lines were generated using an Affymetrix HG-U133 Plus2 chip. We read a matrix that contains normalized expression values after background correction using RMA (Irizarry et al. 2003), cross-chip quantile normalization and expression level summary using medianpolish (all performed using this affy package (Gautier et al. 2004)).

```
data('adai_exprs', package='CMSYM1552018')
```

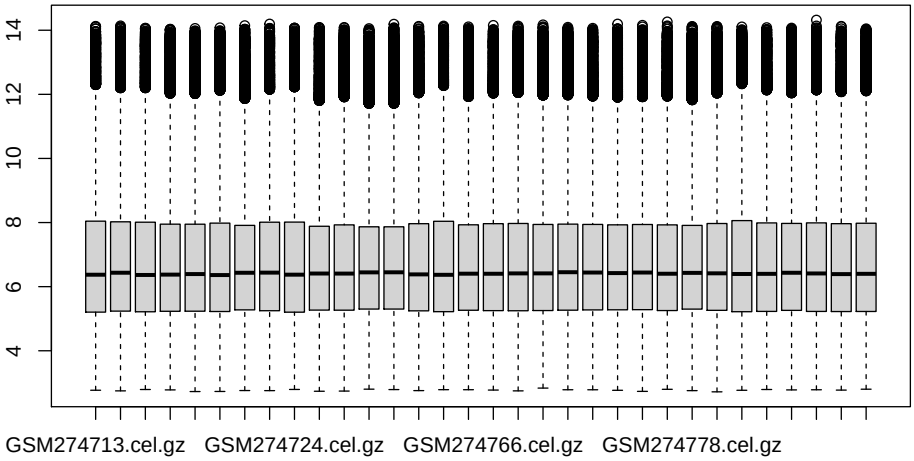


Figure 1: **Normalized expression levels across Adai cell line samples**

For classification of subtypes we need gene level expression values. We select for each gene the probe with the highest median expression value across samples hypothesizing that this probe corresponds to the 'main' transcript of the gene. We first generate a function for this that can then be applied to other data sets as well.

```
get_cms_gene_lvl2 <- function(eset, chip, filter_ref = T){
  ## make probe mapping
  gpl <- probe_map %>% dplyr::select(matches(chip), entrez) %>%
    drop_na() %>% distinct() %>%
    group_by_at(chip) %>% dplyr::slice(1) %>% ungroup() %>%
    as.data.frame() %>% column_to_rownames(chip)

  ## aggregate per entrez
  eset_aggr <- probesToEntrez(eset, gpl, entrez = 'entrez')

  ## convert to long format
  df_long <- eset_aggr %>% as_tibble(rownames='entrez') %>%
    gather(sample, expr, -entrez)
```

```
  if(filter_ref){
    df_long <- df_long %>% filter(entrez %in% entrez_ref)
  }

  ## return results
  return(df_long)
}
```

Next we can apply this function to extract the required data from the Adai expression set. The 'chip' argument to the extraction function refers to the column names of the probe_map object.

```
adai_for_cms <- get_cms_gene_lvl2(adai_exprs, 'Affy HG U133-PLUS-2 probeset') %>%
  mutate(dataset='Adai', platform = 'HG_U133_Plus2',
         sample=gsub('.cel.gz', '', sample))
```

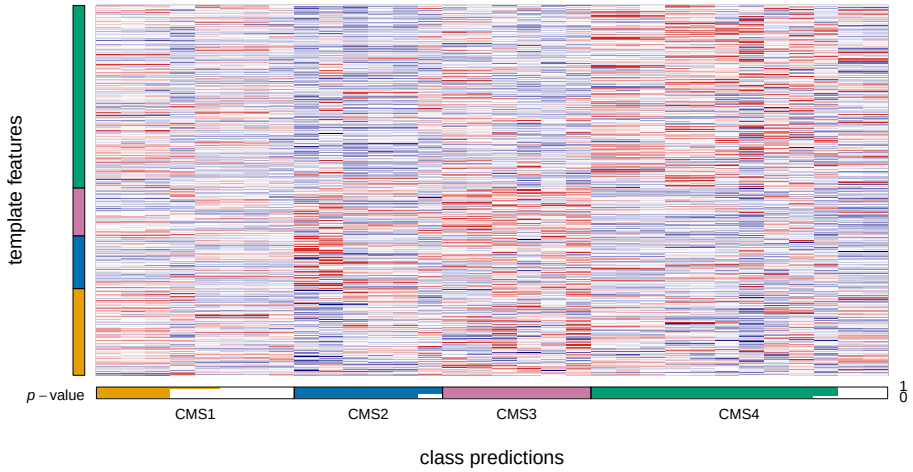
Finally we need to know for the purpose of downstream analysis which sample corresponds to which CRC cell lines. We downloaded information about he samples from GEO which we load from an R data file and which we are going to use for this purpose.

```
data('adai_sample_anno', package='CMSYM1552018')
```

```
## add to adai_for_cms data, exclude misclassified lines
adai_for_cms <- adai_for_cms %>% left_join(adai_sample_anno) %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS'))
```

Now we classify the CRC subtypes of the Adai cell line data set using the CMSclassifier (Guinney et al. 2015).

```
## annotation frame and expression matrix for classification
adai_mat <- adai_for_cms %>% dplyr::select(entrez, sample, expr) %>%
  group_by(sample, entrez) %>% summarise(expr=mean(expr)) %>% ungroup() %>%
  spread(sample, expr) %>% `rownames<-`(NULL) %>%
  data.frame() %>% column_to_rownames('entrez')
adai_anno <- adai_for_cms %>% distinct(sample, cellline)

## classify using RF, 1000 random samples
adai_cms <- map_df(1:1000, function(i){
  classifyCMS.RF(as.data.frame(adai_mat)[,sample(1:ncol(adai_mat), sample(5:ncol(adai_mat), 1))]) %>%
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)
})

## annotate samples with CMS predictions
adai_cms <- adai_anno %>%
  left_join(adai_cms %>% `colnames<-`(gsub('.posteriorProb', '', gsub('RF.', '', colnames(.))))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  group_by(sample, cellline) %>% count(predictedCMS) %>% ungroup()

## Cell-line CMS classifier
adai_cms_preclin <- CMScaller(adai_mat)
```




```
adai_cms_preclin <- adai_cms_preclin %>% as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction) %>%
  inner_join(adai_anno)
adai_cms <- adai_cms %>% inner_join(adai_cms_preclin)
```

4.2 Wagner cell line

The Wagner cell line dataset (Wagner et al. 2007) is another Oncomine dataset that includes colorectal cancer cell lines similar to the Adai data set. Again an Affymetrix HG-U133 Plus2 chip was used to characterize gene expression in cell lines. We proceed similar to above.

```
data('wagner_exprs', package='CMSYM1552018')
```

We select the required data from all probe level expression values and aggregate to gene level using the function defined for this purpose above.

```
wagner_for_cms <- get_cms_gene_lvl2(wagner_exprs, 'Affy HG U133-PLUS-2 probeset') %>%
  mutate(dataset='Wagner', platform = 'HG_U133_Plus2') %>%
  mutate(sample=gsub('.CEL.gz', '', sample)) %>%
  mutate(sample=gsub('_\\d$', '', sample))
```

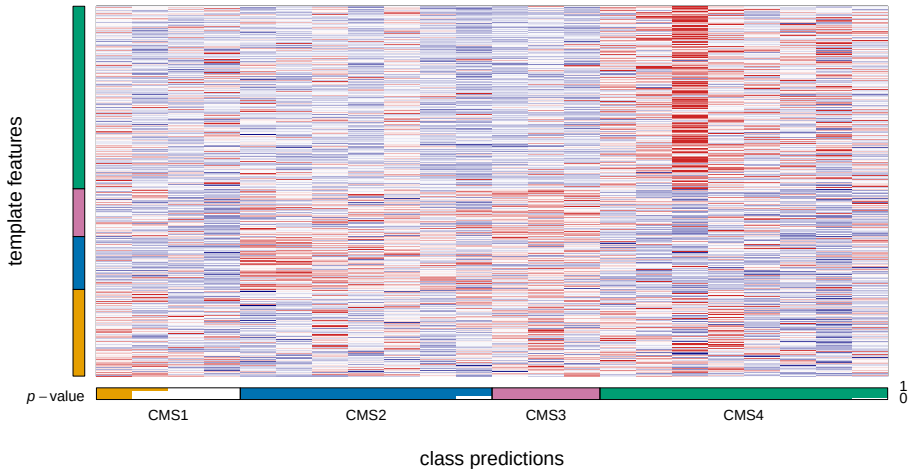
Again we annotate the cell line for each sample.

```
data('wagner_sample_info', package='CMSYM1552018')

## add to wagner_for_cms data
wagner_for_cms <- wagner_for_cms %>% left_join(wagner_sample_info) %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS'))
```

Now we classify the CRC subtypes of the Wagner cell line data set using the CMSclassifier.

```
## annotation frame and expression matrix for classification
wagner_mat <- wagner_for_cms %>% dplyr::select(entrez, sample, expr) %>%
  group_by(sample, entrez) %>% summarise(expr=mean(expr)) %>% ungroup() %>%
  spread(sample, expr) %>% `rownames<-`(NULL) %>%
  data.frame() %>% column_to_rownames('entrez')
wagner_anno <- wagner_for_cms %>% distinct(sample, cellline)
```

```
## classify using RF, 1000 random samples
wagner_cms <- map_df(1:1000, function(i){
  classifyCMS.RF(as.data.frame(wagner_mat)[,sample(1:ncol(wagner_mat), sample(5:ncol(wagner_mat), 1))]) %>%
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)
})

## annotate samples with CMS predictions
wagner_cms <- wagner_anno %>%
  left_join(wagner_cms %>% `colnames<-`(gsub('.posteriorProb', '', gsub('RF.', '', colnames(.))))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  group_by(sample, cellline) %>% count(predictedCMS) %>% ungroup()

## Cell-line CMS classifier
wagner_cms_preclin <- CMScaller(wagner_mat)
```




```
wagner_cms_preclin <- wagner_cms_preclin %>% as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction) %>%
  inner_join(wagner_anno)
wagner_cms <- wagner_cms %>% inner_join(wagner_cms_preclin)
```

4.3 Garnett cell line

For the next dataset (Garnett et al. 2012) we proceed as above.

```
data('garnett_exprs', package='CMSYM1552018')
```

We select the required data from all probe level expression values and aggregate to gene level using the function defined for this purpose above. A small difference is, that in comparison to the other data sets the Affymetrix HG-U133A platform was used for the experiments.

```
garnett_for_cms <- get_cms_gene_lvl2(garnett_exprs, 'Affy HG U133A probeset') %>%
  mutate(dataset='Garnett', platform = 'HG_U133A')
```

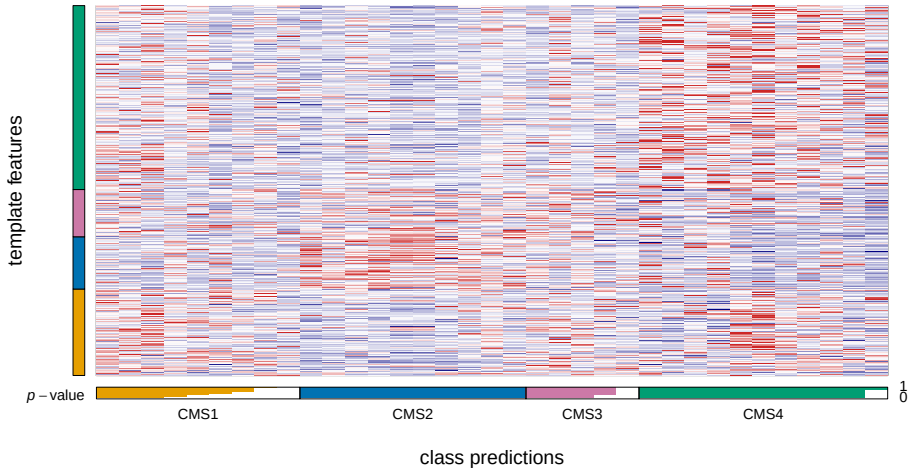
Again we annotate the cell line for each sample.

```
data('garnett_sample_info', package='CMSYM1552018')
```

```
## add to wagner_for_cms data
garnett_for_cms <- garnett_for_cms %>% left_join(garnett_sample_info) %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS'))
```

Now we classify the CRC subtypes of the Garnett cell line data set using the CMSclassifier.

```
## annotation frame and expression matrix for classification
garnett_mat <- garnett_for_cms %>% dplyr::select(entrez, sample, expr) %>%
  group_by(sample, entrez) %>% summarise(expr=mean(expr)) %>% ungroup() %>%
  spread(sample, expr) %>% `rownames<-`(NULL) %>%
  data.frame() %>% column_to_rownames('entrez')
garnett_anno <- garnett_for_cms %>% distinct(sample, cellline)

## classify using RF, 1000 random samples
garnett_cms <- map_df(1:1000, function(i){
  classifyCMS.RF(as.data.frame(garnett_mat)[,sample(1:ncol(garnett_mat), sample(5:ncol(garnett_mat), 1))]) %>
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)
})

## annotate samples with CMS predictions
garnett_cms <- garnett_anno %>% mutate(sample = paste0('X', sample)) %>%
  left_join(garnett_cms %>% `colnames<-`(gsub('.posteriorProb', '', gsub('RF.', '', colnames(.))))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  group_by(sample, cellline) %>% count(predictedCMS) %>% ungroup()

## Cell-line CMS classifier
garnett_cms_preclin <- CMScaller(garnett_mat)
```




```
garnett_cms_preclin <- garnett_cms_preclin %>% as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction) %>%
  inner_join(garnett_anno %>% mutate(sample = paste0('X', sample)))
garnett_cms <- garnett_cms %>% inner_join(garnett_cms_preclin)
```

4.4 Barettina cell line

This cell line data set is the older microarray based expression data set of the Cancer Cell Line Encyclopedia (CCLE) (Barretina et al. 2012). We process as above. The HG U133 Plus2 platform was used.

```
data('ccle_exprs', package='CMSYM1552018')
```

We select the required data from all probe level expression values and aggregate to gene level.

```
ccle_for_cms <- get_cms_gene_lvl2(ccle_exprs, 'Affy HG U133-PLUS-2 probeset') %>%
  mutate(dataset='CCLE', platform = 'HG_U133_Plus2',
         sample = gsub('.CEL', '', sample))
```

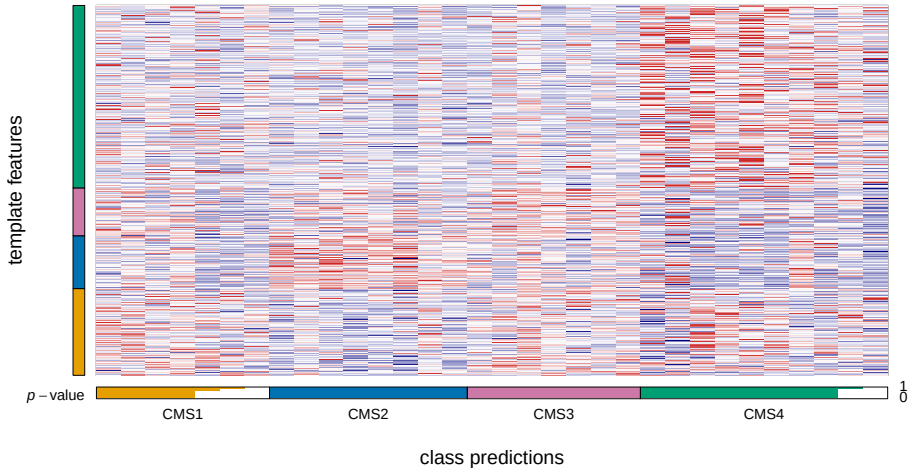
Again we annotate the cell line for each sample.

```
data('ccle_sample_info', package='CMSYM1552018')
```

```
## add to wagner_for_cms data
ccle_for_cms <- ccle_for_cms %>% mutate(sample = gsub('\\.', '-', sample)) %>%
  inner_join(ccle_sample_info) %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS'))
```

Now we classify the CRC subtypes of the CCLE cell line data set using the CMSclassifier.

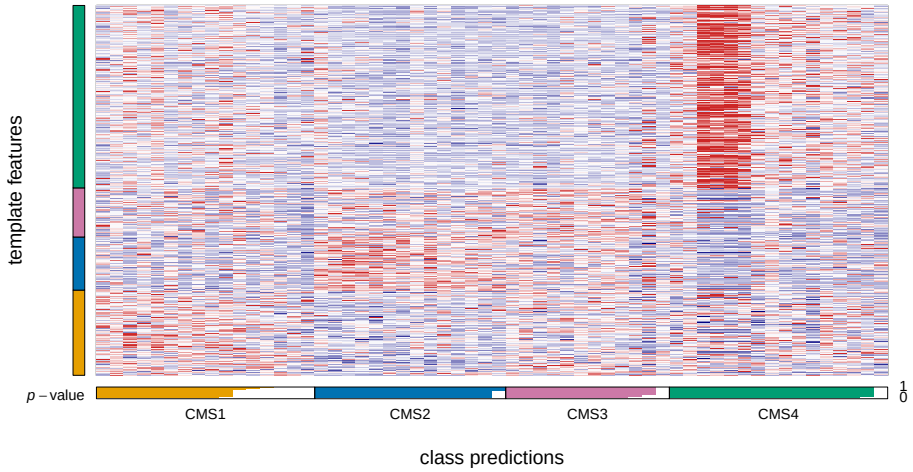
```
## annotation frame and expression matrix for classification
ccle_mat <- ccle_for_cms %>% dplyr::select(entrez, sample, expr) %>%
  group_by(sample, entrez) %>% summarise(expr=mean(expr)) %>% ungroup() %>%
  spread(sample, expr) %>% `rownames<-`(NULL) %>%
  data.frame() %>% column_to_rownames('entrez')
ccle_anno <- ccle_for_cms %>% distinct(sample, cellline) %>%
  mutate(sample = gsub('-', '.', sample))

## classify using RF, 1000 random samples
ccle_cms <- map_df(1:1000, function(i){
  classifyCMS.RF(as.data.frame(ccle_mat)[,sample(1:ncol(ccle_mat), sample(5:ncol(ccle_mat), 1))]) %>%
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)
})

## annotate samples with CMS predictions
ccle_cms <- ccle_anno %>%
  left_join(ccle_cms %>% `colnames<-`(gsub('.posteriorProb', '', gsub('RF.', '', colnames(.))))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  group_by(sample, cellline) %>% count(predictedCMS) %>% ungroup()

## Cell-line CMS classifier
ccle_cms_preclin <- CMScaller(ccle_mat)
```




```
ccle_cms_preclin <- ccle_cms_preclin %>% as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction) %>%
  inner_join(ccle_anno)
ccle_cms <- ccle_cms %>% inner_join(ccle_cms_preclin)
```

Recently there has also been a large RNAseq data set with colorectal cell lines released (Meyers et al. 2017). We can also try to classify based on this set to see if the results are coherent.

```
data('ccle_rnaseq_mat', package='CMSYM1552018')
ccle_rnaseq_mat <- ccle_rnaseq_mat[,!colnames(ccle_rnaseq_mat) %in% c('COLO741', 'NCIH630', 'FHC', 'FHS')]

ccle_rnaseq_cms <- map_df(1:1000, function(i){
  ## sample columns for classification
  sample_cols <- sample(1:ncol(ccle_rnaseq_mat),
                        sample(5:ncol(ccle_rnaseq_mat), 1))

  ## determine cms for this iteration
  cms_it <- classifyCMS.RF(as.data.frame(ccle_rnaseq_mat)[,sample_cols]) %>%
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)

  return(cms_it)
})

## classify subtypes
ccle_rnaseq_cms <- ccle_rnaseq_cms %>%
  `colnames<-`(gsub('RF.', '', gsub('.posteriorProb', '', colnames(.)))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  count(sample, predictedCMS)

## Cell-line CMS classifier
ccle_rnaseq_cms_preclin <- CMScaller(ccle_rnaseq_mat)
```




```
ccle_rnaseq_cms_preclin <- ccle_rnaseq_cms_preclin %>%
  as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction)
ccle_rnaseq_cms <- ccle_rnaseq_cms %>% inner_join(ccle_rnaseq_cms_preclin)
```

4.5 Medico cell line

This data set was published by Medico and colleagues (Medico et al. 2015) is an especially rich resource of mRNA expression in colorectal cancer cell lines. In contrast to the data sets processed so far, an Illumina microarry platform was used to generate the data so the data cannot be processed exactly as the others. We used the GEOquery package to download the data from GEO (Edgar, Domrachev, and Lash 2002) and can now load the expression values. As we do not have access to the probe level data we will just quantile normalize log-transformed expression values.

```
data('medico_exprs', package='CMSYM1552018')

## quantile normalize
medico_norm <- normalize.quantiles(medico_exprs) %>% log2()
colnames(medico_norm) <- colnames(medico_exprs)
rownames(medico_norm) <- rownames(medico_exprs)
```

Now we can proceed as we did with the other samples.

```
medico_for_cms <- get_cms_gene_lvl2(medico_norm, 'Illumina Human HT 12 V4 probe') %>%
  mutate(dataset='Medico', platform = 'Illumina_HT12')
```

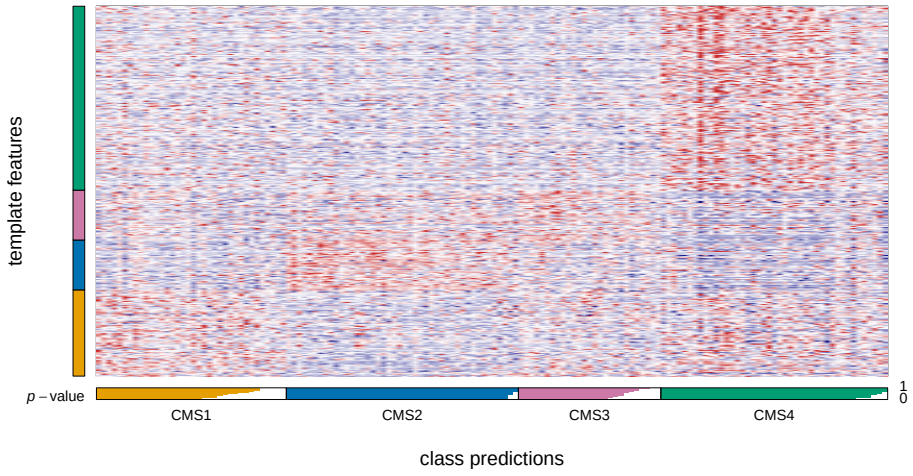
Finally, we annotate the cell line corresponding to each sample.

```
data('medico_samples', package='CMSYM1552018')
```

```
## merge sample info with data, exclude misclassified lines
medico_for_cms <- medico_for_cms %>% inner_join(medico_samples) %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS', 'HUTU80'))
```

