Predicting drug response based on molecular data in pancreatic organoids

Gergana Bounova and Kat Moore

Contents

Summary

This report aims to integrate multiple types of molecular data (SNP/indel, copy number, RNA expression) into a drug response prediction model for single agents and drug combinations. We implement a conventional elastic net model as well as the two-step TANDEM model to emphasize upstream molecular features.

Samples meta data

HUB IDs to manuscript IDs

The HUB IDs are what's used in the drug screens.

```
sampleMap = readxl::read_excel(here("external", "HUB vs PDO nummer v2.xlsx"),
                               col_names = F)
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
```

```
colnames(sampleMap)[c(1,3)] = c("HUB.ID", "PDO.ID")
sampleMap = sampleMap %>% dplyr::select(HUB.ID, PDO.ID)
sampleMap = sampleMap %>% na.omit()
head(sampleMap)
```

```
## # A tibble: 6 x 2
##   HUB.ID         PDO.ID
##   <chr>          <chr>
## 1 HUB-08-B2-002A PDO1
## 2 HUB-08-B2-007B PDO2
## 3 HUB-08-B2-010A PDO4
## 4 HUB-08-B2-010B PDO5
## 5 HUB-08-B2-013A PDO6
## 6 HUB-08-B2-013B PDO7
```

The PDO IDs correspond 1:1 with the manuscript IDs, so that PDO1=T1, PDO2=T2, etc.

```
sampleMap = sampleMap %>%
  mutate(manuscript_id = str_replace(PDO.ID, "PDO", "T"))
```

The rest of the metadata is in a different file.

Create a function to read in all sheets of an Excel workbook as a named list of data frames.

```
read_excel_tabs = function(path){
  require(tidyverse)
  tab_names = readxl::excel_sheets(path)
  ldf = lapply(tab_names, function(x) readxl::read_excel(path, sheet = x))
  names(ldf) = tab_names
  return(ldf)
}
```

Manuscript IDs to RNAseq

```
id_info = read_excel_tabs(here("drug_screen_analysis","IDs_info_detailed.xlsx"))
```

```
## New names:
## * `` -> ...17
```

```
names(id_info)[2] = "QC"
#lapply(id_info, head)
```

Combine the RNAseq ID into the sample map

```
sampleMap = left_join(rename(id_info$Overview, manuscript_id=SampleID_short),
                      sampleMap, by = "manuscript_id")
sampleMap = sampleMap %>%
  select(manuscript_id, HUB.ID, RNAseq_ID, sampleId, everything())

head(sampleMap)
```

```
## # A tibble: 6 x 22
##   manuscript_id HUB.ID RNAseq_ID sampleId SampleId_nr RNAseq `IHC panel`
##   <chr>         <chr>  <chr>     <chr>          <dbl> <lgl>  <lgl>
## 1 T1            HUB-0~ HUB.HUB.~ FR11123~           1 NA     NA
## 2 T2            HUB-0~ HUB.WCB.~ FR11123~           2 NA     NA
## 3 T3            <NA>   HUB.HUB.~ FR11124~           3 NA     NA
## 4 T4            HUB-0~ HUB.HUB.~ FR11124~           4 NA     NA
## 5 T5            HUB-0~ HUB.HUB.~ FR11124~           5 NA     NA
## 6 T6            HUB-0~ HUB.HUB.~ FR11123~           6 NA     NA
## # ... with 15 more variables: `originally expanded on` <chr>, `Biobank
## #   code` <chr>, `P-number` <chr>, `E/W number` <chr>, `Date of birth` <dbl>,
## #   Gender <chr>, `Pathology results` <chr>, `Definitive Pathology
## #   diagnosis` <chr>, medium2 <chr>, medium <chr>, `matched normal` <chr>,
## #   ...17 <chr>, germline_sample <chr>, germline_sample_ID <chr>, PDO.ID <chr>
```

Which samples lack HUB.IDs (i.e. don't have drug data)?

```
sampleMap %>% filter(is.na(HUB.ID)) %>% pull(manuscript_id)
```

```
## [1] "T3"  "T8"  "T11" "T12" "T13" "T27" "T29"
```

Whole Genome Sequencing

WGS data analyzed by Arne van Hoeck.

Read in variant data

```
muts = read_excel_tabs(here("drug_screen_analysis", "driver_muts_overview.xlsx"))
```

```
sv = inner_join(muts$driver_overview, muts$`list of driver genes`,
                by = c("gene"="Gene"))
head(sv)
```

```
## # A tibble: 6 x 15
##   sampleid gene  chromosome position ref   alt   type  effect impact impact2
```

```
##   <chr>    <chr> <chr>         <dbl> <chr> <chr> <chr> <chr>  <chr>
## 1 T25      ABL1  9           1.34e8 TC    T     INDEL frame~ no-SNV INDEL
## 2 T4       ABL1  9           1.34e8 A     G     SNP   struc~ Synon~ Synony~
## 3 T10      ACVR~ 12          5.24e7 C     T     SNP   misse~ Misse~ Missen~
## 4 T9       ACVR~ 12          5.24e7 C     T     SNP   misse~ Misse~ Missen~
## 5 T25      ACVR~ 2           1.49e8 TA    T     INDEL frame~ no-SNV INDEL
## 6 T4       ACVR~ 2           1.49e8 T     C     SNP   synon~ Synon~ Synony~
## # ... with 5 more variables: LOH <dbl>, Cancer <chr>, KEY <chr>,
## #   suppressor_or_oncogene <chr>, Gene_MoA <chr>
```

Overview of variants detected. Explanations can be found here(https://www.targetvalidation.org/variants).

```
table(sv$effect, sv$impact)
```

```
##
##                                 Essential_Splice Missense Nonsense no-SNV
##   frameshift variant                           0        0        0     62
##   inframe deletion                             0        0        0      6
##   inframe insertion                            0        0        0      6
##   intron variant                               1        0        0      0
##   missense variant                             0      147        0     15
##   protein protein contact                      0        4        0      0
##   sequence feature                             0        0        0      0
##   splice acceptor variant                      1        0        0      0
##   splice region variant                        0        1        0      0
##   start lost                                   0        1        0      0
##   stop gained                                  0        4       25      4
##   structural interaction variant               0       30        0      4
##   synonymous variant                           0        0        0      6
##
##                                 Synonymous
##   frameshift variant                     0
##   inframe deletion                       0
##   inframe insertion                      0
##   intron variant                         1
##   missense variant                       5
##   protein protein contact                5
##   sequence feature                       1
##   splice acceptor variant                0
##   splice region variant                  3
##   start lost                             0
##   stop gained                            0
##   structural interaction variant        36
##   synonymous variant                   597
```

Remove synonymous mutants

```
sv = sv %>% filter(impact != "Synonymous")
```

Total number of pancreatic cancer genes:

```
length(unique(sv$gene))
```

```
## [1] 83
```

Total number of samples with WGS data:

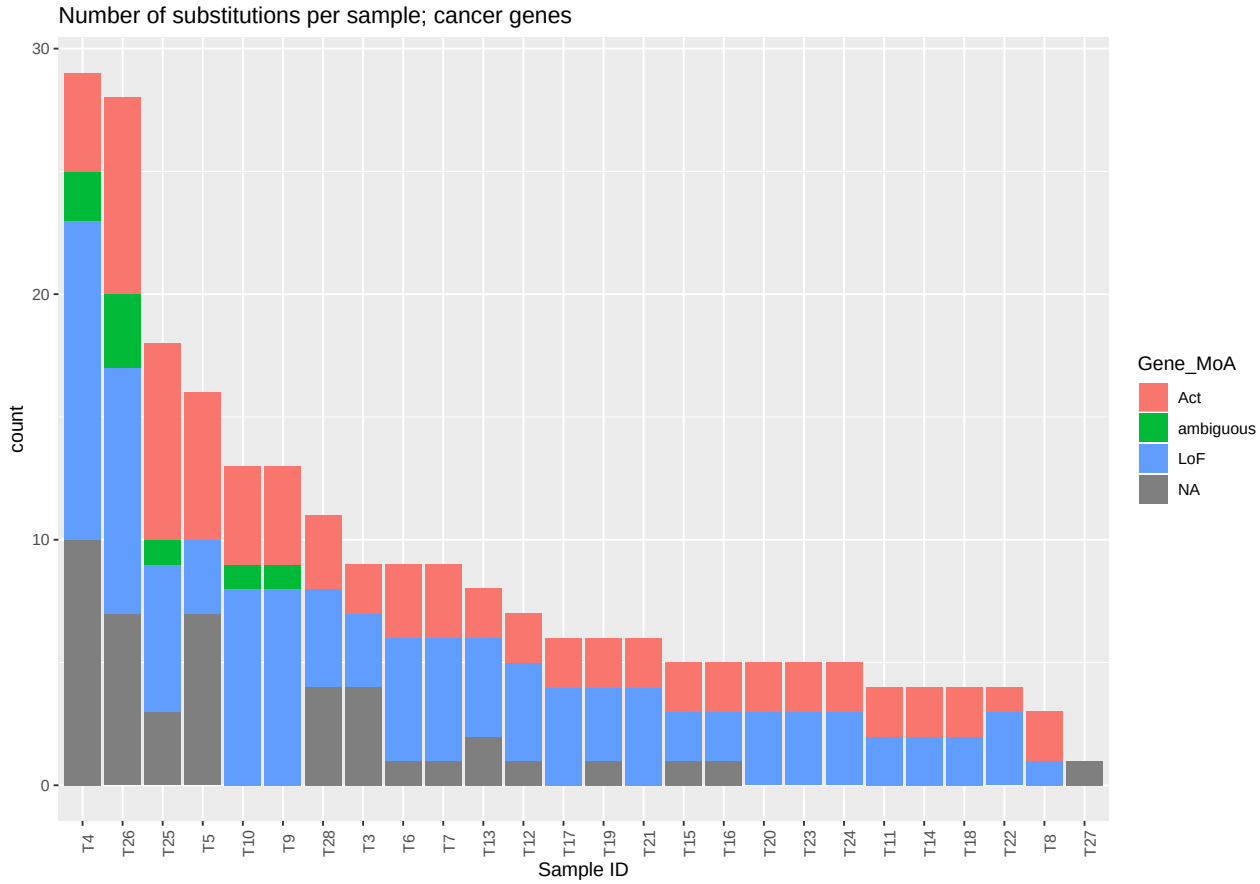
```
sv = sv %>% rename(manuscript_id = sampleid)
length(unique(sv$manuscript_id))
```

```
## [1] 26
```

Variant plots

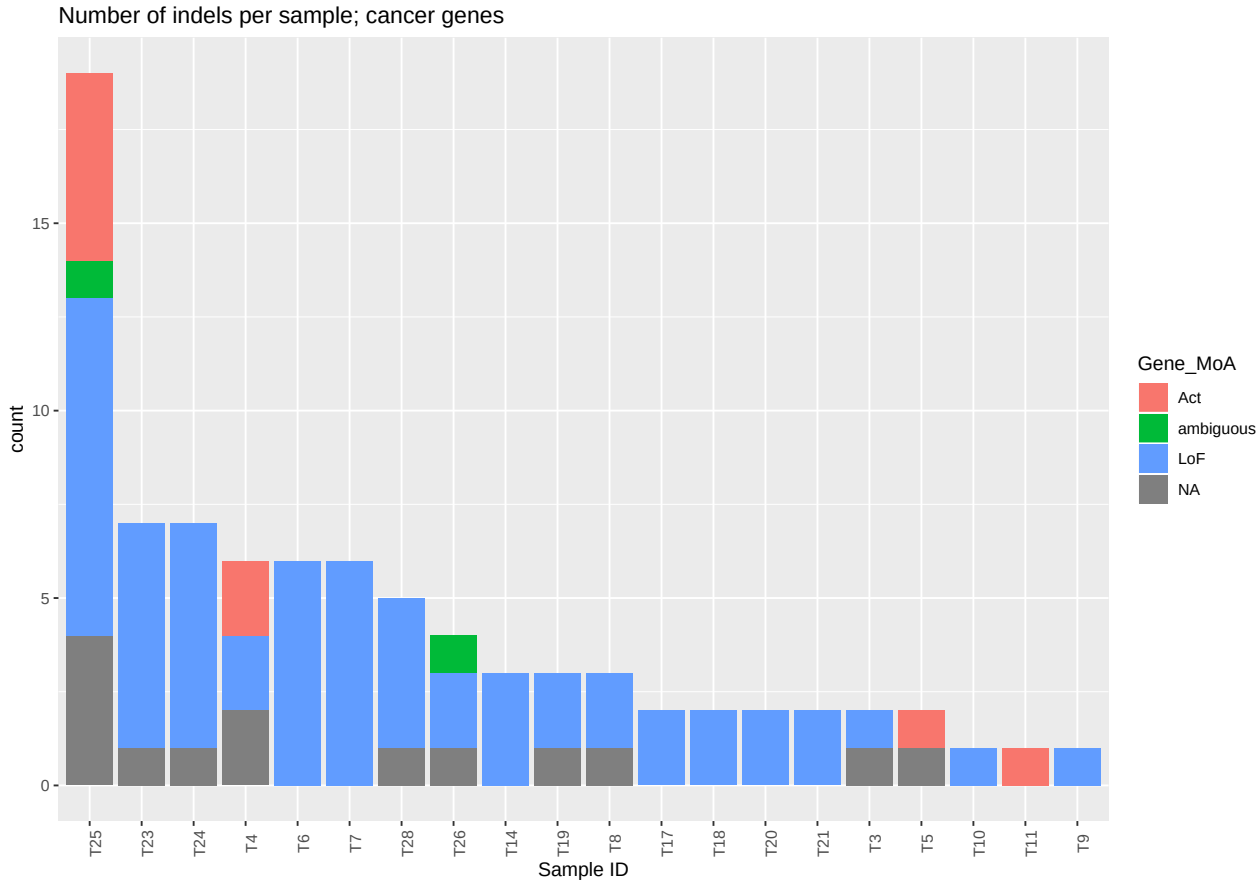
Substitutions per sample

```
sv %>% filter(type %in% c("SNP", "MNP")) %>%
           ggplot(aes(x=factor(manuscript_id,
                               levels=(sv %>%
                                         filter(type %in% c("SNP", "MNP"))%>%
                                         group_by(manuscript_id) %>%
                                         summarise(n=n()) %>%
                                         arrange(desc(n)) %>%
                                         pull(manuscript_id))),
                      fill=Gene_MoA)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Number of substitutions per sample; cancer genes") +
  xlab("Sample ID")
```