Now we classify the CRC subtypes of the CCLE cell line data set using the CMSclassifier.

```
## annotation frame and expression matrix for classification
medico_mat <- medico_for_cms %>% dplyr::select(entrez, sample, expr) %>%
  group_by(sample, entrez) %>% summarise(expr=mean(expr)) %>% ungroup() %>%
  spread(sample, expr) %>% `rownames<-`(NULL) %>%
  data.frame() %>% column_to_rownames('entrez')
medico_anno <- medico_for_cms %>% distinct(sample, cellline) %>%
  mutate(sample = gsub('-', '.', sample))

## classify using RF, 1000 random samples
medico_cms <- map_df(1:1000, function(i){
  classifyCMS.RF(as.data.frame(medico_mat)[,sample(1:ncol(medico_mat), sample(5:ncol(medico_mat), 1))]) %>%
    as_tibble(rownames = 'sample') %>%
    mutate(iteration = i)
})

## annotate samples with CMS predictions
medico_cms <- medico_anno %>%
  left_join(medico_cms %>% `colnames<-`(gsub('.posteriorProb', '', gsub('RF.', '', colnames(.))))) %>%
  mutate(predictedCMS = ifelse(is.na(predictedCMS), 'np', predictedCMS)) %>%
  group_by(sample, cellline) %>% count(predictedCMS) %>% ungroup()

## Cell-line CMS classifier
medico_cms_preclin <- CMScaller(medico_mat)
```




```
medico_cms_preclin <- medico_cms_preclin %>%
  as_tibble(rownames='sample') %>%
  dplyr::select(sample, CMSpreclinical = prediction) %>%
  inner_join(medico_anno)
medico_cms <- medico_cms %>% inner_join(medico_cms_preclin)
```

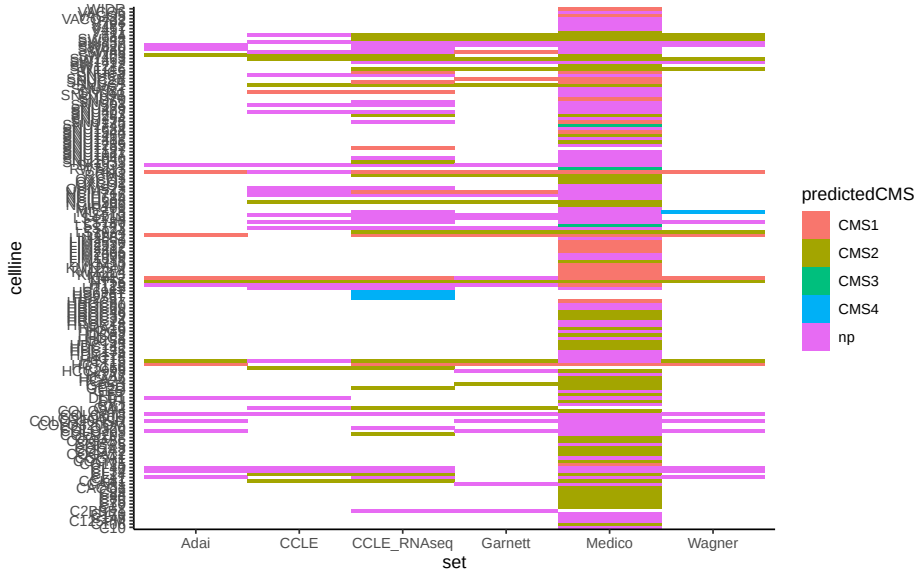
5 Aggregation of subtype data

Now that we have classified all datasets for molecular subtypes we aggregate them into one table to choose the consensus molecular subtypes. We further annotate information about the microsatelite instability status of each cell line as it correlates well with molecular subtype 1 and might be an interesting biological covariate for downstream analyses.

```
data('msi_status', package='CMSYM1552018')

## combine classification results of individual datasets
all_cms <- adai_cms %>% mutate(set = 'Adai') %>%
  bind_rows(wagner_cms %>% mutate(set = 'Wagner')) %>%
  bind_rows(garnett_cms %>% mutate(set = 'Garnett')) %>%
  bind_rows(ccle_cms %>% mutate(set = 'CCLE')) %>%
  bind_rows(medico_cms %>% mutate(set = 'Medico')) %>%
  bind_rows(ccle_rnaseq_cms %>% mutate(set = 'CCLE_RNAseq', cellline = sample)) %>%
  mutate(cellline = toupper(cellline),
         CMSpreclinical = ifelse(is.na(CMSpreclinical), 'np', as.character(CMSpreclinical))) %>%
  filter(n >= 150) %>%
  group_by(set, cellline, sample) %>% arrange(predictedCMS) %>% dplyr::slice(1) %>% ungroup() %>%
  group_by(cellline, set) %>%
  summarise(predictedCMS = names(sort(table(predictedCMS), decreasing=T))[1],
            CMSpreclinical = names(sort(table(CMSpreclinical), decreasing=T))[1]) %>%
  ungroup() %>%
  group_by(cellline) %>%
  summarise(cms_profile = paste(predictedCMS, collapse=','),
            cms_profile_preclin = paste(CMSpreclinical, collapse=',')) %>%
  ungroup()

## add MSI info
all_cms <- all_cms %>% left_join(msi_status)
```

We can also generate a visualization of the predicted subtypes across data sets.

```
adai_cms %>% mutate(set = 'Adai') %>%
  bind_rows(wagner_cms %>% mutate(set = 'Wagner')) %>%
  bind_rows(garnett_cms %>% mutate(set = 'Garnett')) %>%
  bind_rows(ccle_cms %>% mutate(set = 'CCLE')) %>%
  bind_rows(medico_cms %>% mutate(set = 'Medico')) %>%
  bind_rows(ccle_rnaseq_cms %>% mutate(set = 'CCLE_RNAseq', cellline = sample)) %>%
  mutate(cellline = toupper(cellline)) %>%
  group_by(set, cellline, sample) %>% arrange(desc(n)) %>% dplyr::slice(1) %>% ungroup() %>%
  dplyr::select(cellline, set, predictedCMS) %>% distinct() %>%
  filter(!cellline %in% c('COLO741', 'NCIH630', 'FHC', 'FHS')) %>%
  ggplot(aes(set, cellline, fill = predictedCMS)) +
  geom_tile() + theme_classic()
```



There are no disagreements which is nice to see. It can, however, happen that based on one sample a sub-type could not be assigned where based on another sample classification was possible. In these cases we select the most frequent classification result. Based on this we firmly assign a subtype to each cell line. We are then interest in observing the distribution of predicted subtypes.

```
## by sorting alphabetically we can retrieve a
## subtype if possible as 'C' comes before 'n'
cms_cl <- all_cms %>%
  mutate(cms = map(cms_profile,
                   function(x) strsplit(x, ',') %>% unlist() %>%
                     table() %>% .[. == max(.)] %>% names() %>%
                     sort() %>% .[1]),
         cms_preclin = map(cms_profile_preclin,
                   function(x) strsplit(x, ',') %>% unlist() %>%
                     table() %>% .[. == max(.)] %>% names() %>%
                     sort() %>% .[1])) %>%
  unnest(c(cms, cms_preclin))
```

In agreement with other reports in tumour organoids we observe a strong tendency towards classification into CMS1 or CMS2, whereas CMS3 and CMS4 are only rarely observed.

Based on the paper we would assume that there is a significant relationship between the MSI/MSS status and the molecular subtype, wehere subtype 1 should be more commonly MSI.

```
cms_cl %>% filter(cms %in% c('CMS1', 'CMS2')) %>%
  dplyr::select(-c(cellline, cms_profile, cms_profile_preclin, cms_preclin)) %>%
  table() %>% fisher.test()

## for preclinical classification
cms_cl %>% filter(cms_preclin %in% c('CMS1', 'CMS2')) %>%
  dplyr::select(-c(cellline, cms_profile, cms_profile_preclin, cms)) %>%
  table() %>% fisher.test()
```

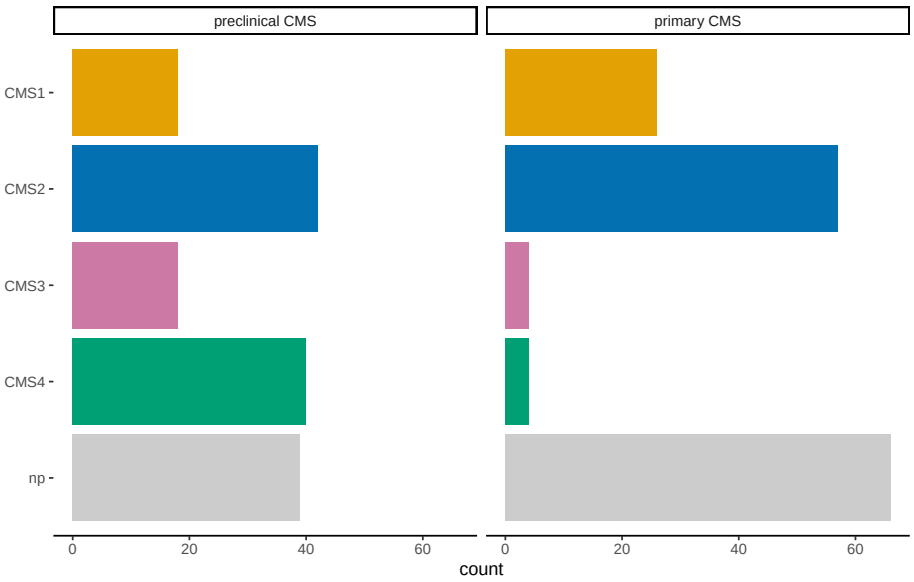


Figure 2: **Distribution of CRC cell line subtypes**

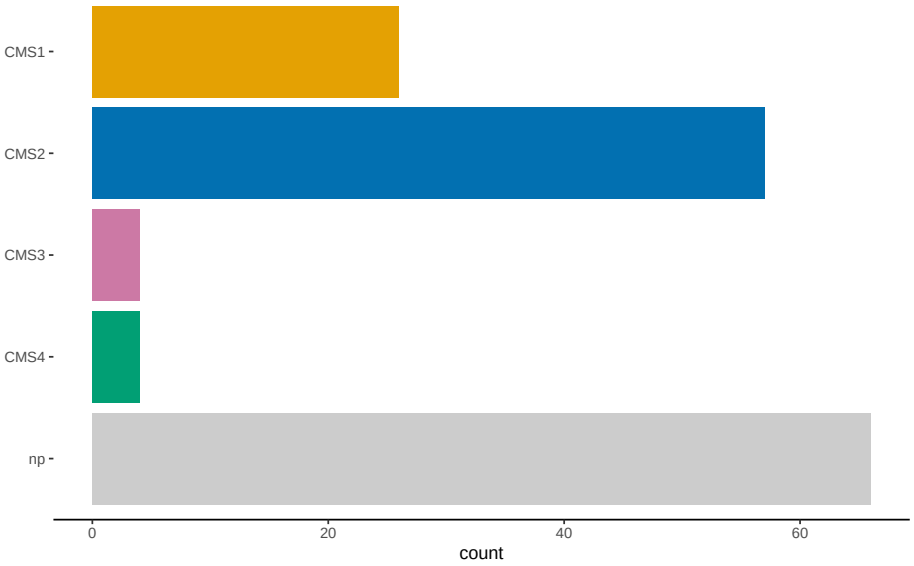


Figure 3: **Distribution of CRC cell line subtypes**

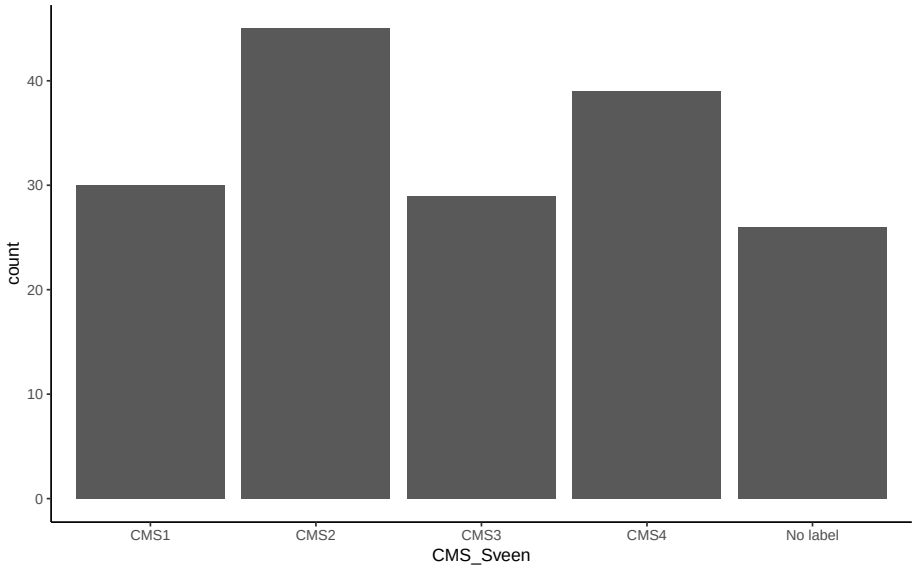
6 Comparison with other studies

Recently, studies have been published where CMS classification was performed for CRC cell lines (Sveen et al. 2018) (Linnekamp et al. 2018).

6.1 Sveen et al

Here a specific CMS classifier for CRC cell lines was derived based on a smaller set (~450) of genes highly expressed in cell lines. We want to see how consistent these results are with our classification results where the CMS classifier was used as published based on primary tumour samples. We read the resulting Sveen et al classifications into R and compare them to the CMSclassifier class labels.

```
## load sveen cms data
data('sveen_cms', package='CMSYM1552018')

## bar plot of cms label distribution
sveen_cms %>% ggplot(aes(CMS_Sveen)) + geom_bar() + theme_classic()
```



##Linnekamp et al

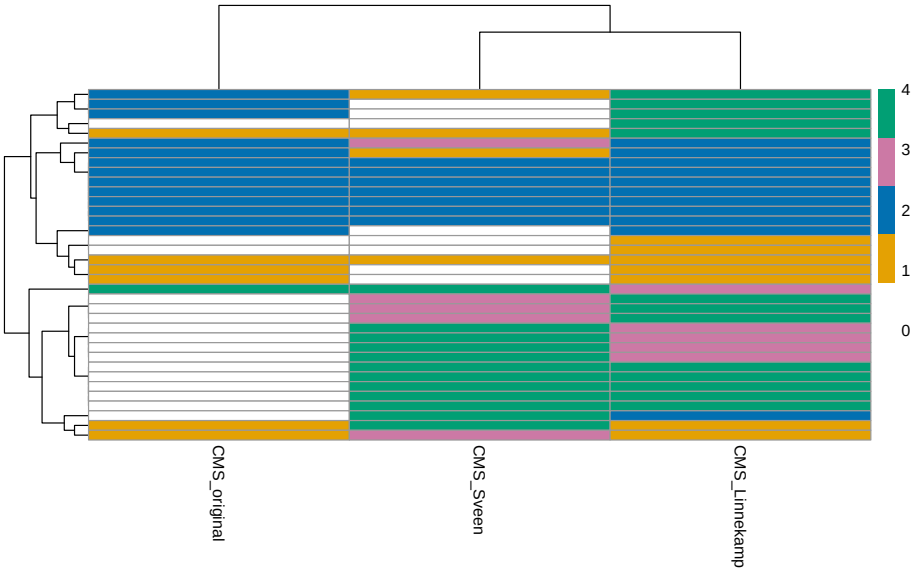
A suppmentary table with suptype predictions is not available from the paper so we type in the predicitons manually based on Figure 2.

```
lk_cms <- list(
  CMS1 = c('SNUC2B', 'LS411N', 'LOVO', 'HCT116', 'RKO', 'KM12', 'SW48'),
  CMS2 = c('CCK81', 'GP5D', 'SW1417', 'NCIH630', 'SW1463', 'SW948', 'T84',
           'SNUC1', 'RCM1', 'SW1116', 'HT55', 'LS1034'),
  CMS3 = c('LS123', 'OUMS23', 'NCIH716', 'COLO320HSR', 'HUTU80', 'MDST8', 'CAR1'),
  CMS4 = c('SW620', 'C2BBE1', 'COLO678', 'NCIH747', 'HCT15', 'CL11', 'SW837',
           'HT29', 'SKCO1', 'HT115', 'HCC2998', 'COLO205', 'CW2')
)

linne_cms <- map2(names(lk_cms), 1:length(lk_cms),
                  function(x, y) tibble(CMS_Linnekamp=x, cellline=lk_cms[[y]])) %>%
  bind_rows()
```

6.2 Comparison

We compare the classification labels of our cell lines and the other cell line CMS predicitions. We generate both a heat map and a bar plot to visualize overlap.

```
sveen_cms %>% full_join(cms_cl %>% dplyr::select(cellline, CMS_original=cms)) %>%
  full_join(linne_cms) %>% drop_na() %>% mutate_all(~gsub('CMS', '', .)) %>%
  mutate_all(~ifelse(. %in% c('np', 'No label'), 0, .)) %>%
  data.frame() %>% `rownames<-`(NULL) %>% column_to_rownames('cellline') %>%
  apply(2, as.integer) %>%
  pheatmap(color=c('#ffffff', '#e4a103','#0270b1','#cc79a5','#019f74'))
```

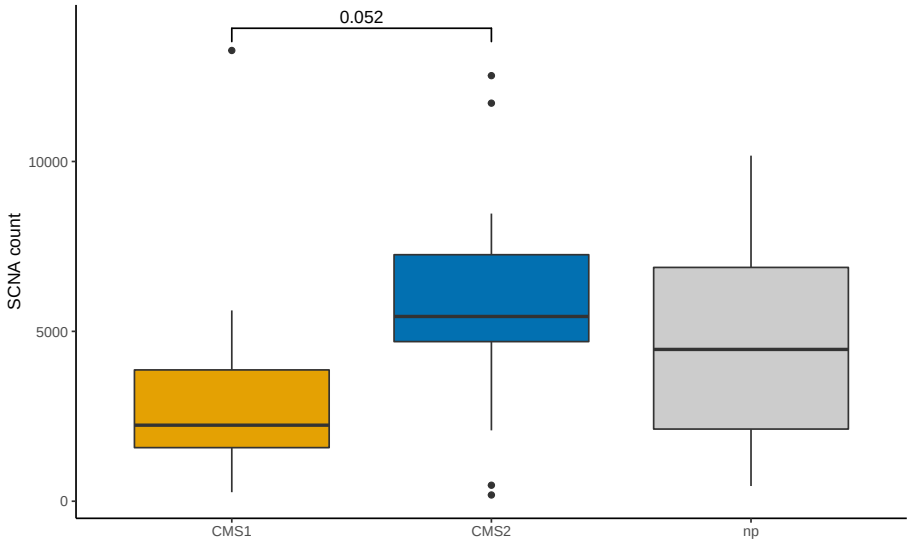


7 Characterization of predicted subytpes

We next try to recreate the copy number and mutation plots shown in the paper by Guinney et al., to observe if cell lines behave similarly. To do this, we use data from the COSMIC database (Forbes et al. 2015). Mutations include all non-silent mutation events. Copy number changes of X and Y chromosomes are disregarded.

```
## read and preprocess data exported from CCLE
data('ccle_cnv', package='CMSYM1552018')
ccle_cnv <- ccle_cnv %>% dplyr::select(EGID:value) %>%
  inner_join(cms_cl)

## plot cna across groups (boxplot)
scna_plot <- ccle_cnv %>% filter(tissue == 'LARGE_INTESTINE') %>%
  mutate(cnv=round((2^value)*2, digits=0)) %>%
  filter(!cnv %in% 2, CHR %in% as.character(1:22)) %>%
  gather(cms_type, cms, cms, cms_preclin) %>%
  count(cms_type, cms, cellline) %>%
  filter(cms != 'np') %>%
  ggplot(aes(cms, n, fill=cms)) + geom_boxplot() + theme_classic() +
```

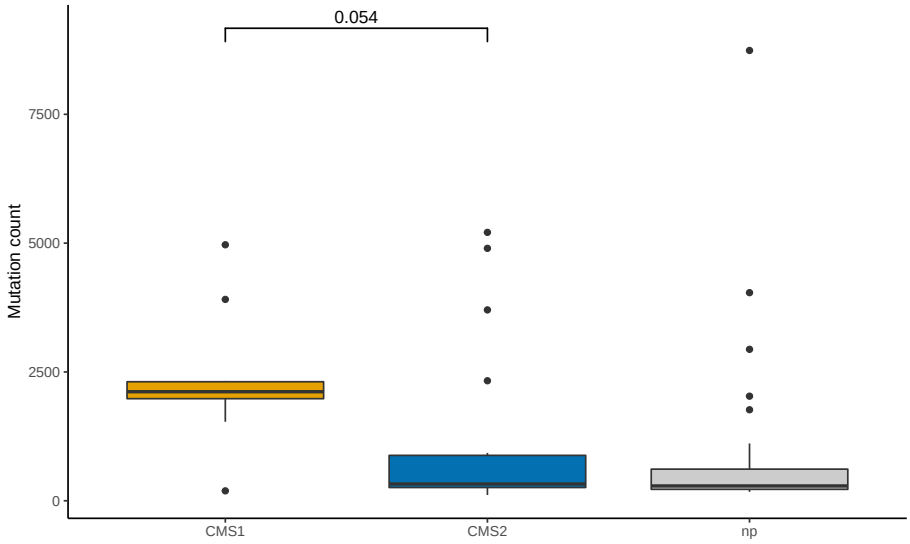
```
  geom_signif(comparisons = list(c('CMS1', 'CMS2'))) +
  facet_wrap(~cms_type, ncol=1) +
  xlab('') + ylab('SCNA count') +
  scale_fill_manual(values=c('#e4a103','#0270b1','#cc79a5',
                             '#019f74', '#cccccc')) +
  theme(legend.position='none')

ccle_cnv %>% filter(tissue == 'LARGE_INTESTINE') %>%
  mutate(cnv=round((2^value)*2, digits=0)) %>%
  filter(!cnv %in% 2, CHR %in% as.character(1:22)) %>%
  gather(cms_type, cms, cms, cms_preclin) %>%
  count(cms_type, cms, cellline) %>%
  filter(cms_type == 'cms', cms %in% c('CMS1', 'CMS2', 'np')) %>%
  ggplot(aes(cms, n, fill=cms)) + geom_boxplot() + theme_classic() +
  geom_signif(comparisons = list(c('CMS1', 'CMS2'))) +
  xlab('') + ylab('SCNA count') +
  scale_fill_manual(values=c('#e4a103','#0270b1', '#cccccc')) +
  theme(legend.position='none')
```



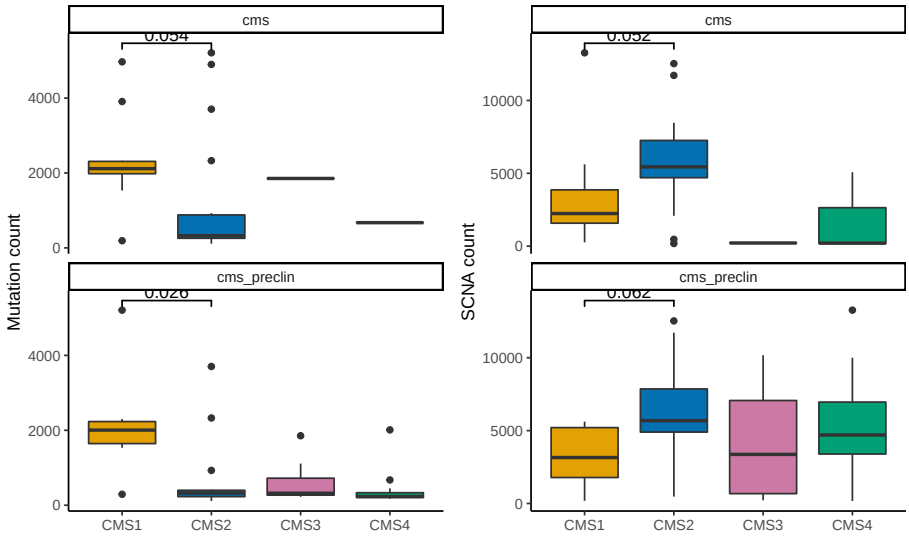
We further aim to reproduce the mutational characteristics described in the original CMS study (Guinney et al. 2015).

```
## read mutation data derived from COSMIC
data('cosmic_mut_crc', package='CMSYM1552018')

## boxplot of mutations across subtypes
mut_plot <- cosmic_mut_crc %>% dplyr::select(cellline, symbol) %>%
  distinct() %>% inner_join(cms_cl) %>%
  gather(cms_type, cms, cms, cms_preclin) %>%
  count(cms_type, cellline, cms) %>%
  filter(cms != 'np') %>%
  dplyr::select(`Molecular subtype`=cms, Count=n, everything()) %>%
  ggplot(aes(`Molecular subtype`, Count, fill=`Molecular subtype`)) +
  geom_boxplot() + facet_wrap(~cms_type, ncol=1) +
```

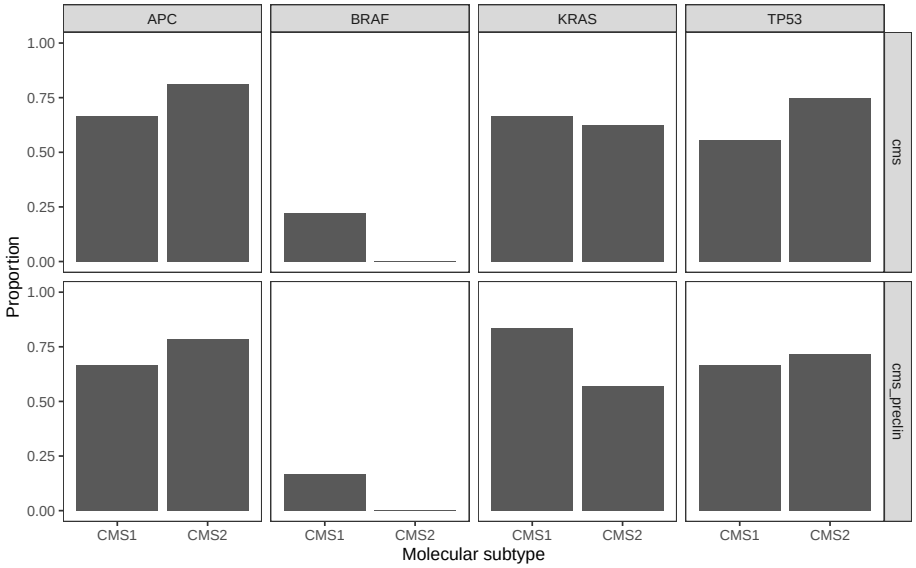
```
  theme_classic() +
  scale_fill_manual(values=c('#e4a103','#0270b1','#cc79a5',
                             '#019f74', '#cccccc')) +
  geom_signif(comparisons=list(c('CMS1', 'CMS2'))) +
  xlab('') + ylab('Mutation count') +
  theme(legend.position='none')

cosmic_mut_crc %>% dplyr::select(cellline, symbol) %>%
  distinct() %>% inner_join(cms_cl) %>%
  gather(cms_type, cms, cms, cms_preclin) %>%
  count(cms_type, cms, cellline) %>%
  filter(cms_type == 'cms', cms %in% c('CMS1', 'CMS2', 'np')) %>%
  ggplot(aes(cms, n, fill=cms)) +
  geom_boxplot() + theme_classic() +
  scale_fill_manual(values=c('#e4a103','#0270b1', '#cccccc')) +
  geom_signif(comparisons=list(c('CMS1', 'CMS2'))) +
  xlab('') + ylab('Mutation count') +
  theme(legend.position='none')
```




```
## plot
mut_plot + scna_plot
```

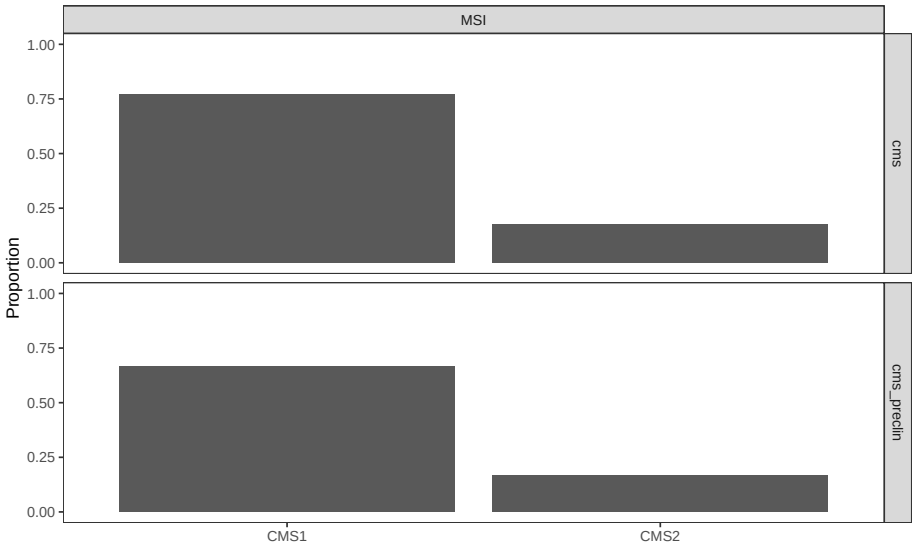



```
## barplot of mutation frequencey for known oncogenes
cosmic_mut_crc %>% filter(symbol %in% c('KRAS', 'BRAF', 'APC', 'TP53')) %>%
  dplyr::select(symbol, cellline, `Mutation AA`) %>% distinct() %>%
  inner_join(cms_cl) %>%
  gather(cms_type, cms, cms, cms_preclin) %>%
  group_by(cellline, cms, cms_type) %>% summarise(
    APC=ifelse('APC' %in% symbol, T, F),
    TP53=ifelse('TP53' %in% symbol, T, F),
    KRAS=ifelse('KRAS' %in% symbol, T, F),
    BRAF=ifelse(('BRAF' %in% symbol) &
                  ('p.V600E' %in% `Mutation AA`), T, F)) %>% ungroup() %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  group_by(cms, cms_type) %>% summarise(
    APC=length(which(APC))/n(),
    TP53=length(which(TP53))/n(),
    KRAS=length(which(KRAS))/n(),
    BRAF=length(which(BRAF))/n()) %>% ungroup() %>%
  gather(oncogene, `Mutation rate`, -c(cms, cms_type)) %>%
  dplyr::select(`Molecular subtype`=cms, everything()) %>%
  ggplot(aes(x=`Molecular subtype`, y=`Mutation rate`)) +
  geom_bar(stat='identity') +
  ylim(c(0,1)) + theme_bw() +
  facet_grid(cms_type~oncogene) + ylab('Proportion') +
  theme(panel.grid=element_blank())
```



Finally, we compare microsatellite instability status.

```
cms_cl %>% gather(cms_type, cms, cms, cms_preclin) %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  group_by(cms_type, cms) %>%
  summarise(Proportion=sum(msi_status == 'MSI', na.rm=T)/n()) %>%
  ungroup() %>% mutate(type='MSI') %>%
  ggplot(aes(cms, Proportion)) + geom_bar(stat='identity') +
  theme_bw() + theme(panel.grid=element_blank()) +
  facet_grid(cms_type~type) + xlab('') + ylim(c(0,1))
```



8 CMS-dependent drug response drived from public drug screening data

8.1 Identification of candidate substances

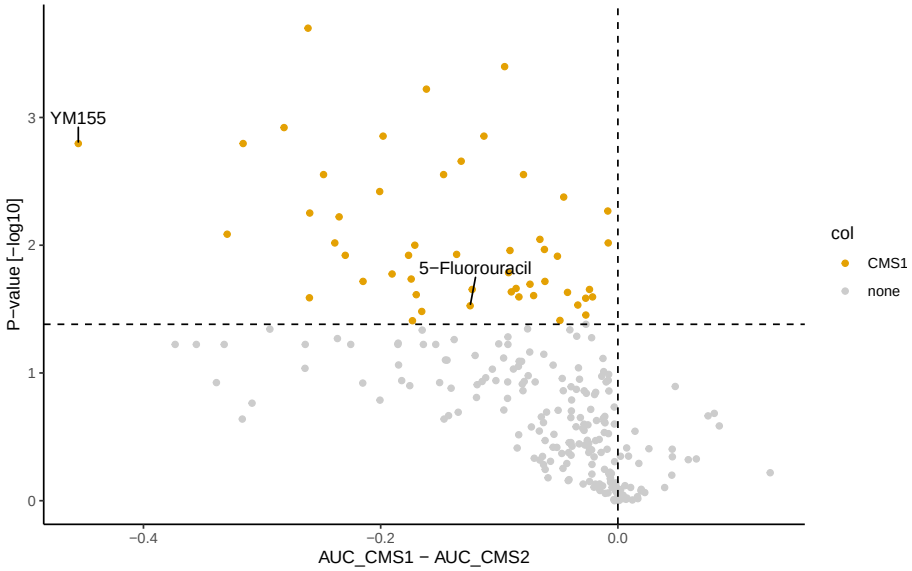
We read drug response AUC data from the 1001 cell lines study published in 2016 (Iorio et al. 2016). We select drug response data for CRC cell lines where subtype classifications are available.

```
## read auc data (Table S4; original publication)
data('drug_resp', package='CMSYM1552018')

## drug id-to-name mapping
drug_map <- drug_resp %>% dplyr::slice(1) %>% as.list %>% unlist
drug_map <- data.frame(drug_id=names(drug_map), name=drug_map) %>%
  tbl_df %>% drop_na()
```

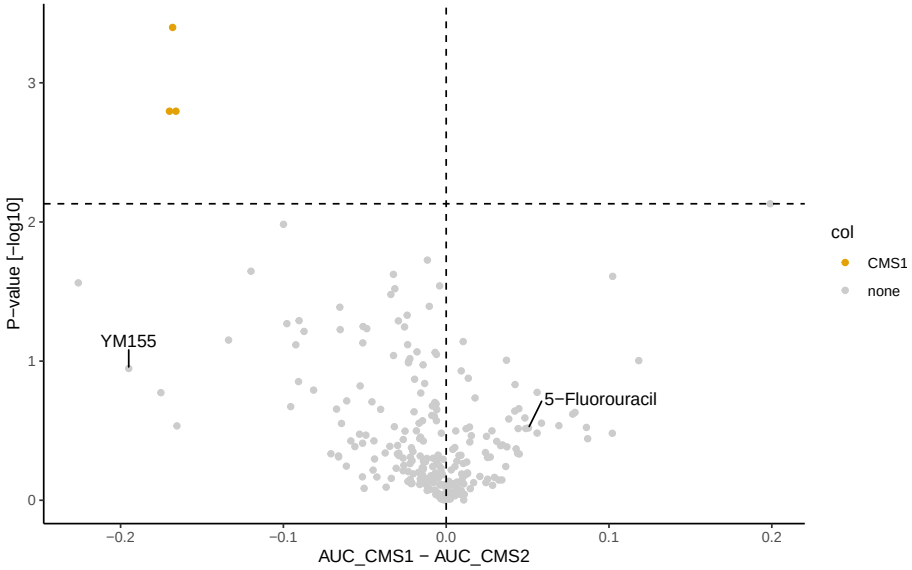
Statistical testing for subtype specific drugs. For testing we use a permutation test as implemented in the 'permTS' R package with 10,000 Monte Carlo resamplings.

```
## reformat cell line names and filter sub-types
resp_auc <- drug_resp %>% dplyr::slice(-1) %>% tbl_df %>%
  dplyr::select(-Cell.line.cosmic.identifiers) %>%
  mutate(Sample.Names=gsub('-', '',
                      gsub('\\.', '', toupper(Sample.Names)))) %>%
  dplyr::select(cellline=Sample.Names, everything()) %>%
  filter(cellline %in% cms_cl$cellline)
## type cast drug effects to numeric
resp_auc <- cbind(resp_auc[,1], apply(resp_auc[,2:ncol(resp_auc)], 2, as.numeric)) %>% tbl_df
## annotate drugs and cms
resp_auc <- resp_auc %>% melt %>% tbl_df %>%
  dplyr::select(drug_id=variable, everything()) %>%
  inner_join(drug_map) %>% inner_join(cms_cl)

## permutation test
drug_results_cms12_primary <- resp_auc %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  nest(-name) %>%
  mutate(test = map(data, ~ permTS(value ~ cms, data=.x, method="exact.mc",
                                   control=permControl(nmc=10^4)))) %>%
  mutate(res = map(test, ~ tibble(p.value = .x$p.value,
                                  delta_auc = .x$estimate))) %>%
  unnest(res) %>% mutate(fdr = p.adjust(p.value, method='BH')) %>%
  arrange(delta_auc)

## repeat the above analysis with MSI status
drug_results_msi_primary <- resp_auc %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  nest(-name) %>%
  mutate(test = map(data, ~ permTS(value ~ msi_status, data=.x, method="exact.mc",
                                   control=permControl(nmc=10^4)))) %>%
```

```
  mutate(res = map(test, ~ tibble(p.value = .x$p.value,
                                  delta_auc = .x$estimate))) %>%
  unnest(res) %>% mutate(fdr = p.adjust(p.value, method='BH')) %>%
  arrange(delta_auc)

drug_results_cms14_preclin <- resp_auc %>%
  mutate(cms = ifelse(cms_preclin %in% c('CMS1', 'CMS4'), 'CMS14',
               ifelse(cms_preclin %in% c('CMS2', 'CMS3'), 'CMS23', 'np'))) %>%
  filter(cms != 'np') %>%
  nest(-name) %>%
  mutate(test = map(data, ~ permTS(value ~ cms, data=.x, method="exact.mc",
                                   control=permControl(nmc=10^4)))) %>%
  mutate(res = map(test, ~ tibble(p.value = .x$p.value,
                                  delta_auc = .x$estimate))) %>%
  unnest(res) %>% mutate(fdr = p.adjust(p.value, method='BH')) %>%
  arrange(delta_auc)

## visualize as volcano plot
plot_drug_volcano_ym <- function(df){
  ## approximate 20% FDR cutoff
  fdr10 <- df %>% filter(fdr > 0.2) %>% arrange(p.value) %>%
    .$p.value %>% .[1] %>% log10() %>% `*`(-1)

  ## plot volcano
  df %>% mutate(col=ifelse(fdr < 0.2 & delta_auc > 0, 'CMS2',
                    ifelse(fdr < 0.2 & delta_auc < 0, 'CMS1', 'none'))) %>%
    mutate(label = ifelse(name %in% c('YM155', 'SN-38', '5-Fluorouracil'),
                          as.character(name), '')) %>%
    ggplot(aes(delta_auc, -log10(p.value), colour=col)) + geom_point() +
    geom_vline(xintercept=0, linetype = 'dashed') +
    geom_hline(yintercept = fdr10, linetype='dashed') +
    geom_text_repel(aes(label=label), nudge_y = 0.2, colour='black',
                    min.segment.length = 0) +
    scale_colour_manual(values = c('#e4a103', '#cccccc')) +
    xlab('AUC_CMS1 - AUC_CMS2') + ylab('P-value [-log10]') +
    theme(legend.position = 'none') +
    theme_classic()
}

plot_drug_volcano_ym(drug_results_cms12_primary)
```



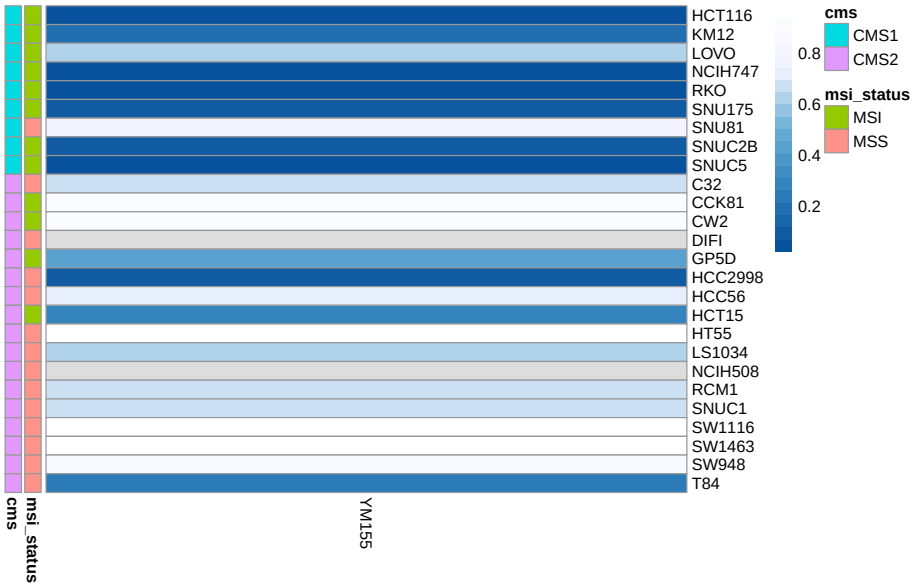

```
plot_drug_volcano_ym(drug_results_cms14_preclin)
```



We further plot a small heat map of drug response in individual cell lines.

```
## annotate rownames of heat map to show MSI status
row_anno <- resp_auc %>% filter(cms %in% c('CMS1', 'CMS2')) %>%
  distinct(cellline, cms, msi_status) %>%
  as.data.frame() %>% column_to_rownames('cellline')

resp_auc %>% filter(name %in% c('YM155'),
                    cms %in% c('CMS1', 'CMS2')) %>% arrange(cms, cellline) %>%
  mutate(cellline = factor(cellline, levels = unique(cellline))) %>%
  reshape2::acast(cellline ~ name, value.var = 'value') %>%
  pheatmap(cluster_rows = F, cluster_cols = F,
           annotation_row = row_anno,
```

```
                color = colorRampPalette(rev(c('#ffffff', '#eff3ff', '#6baed6','#3182bd','#08519c'))))(20))
```



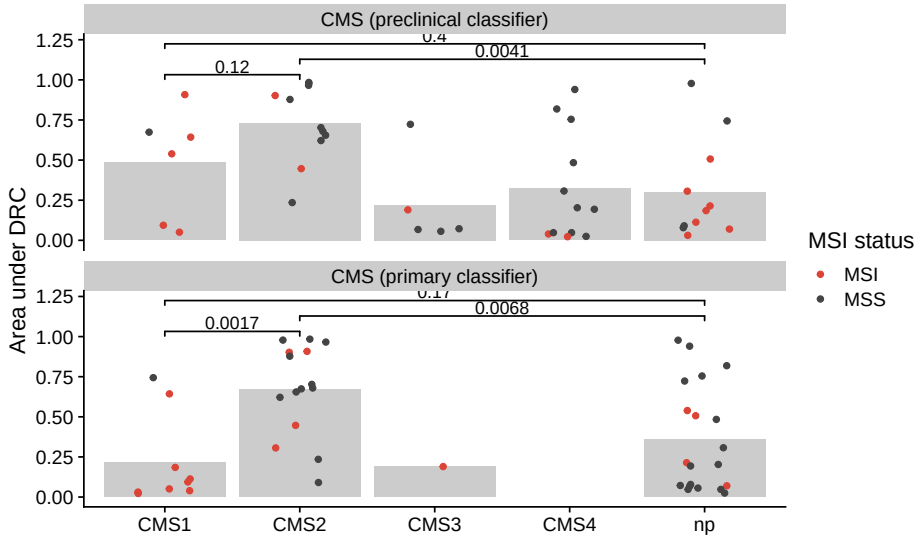
To investigate whether MSI is an equally good predictor for YM-155 sensitivity we perform a comparable test for MSI.

```
drug_results_msi_primary <- resp_auc %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  nest(-name) %>%
  mutate(test = map(data, ~ permTS(value ~ as.factor(msi_status),
                                   data=.x, method="exact.mc",
                                   control=permControl(nmc=10^4)))) %>%
  mutate(res = map(test, ~ tibble(p.value = .x$p.value,
                                  delta_auc = .x$estimate))) %>%
  unnest(res) %>% mutate(fdr = p.adjust(p.value, method='BH')) %>%
  arrange(delta_auc)
```

We also show a boxplot of YM155 response in CMS1,2 vs non-classified lines based on the primary classifier, and a similar plot comparing YM155 across subtypes based on the preclinical classifier.