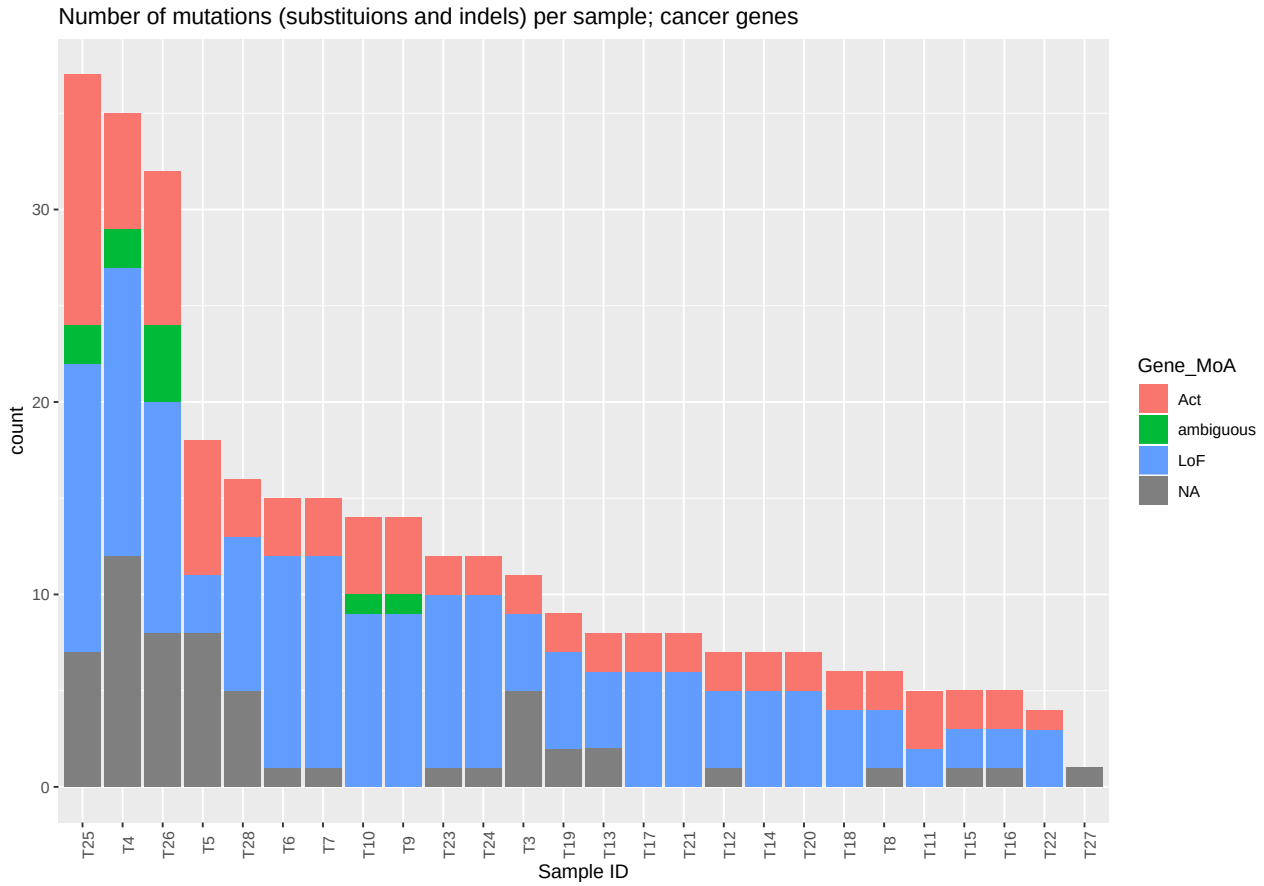
Indels per sample

```
sv %>% filter(type == "INDEL") %>%
          ggplot(aes(x=factor(manuscript_id,
                              levels= sv %>% filter(type == "INDEL") %>%
                                group_by(manuscript_id) %>% summarise(n=n()) %>%
                                arrange(desc(n)) %>% pull(manuscript_id)),
                     fill=Gene_MoA)) + geom_bar() +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Number of indels per sample; cancer genes") +
  xlab("Sample ID")
```



Mutations per sample

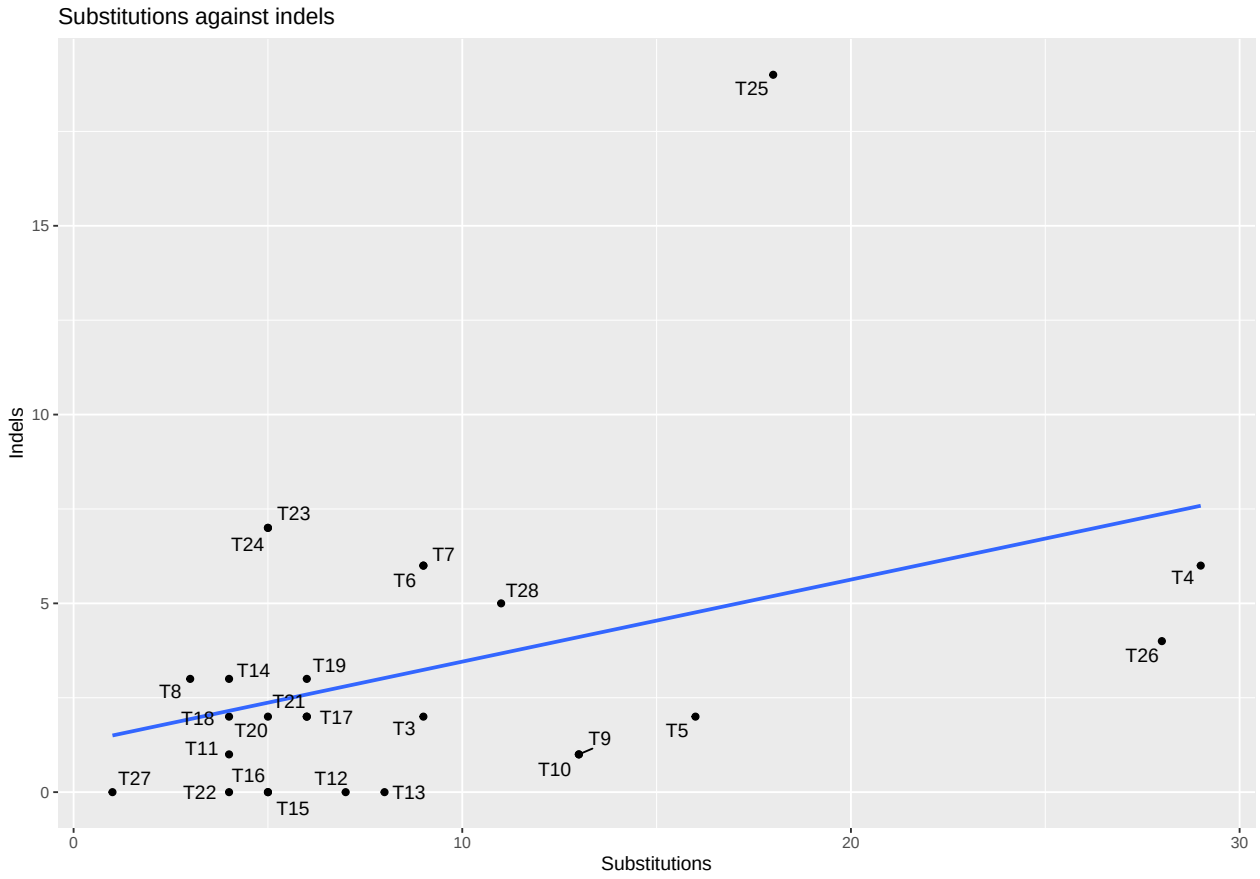
```
sv %>% ggplot(aes(x=factor(manuscript_id,
                              levels= sv %>%
                                group_by(manuscript_id) %>% summarise(n=n()) %>%
                                arrange(desc(n)) %>% pull(manuscript_id)),
                     fill=Gene_MoA)) + geom_bar() +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Number of mutations (substituions and indels) per sample; cancer genes") +
  xlab("Sample ID")
```




```
ggsave(filename = here("figs", "mutations_per_sample.pdf"),
       device = "pdf", width = 10, height =7, units = "in")
```

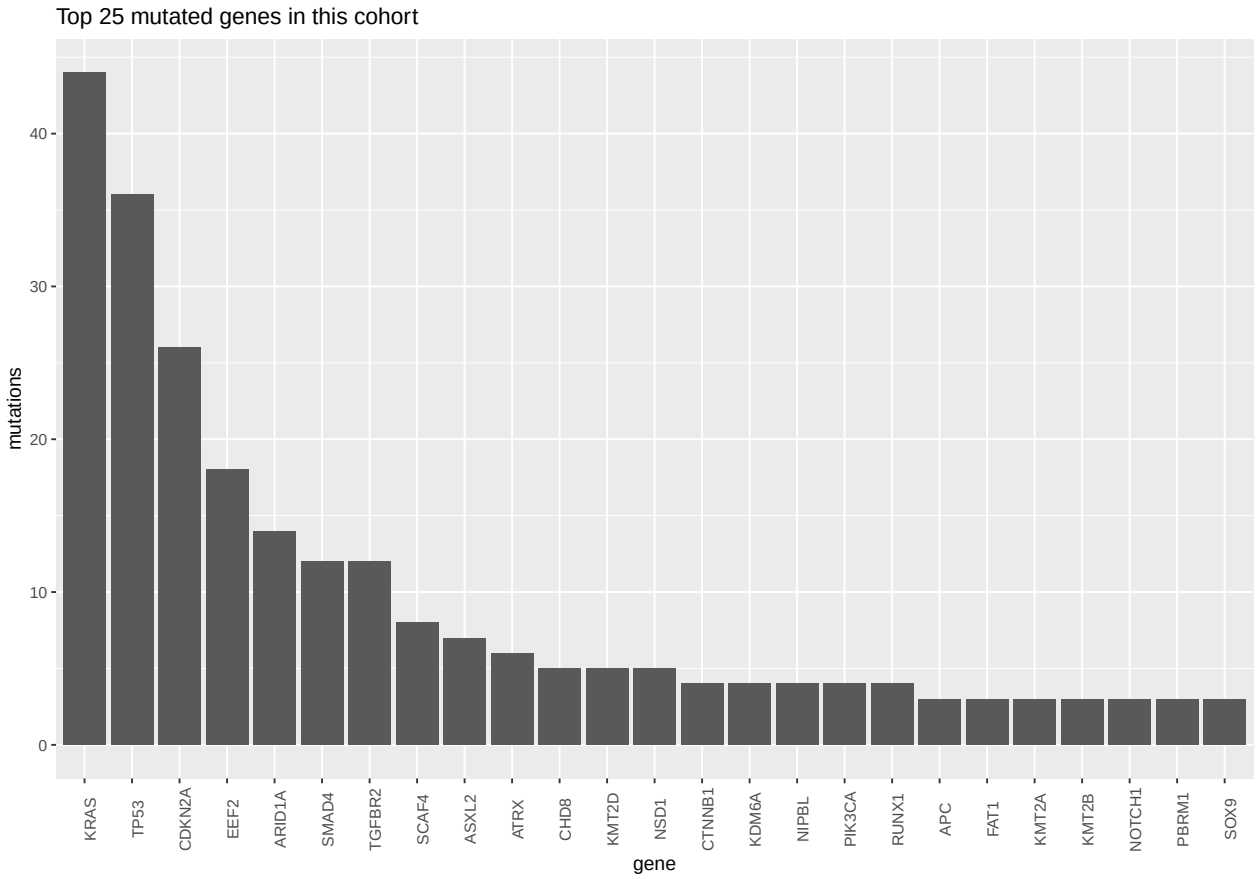
Substitutions vs indels

```
full_join(sv %>% filter(impact != "Synonymous" & type %in% c("SNP", "MNP")) %>%
            group_by(manuscript_id) %>% summarise(Substitutions = n()),
          sv %>% filter(impact != "Synonymous" & type =="INDEL") %>%
            group_by(manuscript_id) %>% summarise(Indels = n()),
          by="manuscript_id") %>%
  mutate(Indels=replace_na(Indels, 0), Substitutions=replace_na(Substitutions, 0)) %>%
  ggplot(aes(x=Substitutions, y = Indels, label=manuscript_id)) +
  geom_point() + geom_smooth(method="lm", se = F) +
  ggrepel::geom_text_repel() +
  ggtitle("Substitutions against indels")
```



Top 25 mutated genes

```
sv %>% group_by(gene) %>% summarise(mutations=n()) %>%
  arrange(desc(mutations)) %>% head(25) %>%
  ggplot(aes(x = factor(gene, levels=gene), y=mutations)) +
  geom_bar(stat="identity") +
  xlab("gene") + ggtitle("Top 25 mutated genes in this cohort") +
  theme(axis.text.x = element_text(angle = 90))
```




```
ggsave(filename = here("figs", "top25_mutated_genes.pdf"),
       device = "pdf", width = 10, height =7, units = "in")
```

Account for gene length.

```
geneLengths = read_tsv(here("external", "geneLengths_GRCh37p13.txt"))
```

```
## Parsed with column specification:
## cols(
##   `Gene stable ID` = col_character(),
##   `Gene description` = col_character(),
##   `Gene start (bp)` = col_double(),
##   `Gene end (bp)` = col_double(),
##   `Gene name` = col_character(),
##   `HGNC symbol` = col_character()
## )
```

```
geneLengths$length = geneLengths$`Gene end (bp)` - geneLengths$`Gene start (bp)`
geneLengths = geneLengths %>% select(geneSymbol = `HGNC symbol`, length) %>%
  filter(!is.na(geneSymbol))
geneLengths = geneLengths[!duplicated(geneLengths[, 'geneSymbol']),]
```

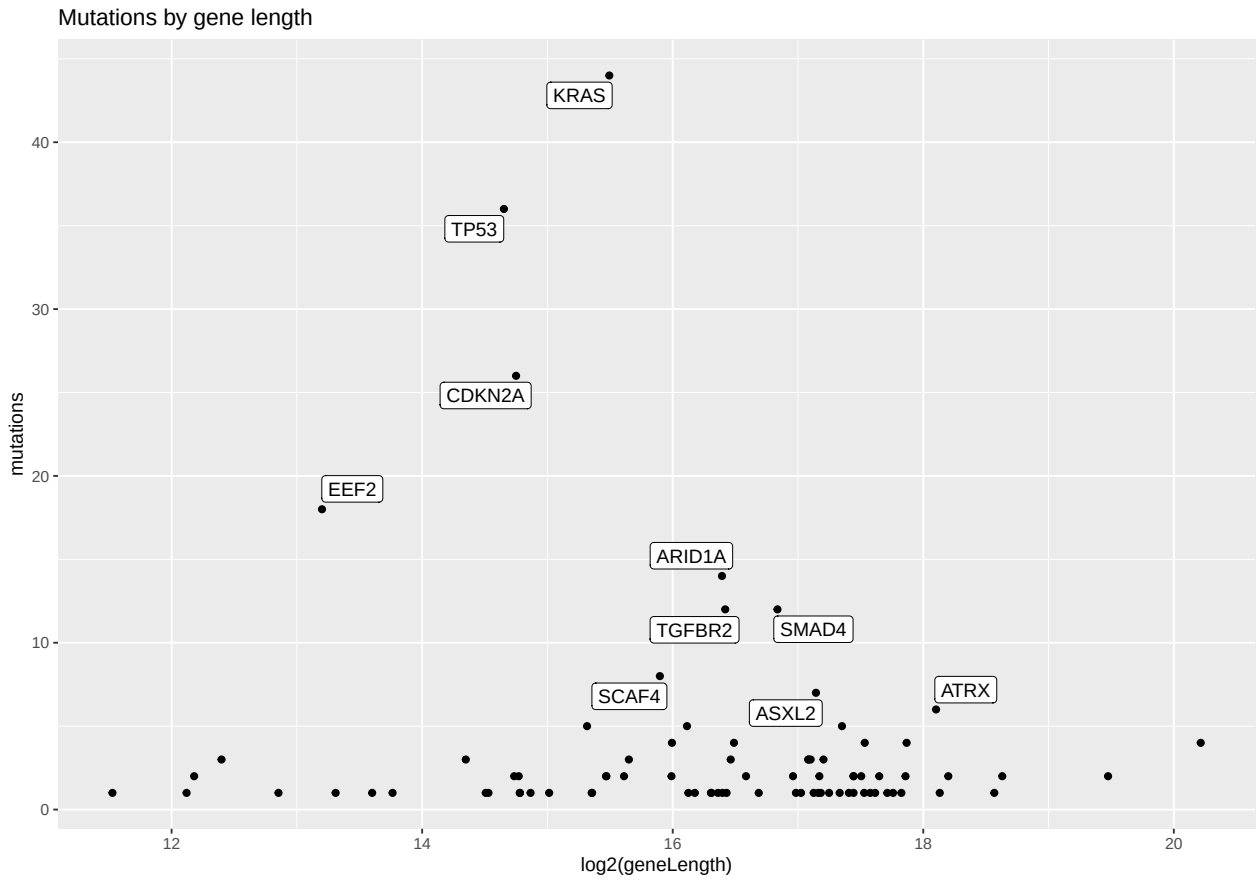
Add gene length data to variant data

```
sv = left_join(sv, geneLengths, by = c("gene" = "geneSymbol"))
stopifnot(sum(is.na(sv$length))==0)
```

Mutations by gene length

```
sv %>% group_by(gene, length) %>% summarise(mutations=n()) %>% ungroup() %>%
  mutate(labels = if_else(mutations > 5, gene, NULL)) %>%
  ggplot(aes(x=log2(length), y = mutations, label=labels)) + geom_point() +
  ggrepel::geom_label_repel() + xlab("log2(geneLength)") +
  ggtitle("Mutations by gene length")
```

```
## Warning: Removed 73 rows containing missing values (geom_label_repel).
```




```
ggsave(filename = here("figs", "mutations_by_gene_length.pdf"),
       device = "pdf", width = 10, height =7, units = "in")
```

```
## Warning: Removed 73 rows containing missing values (geom_label_repel).
```

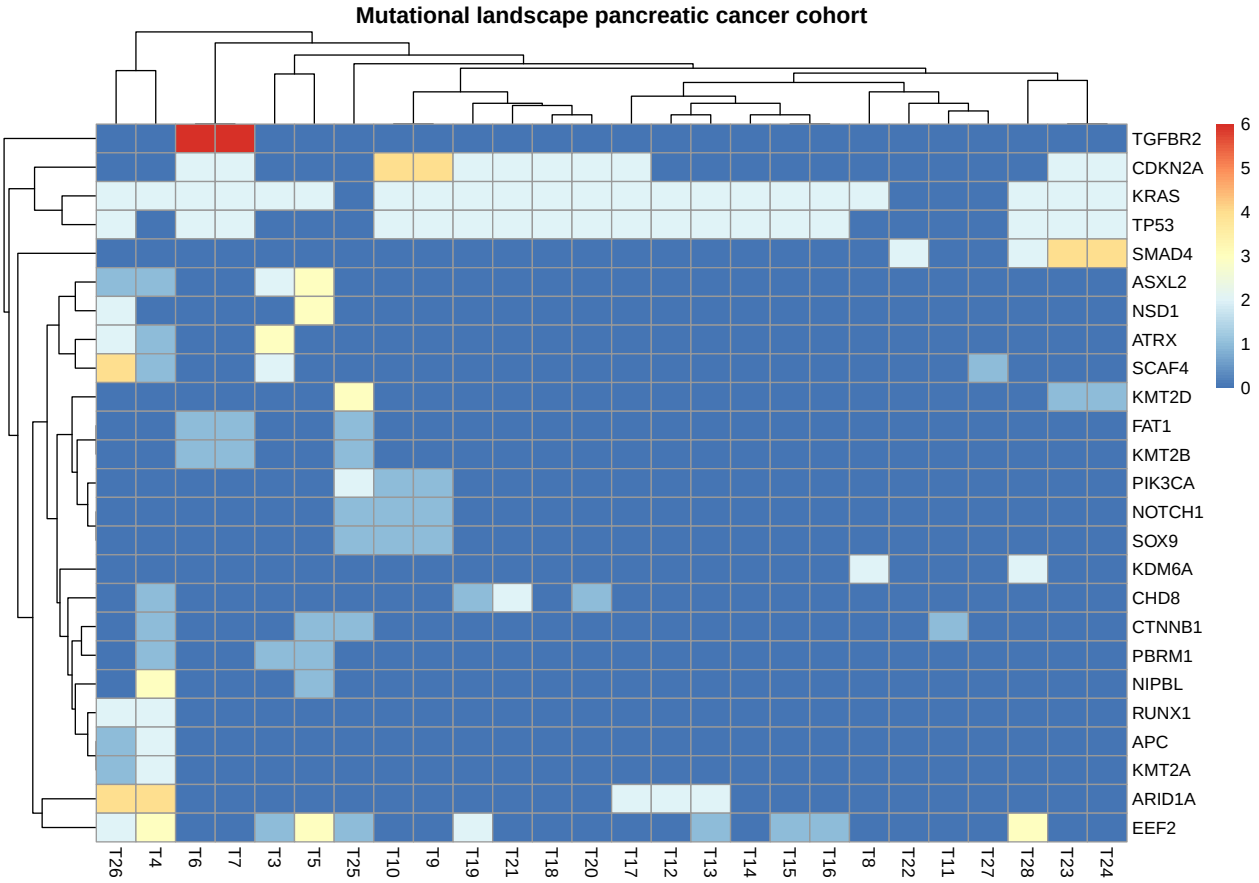
Mutational landscape

Set up a heatmap for top 25 mutations by sample

```
hm_mut = sv %>% group_by(manuscript_id, gene) %>% summarise(mutations=n()) %>%
  filter(gene %in% (sv %>% group_by(gene) %>% summarise(mutations=n()) %>%
                      arrange(desc(mutations)) %>% head(25) %>% pull(gene))) %>%
  spread(key = manuscript_id, value = mutations)

hm_mut[is.na(hm_mut)] = 0
```

```
hm_mut = hm_mut %>% column_to_rownames("gene")
```

```
pheatmap(hm_mut, main = "Mutational landscape pancreatic cancer cohort")
```



Summarize mutations by gene

This will be the input for the elastic net models later in the report.

```
mutMatrix = sv %>% group_by(manuscript_id, gene) %>% summarise(n=n()) %>%
  spread(key=manuscript_id, value=n, fill=0) %>%
  mutate(gene=paste0("mut",gene)) %>%
  column_to_rownames("gene") %>% as.matrix()

mutMatrix[1:10,1:10]
```

```
##           T10 T11 T12 T13 T14 T15 T16 T17 T18 T19
## mutABL1     0   0   0   0   0   0   0   0   0   0
## mutACVR1B   1   0   0   0   0   0   0   0   0   0
## mutACVR2A   0   0   0   0   0   0   0   0   0   0
## mutALK      0   0   0   0   0   0   0   0   0   0
## mutAPC      0   0   0   0   0   0   0   0   0   0
## mutAR       0   0   0   0   0   0   0   0   0   0
## mutARID1A   0   0   2   2   0   0   0   2   0   0
## mutARID2    0   1   0   0   0   0   0   0   0   0
## mutASXL1    0   0   0   0   0   0   0   0   0   0
```

```
## mutASXL2    0   0   0   0   0   0   0   0   0   0
```

Copy number data

Copy numbers were called in a genewise rather than region-wise fashion. Both mutational events and copy number events are combined within the same file, so that only the first 5 columns will be filled in if the copy number is being reported, whereas the remaining columns are applicable to mutations.

The adjustedcopynumber refers to the reported mutation. For instance, the C>T mutation in PIK3CB has a CN of 4.44. This could be different for multiple mutations within the same gene. If you would like to use the "overall" copy number level of a gene, it is advisable to take the mincopynumber. The minCN and maxCN are mostly the same, unless there is a breakpoint detected within the gene region.

```
cancer_cnv = read_tsv(file = here("drug_screen_analysis",
                                  "cancer_genes_raw_data.txt")) #Called by gene
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   sampleid = col_character(),
##   gene = col_character(),
##   mincopynumber = col_double(),
##   maxcopynumber = col_double(),
##   adjustedcopynumber = col_double(),
##   adjustedvaf = col_double(),
##   minoralleleploidy = col_double(),
##   loh = col_double(),
##   chromosome = col_character(),
##   position = col_double(),
##   ref = col_character(),
##   alt = col_character(),
##   clonality = col_character(),
##   type = col_character(),
##   effect = col_character()
## )
```

```
head(cancer_cnv)
```

```
## # A tibble: 6 x 16
##      X1 sampleid gene  mincopynumber maxcopynumber adjustedcopynum~ adjustedvaf
##   <dbl> <chr>    <chr>         <dbl>         <dbl>            <dbl>       <dbl>
## 1     1 FR11123~ POLQ           4.44          4.44            NA           NA
## 2     2 FR11123~ PIK3~          4.44          4.44             4.44         0.5
## 3     3 FR11123~ ATR            4.44          4.44            NA           NA
## 4     4 FR11123~ TBL1~          4.43          4.43            NA           NA
## 5     5 FR11123~ PIK3~          4.43          4.43            NA           NA
## 6     6 FR11123~ FGFR3          2.97          2.97            NA           NA
## # ... with 9 more variables: minoralleleploidy <dbl>, loh <dbl>,
## #   chromosome <chr>, position <dbl>, ref <chr>, alt <chr>, clonality <chr>,
## #   type <chr>, effect <chr>
```

Join cnv data with region data and sample metadata

```
cancer_cnv =
left_join(cancer_cnv, select(sampleMap, manuscript_id,
                             HUB.ID, RNAseq_ID, sampleId),
          by = c("sampleid"="sampleId")) %>%
  select(manuscript_id, gene,
         copynumberlevel=mincopynumber, everything()) %>%
  select(-X1)

head(cancer_cnv)
```

```
## # A tibble: 6 x 18
##   manuscript_id gene  copynumberlevel sampleid maxcopynumber adjustedcopynum~
##   <chr>         <chr>           <dbl> <chr>            <dbl>            <dbl>
## 1 T6            POLQ             4.44 FR11123~          4.44             NA
## 2 T6            PIK3~            4.44 FR11123~          4.44              4.44
## 3 T6            ATR              4.44 FR11123~          4.44             NA
## 4 T6            TBL1~            4.43 FR11123~          4.43             NA
## 5 T6            PIK3~            4.43 FR11123~          4.43             NA
## 6 T6            FGFR3            2.97 FR11123~          2.97             NA
## # ... with 12 more variables: adjustedvaf <dbl>, minoralleleploidy <dbl>,
## #   loh <dbl>, chromosome <chr>, position <dbl>, ref <chr>, alt <chr>,
## #   clonality <chr>, type <chr>, effect <chr>, HUB.ID <chr>, RNAseq_ID <chr>
```

Samples for which cn data is available:

```
sort(unique(cancer_cnv$manuscript_id))
```

```
##  [1] "T10" "T11" "T12" "T13" "T14" "T15" "T16" "T17" "T18" "T19" "T20" "T21"
## [13] "T22" "T23" "T24" "T25" "T26" "T27" "T28" "T3"  "T4"  "T5"  "T6"  "T7"
## [25] "T8"  "T9"
```

```
stopifnot(sum(is.na(cancer_cnv$manuscript_id))==0)
```

We have copy number data for all samples

```
stopifnot(length(setdiff(unique(cancer_cnv$manuscript_id),sampleMap$manuscript_id))==0)
```

Number of cancer-related genes with cn levels reported:

```
length(unique(cancer_cnv$gene))
```

```
## [1] 202
```

There is generally only one entry per gene:sample combination, however, some gene:sample combinations have many entries (up to 77).

```
cancer_cnv %>% group_by(gene, manuscript_id) %>%
  summarise(n=n()) %>% pull(n) %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.755   1.000  77.000
```

This occurs when there are multiple different mutations detected for that gene within a given sample, see ARID1A as an example.

```
cancer_cnv %>% filter(gene=="ARID1A" & manuscript_id == "T17") %>%
  select(manuscript_id, gene, copynumberlevel, position,
         ref, alt, type, effect, everything())
```

```
## # A tibble: 11 x 18
```

```
##    manuscript_id gene  copynumberlevel position ref   alt   type  effect
##    <chr>         <chr>           <dbl>    <dbl> <chr> <chr> <chr> <chr>
##  1 T17           ARID~            2.06 27024789 C     T     SNP   intro~
##  2 T17           ARID~            2.06 27025162 TGG   TCGT  INDEL intro~
##  3 T17           ARID~            2.06 27025171 G     T     SNP   intro~
##  4 T17           ARID~            2.06 27106518 CA    TG    MNP   misse~
##  5 T17           ARID~            2.06 27107001 C     T     SNP   synon~
##  6 T17           ARID~            2.06 27107028 C     A     SNP   synon~
##  7 T17           ARID~            2.06 27107817 C     T     SNP   UTR v~
##  8 T17           ARID~            2.06 27107884 A     AATG  INDEL UTR v~
##  9 T17           ARID~            2.06 27107886 C     CGAG  INDEL UTR v~
## 10 T17           ARID~            2.06 27107941 C     T     SNP   UTR v~
## 11 T17           ARID~            2.06 27108038 A     G     SNP   UTR v~
## # ... with 10 more variables: sampleid <chr>, maxcopynumber <dbl>,
## #   adjustedcopynumber <dbl>, adjustedvaf <dbl>, minoralleleploidy <dbl>,
## #   loh <dbl>, chromosome <chr>, clonality <chr>, HUB.ID <chr>, RNAseq_ID <chr>
```

The copynumberlevel in this case refers to the overal gene expression and does not vary between multiple reported mutations within the same gene:sample combination. We can see this is the case for all genes in the dataset by calculating the copy number standard deviation for every gene:sample combination. It will be 0 when there are multiple mutations reported and NA when only one event is reported.

```
#Total number of trues should be equal to the length of the vector
stopifnot(
  (cancer_cnv %>% group_by(manuscript_id, gene) %>%
     summarise(sd = sd(copynumberlevel)) %>% pull(sd) %in% c(0,NA) %>% sum()) ==
    (cancer_cnv %>% group_by(manuscript_id, gene) %>%
       summarise(sd = sd(copynumberlevel)) %>% pull(sd) %>% length())
  )
```

For this reason, when looking at copy number, we can reduce the dataset down to unique gene:sample combinations.

```
cancer_cnv = cancer_cnv %>%
  mutate(genebysample = paste(manuscript_id, gene, sep=":")) %>%
  filter(!duplicated(genebysample)) %>%
  select(manuscript_id, gene, copynumberlevel)
```