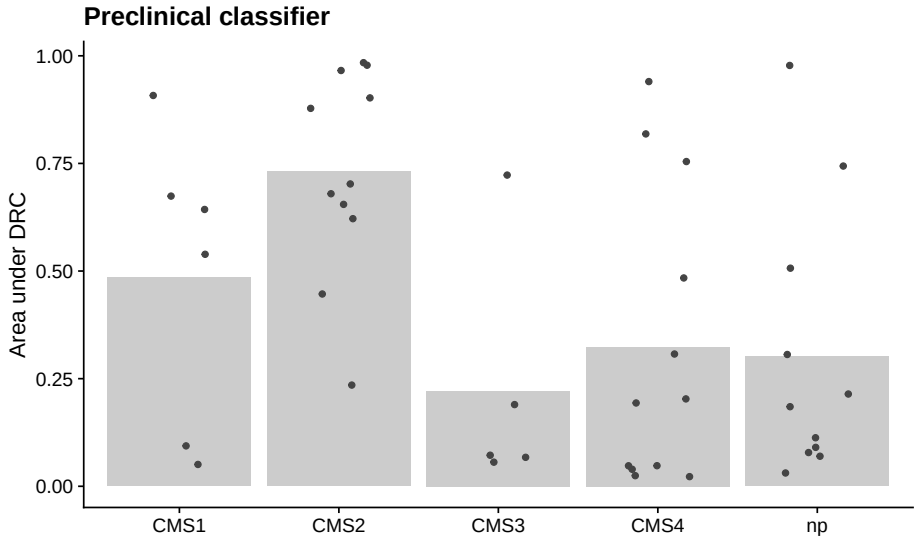
```
## YM155 response across all subtypes in both classifiers
resp_auc %>% filter(name == 'YM155') %>%
  dplyr::select(cellline, value, msi_status, cms, cms_preclin) %>%
  gather(classifier, cms, cms, cms_preclin) %>%
  mutate(classifier = ifelse(classifier == 'cms', 'CMS (primary classifier)', 'CMS (preclinical classifier)'
  ggplot(aes(cms, value)) +
  stat_summary(fun.y = 'mean', geom = 'bar', fill = '#cccccc') +
  geom_jitter(aes(color = msi_status), width = 0.2) +
  ggsignif::geom_signif(comparisons = list(c('CMS1', 'CMS2'), c('CMS2', 'np'), c('CMS1', 'np')),
                        step_increase = 0.1) +
  facet_wrap(~ classifier, ncol = 1) +
  cowplot::theme_cowplot() +
  labs(color = 'MSI status') +
```

```
  scale_color_manual(values = c('#db4437', '#444444')) +
  xlab('') + ylab('Area under DRC')
```

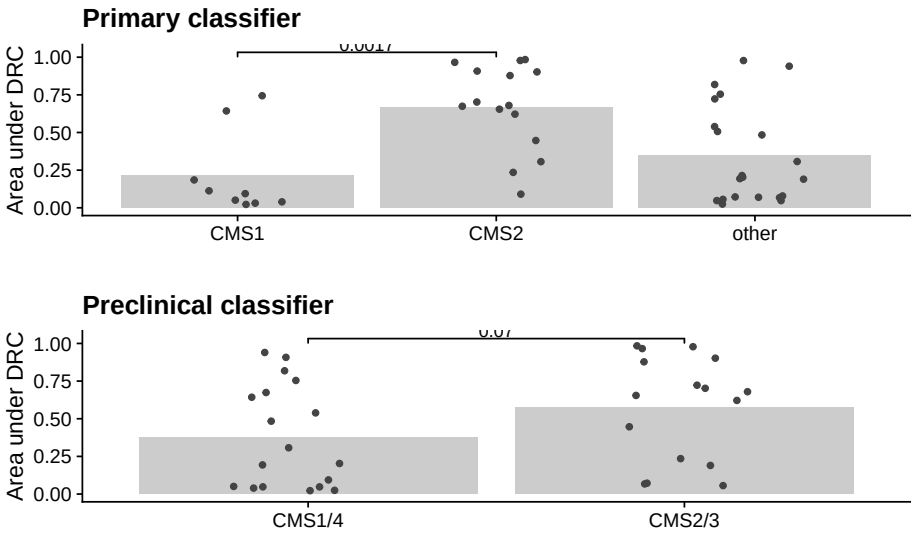



```
## CMS1/4 vs CMS2/3 preclinical
df <- resp_auc %>% filter(name == 'YM155') %>%
  dplyr::select(cellline, value, msi_status, cms, cms_preclin) %>%
  gather(classifier, cms, cms, cms_preclin) %>%
  mutate(cms = ifelse(classifier == 'cms' & !cms %in% c('CMS1', 'CMS2'), 'other', cms),
         cms = ifelse(classifier == 'cms_preclin' & cms %in% c('CMS1', 'CMS4'), 'CMS1/4',
                      ifelse(classifier == 'cms_preclin' & cms %in% c('CMS2', 'CMS3'), 'CMS2/3', cms))) %>%
  mutate(classifier = ifelse(classifier == 'cms', 'CMS (primary classifier)', 'CMS (preclinical classifier)')
  filter(cms != 'np')

p1 <- df %>% filter(classifier == 'CMS (primary classifier)') %>%
  ggplot(aes(cms, value)) +
  stat_summary(fun.y = 'mean', geom = 'bar', fill = '#cccccc') +
  geom_jitter(color = '#444444', width = 0.2) +
  ggsignif::geom_signif(comparisons = list(c('CMS1', 'CMS2')),
                        step_increase = 0.1) +
  cowplot::theme_cowplot() +
  xlab('') + ylab('Area under DRC') +
  ggtitle('Primary classifier')

## permutation p-value
permTS(value ~ cms,
       df %>% filter(classifier == 'CMS (primary classifier)',
                     cms %in% c('CMS1', 'CMS2')),
       method = 'exact.mc', control = permControl(nmc = 10^4))

p2 <- df %>% filter(classifier != 'CMS (primary classifier)') %>%
  ggplot(aes(cms, value)) +
  stat_summary(fun.y = 'mean', geom = 'bar', fill = '#cccccc') +
  geom_jitter(color = '#444444', width = 0.2) +
```

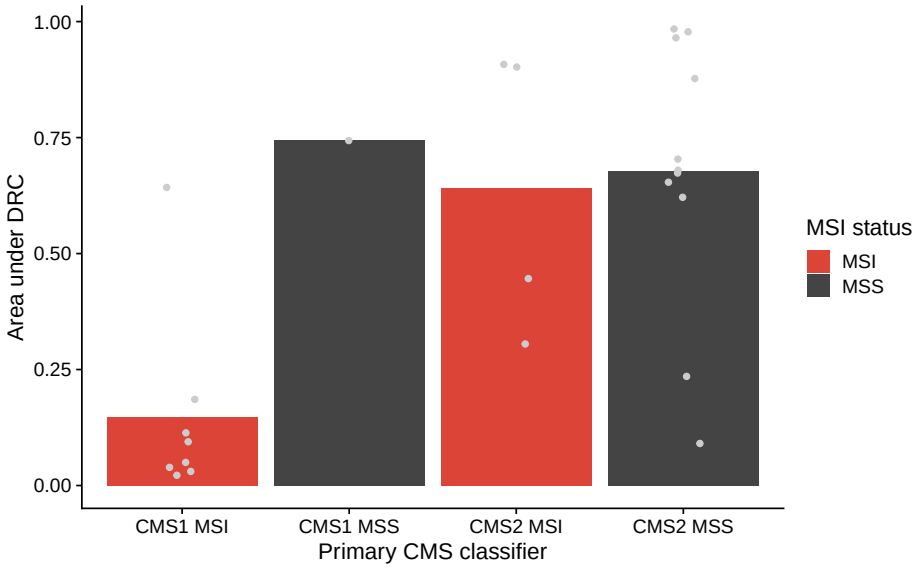
```
  ggsignif::geom_signif(comparisons = list(c('CMS1/4', 'CMS2/3')),
                        step_increase = 0.1) +
  cowplot::theme_cowplot() +
  xlab('') + ylab('Area under DRC') +
  ggtitle('Preclinical classifier')

permTS(value ~ cms,
       df %>% filter(classifier != 'CMS (primary classifier)',
                     cms %in% c('CMS1/4', 'CMS2/3')),
       method = 'exact.mc', control = permControl(nmc = 10^4))

resp_auc %>% filter(name == 'YM155') %>%
  dplyr::select(cellline, value, msi_status, cms, cms_preclin) %>%
  gather(classifier, cms, cms, cms_preclin) %>%
  filter(classifier == 'cms_preclin') %>%
  ggplot(aes(cms, value)) +
  stat_summary(fun.y = 'mean', geom = 'bar', fill = '#cccccc') +
  geom_jitter(color = '#444444', width = 0.2) +
  ggsignif::geom_signif(comparisons = list(c('CMS1/4', 'CMS2/3')),
                        step_increase = 0.1) +
  cowplot::theme_cowplot() +
  xlab('') + ylab('Area under DRC') +
  ggtitle('Preclinical classifier')
```




```
p1 + p2 + plot_layout(ncol = 1)
```



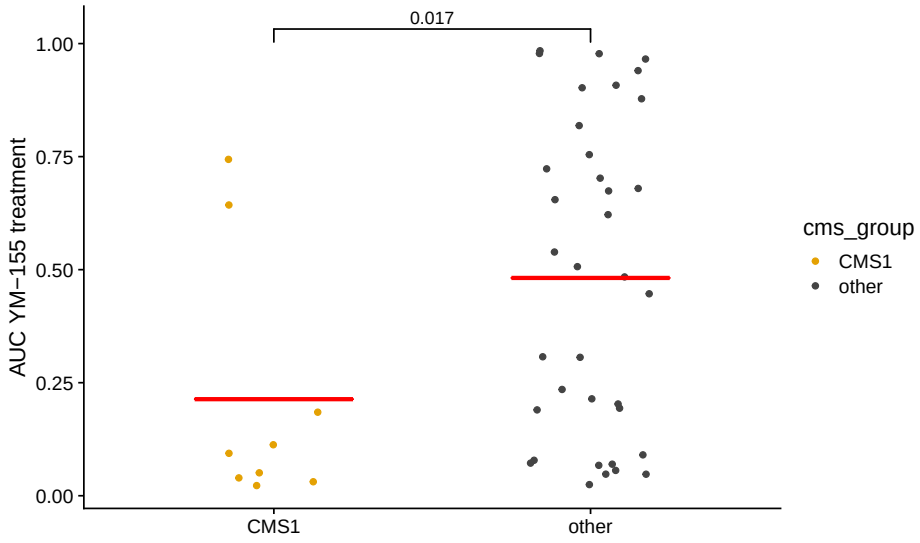

```
resp_auc %>% filter(cms %in% c('CMS1', 'CMS2'), name == 'YM155') %>%
  mutate(cms_msi = paste(cms, msi_status)) %>%
  ggplot(aes(cms_msi, value)) +
  stat_summary(fun.y = 'mean', geom='bar', position = 'dodge', aes(fill = msi_status)) +
  geom_jitter(width = 0.1, color = '#cccccc') +
  cowplot::theme_cowplot() +
  scale_fill_manual(values = c('#db4437', '#444444')) +
  labs(fill = 'MSI status') +
  xlab('Primary CMS classifier') + ylab('Area under DRC')
```



We compare YM-155 efficiency in correlation to other features in order to determine if CMS1 is best.

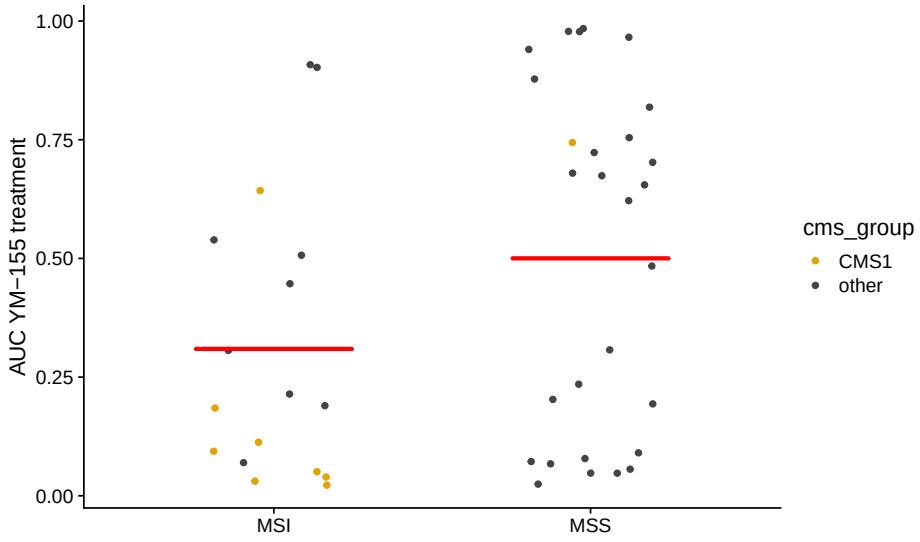
```
ym155_response <- resp_auc %>% filter(name == 'YM155')

## plots
```

```
## cms1 vs all other
ym155_response %>% mutate(cms_group = ifelse(cms == 'CMS1', 'CMS1', 'other')) %>%
  ggplot(aes(cms_group, value, color = cms_group)) + geom_jitter(width = 0.2) +
  stat_summary(fun = 'mean', fun.min = 'mean',
               fun.max = 'mean', color = 'red',
               width = 0.5, geom = 'crossbar') +
  ggsignif::geom_signif(comparisons = list(c('CMS1', 'other')),
                        color = 'black') +
  cowplot::theme_cowplot() +
  scale_color_manual(values=c('#e4a103', '#444444')) +
  ylab('AUC YM-155 treatment') + xlab('')
#> Warning: Removed 5 rows containing non-finite values (stat_summary).
#> Warning: Removed 5 rows containing non-finite values (stat_signif).
#> Warning: Removed 5 rows containing missing values (geom_point).
```




```
permTS(value ~ cms_group,
       ym155_response %>% mutate(cms_group = ifelse(cms == 'CMS1', 'CMS1', 'other')),
       method = 'exact.mc', control = permControl(nmc = 10^4))
#>
#>  Exact Permutation Test Estimated by Monte Carlo
#>
#> data:  value by cms_group
#> p-value = 0.0314
#> alternative hypothesis: true mean cms_group=CMS1 - mean cms_group=other is not equal to 0
#> sample estimates:
#> mean cms_group=CMS1 - mean cms_group=other
#>                                -0.268176
#>
#> p-value estimated from 10000 Monte Carlo replications
#> 99 percent confidence interval on p-value:
#>  0.02517840 0.03816264

## msi vs mss
```

```
ym155_response %>% mutate(cms_group = ifelse(cms == 'CMS1', 'CMS1', 'other')) %>%
  ggplot(aes(msi_status, value, color = cms_group)) +
  geom_jitter(width = 0.2) +
  stat_summary(fun = 'mean', fun.min = 'mean',
               fun.max = 'mean', color = 'red',
               width = 0.5, geom = 'crossbar') +
  ggsignif::geom_signif(comparisons = list(c('MSI', 'MSS')), color = 'black',
                        test = 'permTS',
                        test.args = list(method = "exact.mc",
                                         control = permControl(nmc=10^4))) +
  cowplot::theme_cowplot() +
  scale_color_manual(values=c('#e4a103', '#444444')) +
  ylab('AUC YM-155 treatment') + xlab('')
#> Warning: Removed 5 rows containing non-finite values (stat_summary).
#> Warning: Removed 5 rows containing non-finite values (stat_signif).
#> Warning: Computation failed in `stat_signif()`:
#> missing value where TRUE/FALSE needed
#> Warning: Removed 5 rows containing missing values (geom_point).
```




```
permTS(value ~ msi_status,
       ym155_response %>% mutate(cms_group = ifelse(cms == 'CMS1', 'CMS1', 'other')) %>%
         filter(msi_status %in% c('MSI', 'MSS')),
       method = 'exact.mc', control = permControl(nmc = 10^4))
#>
#>  Exact Permutation Test Estimated by Monte Carlo
#>
#> data:  value by msi_status
#> p-value = 0.06839
#> alternative hypothesis: true mean msi_status=MSI - mean msi_status=MSS is not equal to 0
#> sample estimates:
#> mean msi_status=MSI - mean msi_status=MSS
#>                                -0.1906729
#>
```

```
#> p-value estimated from 10000 Monte Carlo replications
#> 99 percent confidence interval on p-value:
#>  0.05919600 0.07810324
```

8.1.1 CCLE screens

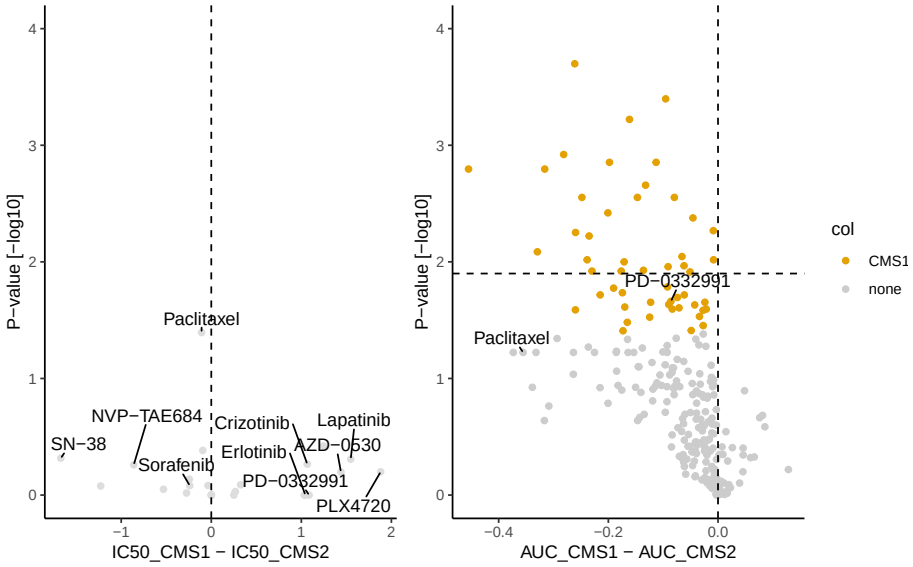
Similar screens to the above were performed by the Cancer Cell Line Encyclopedia consortium. We hence repeat the above analysis for these data.

```
data('resp_ic50_ccle', package = 'CMSYM1552018')

## primary tumour classifier
resp_ic50_ccle <- resp_ic50_ccle %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  nest(-drug) %>%
  mutate(test = map(data, ~ permTS(ic50 ~ cms, data=.x, method="exact.mc",
                                   control=permControl(nmc=10^4)))) %>%
  mutate(res = map(test, ~ tibble(p.value = .x$p.value,
                                  delta_auc = .x$estimate))) %>%
  unnest(res) %>% mutate(fdr = p.adjust(p.value, method='BH')) %>%
  arrange(delta_auc)
```

Here we do not find any siginificant differences. We make a plot that shows overlapping drugs from both screening projects next to each other.

```
## map CCLE drug names to cancerXgene
drug_id_map <- tribble(
  ~drug_ccle, ~name,
  'Irinotecan', 'SN-38',
  'TAE684', 'NVP-TAE684',
  'LBW242', 'LBW242',
  'PD-0325901', 'PD-0325901',
  'AZD6244', 'AZD6244',
  'TKI258', 'TKI258',
  'Topotecan', 'Topotecan',
  'L-685458', 'L-685458',
  'Sorafenib', 'Sorafenib',
  'Paclitaxel', 'Paclitaxel',
  'Panobinostat', 'Panobinostat',
  'Erlotinib', 'Erlotinib',
  '17-AAG', '17-AAG',
  'RAF265', 'RAF265',
  'PHA-665752', 'PHA-665752',
  'Nutlin-3', 'Nutlin-3a',
  'Nilotinib', 'Nilotinib',
  'AEW541', 'AEW541',
  'ZD-6474', 'ZD-6474',
  'PF2341066', 'Crizotinib',
  'PD-0332991', 'PD-0332991',
  'AZD0530', 'AZD-0530',
  'Lapatinib', 'Lapatinib',
```

```
    'PLX4720', 'PLX4720'
  )

  ccle_drug_result <- resp_ic50_ccle %>%
    dplyr::select(drug_ccle = drug, everything()) %>%
    left_join(drug_id_map) %>%
    mutate(in_cosmic = ifelse(name %in% drug_results_cms12_primary$name, T, F))

  ## plot volcano
  ccle_p1 <- ccle_drug_result %>%
    mutate(col=ifelse(fdr < 0.2 & delta_auc > 0, 'CMS2',
                ifelse(fdr < 0.2 & delta_auc < 0, 'CMS1', 'none'))) %>%
    mutate(label = ifelse(in_cosmic,
                          as.character(name), '')) %>%
    ggplot(aes(delta_auc, -log10(p.value))) + geom_point(colour = '#dddddd') +
    geom_vline(xintercept=0, linetype = 'dashed') +
    geom_text_repel(aes(label=label), nudge_y = 0.1, colour='black',
                    min.segment.length = 0) +
    scale_colour_manual(values = c('#e4a103', '#cccccc')) +
    ylim(c(0,4)) +
    xlab('IC50_CMS1 - IC50_CMS2') + ylab('P-value [-log10]') +
    theme(legend.position = 'none') +
    theme_classic()

  fdr20 <- drug_results_cms12_primary %>%
    filter(fdr > 0.2) %>% arrange(p.value) %>%
    .$p.value %>% .[1] %>% log10() %>% `*`(-1)
  ccle_p2 <- drug_results_cms12_primary %>%
    mutate(col=ifelse(fdr < 0.2 & delta_auc > 0, 'CMS2',
                ifelse(fdr < 0.2 & delta_auc < 0, 'CMS1', 'none')),
           in_cosmic = ifelse(name %in% ccle_drug_result$name, T, F)) %>%
    mutate(label = ifelse(in_cosmic,
                          as.character(name), '')) %>%
    ggplot(aes(delta_auc, -log10(p.value), colour=col)) + geom_point() +
    geom_vline(xintercept=0, linetype = 'dashed') +
    geom_hline(yintercept = 1.9, linetype='dashed') +
    geom_text_repel(aes(label=label), nudge_y = 0.1, colour='black',
                    min.segment.length = 0) +
    scale_colour_manual(values = c('#e4a103', '#cccccc')) +
    ylim(c(0,4)) +
    xlab('AUC_CMS1 - AUC_CMS2') + ylab('P-value [-log10]') +
    theme(legend.position = 'none') +
    theme_classic()

  ccle_p1 + ccle_p2
```



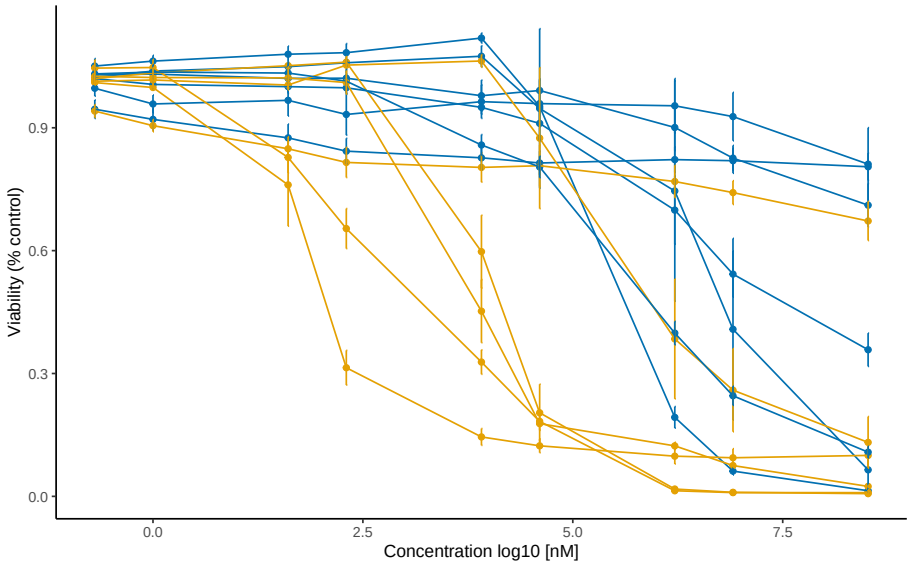
8.2 Validation of YM-155

We load the data from proliferation assays that we performed to validate the YM155 effect experimentally in additional cell lines. We define a function that can plot a dose-response curve to use as a figure. We finally define a function that draws a jitter plot comparing area-under-the curve values of cell lines of CMS1 versus CMS2.