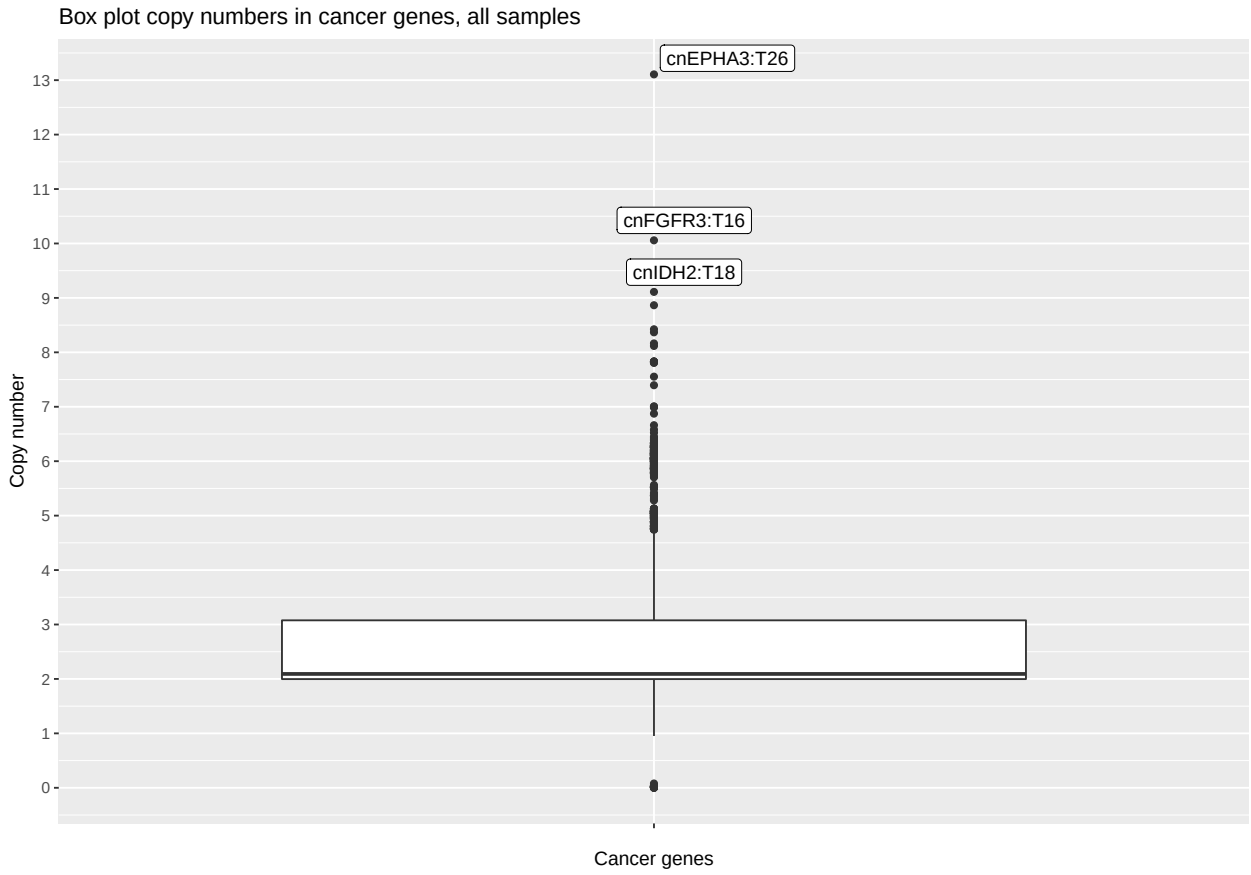
Add "CN" as a prefix so that we can easily ascertain that we're looking at copy number data later on.

```
cancer_cnv = cancer_cnv %>% mutate(gene = paste0("cn", gene))
```

Boxplot adjusted copy number

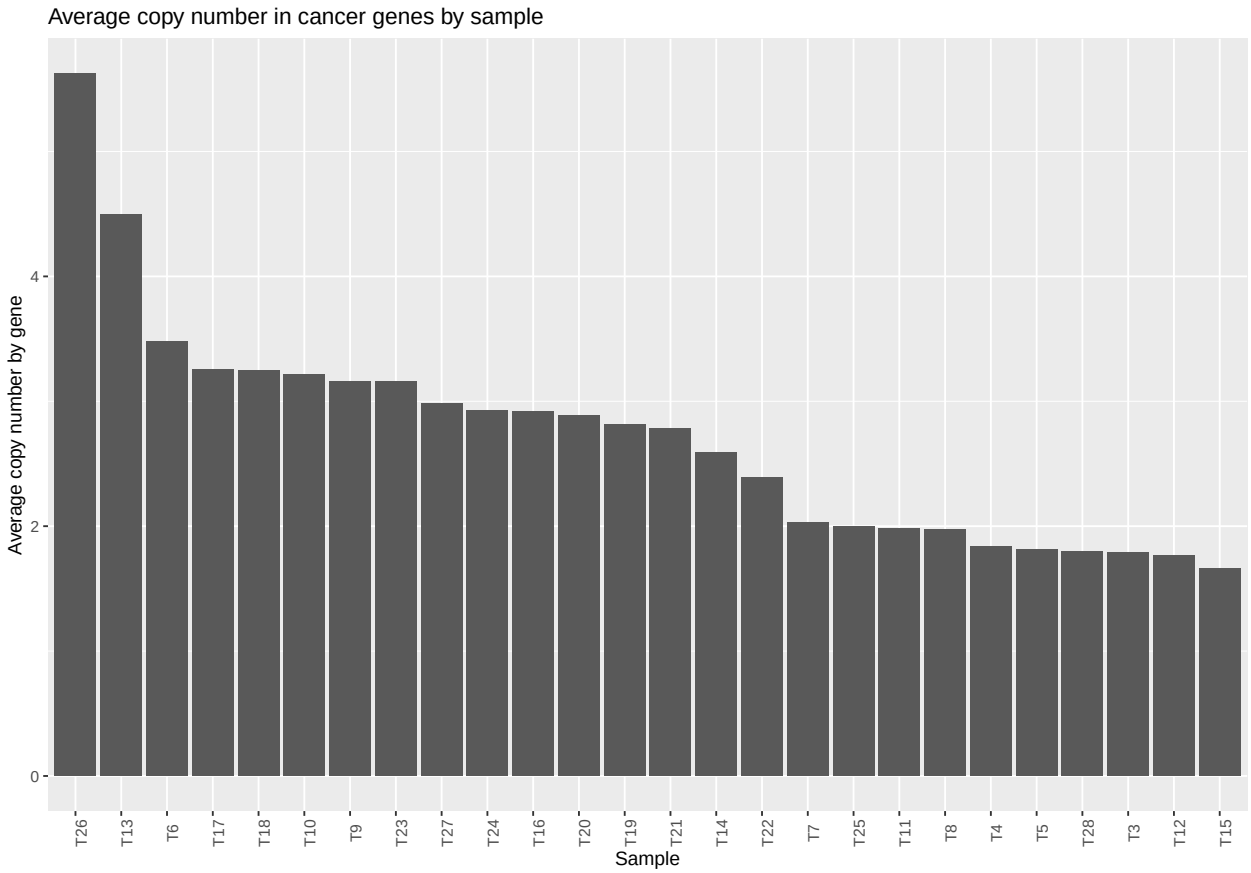
```
cancer_cnv %>%
  mutate(label=if_else(copynumberlevel >=9,
                       paste(gene, manuscript_id, sep=":"), NULL)) %>%
  ggplot(aes(x="", y = copynumberlevel, label=label)) +
  geom_boxplot() +
  scale_y_continuous(breaks = seq(0, 20, by=1)) +
  ylab("Copy number") + xlab("Cancer genes") +
  ggtitle("Box plot copy numbers in cancer genes, all samples") +
  ggrepel::geom_label_repel()
```

```
## Warning: Removed 5249 rows containing missing values (geom_label_repel).
```



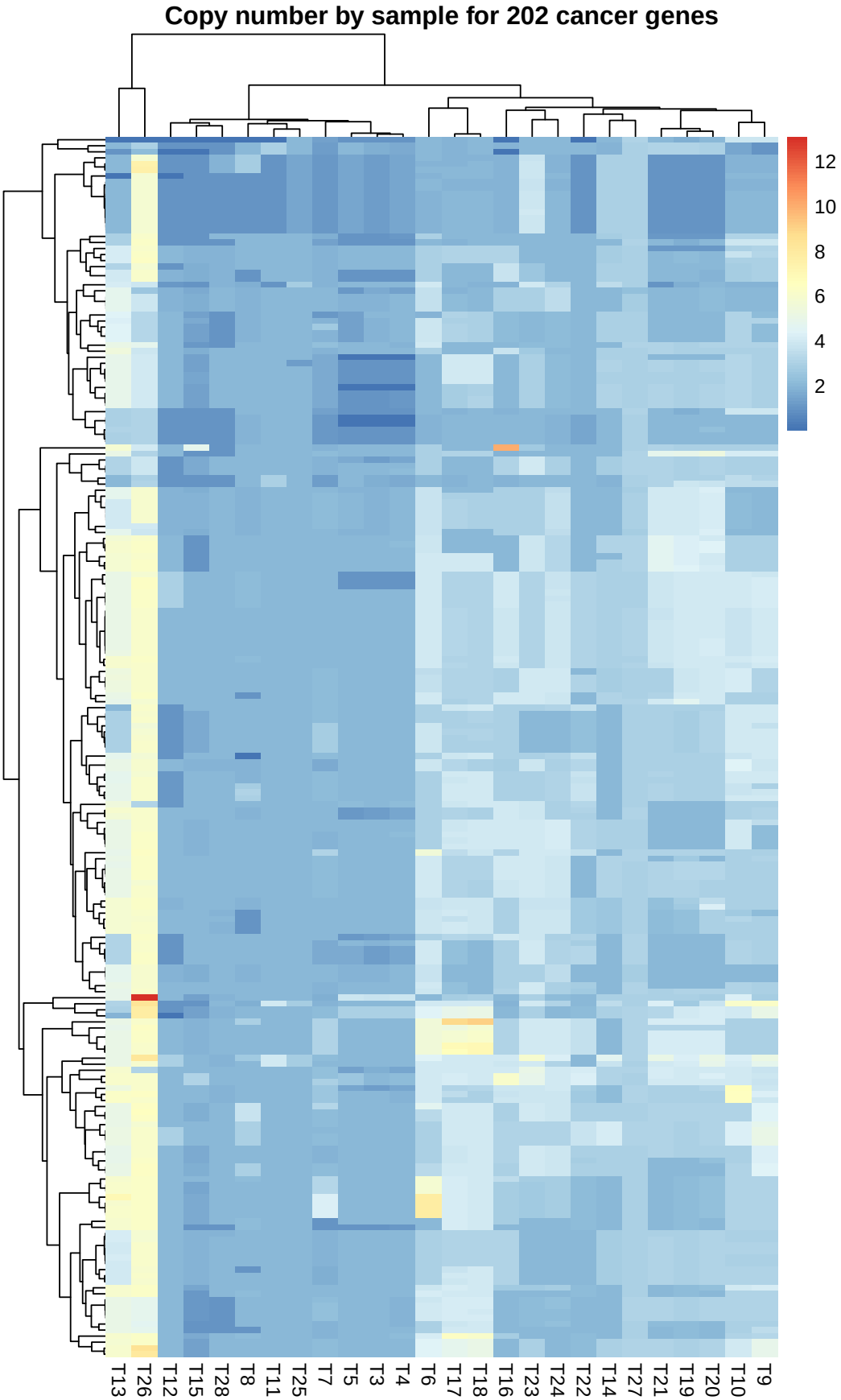
Sample CNV bar plot

```
cancer_cnv %>%
  group_by(manuscript_id) %>% summarise(mean_cn = mean(copynumberlevel)) %>%
  arrange(desc(mean_cn)) %>% #mutate(manuscript_id = factor(manuscript_id)) %>%
  ggplot(aes(x=factor(manuscript_id, levels=manuscript_id), y = mean_cn)) +
  geom_bar(stat="identity") +
  xlab("Sample") + ylab("Average copy number by gene") +
  ggtitle("Average copy number in cancer genes by sample") +
  theme(axis.text.x = element_text(angle = 90))
```



Heatmap copy number

```
cancer_cnv %>% spread(key=manuscript_id, value=copynumberlevel) %>%
  column_to_rownames("gene") %>%
  pheatmap(show_rownames = F, scale="none",
           main=paste("Copy number by sample for",
                      length(unique(cancer_cnv$gene)), "cancer genes"))
```

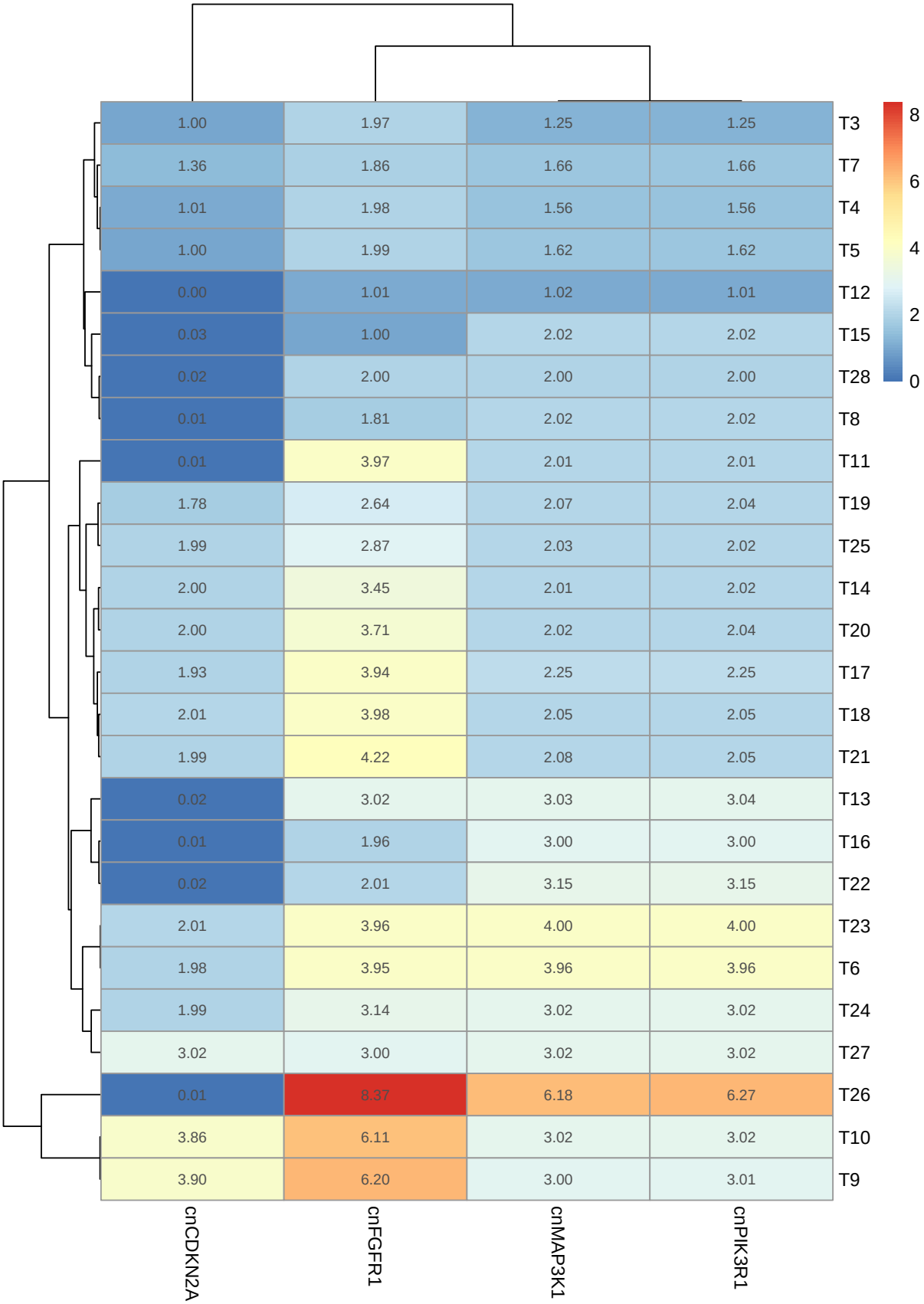

Copy number by sample for 202 cancer genes
12
10
8
6
4
2
T13
T26
T12
T15
T28
T8
T11
T25
T7
T5
T3
T4
T6
T17
T18
T16
T23
T24
T22
T14
T27
T21
T19
T20
T10
T9

```
#Also save as image
cancer_cnv %>% spread(key=manuscript_id, value=copynumberlevel) %>%
  column_to_rownames("gene") %>%
  pheatmap(show_rownames = F, scale="none",
           main=paste("Copy number by sample for",
                      length(unique(cancer_cnv$gene)), "cancer genes"),
           filename = here("figs", "hm_copyn_by_sample.pdf"),
           height=10, width=6)
```

Heatmap known CN genes

Examine CNVs for a few known genes with frequently occuring events. Notable is that cnCDKN2A normally is deleted in cancer, which is not universally the case for all of the organoids in the screen.

```
cancer_cnv %>%
  filter(gene %in% c("cnCDKN2A", "cnFGFR1", "cnPIK3R1", "cnMAP3K1")) %>%
  mutate(copynumberlevel = round(copynumberlevel, 2)) %>%
  spread(key = "gene", value = "copynumberlevel") %>%
  column_to_rownames("manuscript_id") %>%
  pheatmap(., display_numbers = T,
           main = "Copy numbers for select genes with known pancreatic cancer aberrations")
```

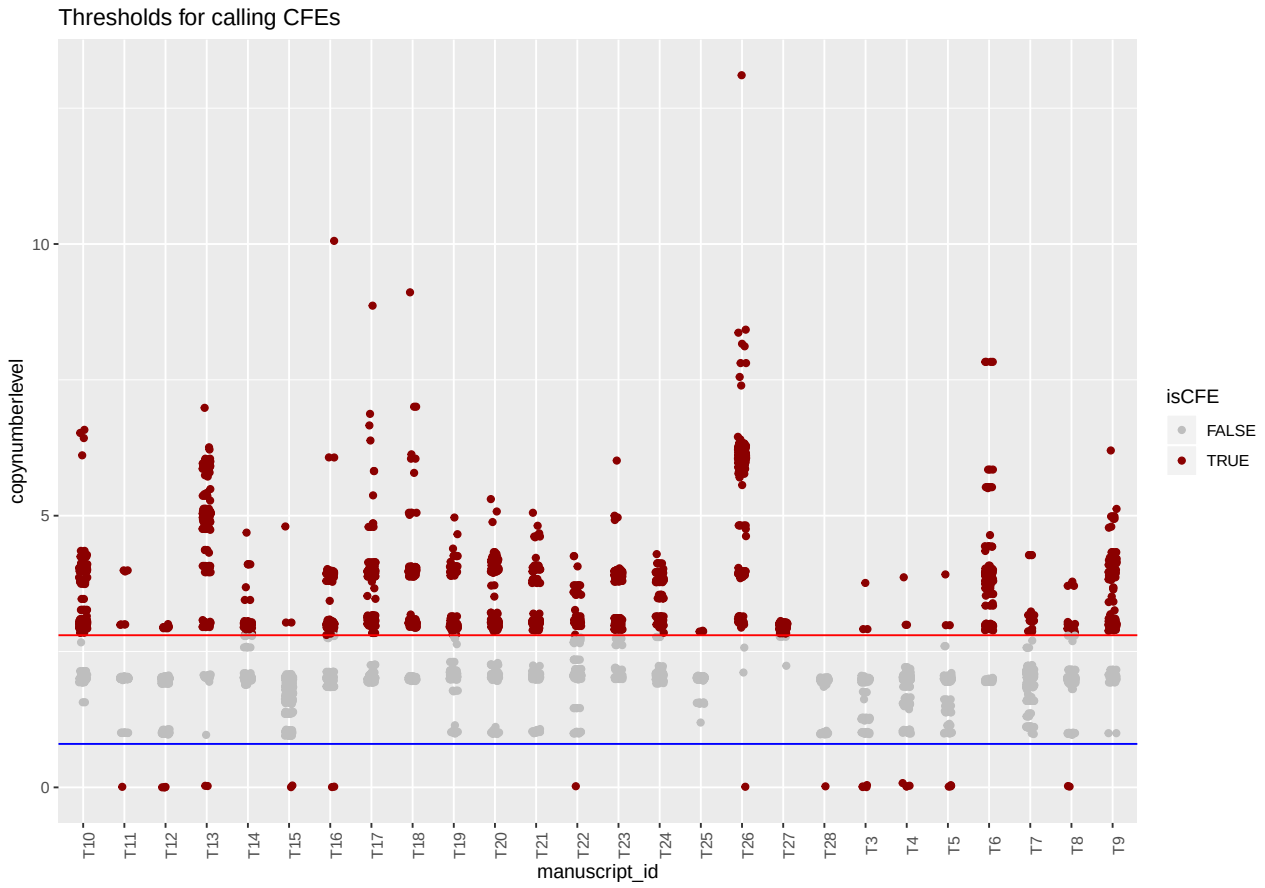

Copy numbers for select genes with known pancreatic cancer aberrations
1.00 1.97 1.25 1.25 T3
1.36 1.86 1.66 1.66 T7
1.01 1.98 1.56 1.56 T4
1.00 1.99 1.62 1.62 T5
0.00 1.01 1.02 1.01 T12
0.03 1.00 2.02 2.02 T15
0.02 2.00 2.00 2.00 T28
0.01 1.81 2.02 2.02 T8
0.01 3.97 2.01 2.01 T11
1.78 2.64 2.07 2.04 T19
1.99 2.87 2.03 2.02 T25
2.00 3.45 2.01 2.02 T14
2.00 3.71 2.02 2.04 T20
1.93 3.94 2.25 2.25 T17
2.01 3.98 2.05 2.05 T18
1.99 4.22 2.08 2.05 T21
0.02 3.02 3.03 3.04 T13
0.01 1.96 3.00 3.00 T16
0.02 2.01 3.15 3.15 T22
2.01 3.96 4.00 4.00 T23
1.98 3.95 3.96 3.96 T6
1.99 3.14 3.02 3.02 T24
3.02 3.00 3.02 3.02 T27
0.01 8.37 6.18 6.27 T26
3.86 6.11 3.02 3.02 T10
3.90 6.20 3.00 3.01 T9
cnCDKN2A
cnFGFR1
cnMAP3K1
cnPIK3R1
8
6
4
2
0

Copy number thresholds

The thresholds are defined as follows:

- cn > 2.8 = amplification
- cn < 0.8 = deletion

```
cancer_cnv =
cancer_cnv %>%
  mutate(isCFE = copynumberlevel > 2.8 | copynumberlevel < 0.8) %>%
  select(manuscript_id, gene, isCFE, copynumberlevel, everything())

cancer_cnv %>%
  ggplot(aes(x = manuscript_id, y = copynumberlevel, color=isCFE)) +
  geom_jitter(height=0, width = 0.1) +
  geom_hline(yintercept = 2.8, color = "red") +
  geom_hline(yintercept = 0.8, color="blue") +
  scale_color_manual(values=c("gray", "darkred")) +
  ggtitle("Thresholds for calling CFEs") +
  theme(axis.text.x = element_text(angle=90))
```



The data shows that overall, amplification events are far more common than deletion events among this cohort of pancreatic organoids.

This removes about half of the data.

```
cancer_cnv %>% select(isCFE) %>% table()
```

```
## .
```

```
## FALSE  TRUE
##  2869  2383
```

Summarize CN matrix

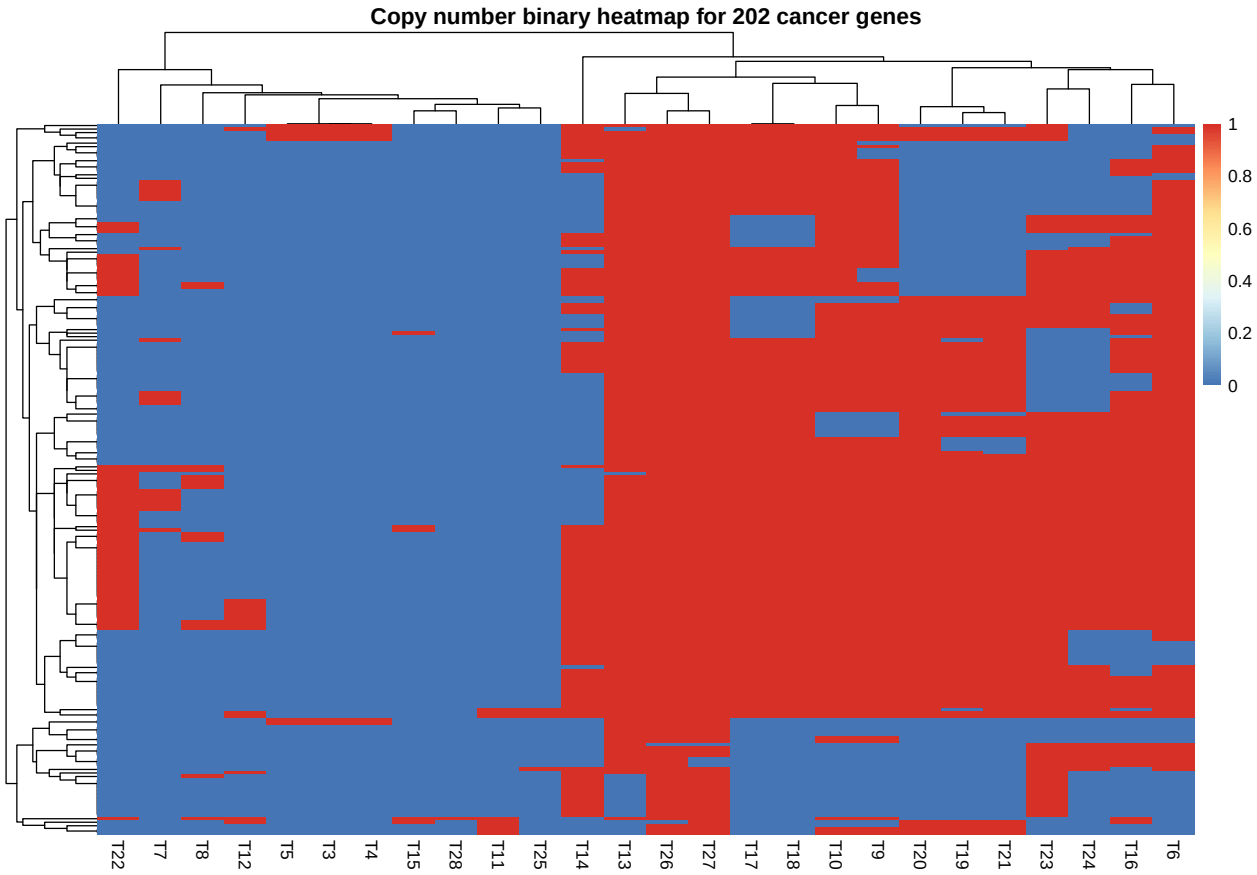
A binary yes/no as to whether a gene/sample exceeds the copy number threshold. This will be the input for the elastic net model later on.

```
cnMatrix =
cancer_cnv %>%  mutate(bin = if_else(isCFE==T, 1, 0)) %>%
  select(bin,manuscript_id,gene) %>% spread(key=manuscript_id, value=bin) %>%
  column_to_rownames("gene") %>% as.matrix()

cnMatrix[1:4,1:4]
```

```
##          T10 T11 T12 T13
## cnABL1     1   0   0   1
## cnACVR1B   1   0   0   1
## cnACVR2A   1   0   0   1
## cnAJUBA    1   0   0   1
```

Heatmap copy number events post threshold

```
pheatmap(cnMatrix, show_rownames = F,
         main = paste("Copy number binary heatmap for",
                      nrow(cnMatrix), "cancer genes"))
```



RNAseq data

```
rna_raw = read_tsv(here("drug_screen_analysis",
                        "RNAseq_data", "SU2C_raw_counts.txt"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
rna_raw = rna_raw %>% column_to_rownames("X1")
rna_raw = as.matrix(rna_raw)
```

Ensure we're not missing ID mappings for any RNA samples.

```
stopifnot(length(setdiff(colnames(rna_raw), sampleMap$RNAseq_ID))==0 &
            length(setdiff(sampleMap$RNAseq_ID,colnames(rna_raw))) == 0)
```

Translate RNAseq IDs to manuscript IDs.

```
colnames(rna_raw) = sampleMap$manuscript_id[match(colnames(rna_raw),
                                                  sampleMap$RNAseq_ID)]
```

Are there duplicates/replicates here?

```
stopifnot(sum(duplicated(colnames(rna_raw)))==0)
```

Convert IDs to geneSymbol

This function:

loadOrganoidHubrechtRNAseq(Log = TRUE, geneSymbol = TRUE, batchNorm = TRUE, aveReps = TRUE, removeZeros = TRUE)

is not part of our legacy code, but the options give us some idea of what needs to be done.