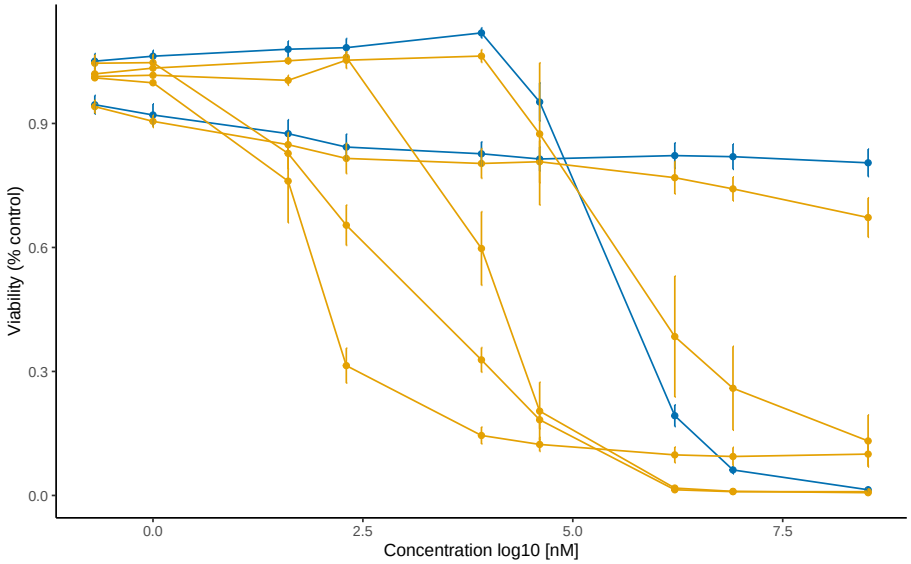
```
cell_lines <- c('HCT116', 'RKO', 'LIM2405', 'WiDr', 'LoVo',
                'SNU-C2A',
                'CaCo2', 'HT55', 'GP2d', 'HCA-24', 'CL-14', 'SW403', 'CCK-81')
start_row <- c(3, 18, 33, 48, 63, 78, 93, 108, 123, 138, 153, 168, 183)

cms1 <- c('RKO', 'HCT116', 'LoVo', 'SNU-C2A', 'WiDr', 'LIM2405')
cms2 <- c('HT55', 'CaCo', 'CL-14', 'SW403', 'GP2d', 'HCA-24', 'CCK-81')

plot_prolif <- function(df){
  df %>% ggplot(aes(conc_log, vmean, colour=CMS, group=cellline)) +
    geom_point() + geom_line() +
    geom_errorbar(aes(ymin = ymin, ymax = ymax), width = 0) + theme_classic() +
    xlab('Concentration log10 [nM]') + ylab('Viability (% control)') +
    scale_colour_manual(values = c('#e4a103','#0270b1')) +
    theme(legend.position = 'none')
}

compare_plot <- function(df){
  df %>% group_by(cellline, CMS) %>% arrange(conc_log) %>%
    mutate(conc_rank = 1:n()) %>%
    summarise(AUC = pracma::trapz(conc_rank, vmean)) %>% ungroup() %>%
    ggplot(aes(CMS, AUC)) + geom_jitter(width=0.2) +
    stat_summary(fun.y = 'mean', fun.ymin = 'mean',
                 fun.ymax = 'mean', geom='crossbar',
                 width =0.5, colour = 'red') + theme_classic() +
    ggsignif::geom_signif(comparisons = list(c('CMS1', 'CMS2')),
```

```
                              test = 't.test',
                              test.args = list(var.equal = T))
    }
```

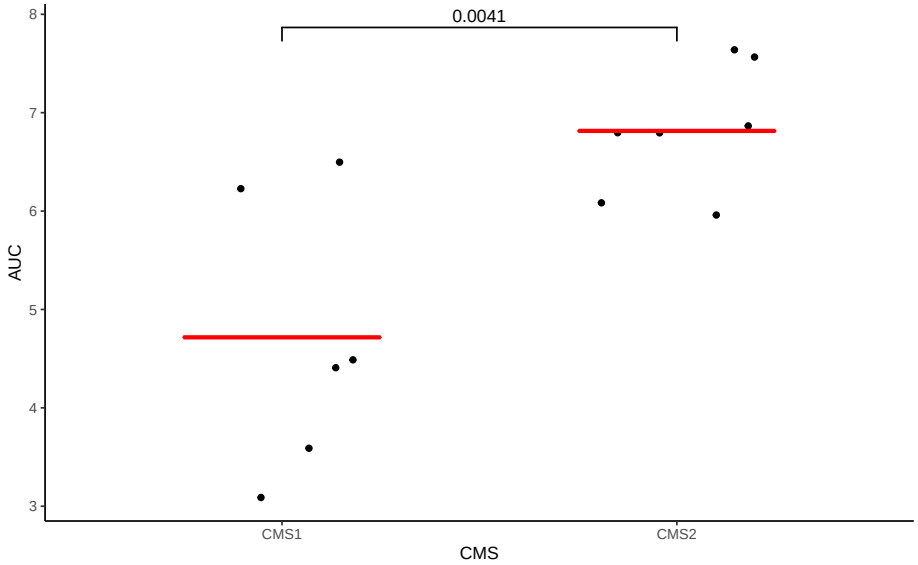
```
    data('ym155', package='CMSYM1552018')

    plot_prolif(ym155)
```




```
    msi_lines <- c('HCT116', 'RKO', 'LIM2405', 'LoVo', 'SNU-C2A',
                   'GP2d', 'CCK-81')
    plot_prolif(ym155 %>% filter(cellline %in% msi_lines))
```

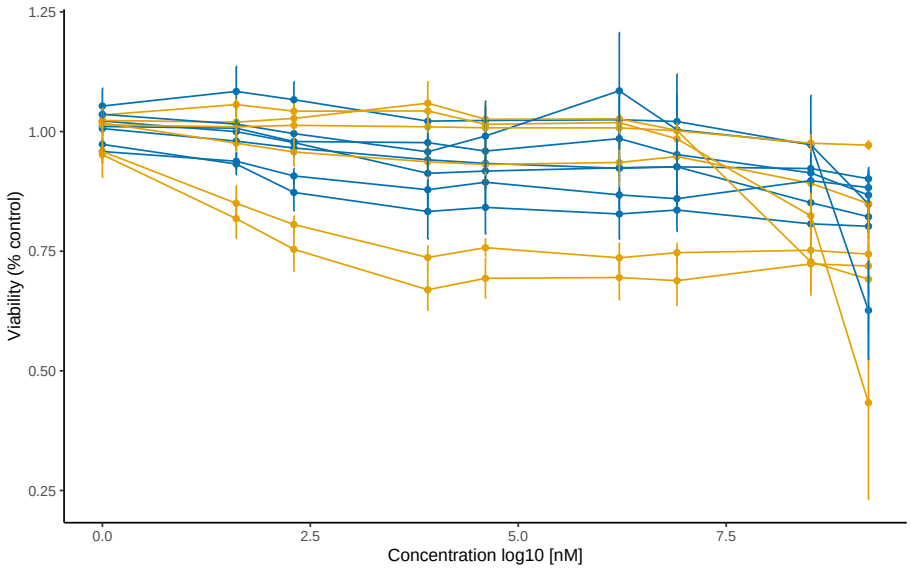


```
compare_plot(ym155)
```

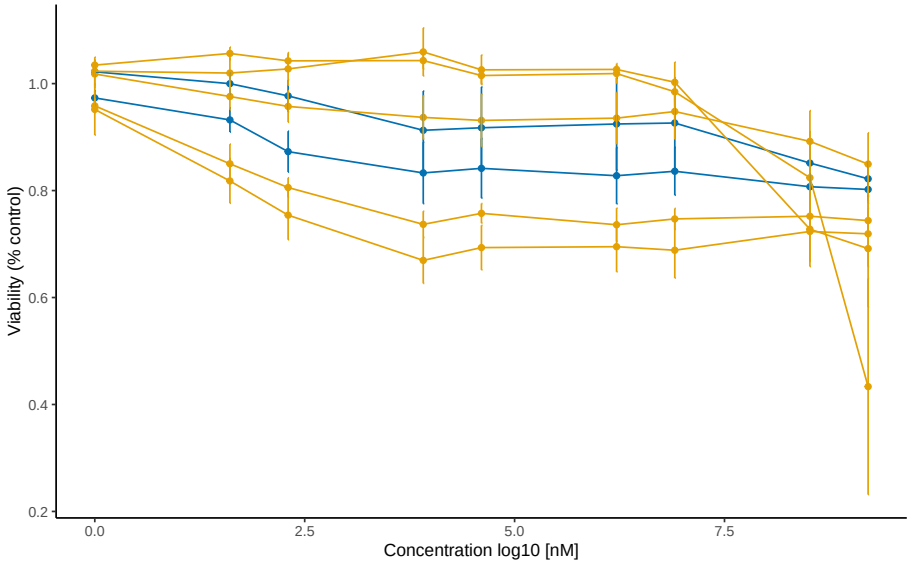


8.3 Validation of 5-FU

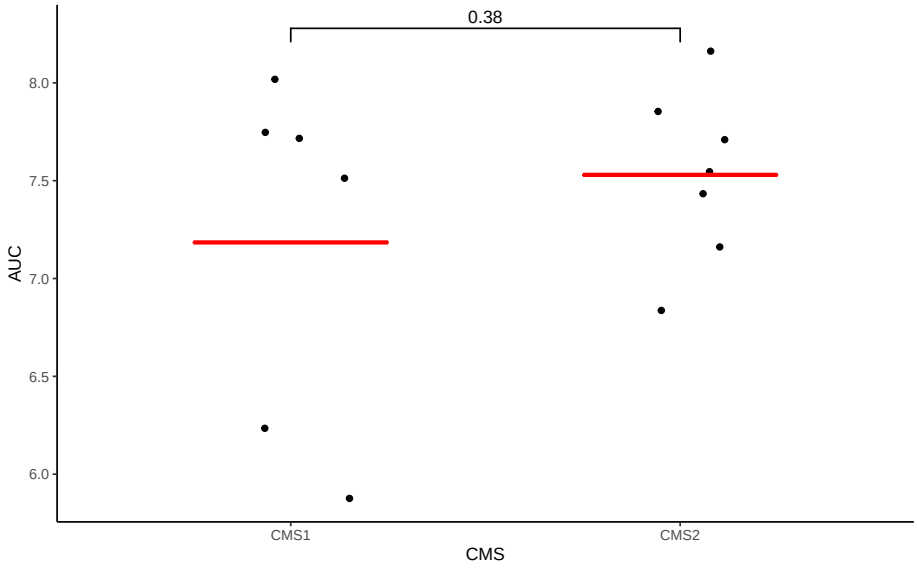
```
data('five_fu', package='CMSYM1552018')

plot_prolif(five_fu)
```




```
plot_prolif(five_fu %>% filter(cellline %in% msi_lines))
```

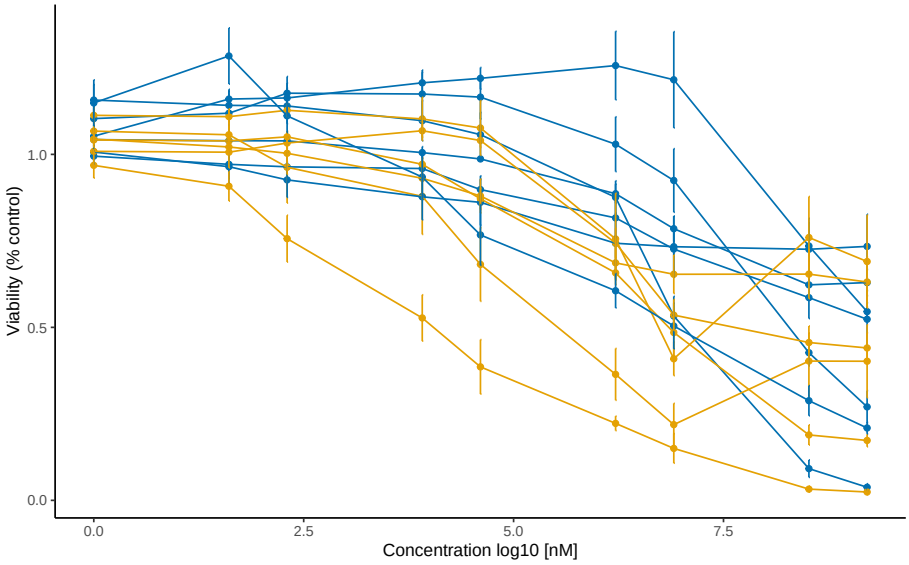



```
compare_plot(five_fu)
```

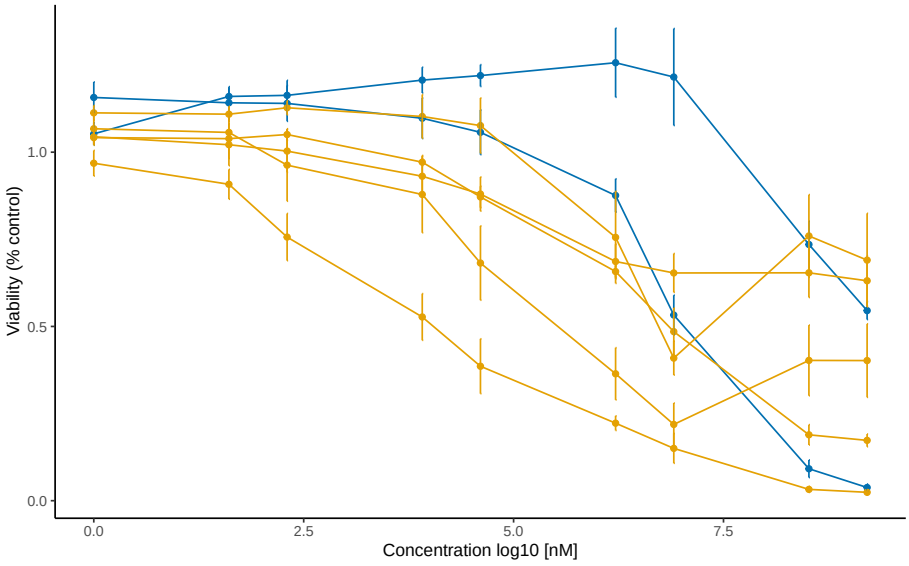


8.4 Validation of SN38 (Irinotecan)

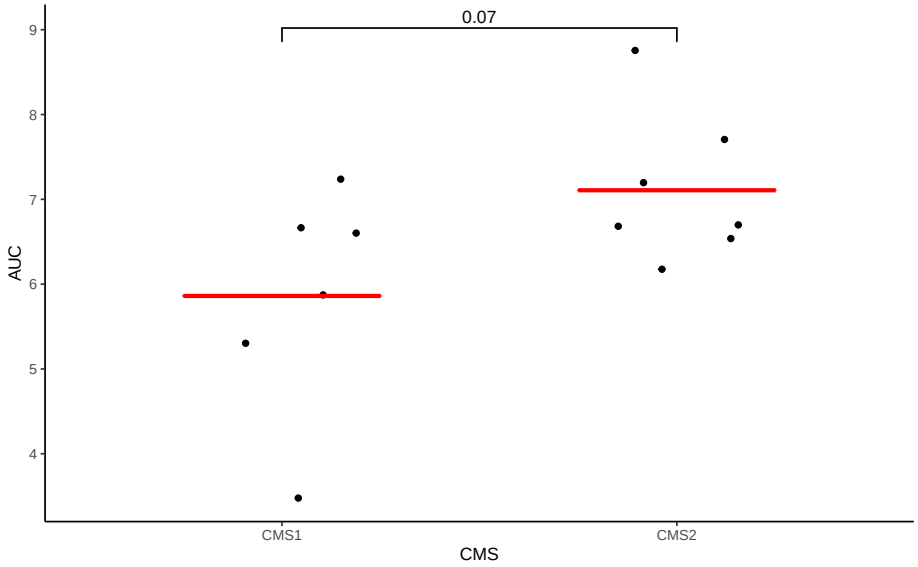
```
data('sn38', package='CMSYM1552018')

plot_prolif(sn38)
```




```
plot_prolif(sn38 %>% filter(cellline %in% msi_lines))
```




```
compare_plot(sn38)
```



8.5 Validation of cell line CMS

In order to confirm that the cell lines we analyzed in our lab have the right CMS we performed microarray experiments using Affymetrix arrays. We load data that was normalized as described above.

```
data('exprs_cms_val', package='CMSYM1552018')
```

We analyze CMS status as above.

```
exprs_val <- get_cms_gene_lvl2(exprs_cms_val, 'Affy HG U133-PLUS-2 probeset') %>%
  mutate(dataset='CCLE', platform = 'HG_U133_Plus2',
         sample = gsub('.CEL', '', sample))

cms_val <- acast(exprs_val, entrez ~ sample, value.var = 'expr') %>%
  as.data.frame() %>%
  classifyCMS.RF(minPosterior = 0.35)
```

The CMS of all cell lines can be confirmed if a slightly more relaxed posterior probability threshold of 0.35 is applied. There are no cell lines for which we observe a class label that disagrees with our previous results.