Read gene ids in from our metadata.

```
geneIDs = read_tsv(here("external", "geneLengths_GRCh37p13.txt"))
```

```
## Parsed with column specification:
## cols(
##   `Gene stable ID` = col_character(),
##   `Gene description` = col_character(),
##   `Gene start (bp)` = col_double(),
##   `Gene end (bp)` = col_double(),
##   `Gene name` = col_character(),
##   `HGNC symbol` = col_character()
## )
```

```
geneIDs = geneIDs %>% select(ensembl_gene_id=`Gene stable ID`,
                             description = `Gene description`,
                             gene_name = `Gene name`,
                             geneSymbol = `HGNC symbol`) %>%
  distinct()
```

Convert the ensembl IDs to gene symbols.

```
geneEx = as.data.frame(rna_raw) %>% rownames_to_column("ensembl_gene_id") %>%
  left_join(.,select(geneIDs, ensembl_gene_id, geneSymbol),
            by = "ensembl_gene_id") %>%
  select(-ensembl_gene_id) %>% select(geneSymbol, everything()) %>% na.omit()
```

Mean expression duplicate genes

Gene names often have multiple ensembl IDs, resulting in duplicates. Aggregate duplicate gene symbols by taking the average.

```
summarize_expression_duplicate_ids <- function(
  mat, id_column, f=colMeans, final_gene_symbol_colname="GeneSymbol"
  ){
  require(dplyr)

  print(paste("Starting with gene expression matrix containing",
              nrow(mat), "rows."))

  #Easiest way to write functions with dplyr is to standardize the column name

  if(id_column != "symbol"){
```

```
  colnames(mat)[colnames(mat)==id_column] <- "symbol"
}

#Make frequency table
id_table <- as.data.frame(table(mat$symbol))

#Identify duplicate genes
dups <- id_table$Var1[id_table$Freq > 1]
stopifnot(length(dups) == length(unique(dups)))
print(paste("Number of genes with duplicate names:", length(dups)))

#Set aside rows with unique gene names
nodup_df <- mat[!mat$symbol %in% dups,]

#Set aside rows with duplicate ids
dup_df <- mat[mat$symbol %in% dups,]
stopifnot(nrow(nodup_df) + nrow(dup_df) == nrow(mat))

#Sort by recurring id
dup_df <- dup_df[order(dup_df$symbol),]

print(paste("Number of rows with duplicate gene ids:", nrow(dup_df)))

#Mean expression fpkm of genes with the same symbol
mean_exps <- matrix(ncol = ncol(dup_df)-1,
                    nrow=0) #Empty matrix, -1 gene symbol column

for (i in 1:length(unique(dup_df$symbol))){
  #Subset rows with same symbol, discard symbol column...
  #...then apply aggregate function
  exp <- f(as.matrix(dup_df[dup_df$symbol==unique(dup_df$symbol)[i], -1]))
  mean_exps <- rbind(mean_exps, exp)
}
stopifnot(nrow(mean_exps) == length(unique(dup_df$symbol)))

rownames(mean_exps) <- unique(dup_df$symbol)
mean_exps <- as.data.frame(mean_exps) %>% rownames_to_column("symbol")

dedupped_df <- rbind(mean_exps, nodup_df)
dedupped_df <- dedupped_df[order(dedupped_df$symbol),]


#All symbols should not be unique
stopifnot(length(unique(dedupped_df$symbol))==length(dedupped_df$symbol))
stopifnot(nrow(mat) - #starting number
            #rows with duplicate genes...
            nrow(dup_df) +
            #...which condense down into this many unique genes...
            length(dups) ==
            #...should equal the number of rows in the final matrix
            nrow(dedupped_df))

print(paste("Number of genes after applying", substitute(f),
```

```
              "to duplicate ids:", nrow(dedupped_df)))

  #For estimate, the column with identifiers HAS to be called
  #GeneSymbol or EntrezGeneID
  colnames(dedupped_df)[colnames(dedupped_df)=="symbol"] <- final_gene_symbol_colname

  return(dedupped_df)
}
```

```
geneEx = summarize_expression_duplicate_ids(
  geneEx, id_column = "geneSymbol",
  f = colMeans, final_gene_symbol_colname="GeneSymbol")
```

```
## [1] "Starting with gene expression matrix containing 37308 rows."
## [1] "Number of genes with duplicate names: 2004"
## [1] "Number of rows with duplicate gene ids: 5642"
## [1] "Number of genes after applying colMeans to duplicate ids: 33670"
```

The row names are numeric integers right now, we need to delete them before we can set the gene symbol as the row name.

```
rownames(geneEx) = NULL
geneEx = geneEx %>% column_to_rownames("GeneSymbol") %>% as.matrix()
```

Remove zero rows

```
print(paste("Genes before moving zero rows:", nrow(geneEx)))
```

```
## [1] "Genes before moving zero rows: 33670"
```

```
geneEx = geneEx[rowSums(geneEx) != 0,]
print(paste("Genes after removing zero rows:", nrow(geneEx)))
```

```
## [1] "Genes after removing zero rows: 25201"
```

Log transform

Pseudocount of 0.1 to avoid taking log of 0.

```
geneEx = log2(geneEx + 0.1)
dim(geneEx)
```

```
## [1] 25201    31
```

Heatmap RNA expression

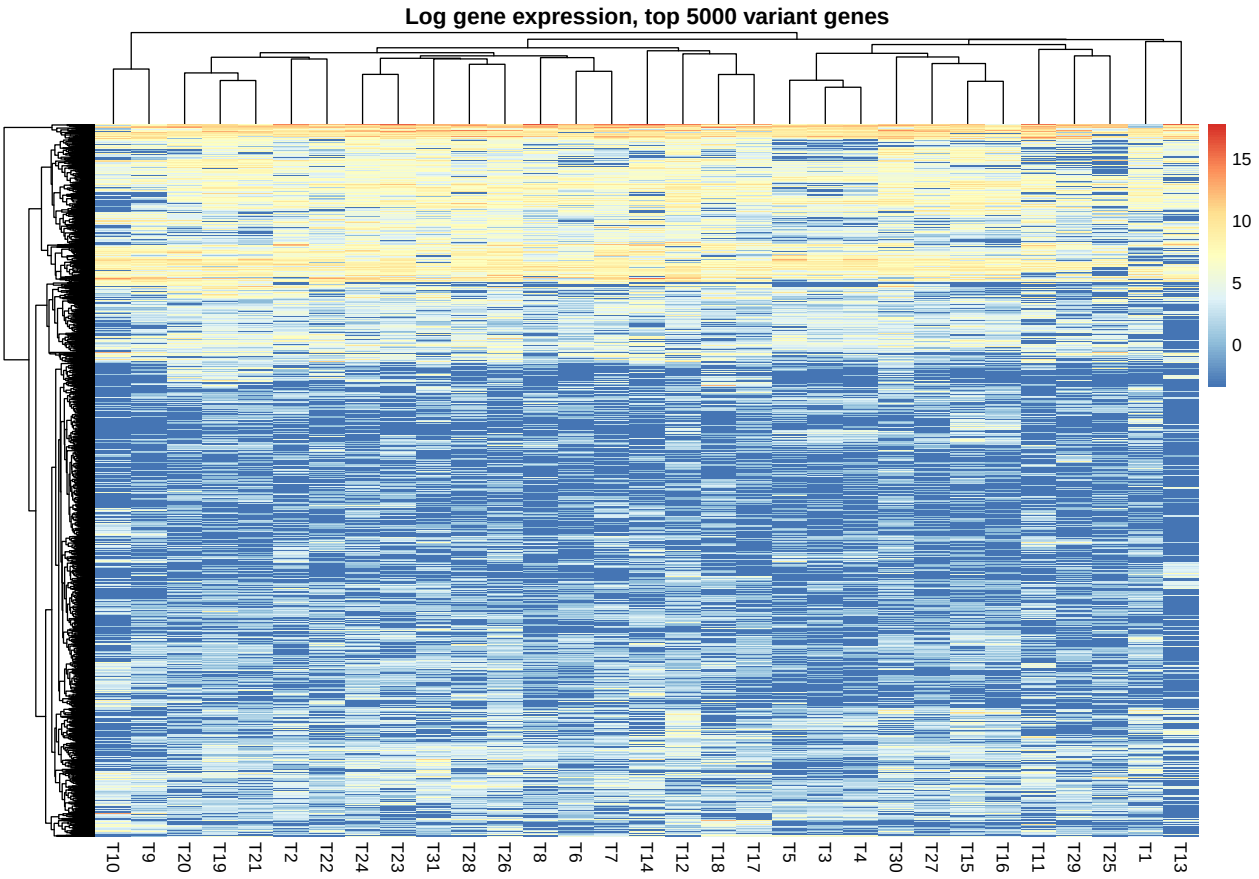
Calculate genewise variance.

```
gexvar = apply(geneEx, 1, var)
gexvar = sort(gexvar, decreasing = T)
head(gexvar, 10)
```

```
##     XIST    BTBD6   CDKN2A   RPS4Y1    DDX3Y     STC2  TXLNG2P    TUSC3
## 38.35824 37.96102 37.84848 37.39365 35.34645 35.27149 34.26816 33.89242
```

```
##     LDHB  SLC14A1
## 30.35175 29.58986
```

```
pheatmap(geneEx[rownames(geneEx) %in% names(gexvar[1:5000]),],
         show_rownames = F,
         main = "Log gene expression, top 5000 variant genes")
```



Single drug response

Read in the single drug response data from Sanger.

```
single_drug = read_csv(here("HUB_Org_Panc_28Jan19",
                            "single_agent_data",
                            "fitted_data_GDSC_Org_Panc_GDSC_28Jan19_1159.csv"))
```

```
## Parsed with column specification:
## cols(
##   MASTER_CELL_ID = col_double(),
##   CELL_LINE_NAME = col_character(),
##   TCGA_DESC = col_logical(),
##   DRUG_ID_lib = col_double(),
##   DATE_CREATED = col_date(format = ""),
##   DRUG_NAME = col_character(),
##   PUTATIVE_TARGET = col_character(),
##   max_conc_micromolar = col_double(),
##   IC50_nat_log = col_double(),
```

```
##   AUC = col_double(),
##   RMSE = col_double()
## )
```

```
single_drug %>% dplyr::group_by(DRUG_NAME, PUTATIVE_TARGET) %>% head(10)
```

```
## # A tibble: 10 x 11
##    MASTER_CELL_ID CELL_LINE_NAME TCGA_DESC DRUG_ID_lib DATE_CREATED DRUG_NAME
##             <dbl> <chr>          <lgl>           <dbl> <date>       <chr>
##  1           2555 HUB-08-B2-002A NA               1011 2018-09-03   Navitocl~
##  2           2555 HUB-08-B2-002A NA               1011 2018-09-18   Navitocl~
##  3           2556 HUB-08-B2-007B NA               1011 2018-10-16   Navitocl~
##  4           2556 HUB-08-B2-007B NA               1011 2018-11-06   Navitocl~
##  5           2558 HUB-08-B2-010A NA               1011 2018-09-03   Navitocl~
##  6           2558 HUB-08-B2-010A NA               1011 2018-10-02   Navitocl~
##  7           2559 HUB-08-B2-010B NA               1011 2018-09-03   Navitocl~
##  8           2559 HUB-08-B2-010B NA               1011 2018-09-18   Navitocl~
##  9           2560 HUB-08-B2-013A NA               1011 2018-11-02   Navitocl~
## 10           2563 HUB-08-B2-019A NA               1011 2018-08-03   Navitocl~
## # ... with 5 more variables: PUTATIVE_TARGET <chr>, max_conc_micromolar <dbl>,
## #   IC50_nat_log <dbl>, AUC <dbl>, RMSE <dbl>
```

Drug overview

```
single_drug %>% select(DRUG_NAME, PUTATIVE_TARGET) %>%
  distinct() %>% arrange(DRUG_NAME)
```

```
## # A tibble: 76 x 2
##    DRUG_NAME      PUTATIVE_TARGET
##    <chr>          <chr>
##  1 5-Fluorouracil Antimetabolite (DNA & RNA)
##  2 Afatinib       ERBB2, EGFR
##  3 Alisertib      AURKA
##  4 Alpelisib      PI3Kalpha
##  5 Avagacestat    Amyloid beta20, Amyloid beta40
##  6 Axitinib       PDGFR, KIT, VEGFR
##  7 AZD0156        ATM
##  8 AZD4547        FGFR1, FGFR2, FGFR3
##  9 AZD6482        PI3Kbeta
## 10 AZD6738        ATR
## # ... with 66 more rows
```

Fix a typo

```
single_drug[single_drug$PUTATIVE_TARGET=="DNA akylating agent",
            "PUTATIVE_TARGET"] <- "DNA alkylating agent"
```

General overview of drugs in screen:

```
print(paste("Total number of single drugs in screen:",
            length(unique(single_drug$DRUG_NAME))))
## [1] "Total number of single drugs in screen: 76"
print(paste("Unique targets in single drug screen:",
            length(unique(single_drug$PUTATIVE_TARGET))))
## [1] "Unique targets in single drug screen: 65"
```

```
print(paste("Cell lines in single drug screen:",
            length(unique(single_drug$CELL_LINE_NAME))))
## [1] "Cell lines in single drug screen: 24"
```

Convert to manuscript IDs

Join the IDs that we have to the manuscript IDs

```
single_drug = right_join(select(sampleMap, HUB.ID, manuscript_id),
          dplyr::rename(single_drug, HUB.ID = CELL_LINE_NAME),
          by = "HUB.ID") %>% select(manuscript_id, everything()) %>%
  arrange(manuscript_id)

#Ensure we haven't lost any samples
stopifnot(nrow(filter(single_drug, is.na(HUB.ID)))==0)
```

Out of the 31 manuscript IDs, these are the ones without single_drug data

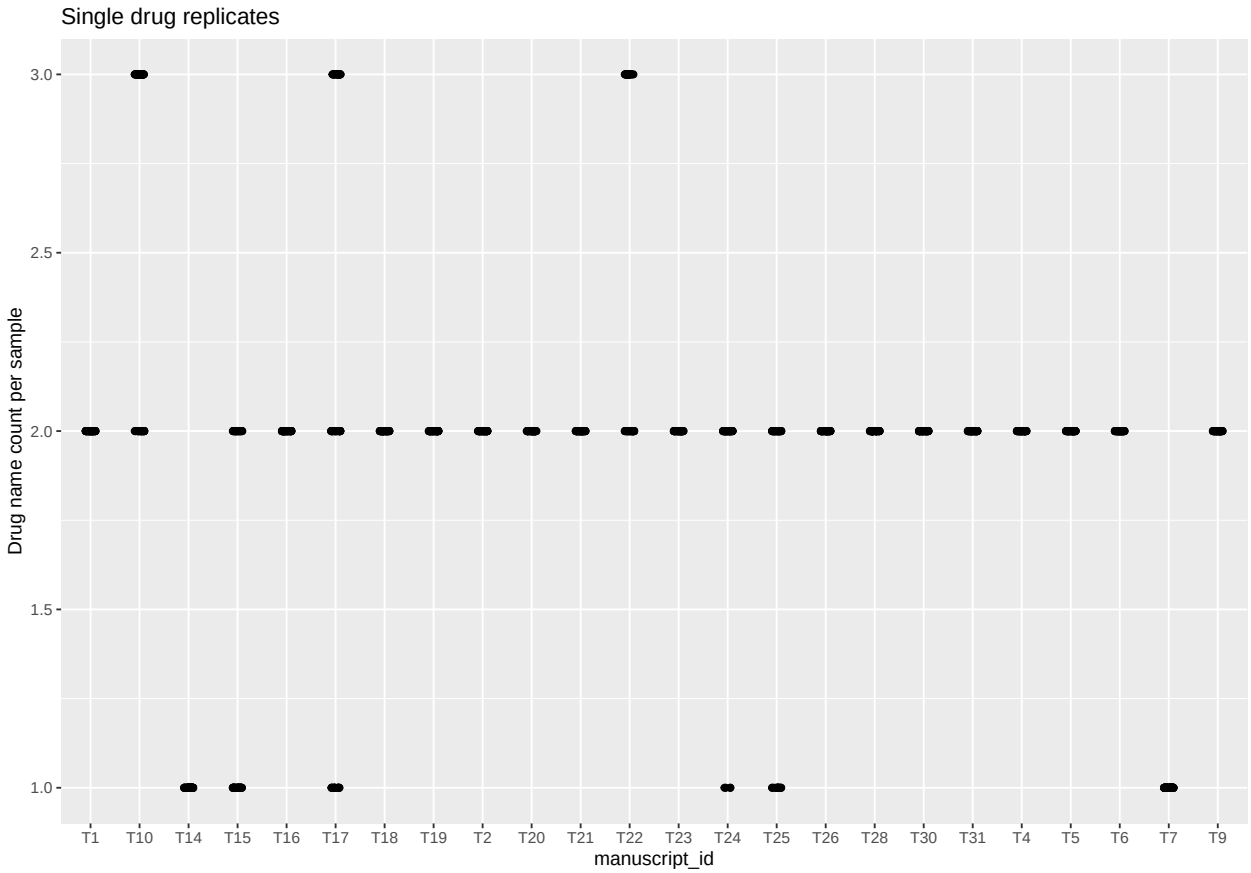
```
setdiff(sampleMap$manuscript_id,single_drug$manuscript_id)
```

```
## [1] "T3"  "T8"  "T11" "T12" "T13" "T27" "T29"
```

Replicate visualization

Most drugs have two replicates per sample, a few have 1 or 3.

```
single_drug %>% group_by(manuscript_id, DRUG_NAME) %>%
  summarise(replicate=n()) %>%
  ggplot(aes(x=manuscript_id, y = replicate)) +
  geom_jitter(height=0, width=0.1) +
  ggtitle("Single drug replicates") + ylab("Drug name count per sample")
```



Do the replicates all have the same drug concentration?

```
single_drug %>% group_by(manuscript_id, DRUG_NAME) %>%
  summarise(n=n(), mean_conc = mean(max_conc_micromolar),
            stdev_conc=sd(max_conc_micromolar)) %>%
  ungroup() %>%  select(stdev_conc) %>% summary()
```

```
##    stdev_conc
##  Min.   :0
##  1st Qu.:0
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
##  NA's   :220
```

Stdev of the concentration is either zero (all concentrations identical) or NA (no replicate), so it would appear that all replicates have the same drug concentration.