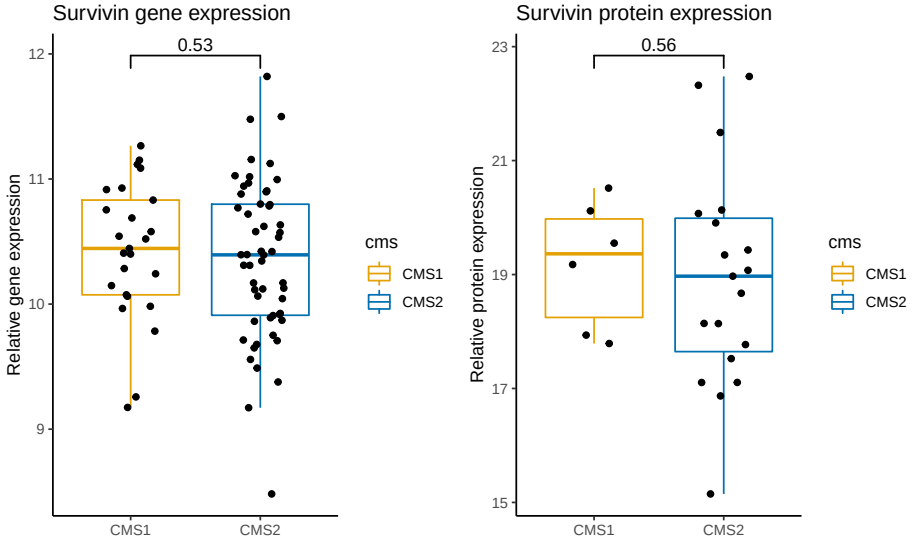
9 YM-155 induces apoptosis independent of Survivin

9.1 Gene and protein expression of Survivin

Protein expression data are derrived from Frejno et al. (Frejno et al. 2017).

```
## gene expression of surivin across CMS1 and CMS2
survivin_gene <- medico_for_cms %>% filter(entrez == 332) %>% inner_join(cms_cl) %>%
  filter(cms %in% c('CMS1', 'CMS2')) %>%
  ggplot(aes(cms, expr)) + geom_boxplot(aes(colour=cms), fill = NA,
```

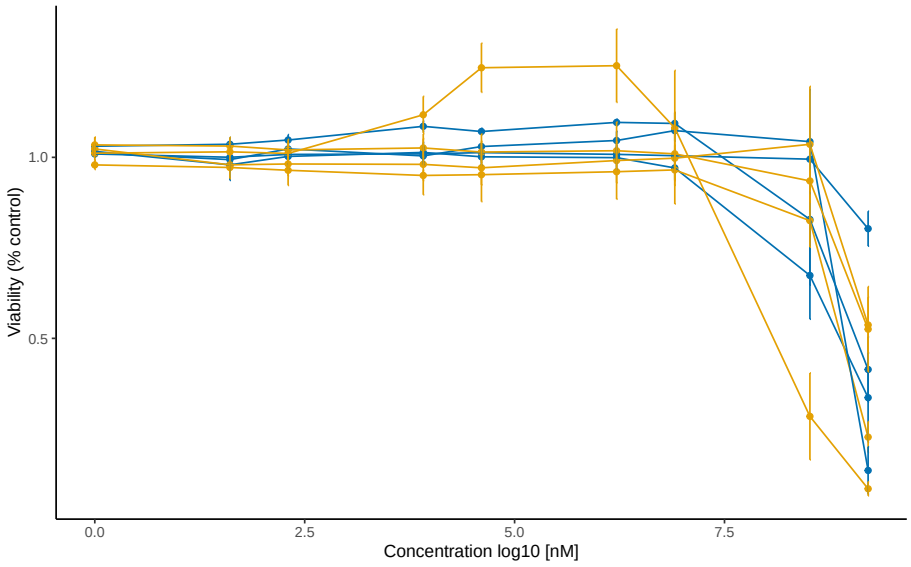
```
                                        outlier.colour = '#ffffff') +
  geom_jitter(width=0.2) +
  scale_colour_manual(values = c('#e4a103','#0270b1')) +
  ylab('Relative gene expression') + xlab('') +
  ggsignif::geom_signif(comparisons = list(c('CMS1', 'CMS2'))) +
  ggtitle('Survivin gene expression') +
  theme(legend.position = 'none')

## protein expression Survivin vs YM155 response
data('protein_expr', package='CMSYM1552018')

## protein expression Survivin vs YM155 response
survivin_protein <- protein_expr %>% filter(symbol == 'BIRC5') %>%
  mutate(prexpr=as.numeric(prexpr)) %>%
  inner_join(cms_cl) %>% filter(cms %in% c('CMS1', 'CMS2')) %>%
  drop_na() %>% ggplot(aes(cms, prexpr)) +
  geom_boxplot(aes(colour=cms), outlier.colour = '#ffffff') +
  geom_jitter(width=0.2) +
  scale_colour_manual(values = c('#e4a103','#0270b1')) +
  ylab('Relative protein expression') + xlab('') +
  ggsignif::geom_signif(comparisons = list(c('CMS1', 'CMS2'))) +
  ggtitle('Survivin protein expression') +
  theme(legend.position = 'none')

## plot both with patchwork
survivin_gene + theme_classic() +
  survivin_protein + theme_classic()
```



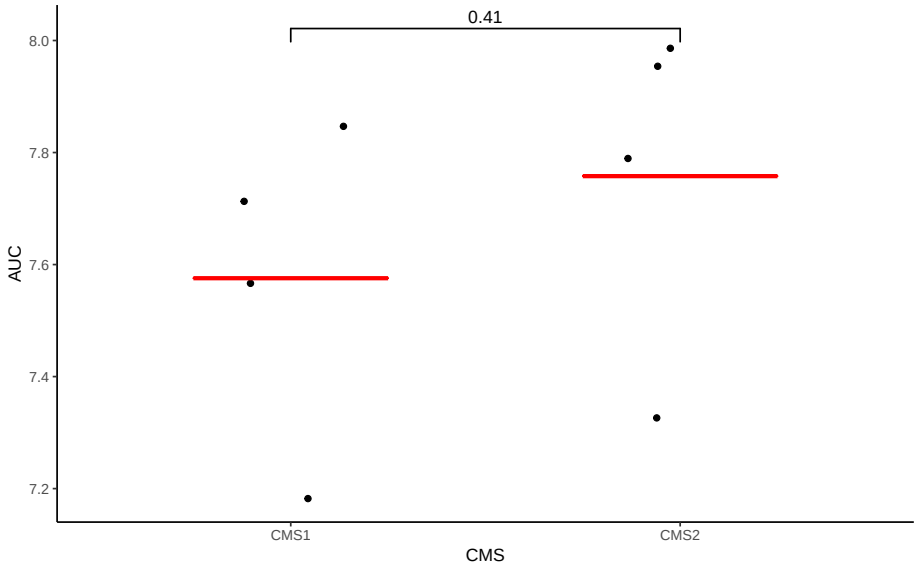
9.2 Validation of Navitoclax

Proliferation assays were performed similar to above for YM155, 5-FU and SN-38. We load the data and plot dose response curves.

```
data('navitoclax', package='CMSYM1552018')

plot_prolif(navitoclax)
```




```
compare_plot(navitoclax)
```



9.3 Differential expression after YM-155 inhibition

To assess whether there are transcriptomic changes associated with YM-155 inhibtion we treated cells with 100 ul of YM-155 and measured gene expression.

9.3.1 Loading array data

The data were generated using an Illumina BeadChip array. We use the lumi package (Du, Kibbe, and Lin 2008) to process the expression measurements. We start by loading the raw data in form of a lumi batch object.

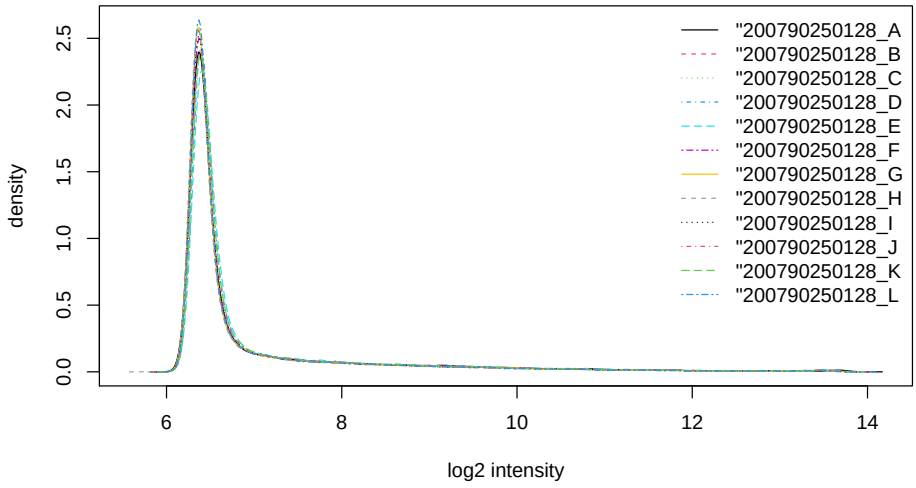
```
data('lumi_raw', package='CMSYM1552018')
data('probe_mapping_ilmn', package='CMSYM1552018')
```

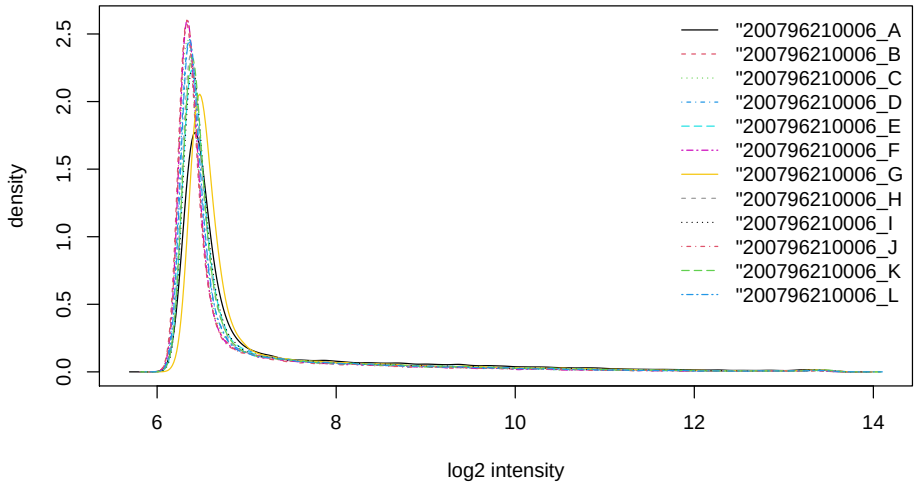
Lumi comes with a number of quality control features that we can use to determine microarray quality. First we report some summary statistics.

```
walk(lumi_raw, summary, 'QC')
```

Next we generate sample density plots.

```
walk(lumi_raw, density)
```





9.3.2 Normalization

We use RMA and quantile normalization to normalize the data.

```
lumi_norm <- lumiExpresso(combine(lumi_raw[[1]], lumi_raw[[2]]))
```

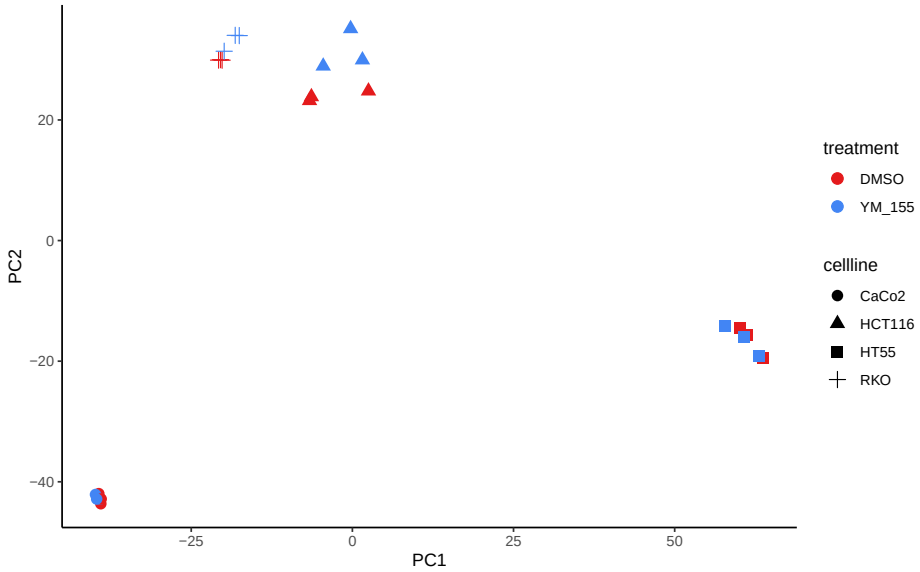
9.3.3 Quality control

By PCA and clustering we want to understand whether the normalized data looks reasonable. First we annotate the sample columns for easy interpretation.

```
## load sample info to annotate samples
data('sample_sheet', package='CMSYM1552018')

## extract expression matrix
lumi_norm_exprs <- exprs(lumi_norm)

## set sample names
sn_ind <- match(paste0('"', sample_sheet$sample_id), colnames(lumi_norm_exprs))
colnames(lumi_norm_exprs) <- sample_sheet$Sample_Name[sn_ind]
```

Then we run a PCA on the data and plot the first two principal components.

```
pca <- lumi_norm_exprs %>% t() %>% prcomp()
pca$x %>% as_tibble(rownames='sample') %>%
  separate(sample, c('cellline', 'replicate', 'treatment'), sep=' ', remove=F) %>%
  ggplot(aes(PC1, PC2, colour=treatment, shape=cellline)) +
  geom_point(size=3) + theme_classic() +
  scale_colour_manual(values = c('#E41A1C', '#4285f4'))
```



This looks as expected. CMS1 cell lines are closer together than to the CMS 2 cell lines. Differences between cell lines dominate the YM155 treatment which we would expect. Nevertheless differences can be seend between treated and untreated samples and these seem to be somewhat larger for the CMS1 lines.

9.3.4 Differential gene expression analysis

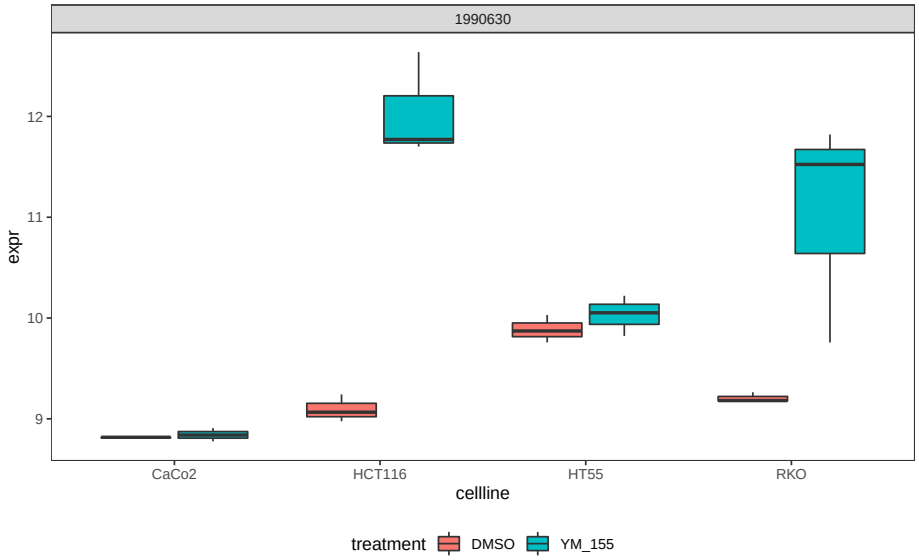
We now use limma (Ritchie et al. 2015) to find genes that are differentially regulated upon YM155 treatment in CMS1 and CMS2 cell lines.

```
## sample annotation for expression matrix columns
sample_anno <- sample_sheet %>%
  separate(Sample_Name, c('cellline', 'replicate', 'treatment'), sep=' ', remove=F) %>%
  mutate(cms = ifelse(cellline %in% c('CaCo2', 'HT55'), 'CMS2', 'CMS1')) %>%
  mutate(treatment_cms = ifelse(treatment == 'DMSO', 'DMSO',
                          ifelse((treatment == 'YM_155') & (cms == 'CMS1'), 'YM155_CMS1',
                                 'YM155_CMS2')))

## check that sample anno matches to expr matrix
identical(colnames(lumi_norm_exprs), sample_anno$Sample_Name)

## define model matrix
mm <- model.matrix(~ cellline + treatment_cms, data = sample_anno)

fit <- lmFit(lumi_norm_exprs, mm)
fit_eb <- eBayes(fit)

## results for cms1
dge_res_cms1 <- topTable(fit_eb, coef = 5, n=Inf) %>%
  as_tibble(rownames='ProbeID') %>%
  left_join(probe_mapping_ilmn) %>%
  filter(!is.na(Symbol))

## results for cms2
dge_res_cms2 <- topTable(fit_eb, coef = 6, n=Inf) %>%
  as_tibble(rownames='ProbeID') %>%
  left_join(probe_mapping_ilmn) %>%
  filter(!is.na(Symbol))
```

To explore the results we first generate a function that can plot expression levels for all probes targeting a gene of interest across different groups.

```
expr_long <- lumi_norm_exprs %>% as_tibble(rownames='ProbeID') %>%
  left_join(probe_mapping_ilmn) %>%
  gather(sample, expr, -c(ProbeID, Entrez_Gene_ID, Symbol)) %>%
  filter(!is.na(Symbol)) %>%
  separate(sample, c('cellline', 'replicate', 'treatment'), sep=' ', remove=F)

plot_gene_expr <- function(symbol){
  expr_long %>% filter(Symbol == symbol) %>%
    ggplot(aes(cellline, expr, fill = treatment)) +
    geom_boxplot() + facet_wrap(~ProbeID) +
    theme_bw() + theme(panel.grid = element_blank(), legend.position = 'bottom')
}

## Plot gene expression of TRIB3
## - a key player in ER-stress induced apoptosis
```

```
plot_gene_expr('TRIB3')
```



We find no differentially expressed genes in CMS2 samples why nice signal is present for CMS1. It seems from the data that especially ER-stress induced apopotosis and NFkappaB signaling are induced by the YM-155 treatment. We can confirm this by gene set enrichment analysis (Sergushichev 2016).

```
## gene ontology annotation
go_ids <- as_tibble(as.data.frame(GOTERM)[,-1]) %>%
  filter(Ontology == 'BP') %>% distinct(go_id, Term)
src <- src_organism('TxDb.Hsapiens.UCSC.hg38.knownGene')
go_map <- tbl(src, 'id_go') %>% filter(ontology == 'BP') %>%
  distinct(entrez, go) %>% collect(n=Inf) %>%
  inner_join(go_ids %>% dplyr::select(go = go_id, term = Term)) %>%
  dplyr::select(-go) %>% split(.$term) %>% map(~ .x$entrez)

## ranked list
ranks <- setNames(dge_res_cms1$t, dge_res_cms1$Entrez_Gene_ID) %>%
  sort(decreasing=T)
ranks2 <- setNames(dge_res_cms2$t, dge_res_cms2$Entrez_Gene_ID) %>%
  sort(decreasing=T)
fgsea_res_go <- fgsea(go_map, ranks, nperm=1e5, maxSize=500) %>%
  as_tibble() %>% arrange(pval)
fgsea_res_cms2 <- fgsea(go_map, ranks2, nperm=1e4, maxSize=500) %>%
  as_tibble() %>% arrange(pval)
```

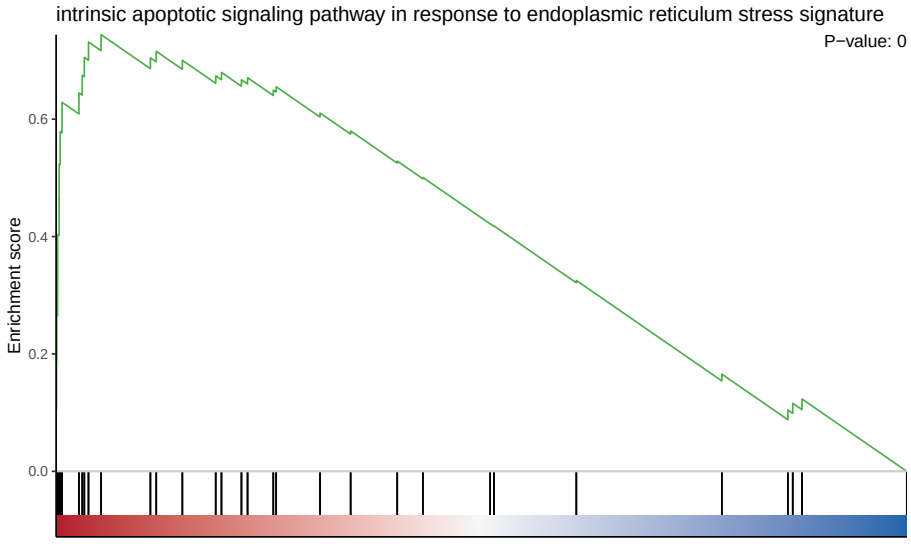
We define a custom function that can draw barcode plots to visualize gene set enrichment.

```
pretty_barcode_plot <- function(stat_vector, sig){
  require(fgsea)
  ## genes in signature
  sig_genes <- sig[[1]]
  if(NA %in% stat_vector){
    warning('Removing NAs from ranked gene list')
```

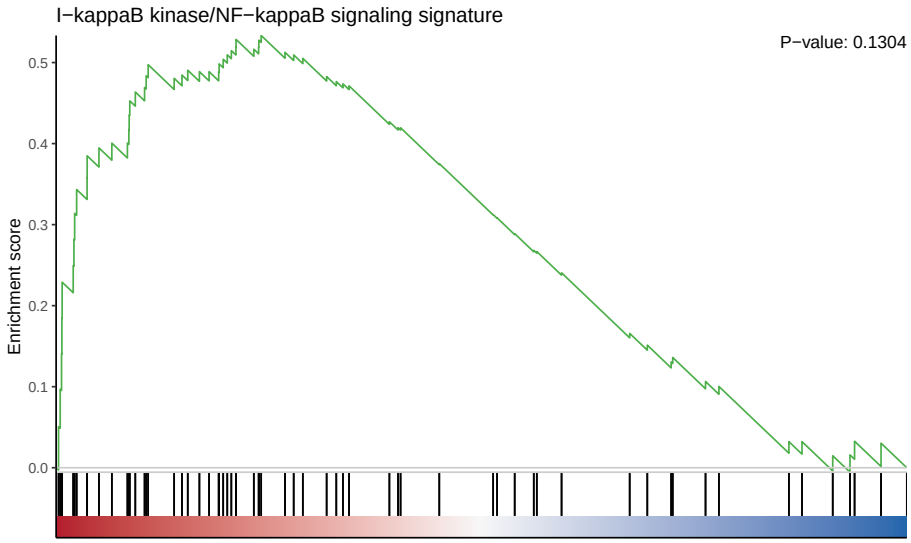
```
    stat_vector <- stat_vector[!is.na(stat_vector)]
  }

  ## 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(stat_vector))

  ## p-value and negative enrichment score
  pval <- limma::cameraPR(stat_vector, sig[[1]])$PValue

  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(paste(names(sig)[1], 'signature')) +
    ylab('Enrichment score') +
    annotate('text', x = Inf, y = Inf, hjust=1, vjust = 1,
             label = paste('P-value:', round(pval, 4)))

  return(bc_plot_custom)
}
```

We draw barcode plots for the mentioned processes of interest. We further draw a volcano plot that can globally visualize expression changes upon YM-155 treatment selecting the probe with the strongest fold change to represent each gene.