Similarly, the concentration of drugs does not vary between samples.

```
single_drug %>% group_by(DRUG_NAME) %>%
  summarise(n=n(), mean_conc = mean(max_conc_micromolar),
            stdev_conc=sd(max_conc_micromolar)) %>%
  ungroup() %>%  select(stdev_conc) %>% summary()
```

```
##    stdev_conc
##  Min.   :0
##  1st Qu.:0
```

```
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

Median of replicates

For association analyses, we use the median values of drug response metrics, computed across organoid-drug replicates.

```
single_drug_median =
single_drug %>% group_by(manuscript_id, DRUG_NAME, PUTATIVE_TARGET) %>%
  summarise(med_IC50_natlog = median(IC50_nat_log), med_AUC = median(AUC),
            med_RMSE = median(RMSE)) %>%
  ungroup()

#Ensure uniqueness
stopifnot(length(unique(
  paste0(single_drug_median$manuscript_id,
         single_drug_median$DRUG_NAME)))==nrow(single_drug_median))
```

IC50 matrix

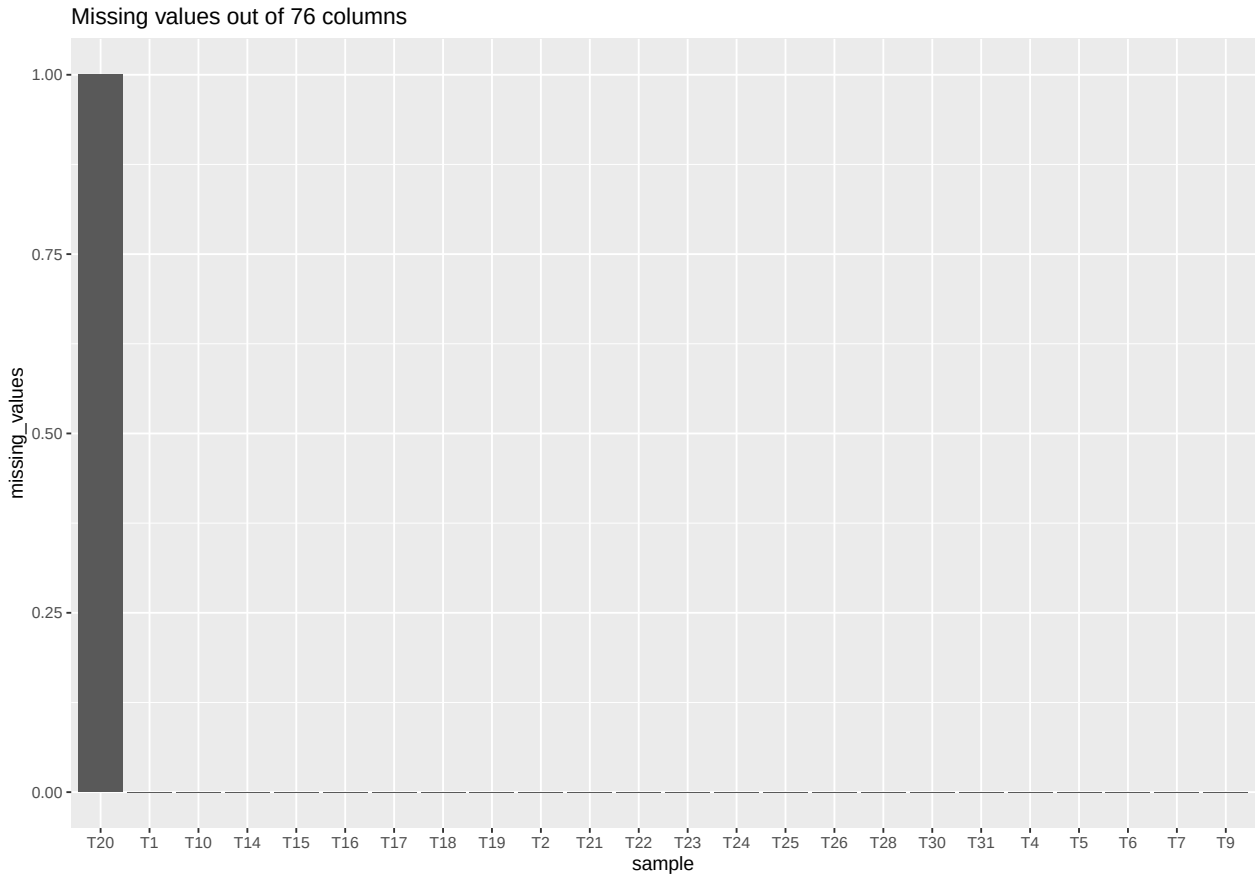
Not all drugs are available for all cell lines. What to do about missing values?

```
IC50 =
single_drug_median %>% select(manuscript_id, DRUG_NAME, med_IC50_natlog) %>%
  spread(key=DRUG_NAME, value=med_IC50_natlog) %>%
  column_to_rownames("manuscript_id") %>%
  as.matrix()

IC50[1:4,1:4]
```

```
##     5-Fluorouracil Afatinib Alisertib Alpelisib
## T1        5.655602 3.357282  4.886056  6.199309
## T10       6.568933 2.184155  4.415300  4.639153
## T14       5.602219 3.545439  5.294620  4.065525
## T15       6.316899 2.712190  4.183508  3.780825
```

Looks like there's only one missing value in the current dataset: T20 for Alisertib.

```
tibble(sample = rownames(IC50), missing_values = rowSums(is.na(IC50))) %>%
  arrange(desc(missing_values)) %>%
  ggplot(aes(x=factor(sample, levels=sample), y = missing_values)) +
  geom_bar(stat="identity") +
  xlab("sample") +
  ggtitle(paste("Missing values out of", ncol(IC50), "columns"))
```



AUC matrix

```
AUC =
single_drug_median %>% select(manuscript_id, DRUG_NAME, med_AUC) %>%
  spread(key=DRUG_NAME, value=med_AUC) %>%
  column_to_rownames("manuscript_id") %>%
  as.matrix()

AUC[1:4,1:4]
```

```
##     5-Fluorouracil  Afatinib Alisertib Alpelisib
## T1       0.9213827 0.9061241 0.9158866 0.9556016
## T10      0.9636951 0.8777542 0.9293995 0.9334333
## T14      0.9328335 0.9194915 0.9427223 0.9058218
## T15      0.9380989 0.8506935 0.8916628 0.8742383
```

ICD 50 heatmap

Half maximal inhibitory concentration: The concentration at which half of all cells die. We're working with natural logs, so we have some value below 0.

```
summary(single_drug_median$med_IC50_natlog)
```

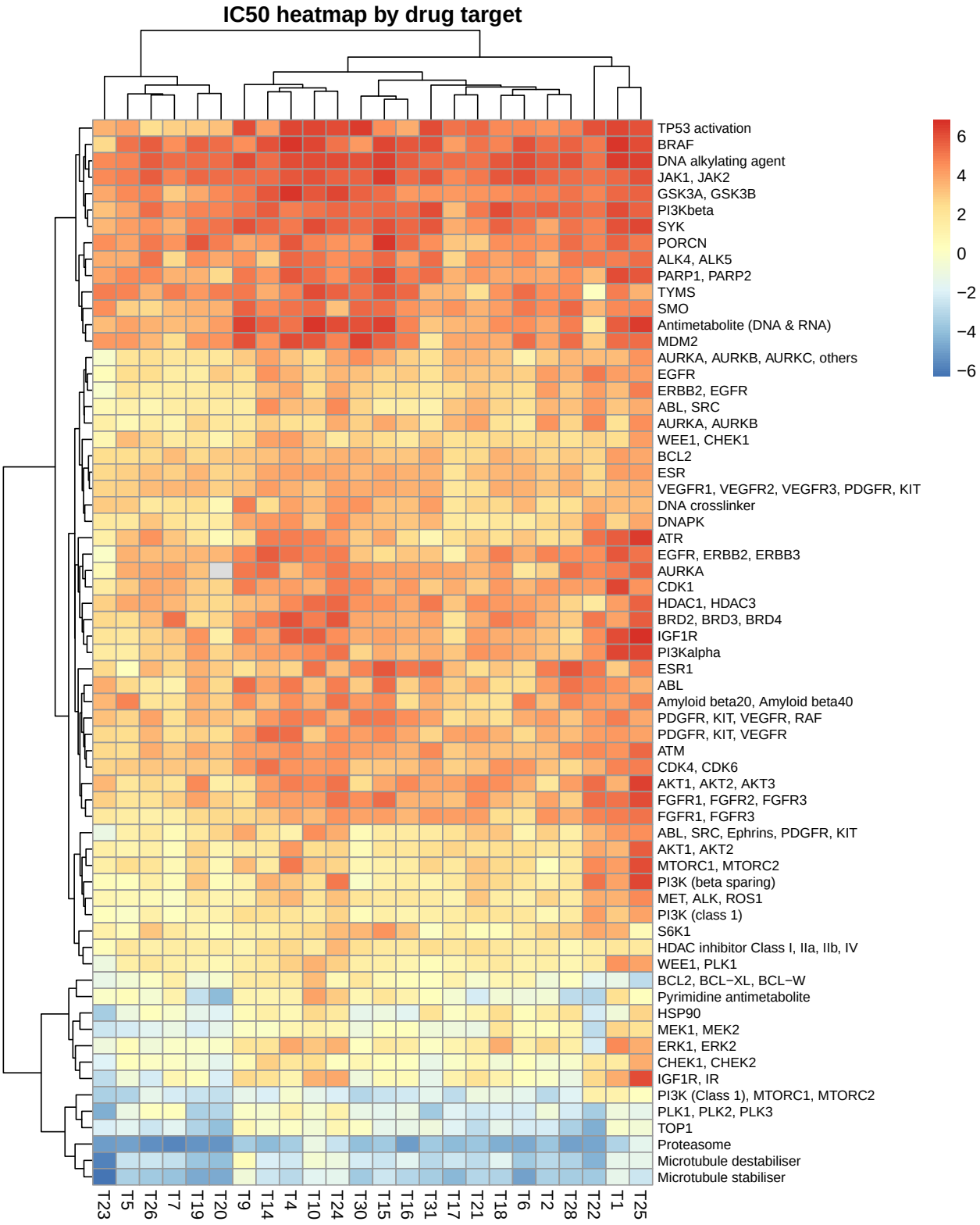
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##  -6.740   1.376   3.098   2.576   4.429   7.661
```

Average the same drug targets together.

```
hm_1drug =
single_drug %>% group_by(manuscript_id, PUTATIVE_TARGET) %>%
  summarise(med_IC50_natlog = median(IC50_nat_log)) %>%
  ungroup %>% select(manuscript_id, PUTATIVE_TARGET, med_IC50_natlog) %>%
  tidyr::spread(key=manuscript_id, value=med_IC50_natlog) %>%
  column_to_rownames("PUTATIVE_TARGET")
```

```
pheatmap(hm_1drug, fontsize_row = 8, main="IC50 heatmap by drug target")
```


IC50 heatmap by drug target
TP53 activation
BRAF
DNA alkylating agent
JAK1, JAK2
GSK3A, GSK3B
PI3Kbeta
SYK
PORCN
ALK4, ALK5
PARP1, PARP2
TYMS
SMO
Antimetabolite (DNA & RNA)
MDM2
AURKA, AURKB, AURKC, others
EGFR
ERBB2, EGFR
ABL, SRC
AURKA, AURKB
WEE1, CHEK1
BCL2
ESR
VEGFR1, VEGFR2, VEGFR3, PDGFR, KIT
DNA crosslinker
DNAPK
ATR
EGFR, ERBB2, ERBB3
AURKA
CDK1
HDAC1, HDAC3
BRD2, BRD3, BRD4
IGF1R
PI3Kalpha
ESR1
ABL
Amyloid beta20, Amyloid beta40
PDGFR, KIT, VEGFR, RAF
PDGFR, KIT, VEGFR
ATM
CDK4, CDK6
AKT1, AKT2, AKT3
FGFR1, FGFR2, FGFR3
FGFR1, FGFR3
ABL, SRC, Ephrins, PDGFR, KIT
AKT1, AKT2
MTORC1, MTORC2
PI3K (beta sparing)
MET, ALK, ROS1
PI3K (class 1)
S6K1
HDAC inhibitor Class I, IIa, IIb, IV
WEE1, PLK1
BCL2, BCL−XL, BCL−W
Pyrimidine antimetabolite
HSP90
MEK1, MEK2
ERK1, ERK2
CHEK1, CHEK2
IGF1R, IR
PI3K (Class 1), MTORC1, MTORC2
PLK1, PLK2, PLK3
TOP1
Proteasome
Microtubule destabiliser
Microtubule stabiliser
T23
T5
T26
T7
T19
T20
T9
T14
T4
T10
T24
T30
T15
T16
T31
T17
T21
T18
T6
T2
T28
T22
T1
T25
6
4
2
0
−2
−4
−6

AUC heatmap

```
single_drug %>% group_by(manuscript_id, PUTATIVE_TARGET) %>%
  summarise(medAUC = median(AUC)) %>%
  ungroup %>% select(manuscript_id, PUTATIVE_TARGET, medAUC) %>%
  tidyr::spread(key=manuscript_id, value=medAUC) %>%
  column_to_rownames("PUTATIVE_TARGET") %>%
  pheatmap(.,fontsize_row = 8, main="AUC heatmap by drug target")
```