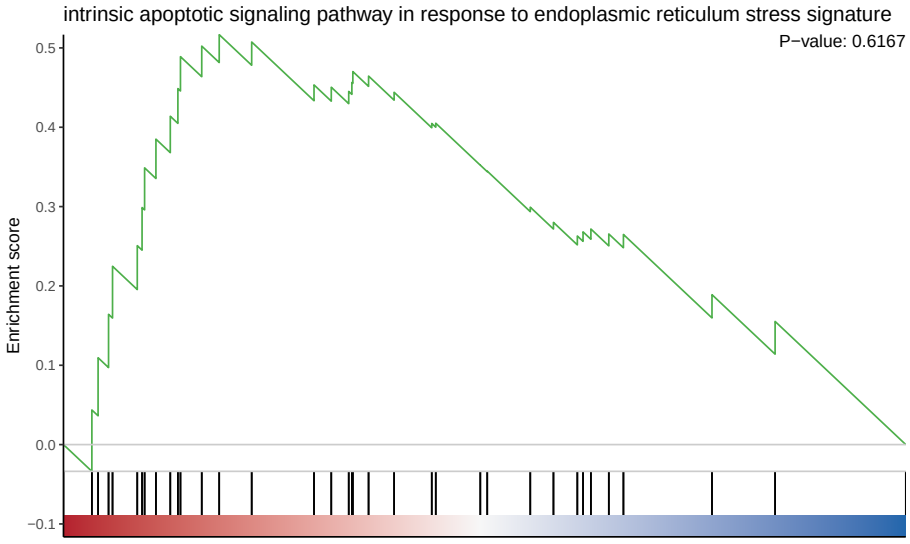
```
## CMS1
pretty_barcode_plot(ranks, go_map['intrinsic apoptotic signaling pathway in response to endoplasmic reticulu
```



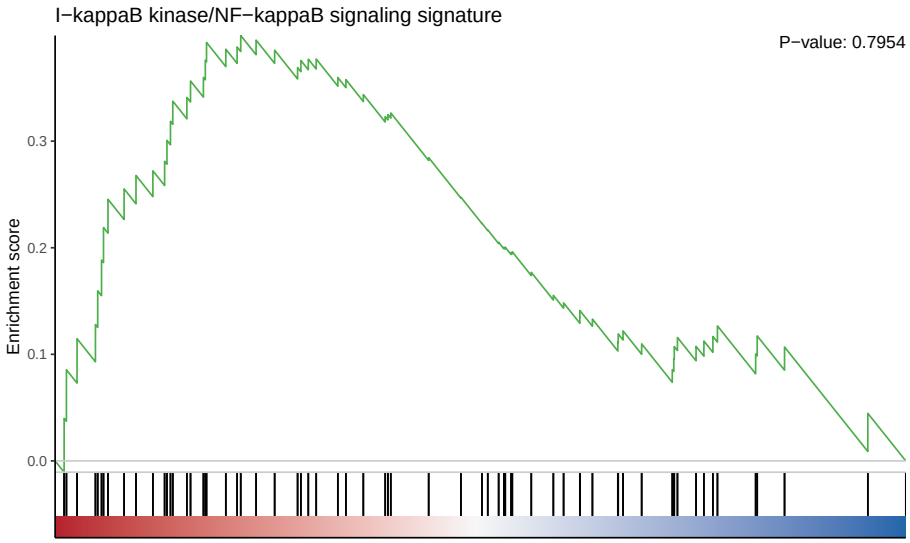

```
pretty_barcode_plot(ranks, go_map['I-kappaB kinase/NF-kappaB signaling'])
```




```
## CMS2
pretty_barcode_plot(ranks2, go_map['intrinsic apoptotic signaling pathway in response to endoplasmic reticulu
```

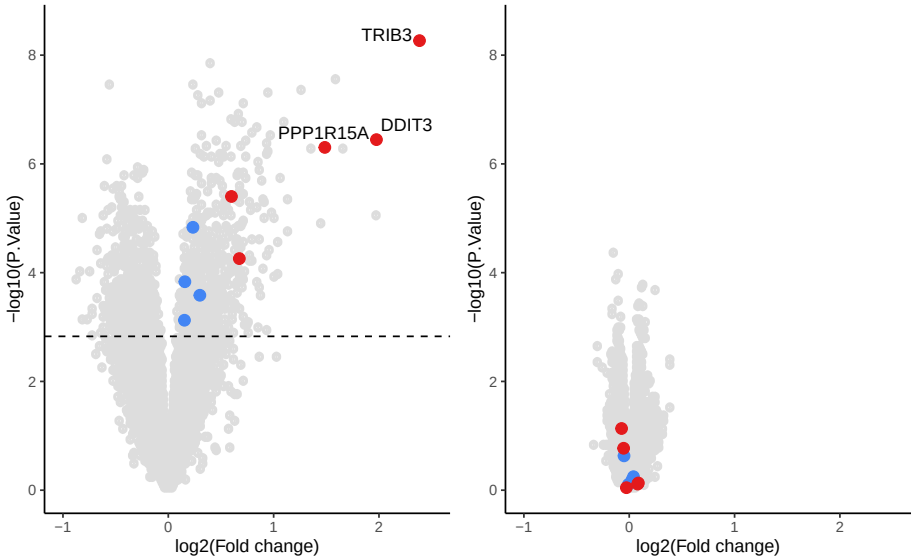



```
pretty_barcode_plot(ranks2, go_map['I-kappaB kinase/NF-kappaB signaling'])
```




```
bc_plot <- function(df, plot_names=T){
  if(plot_names){
    chop <- c('TIRB3', 'DDIT3', 'TNFRSF10B', 'ATF4', 'PPP1R15A', 'TRIB3')
    nfkb <- c('NFKB1', 'IKBKB', 'NFKB2', 'NKIRAS2')
  } else {
    genes <- c()
  }

  df <- df %>% group_by(Symbol) %>% arrange(desc(abs(t))) %>%
    dplyr::slice(1) %>% ungroup() %>%
    mutate(label = ifelse(Symbol %in% c(chop, nfkb), Symbol, ''),
           highlight = ifelse(label %in% chop, 'CHOP',
                      ifelse(label %in% nfkb, 'NFKB', 'none')))
```

```
  fdr5 <- df %>% filter(adj.P.Val < 0.05) %>% arrange(desc(P.Value)) %>%
    pull(P.Value) %>% head(1) %>% log10() %>% `*`(-1)

  df %>% ggplot(aes(logFC, -log10(P.Value), label = label)) +
    geom_hex(data = subset(df, highlight == 'none'), colour = '#dddddd', fill='#dddddd', bins=150, size=2) +
    geom_point(data = subset(df, highlight == 'NFKB'), colour= '#4285f4', size=3) +
    geom_point(data = subset(df, highlight == 'CHOP'), colour= '#e41a1c', size=3) +
    ggrepel::geom_text_repel() +
    geom_hline(yintercept = fdr5, linetype = 'dashed') +
    theme_classic() + theme(legend.position = 'none') +
    xlab('log2(Fold change)') + ylim(c(0, 8.5)) + xlim(c(-1, 2.5))
}

p1 <- bc_plot(dge_res_cms1)
p2 <- bc_plot(dge_res_cms2)

p1 + p2
```



10 A CRISPR screen identifies resistance markers to YM155 treatment in CMS1

We generated read counts from the initial sequencing (fastq) data using CRISPRanalyzeR (Winter et al. 2017). We now load these counts into R to analyze them further.

```
data('counts', package='CMSYM1552018')
```

10.1 Quality control

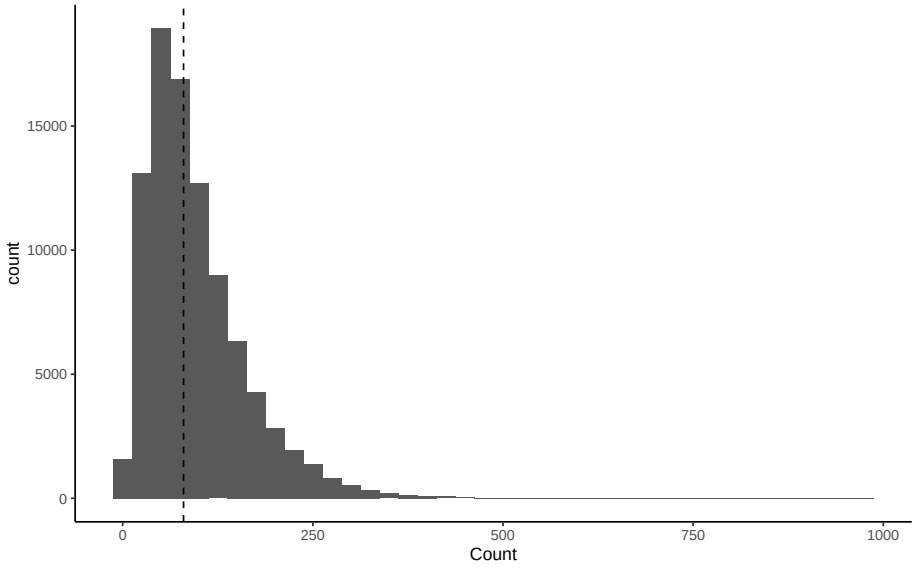
10.1.1 Sequencing depth

First we would like to assess sequencing depth for each sample and compare it to previous screens.

10.1.1.1 Time 0 sample

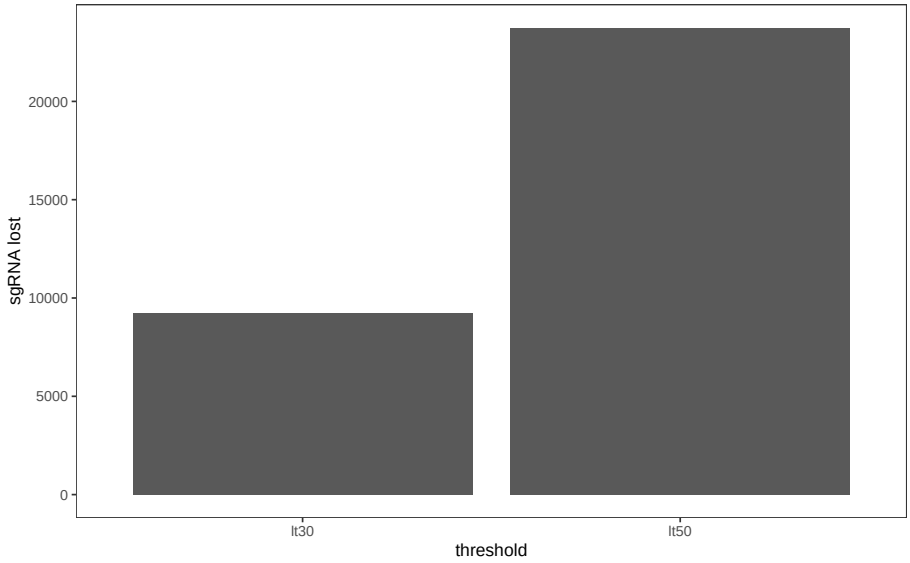
We start with the initial time point (T0).

```
## our T0 sample
median_our <- counts %>% filter(sample == 'd0screenlibrary') %>%
  .$Count %>% median()

counts %>% filter(sample == 'd0screenlibrary') %>%
  ggplot() + geom_histogram(aes(Count), bins=40) +
  geom_vline(xintercept=median_our, linetype='dashed') +
  theme_classic()
```



The sequencing depth should be sufficient to continue with the analysis of the screen.

```
counts %>%
    filter(sample %in% c('rc_initial', 'HCTwtT0', 'd0screenlibrary')) %>%
    mutate(lt30 = Count < 30, lt50 = Count < 50) %>%
    summarise(lt30 = sum(lt30), lt50 = sum(lt50)) %>%
    gather(threshold, `sgRNA lost`, lt30, lt50) %>%
    ggplot(aes(threshold, `sgRNA lost`)) +
    geom_bar(stat='identity') +
    theme_bw() + theme(panel.grid = element_blank())
```

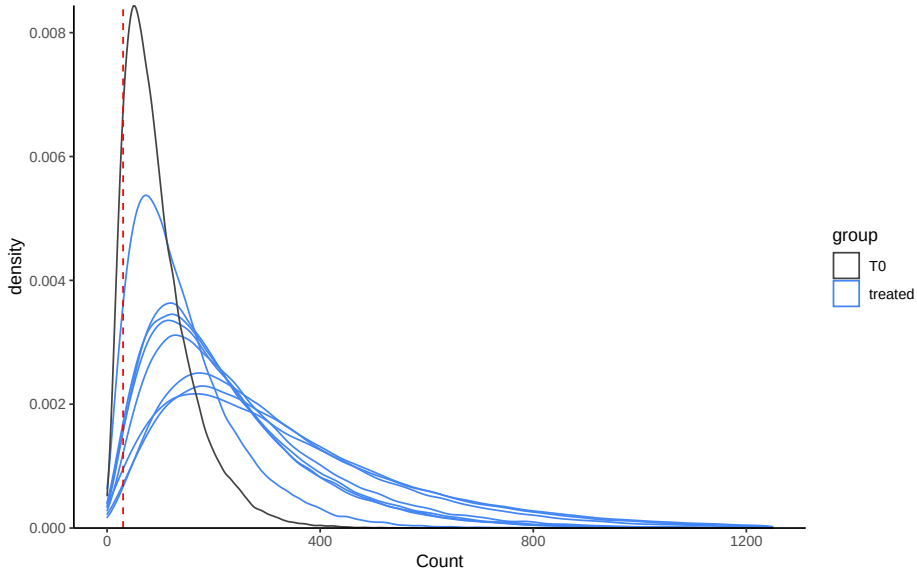


To avoid losing too many sgRNAs due to insufficient coverage we will continue with a less stringent exclusion criterium (< 30), which results in only a loss of ~10% of the sgRNAs.

10.1.1.2 All samples

We plot histograms for read count distributions of all samples.

```
counts %>%
  mutate(group = ifelse(grepl('d0', sample), 'T0', 'treated')) %>%
  ## exclude outlier counts so we can see the distribution
  filter(Count < 1250) %>%
  ggplot() + geom_density(aes(Count, group=sample, colour=group)) +
  geom_vline(xintercept = 30, colour = 'red', linetype='dashed') +
  scale_colour_manual(values=c('#444444', '#4285f4')) +
  scale_y_continuous(expand = c(0,0)) +
  theme_classic()
```



Summary statistics to sum up the sequencing.

```
counts %>%
  group_by(sample) %>%
  summarise(total = sum(Count), mean = mean(Count),
            median = median(Count)) %>% ungroup()
#> # A tibble: 9 x 4
#>   sample                total  mean median
#> * <chr>                 <int> <dbl>  <dbl>
#> 1 Screen1DMSOR1       20080262 220.     182
#> 2 Screen1DMSOR2       20872124 229.     189
#> 3 Screen1NavitoclaxR1 31602078 346.     282
#> 4 Screen1NavitoclaxR2 13021252 143.     118
#> 5 Screen2DMSOR1       22875111 250.     208
#> 6 Screen2DMSOR2       19900023 218.     180
#> 7 Screen2YM155R1      31287008 343.     284
#> 8 Screen2YM155R2      29227176 320.     267
#> 9 d0screenlibrary      8727438  95.6     80
```

10.1.2 Essential genes

We want to get an idea of how strong the phenotypes in our screen are by comparing core-essential and non-essential screens. We first use the edgeR TMM-normalization to normlize samples and the calculate fold changes for each sample, comparing later time points to the T0 samples.

```
## make sure that the T0 sample is the first in the alphabet
data_split <- list()
data_split[[1]] <- counts %>%
  separate(sgRNA, c('symbol', 'sequence'), sep='_') %>%
  mutate(sample = ifelse(grepl('d0|T0|initial', sample), 'aa', sample)) %>%
  unite(sgRNA, symbol, sequence)

## a function to calculate fold changes for each screen (replicates individually)
calc_fc <- function(df){
  ## matrix of counts
  cmat <- df %>% distinct() %>%
    spread(sample, Count) %>%
    data.frame() %>% `rownames<-`(NULL) %>%
    column_to_rownames('sgRNA')

  ## normalize with edgeR
  cmat <- calcNormFactors(DGEList(cmat)) %>% cpm(log=T)

  ## calculate fold changes and convert back to long df
  data.frame(cmat[,-1] - cmat[,1]) %>% rownames_to_column('sgRNA') %>%
    tbl_df %>% gather(sample, log2fc,  -sgRNA) %>%
    separate(sgRNA, c('symbol', 'sequence'), sep='_')
}

## apply function to calculate fc for each screen
data_norm <- data_split %>% map(calc_fc)
## exclude low coverage sgRNAs
```

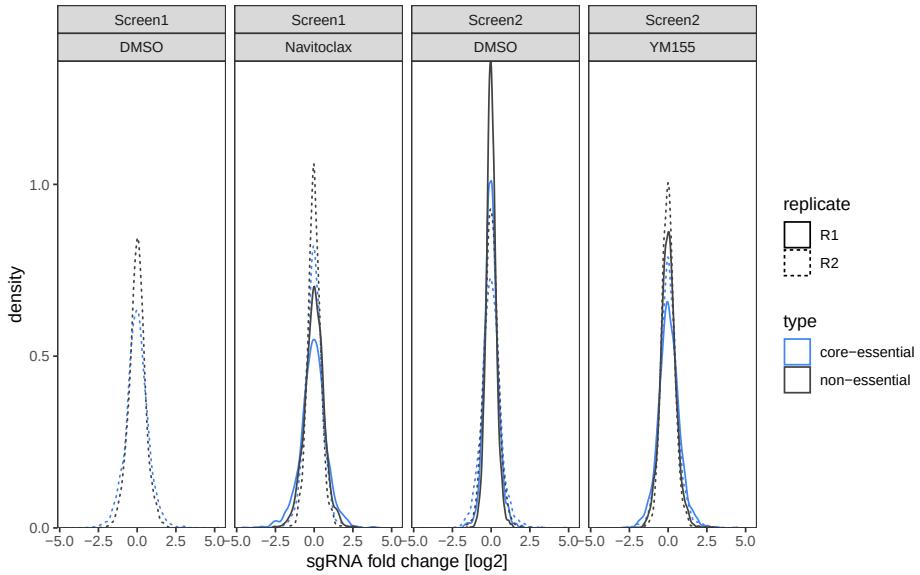
```
data_norm <- data_norm %>% bind_rows() %>%
  inner_join(distinct(counts, sample)) %>%
  anti_join(data_split %>% bind_rows() %>%
              filter(sample == 'aa', Count < 30) %>%
              distinct(sgRNA) %>%
              separate(sgRNA, c('symbol', 'sequence'), sep='_'))
```

Next we load the core- and non-essential gene reference set defined in Hart et al., 2017.

```
## core-essential genes
data('ceg', package='CMSYM1552018')
## non-essential genes
data('neg', package='CMSYM1552018')
```

Now we can compare fold changes between core-essential and non-essential genes for each sample of our screen (Hart et al. 2017).

```
data_norm %>%
  mutate(type = ifelse(symbol %in% ceg, 'core-essential',
                       ifelse(symbol %in% neg, 'non-essential', 'none'))) %>%
  filter(type != 'none') %>%
  extract(sample, c('screen', 'treatment', 'replicate'),
          regex = '(Screen\\d)(.+)(R\\d$)', remove=F) %>%
  ggplot(aes(log2fc, colour = type, linetype = replicate)) +
  geom_density() + facet_wrap(~screen + treatment, nrow=1) +
  scale_colour_manual(values = c('#4285f4', '#444444')) +
  scale_y_continuous(expand = c(0,0)) +
  xlab('sgRNA fold change [log2]') +
  theme(legend.position = 'bottom') +
  theme_bw() + theme(panel.grid = element_blank())
```



We can conclude that there are recognizable phenotypes, separating distributions of sgRNAs targeting core- and non-essential genes, respectively. This indicates that the experiment worked as intended.

10.2 Reproducibility

We next check how reproducible our phenotypes are across replicates. We plot scatter plots and annotate Pearson and Spearman correlation coefficients on log2-fold changes.

```
## disentangle sample column
ym_screen <- data_norm %>%
  extract(sample, c('screen', 'treatment', 'replicate'),
          regex='(Screen\\d)(DMSO|Navitoclax|YM155)(R\\d)')

reproducibility_plot <- function(df, sc, tr){
  df <- df %>% filter(screen == sc, treatment==tr)
  cors <- df %>% spread(replicate, log2fc) %>%
    summarise(pcc = cor(R1, R2, method='pearson'),
              scc = cor(R1, R2, method='spearman')) %>%
    unlist() %>% round(2)

  df <- df %>% unite(experiment, screen, treatment) %>%
    spread(replicate, log2fc)

  df %>% ggplot(aes(R1, R2)) + geom_hex(bins=100) +
    geom_abline(linetype='dashed') +
    geom_hline(yintercept=0, linetype='dashed') +
    geom_vline(xintercept=0, linetype='dashed') +
    theme(legend.position='none') +
    xlab('replicate 1') + ylab('replicate 2') + ggtitle(paste(sc, tr)) +
    theme_classic()
}

## generate plots using above function
rep_plots <- ym_screen %>% distinct(screen, treatment) %>%
  rowwise() %>% do(p = reproducibility_plot(ym_screen, .$screen, .$treatment)) %>%
  ungroup() %>% .$p

## draw to canvas
purrr::reduce(rep_plots, `+`)
```

Correlation coefficients are in the same ballpark as previous studies (based on a GenomeCRISPR meta analysis). It might be worth pointing out that correlations based on log2 fold changes tend to always be considerably lower than coefficients based normalized read counts that are also often presented. In our experience, however, fold change based correlation coefficients tend to be more 'honest' measures of reproducibility. For the sake of completeness we can also show similar scatter plots based on raw counts.

```
our_counts <- counts %>% filter(sample != 'd0screenlibrary') %>%
  extract(sample, c('screen', 'treatment', 'replicate'),
          regex='(Screen\\d)(.*)(R\\d)$') %>%
  ## we need to do this so the reproducibility plot will work, still counts though
```

```
  dplyr::select(log2fc=Count, everything())

our_counts %>% distinct(screen, treatment) %>%
  drop_na() %>% rowwise() %>%
  do(p = reproducibility_plot(our_counts, .$screen, .$treatment)) %>%
  ungroup() %>% .$p %>% purrr::reduce(`+`)
```

10.3 Hit calling

Next we would like to call hits between treated and untreated samples. For each sgRNA in the screen we would like to compare its abundance in the DMSO-treated sample compared to the sample that was treated with the drug of interest. For this purpose we use the MAGeCK software (version 0.5.7) (Li et al. 2014) that implements a negative binomial model to compare abundance of raw counts.

10.3.1 YM155

First we need to create an input file for mageck. This is a tab separated file that contains and sgRNA column, a gene column and additional columsn for each sample.

```
## YM155
ym155_raw <- counts %>% separate(sgRNA, c('symbol', 'sequence'), sep='_') %>%
  filter(sequence %in% data_norm$sequence) %>%
  filter(sample != 'd0screenlibrary',
         grepl('Screen2', sample)) %>%
  dplyr::select(sequence, symbol, everything()) %>%
  spread(sample, Count)

## write mageck input file
ym155_raw %>% write_tsv('YM155.txt')
```

Next we run mageck on the sample file we just generated.

```
system('mageck test -k YM155.txt -t 2,3 -c 0,1 -n YM155')
```

We then load the results back into R.

```
data('ym_results', package='CMSYM1552018')
```

10.3.2 Navitoclax

For the Navitoclax screen, as above, we need to create an input file for mageck. This is a tab separated file that contains and sgRNA column, a gene column and additional columns for each sample.

```
## YM155
navito_raw <- counts %>% separate(sgRNA, c('symbol', 'sequence'), sep='_') %>%
  filter(sequence %in% data_norm$sequence) %>%
  filter(sample != 'd0screenlibrary',
         grepl('Screen1', sample)) %>%
  dplyr::select(sequence, symbol, everything()) %>%
```

```
  spread(sample, Count)

## write mageck input file
navito_raw %>% write_tsv('Navitoclax.txt')
```

Next we run MAGeCK on the sample file we just generated.

```
system('mageck test -k Navitoclax.txt -t 2,3 -c 0,1 -n Navitoclax')
```

We can then read the results file back into R.

```
data('navito_results', package='CMSYM1552018')
```

10.4 Visualization of results

We can now visualize the results in the form of two volcano plots. Mageck performs two sets of tests: one for positive and one for negative enrichment. Separate plots have to be generated for each of those.

10.4.1 YM155

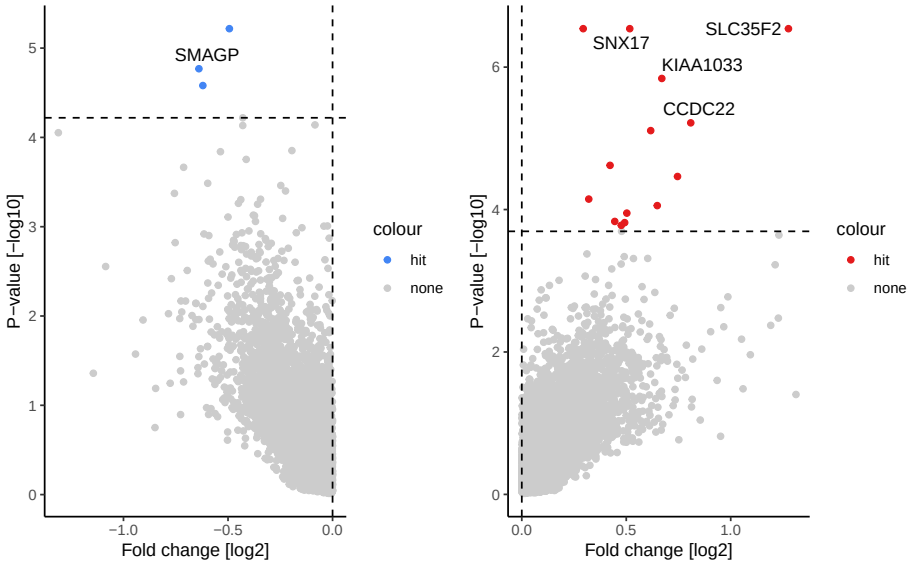
We start with the negative enrichment.

```
mageck_volcano <- function(df, type='negative', highlight=c()){
  ## hit colour
  hit_col <- ifelse(type == 'negative', '#4285f4', '#e41a1c')

  ## fdr 20 cutoff
  fdr20 <- df %>% filter(fdr > 0.2) %>%
    arrange(p.value) %>% .$p.value %>% .[1] %>% log10() %>% `*`(-1)

  ## draw volcano
  df %>% mutate(label = ifelse(symbol %in% highlight, symbol, ''),
                colour = ifelse(fdr < 0.2 & abs(log2fc) > 0.25,
                                'hit', 'none'),
                type = type) %>%
    filter(ifelse(type == 'negative', log2fc < 0, log2fc > 0)) %>%
    ggplot(aes(log2fc, -log10(p.value))) +
    geom_point(aes(colour = colour)) +
    ggrepel::geom_text_repel(aes(label=label)) +
    geom_vline(xintercept=0, linetype = 'dashed') +
    geom_hline(yintercept = fdr20, linetype = 'dashed') +
    scale_colour_manual(values = c(hit_col, '#cccccc')) +
    theme(legend.position = 'none') +
    xlab('Fold change [log2]') + ylab('P-value [-log10]') +
    theme_classic()
}

## list of candidates selected form the Screen
candidates_neg <- c('SMAGP')
candidates_pos <- c('SLC35F2', 'CCDC22', 'SNX17', 'KIAA1033')
```

```
## generate plot
ym_negative <- ym_results %>%
  dplyr::select(symbol=id, log2fc = `neg|lfc`,
                p.value = `neg|p-value`, fdr = `neg|fdr`) %>%
  mageck_volcano(type = 'negative', highlight = candidates_neg)
```

We now add a plot for the positive enrichment and combine them using `patchwork`.

```
## generate plot
ym_positive <- ym_results %>%
  dplyr::select(symbol=id, log2fc = `pos|lfc`,
                p.value = `pos|p-value`, fdr = `pos|fdr`) %>%
  mageck_volcano(type = 'positive', highlight = candidates_pos)

## combine
ym_negative + ym_positive
```



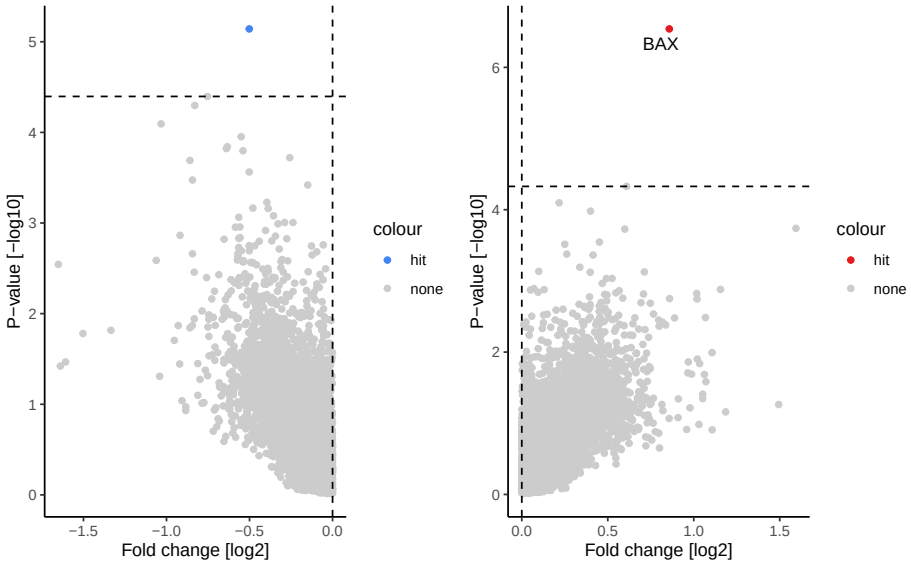
10.4.2 Navitoclax

We proceed as above with the Navitoclax Screen.

```
candidates_pos <- c('BAX')

navito_positive <- navito_results %>%
  dplyr::select(symbol=id, log2fc = `pos|lfc`,
                p.value = `pos|p-value`, fdr = `pos|fdr`) %>%
  mageck_volcano(type = 'positive', highlight = candidates_pos)
```

```
candidates_neg <- c()

navito_negative <- navito_results %>%
  dplyr::select(symbol=id, log2fc = `neg|lfc`,
```

```
                  p.value = `neg|p-value`, fdr = `neg|fdr`) %>%
    mageck_volcano(type = 'negative', highlight = candidates_neg)

## plot
navito_negative + navito_positive
```



11 Session info

```
sessionInfo()
#> R version 4.0.3 (2020-10-10)
#> Platform: x86_64-apple-darwin17.0 (64-bit)
#> Running under: macOS Big Sur 10.16
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.0/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] stats4    parallel  stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] bindrcpp_0.2.2          Rcpp_1.0.6              impute_1.64.0
#>  [4] sva_3.38.0              BiocParallel_1.24.1     genefilter_1.72.1
#>  [7] mgcv_1.8-34             nlme_3.1-152            edgeR_3.32.1
#> [10] Organism.dplyr_1.18.0   AnnotationFilter_1.14.0 GO.db_3.12.1
#> [13] AnnotationDbi_1.52.0    IRanges_2.24.1          S4Vectors_0.28.1
#> [16] fgsea_1.16.0            limma_3.46.0            lumi_2.42.0
```

```
#> [19] readxl_1.3.1            patchwork_1.1.1         perm_1.0-0.0
#> [22] ggrepel_0.9.1           CMScaller_0.99.2        CMSclassifier_1.0.0
#> [25] randomForest_4.6-14     ggsignif_0.6.1          reshape2_1.4.4
#> [28] openxlsx_4.2.3          pheatmap_1.0.12         preprocessCore_1.52.1
#> [31] GEOquery_2.58.0         affy_1.68.0             Biobase_2.50.0
#> [34] BiocGenerics_0.36.0     forcats_0.5.1           stringr_1.4.0
#> [37] dplyr_1.0.4             purrr_0.3.4             readr_1.4.0
#> [40] tidyr_1.1.2             tibble_3.0.6            ggplot2_3.3.3
#> [43] tidyverse_1.3.0         BiocStyle_2.18.1
#>
#> loaded via a namespace (and not attached):
#>   [1] utf8_1.1.4
#>   [2] tidyselect_1.1.0
#>   [3] RSQLite_2.2.3
#>   [4] grid_4.0.3
#>   [5] munsell_0.5.0
#>   [6] codetools_0.2-18
#>   [7] nleqslv_3.3.2
#>   [8] withr_2.4.1
#>   [9] colorspace_2.0-0
#>  [10] TxDb.Hsapiens.UCSC.hg38.knownGene_3.10.0
#>  [11] knitr_1.31
#>  [12] rstudioapi_0.13
#>  [13] labeling_0.4.2
#>  [14] MatrixGenerics_1.2.1
#>  [15] GenomeInfoDbData_1.2.4
#>  [16] farver_2.0.3
#>  [17] bit64_4.0.5
#>  [18] rhdf5_2.34.0
#>  [19] vctrs_0.3.6
#>  [20] generics_0.1.0
#>  [21] xfun_0.21
#>  [22] BiocFileCache_1.14.0
#>  [23] R6_2.5.0
#>  [24] GenomeInfoDb_1.26.2
#>  [25] illuminaio_0.32.0
#>  [26] locfit_1.5-9.4
#>  [27] bitops_1.0-6
#>  [28] rhdf5filters_1.2.0
#>  [29] cachem_1.0.4
#>  [30] reshape_0.8.8
#>  [31] DelayedArray_0.16.1
#>  [32] assertthat_0.2.1
#>  [33] scales_1.1.1
#>  [34] gtable_0.3.0
#>  [35] methylumi_2.36.0
#>  [36] rlang_0.4.10
#>  [37] splines_4.0.3
#>  [38] rtracklayer_1.50.0
#>  [39] lazyeval_0.2.2
#>  [40] hexbin_1.28.2
```

```
#>  [41] broom_0.7.5
#>  [42] BiocManager_1.30.10
#>  [43] yaml_2.2.1
#>  [44] modelr_0.1.8
#>  [45] GenomicFeatures_1.42.1
#>  [46] backports_1.2.1
#>  [47] tools_4.0.3
#>  [48] bookdown_0.21
#>  [49] nor1mix_1.3-0
#>  [50] affyio_1.60.0
#>  [51] ellipsis_0.3.1
#>  [52] RColorBrewer_1.1-2
#>  [53] siggenes_1.64.0
#>  [54] plyr_1.8.6
#>  [55] sparseMatrixStats_1.2.1
#>  [56] progress_1.2.2
#>  [57] zlibbioc_1.36.0
#>  [58] RCurl_1.98-1.2
#>  [59] prettyunits_1.1.1
#>  [60] openssl_1.4.3
#>  [61] cowplot_1.1.1
#>  [62] bumphunter_1.32.0
#>  [63] SummarizedExperiment_1.20.0
#>  [64] haven_2.3.1
#>  [65] fs_1.5.0
#>  [66] magrittr_2.0.1
#>  [67] data.table_1.14.0
#>  [68] reprex_1.0.0
#>  [69] matrixStats_0.58.0
#>  [70] hms_1.0.0
#>  [71] evaluate_0.14
#>  [72] xtable_1.8-4
#>  [73] XML_3.99-0.5
#>  [74] mclust_5.4.7
#>  [75] gridExtra_2.3
#>  [76] compiler_4.0.3
#>  [77] biomaRt_2.46.3
#>  [78] minfi_1.36.0
#>  [79] KernSmooth_2.23-18
#>  [80] crayon_1.4.1
#>  [81] htmltools_0.5.1.1
#>  [82] lubridate_1.7.9.2
#>  [83] DBI_1.1.1
#>  [84] dbplyr_2.1.0
#>  [85] MASS_7.3-53.1
#>  [86] rappdirs_0.3.3
#>  [87] Matrix_1.3-2
#>  [88] cli_2.3.1
#>  [89] quadprog_1.5-8
#>  [90] bindr_0.1.1
#>  [91] GenomicRanges_1.42.0
```

```
#>  [92] pkgconfig_2.0.3
#>  [93] GenomicAlignments_1.26.0
#>  [94] xml2_1.3.2
#>  [95] foreach_1.5.1
#>  [96] annotate_1.68.0
#>  [97] rngtools_1.5
#>  [98] multtest_2.46.0
#>  [99] beanplot_1.2
#> [100] XVector_0.30.0
#> [101] rvest_0.3.6
#> [102] doRNG_1.8.2
#> [103] scrime_1.3.5
#> [104] digest_0.6.27
#> [105] pracma_2.3.3
#> [106] Biostrings_2.58.0
#> [107] rmarkdown_2.7
#> [108] base64_2.0
#> [109] cellranger_1.1.0
#> [110] fastmatch_1.1-0
#> [111] DelayedMatrixStats_1.12.3
#> [112] curl_4.3
#> [113] Rsamtools_2.6.0
#> [114] lifecycle_1.0.0
#> [115] jsonlite_1.7.2
#> [116] Rhdf5lib_1.12.1
#> [117] askpass_1.1
#> [118] fansi_0.4.2
#> [119] pillar_1.5.0
#> [120] lattice_0.20-41
#> [121] fastmap_1.1.0
#> [122] httr_1.4.2
#> [123] survival_3.2-7
#> [124] glue_1.4.2
#> [125] zip_2.1.1
#> [126] iterators_1.0.13
#> [127] bit_4.0.4
#> [128] stringi_1.5.3
#> [129] HDF5Array_1.18.1
#> [130] blob_1.2.1
#> [131] org.Hs.eg.db_3.12.0
#> [132] memoise_2.0.0
```

References

Barretina, Jordi, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, et al. 2012. "The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity." *Nature* 483 (7391): 603–7. https://doi.org/10.1038/nature11003.

Du, Pan, Warren A Kibbe, and Simon M Lin. 2008. "lumi: a pipeline for processing Illumina microarray." *Bioinformatics (Oxford, England)* 24 (13): 1547–8. https://doi.org/10.1093/bioinformatics/btn224.

Edgar, Ron, Michael Domrachev, and Alex E Lash. 2002. "Gene Expression Omnibus: NCBI gene expression and hybridization array data repository." *Nucleic Acids Research* 30 (1): 207–10. http://www.ncbi.nlm.nih.gov/pubmed/11752295%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC99122.

Forbes, Simon A, David Beare, Prasad Gunasekaran, Kenric Leung, Nidhi Bindal, Harry Boutselakis, Minjie Ding, et al. 2015. "COSMIC: exploring the world's knowledge of somatic mutations in human cancer." *Nucleic Acids Research* 43 (Database issue): D805–11. https://doi.org/10.1093/nar/gku1075.

Frejno, Martin, Riccardo Zenezini Chiozzi, Mathias Wilhelm, Heiner Koch, Runsheng Zheng, Susan Klaeger, Benjamin Ruprecht, et al. 2017. "Pharmacoproteomic characterisation of human colon and rectal cancer." *Molecular Systems Biology* 13 (11): 951. https://doi.org/10.15252/msb.20177701.

Garnett, Mathew J., Elena J. Edelman, Sonja J. Heidorn, Chris D. Greenman, Anahita Dastur, King Wai Lau, Patricia Greninger, et al. 2012. "Systematic identification of genomic markers of drug sensitivity in cancer cells." *Nature* 483 (7391): 570–75. https://doi.org/10.1038/nature11005.

Gautier, L., L. Cope, B. M. Bolstad, and R. A. Irizarry. 2004. "affy–analysis of Affymetrix GeneChip data at the probe level." *Bioinformatics* 20 (3): 307–15. https://doi.org/10.1093/bioinformatics/btg405.

Guinney, Justin, Rodrigo Dienstmann, Xin Wang, Aurélien de Reyniès, Andreas Schlicker, Charlotte Soneson, Laetitia Marisa, et al. 2015. "The consensus molecular subtypes of colorectal cancer." *Nature Medicine* 21 (11): 1350–6. https://doi.org/10.1038/nm.3967.

Hart, Traver, Amy Hin Yan Tong, Katie Chan, Jolanda Van Leeuwen, Ashwin Seetharaman, Michael Aregger, Megha Chandrashekhar, et al. 2017. "Evaluation and Design of Genome-Wide CRISPR/SpCas9 Knockout Screens." *G3 (Bethesda, Md.)* 7 (8): 2719–27. https://doi.org/10.1534/g3.117.041277.

Iorio, Francesco, Theo A. Knijnenburg, Daniel J. Vis, Graham R. Bignell, Michael P. Menden, Michael Schubert, Nanne Aben, et al. 2016. "A Landscape of Pharmacogenomic Interactions in Cancer." *Cell* 166 (3): 740–54. https://doi.org/10.1016/j.cell.2016.06.017.

Irizarry, R. A., Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. 2003. "Exploration, normalization, and summaries of high density oligonucleotide array probe level data." *Biostatistics* 4 (2): 249–64. https://doi.org/10.1093/biostatistics/4.2.249.

Li, Wei, Han Xu, Tengfei Xiao, Le Cong, Michael I Love, Feng Zhang, Rafael A Irizarry, Jun S Liu, Myles Brown, and X Shirley Liu. 2014. "MAGeCK enables robust identification of essential genes from genome-scale CRISPR/Cas9 knockout screens." *Genome Biology* 15 (12): 554. https://doi.org/10.1186/s13059-014-0554-4.

Linnekamp, Janneke F., Sander R. van Hooff, Pramudita R. Prasetyanti, Raju Kandimalla, Joyce Y. Buikhuisen, Evelyn Fessler, Prashanthi Ramesh, et al. 2018. "Consensus molecular subtypes of colorectal cancer are recapitulated in in vitro and in vivo models." *Cell Death & Differentiation* 25 (3): 616–33. https://doi.org/10.1038/s41418-017-0011-5.

Medico, Enzo, Mariangela Russo, Gabriele Picco, Carlotta Cancelliere, Emanuele Valtorta, Giorgio Corti, Michela Buscarino, et al. 2015. "The molecular landscape of colorectal cancer cell lines unveils clinically actionable kinase targets." *Nature Communications* 6 (1): 7002. https://doi.org/10.1038/ncomms8002.

Meyers, Robin M, Jordan G Bryan, James M McFarland, Barbara A Weir, Ann E Sizemore, Han Xu, Neekesh V Dharia, et al. 2017. "Computational correction of copy number effect improves specificity of CRISPR–Cas9 essentiality screens in cancer cells." *Nature Genetics* 49 (12): 1779–84. https://doi.org/10.1038/ng.3984.

Rhodes, Daniel R, Jianjun Yu, K Shanker, Nandan Deshpande, Radhika Varambally, Debashis Ghosh, Terrence Barrette, Akhilesh Pandey, and Arul M Chinnaiyan. n.d. "ONCOMINE: a cancer microarray database and integrated data-mining platform." *Neoplasia (New York, N.Y.)* 6 (1): 1–6. http://www.ncbi.nlm.nih.gov/pubmed/15068665%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1635162.

Ritchie, Matthew E, Belinda Phipson, Di Wu, Yifang Hu, Charity W Law, Wei Shi, and Gordon K Smyth. 2015. "limma powers differential expression analyses for RNA-sequencing and microarray studies." *Nucleic Acids Research* 43 (7): e47. https://doi.org/10.1093/nar/gkv007.

Sergushichev, Alexey. 2016. "An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation." *bioRxiv*, June, 060012. https://doi.org/10.1101/060012.

Sveen, Anita, Jarle Bruun, Peter W. Eide, Ina A. Eilertsen, Lorena Ramirez, Astrid Murumägi, Mariliina Arjama, et al. 2018. "Colorectal Cancer Consensus Molecular Subtypes Translated to Preclinical Models Uncover Potentially Targetable Cancer Cell Dependencies." *Clinical Cancer Research* 24 (4): 794–806. https://doi.org/10.1158/1078-0432.CCR-17-1234.

Wagner, Klaus W, Elizabeth A Punnoose, Thomas Januario, David A Lawrence, Robert M Pitti, Kate Lancaster, Dori Lee, et al. 2007. "Death-receptor O-glycosylation controls tumor-cell sensitivity to the proapoptotic ligand Apo2L/TRAIL." *Nature Medicine* 13 (9): 1070–7. https://doi.org/10.1038/nm1627.

Winter, Jan, Marc Schwering, Oliver Pelz, Benedikt Rauscher, Tianzuo Zhan, Florian Heigwer, and Michael Boutros. 2017. "CRISPRAnalyzeR: Interactive analysis, annotation and documentation of pooled CRISPR screens." *bioRxiv*, February, 109967. https://doi.org/10.1101/109967.