AUC heatmap by drug target
PORCN
CDK4, CDK6
GSK3A, GSK3B
ESR
VEGFR1, VEGFR2, VEGFR3, PDGFR, KIT
BCL2
DNA alkylating agent
JAK1, JAK2
BRAF
PI3Kbeta
SYK
PARP1, PARP2
ALK4, ALK5
ATM
MDM2
SMO
TYMS
Antimetabolite (DNA & RNA)
HDAC1, HDAC3
TOP1
HSP90
EGFR, ERBB2, ERBB3
ABL, SRC, Ephrins, PDGFR, KIT
AURKA
AURKA, AURKB, AURKC, others
ESR1
Amyloid beta20, Amyloid beta40
DNA crosslinker
FGFR1, FGFR2, FGFR3
PDGFR, KIT, VEGFR, RAF
CDK1
PDGFR, KIT, VEGFR
BRD2, BRD3, BRD4
IGF1R
PI3Kalpha
AKT1, AKT2, AKT3
ABL
FGFR1, FGFR3
TP53 activation
S6K1
CHEK1, CHEK2
WEE1, PLK1
PI3K (beta sparing)
PI3K (Class 1), MTORC1, MTORC2
MTORC1, MTORC2
AKT1, AKT2
PI3K (class 1)
ATR
DNAPK
WEE1, CHEK1
HDAC inhibitor Class I, IIa, IIb, IV
Microtubule stabiliser
Proteasome
ABL, SRC
EGFR
ERBB2, EGFR
AURKA, AURKB
MET, ALK, ROS1
Pyrimidine antimetabolite
IGF1R, IR
BCL2, BCL−XL, BCL−W
ERK1, ERK2
MEK1, MEK2
Microtubule destabiliser
PLK1, PLK2, PLK3
T20 T23 T7 T26 T5 T30 T15 T16 T17 T2 T31 T18 T6 T19 T28 T22 T10 T24 T9 T14 T4 T21 T1 T25
0.8
0.6
0.4
0.2

Top drugs - Single response data

Most potent single drugs in the screening, according to IC50, are those targeting:

```
tibble(DRUG_NAME = factor(names(sort(colSums(IC50))[1:5]),
                          levels = names(sort(colSums(IC50))[1:5]))) %>%
  left_join(.,select(single_drug_median, DRUG_NAME, PUTATIVE_TARGET),
            by = "DRUG_NAME") %>% unique()
```

```
## Warning: Column `DRUG_NAME` joining factor and character vector, coercing into
## character vector
```

```
## # A tibble: 5 x 2
##   DRUG_NAME   PUTATIVE_TARGET
##   <chr>       <chr>
## 1 Bortezomib  Proteasome
## 2 Docetaxel   Microtubule stabiliser
## 3 Vinblastine Microtubule destabiliser
## 4 Paclitaxel  Microtubule stabiliser
## 5 SN-38       TOP1
```

Most potent single drugs in the screening, according to AUC, are those targeting:

```
tibble(DRUG_NAME = factor(names(sort(colSums(AUC))[1:5]),
                          levels = names(sort(colSums(AUC))[1:5]))) %>%
  left_join(.,select(single_drug_median, DRUG_NAME, PUTATIVE_TARGET),
            by = "DRUG_NAME") %>% unique()
```

```
## Warning: Column `DRUG_NAME` joining factor and character vector, coercing into
## character vector
```

```
## # A tibble: 5 x 2
##   DRUG_NAME   PUTATIVE_TARGET
##   <chr>       <chr>
## 1 Gemcitabine Pyrimidine antimetabolite
## 2 Vinblastine Microtubule destabiliser
## 3 BMS-754807  IGF1R, IR
## 4 Navitoclax  BCL2, BCL-XL, BCL-W
## 5 BI-2536     PLK1, PLK2, PLK3
```

The least effective single drugs, accoring to IC50, are those targeting:

```
tibble(DRUG_NAME = factor(
  names(sort(colSums(IC50), decreasing=T)[1:5]),
  levels = names(sort(colSums(IC50), decreasing=T)[1:5]))
  ) %>%
  left_join(.,select(single_drug_median, DRUG_NAME, PUTATIVE_TARGET),
            by = "DRUG_NAME") %>% unique()
```

```
## Warning: Column `DRUG_NAME` joining factor and character vector, coercing into
## character vector
```

```
## # A tibble: 5 x 2
##   DRUG_NAME    PUTATIVE_TARGET
##   <chr>        <chr>
## 1 Temozolomide DNA alkylating agent
## 2 Ruxolitinib  JAK1, JAK2
## 3 Oxaliplatin  DNA alkylating agent
## 4 Dabrafenib   BRAF
```

```
## 5 Olaparib     PARP1, PARP2
```

The least effective single drugs, accoring to AUC, are those targeting:

```
tibble(DRUG_NAME = factor(
  names(sort(colSums(AUC), decreasing=T)[1:5]),
  levels = names(sort(colSums(AUC), decreasing=T)[1:5]))
  ) %>%
  left_join(.,select(single_drug_median, DRUG_NAME, PUTATIVE_TARGET),
            by = "DRUG_NAME") %>% unique()
```

```
## Warning: Column `DRUG_NAME` joining factor and character vector, coercing into
## character vector
```

```
## # A tibble: 5 x 2
##   DRUG_NAME    PUTATIVE_TARGET
##   <chr>        <chr>
## 1 Fulvestrant  ESR
## 2 Tivozanib    VEGFR1, VEGFR2, VEGFR3, PDGFR, KIT
## 3 Temozolomide DNA alkylating agent
## 4 Ruxolitinib  JAK1, JAK2
## 5 Ribociclib   CDK4, CDK6
```

Combination response data

We use four measures of synergy in drug combinations:

- delta AUC (expected AUC - observed AUC)
- delta fAUC: same as above but the AUC is computed from a curve fit to the data
- z-score: confidence in the shift in the curve in the horizontal direction (IC50 shift)
- delta E-max: change in total cell kill at maximum concentration

For association analyses, we use the median values of the above, computed across organoid-combination replicates.

Load drug combinations

```
combi_input = read_csv(here("HUB_Org_Panc_28Jan19","combination_data",
                            "dfSyn_GDSC_Org_Panc_28Jan19_1201.csv"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   CELL_LINE_NAME = col_character(),
##   drug = col_character(),
##   lib_drug = col_character(),
##   norm_neg_pos = col_character(),
##   CL_SPEC = col_character(),
##   drug_spec = col_character(),
##   time_stamp = col_datetime(format = ""),
##   sw_version = col_character(),
##   RESEARCH_PROJECT = col_character(),
##   AL_UID = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
head(combi_input)
```

```
## # A tibble: 6 x 40
##   CELL_LINE_NAME    CL  maxc drug  DRUGSET_ID lib_drug BARCODE SCAN_ID
##   <chr>          <dbl> <dbl> <chr>      <dbl> <chr>      <dbl>   <dbl>
## 1 HUB-08-B2-002A 10229  2.67 522_~        522 L19        42710   41517
## 2 HUB-08-B2-002A 10229  2.67 522_~        522 L19        42471   41303
## 3 HUB-08-B2-010A 10230  2.67 522_~        522 L19        42467   41310
## 4 HUB-08-B2-010A 10230  2.67 522_~        522 L19        43091   41908
## 5 HUB-08-B2-013A 10231  2.67 522_~        522 L19        44530   43091
## 6 HUB-08-B2-019A 10234  2.67 522_~        522 L19        41707   40558
## # ... with 32 more variables: norm_neg_pos <chr>, CL_SPEC <chr>,
## #   drug_spec <chr>, time_stamp <dttm>, sw_version <chr>,
## #   RESEARCH_PROJECT <chr>, scal <dbl>, DRUG_ID_anch <dbl>, CONC_anch <dbl>,
## #   ANCHOR_VIABILITY <dbl>, ANCHOR_VIABILITY_SD <dbl>, AL_UID <chr>,
## #   LIBRARY_XMID <dbl>, LIBRARY_SCAL <dbl>, LIBRARY_AUC <dbl>,
## #   LIBRARY_RMSE <dbl>, SYNERGY_XMID <dbl>, scale <dbl>, SYNERGY_OBS_AUC <dbl>,
## #   SYNERGY_RMSE <dbl>, SYNERGY_EXP_AUC <dbl>, SYNERGY_DELTA_AUC <dbl>,
## #   LIBRARY_fAUC <dbl>, SYNERGY_EXP_fAUC <dbl>, SYNERGY_OBS_fAUC <dbl>,
## #   SYNERGY_DELTA_fAUC <dbl>, SYNERGY_ZSCORE <dbl>, LIBRARY_EMAX <dbl>,
## #   SYNERGY_EXP_EMAX <dbl>, SYNERGY_OBS_EMAX <dbl>, SYNERGY_DELTA_EMAX <dbl>,
## #   DRUG_ID_lib <dbl>
```

These are the same organoids as the single drug screen.

```
identical(sort(unique(combi_input$CELL_LINE_NAME)),
          sort(unique(single_drug$HUB.ID)))
```

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

Load file for mapping drug IDs to drug names.

```
drug_decode = read_csv(here("HUB_Org_Panc_28Jan19","single_agent_data",
                            "drug_decode_GDSC_Org_Panc_GDSC_28Jan19_1159.csv"))
```

```
## Parsed with column specification:
## cols(
##   DRUG_ID = col_double(),
##   DRUG_NAME = col_character(),
##   PUTATIVE_TARGET = col_character(),
##   OWNED_BY = col_character(),
##   WEBRELEASE = col_character(),
##   DRUG_OWNER = col_double()
## )
```

```
head(drug_decode)
```

```
## # A tibble: 6 x 6
##   DRUG_ID DRUG_NAME  PUTATIVE_TARGET              OWNED_BY WEBRELEASE DRUG_OWNER
##     <dbl> <chr>      <chr>                       <chr>    <chr>           <dbl>
## 1    1011 Navitoclax BCL2, BCL-XL, BCL-W         GDSC     Y                1046
## 2    1053 MK-2206    AKT1, AKT2                  GDSC     Y                1046
## 3    1085 Sorafenib  PDGFR, KIT, VEGFR, RAF      GDSC     N                1046
## 4    1057 Dactolisib PI3K (Class 1), MTORC1, MTO~ GDSC     Y                1046
## 5    1559 Luminespib HSP90                       GDSC     N                1046
## 6    1022 AZD7762    CHEK1, CHEK2                GDSC     Y                1046
```

Add the drug names and targets for both library and anchor, and the manuscript IDs from sampleMap.

```
combi = left_join(combi_input,
                  select(drug_decode, DRUG_ID, LIBRARY_NAME=DRUG_NAME,
                         LIBRARY_TARGET=PUTATIVE_TARGET),
                  by = c("DRUG_ID_lib"="DRUG_ID")) %>%
  left_join(., select(drug_decode, DRUG_ID, ANCHOR_NAME=DRUG_NAME,
                      ANCHOR_TARGET=PUTATIVE_TARGET),
            by = c("DRUG_ID_anch"="DRUG_ID")) %>%
  left_join(.,select(sampleMap, HUB.ID, manuscript_id),
            by=c("CELL_LINE_NAME"="HUB.ID")) %>%
  mutate(combination = paste(LIBRARY_NAME,ANCHOR_NAME, sep=":")) %>%
  select(manuscript_id, combination, maxc, LIBRARY_NAME, ANCHOR_NAME,
         contains("SYNERGY"), everything())

head(combi)
```

```
## # A tibble: 6 x 46
##   manuscript_id combination  maxc LIBRARY_NAME ANCHOR_NAME SYNERGY_XMID
##   <chr>         <chr>       <dbl> <chr>        <chr>              <dbl>
## 1 T1            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         15.3
## 2 T1            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         14.3
## 3 T4            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         15.2
## 4 T4            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         13.8
## 5 T6            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         13.0
## 6 T9            MK-2206:Ge~  2.67 MK-2206      Gemcitabine         11.3
## # ... with 40 more variables: SYNERGY_OBS_AUC <dbl>, SYNERGY_RMSE <dbl>,
## #   SYNERGY_EXP_AUC <dbl>, SYNERGY_DELTA_AUC <dbl>, SYNERGY_EXP_fAUC <dbl>,
## #   SYNERGY_OBS_fAUC <dbl>, SYNERGY_DELTA_fAUC <dbl>, SYNERGY_ZSCORE <dbl>,
## #   SYNERGY_EXP_EMAX <dbl>, SYNERGY_OBS_EMAX <dbl>, SYNERGY_DELTA_EMAX <dbl>,
## #   CELL_LINE_NAME <chr>, CL <dbl>, drug <chr>, DRUGSET_ID <dbl>,
## #   lib_drug <chr>, BARCODE <dbl>, SCAN_ID <dbl>, norm_neg_pos <chr>,
## #   CL_SPEC <chr>, drug_spec <chr>, time_stamp <dttm>, sw_version <chr>,
## #   RESEARCH_PROJECT <chr>, scal <dbl>, DRUG_ID_anch <dbl>, CONC_anch <dbl>,
## #   ANCHOR_VIABILITY <dbl>, ANCHOR_VIABILITY_SD <dbl>, AL_UID <chr>,
## #   LIBRARY_XMID <dbl>, LIBRARY_SCAL <dbl>, LIBRARY_AUC <dbl>,
## #   LIBRARY_RMSE <dbl>, scale <dbl>, LIBRARY_fAUC <dbl>, LIBRARY_EMAX <dbl>,
## #   DRUG_ID_lib <dbl>, LIBRARY_TARGET <chr>, ANCHOR_TARGET <chr>
```

Ensure we aren't missing any manuscript IDs.

```
stopifnot(sum(is.na(combi$manuscript_id))==0)
```

Here the library concentration does vary by cell line.

```
combi %>%
  group_by(combination) %>%
  summarise(mean_maxc = mean(maxc),
            sd_maxc = sd(maxc),
            mean_conc_anch = mean(CONC_anch),
            sd_conc_anch = sd(CONC_anch)) %>%
  ungroup %>% select(contains("sd_")) %>% summary()
```

```
##     sd_maxc        sd_conc_anch
##  Min.   :0.1605   Min.   :0.001605
##  1st Qu.:0.8026   1st Qu.:0.001605
##  Median :1.6051   Median :0.001605
```

```
##  Mean   :1.3863   Mean   :0.001605
##  3rd Qu.:1.6051   3rd Qu.:0.001605
##  Max.   :3.2103   Max.   :0.001605
```

Combination drugs and their targets

Library targets by drug name:

```
combi %>% select(DRUG_NAME=LIBRARY_NAME, LIBRARY_TARGET) %>% distinct()
```

```
## # A tibble: 9 x 2
##   DRUG_NAME      LIBRARY_TARGET
##   <chr>          <chr>
## 1 MK-2206        AKT1, AKT2
## 2 Trametinib     MEK1, MEK2
## 3 Linsitinib     IGF1R
## 4 Lapatinib      ERBB2, EGFR
## 5 MK-1775        WEE1, PLK1
## 6 Taselisib      PI3K (beta sparing)
## 7 5-Fluorouracil Antimetabolite (DNA & RNA)
## 8 Sorafenib      PDGFR, KIT, VEGFR, RAF
## 9 SCH772984      ERK1, ERK2
```

Anchor targets:

```
combi %>% select(ANCHOR_NAME, ANCHOR_TARGET) %>% distinct()
```

```
## # A tibble: 2 x 2
##   ANCHOR_NAME ANCHOR_TARGET
##   <chr>       <chr>
## 1 Gemcitabine Pyrimidine antimetabolite
## 2 Trametinib  MEK1, MEK2
```

Drug:Anchor combinations

```
combi %>% group_by(combination, LIBRARY_TARGET, ANCHOR_TARGET) %>%
  summarise(n=n())
```

```
## # A tibble: 11 x 4
##    combination             LIBRARY_TARGET           ANCHOR_TARGET              n
##    <chr>                   <chr>                    <chr>                  <int>
##  1 5-Fluorouracil:Gemcita~ Antimetabolite (DNA & R~ Pyrimidine antimetabo~    46
##  2 Lapatinib:Gemcitabine   ERBB2, EGFR              Pyrimidine antimetabo~    46
##  3 Linsitinib:Gemcitabine  IGF1R                    Pyrimidine antimetabo~    46
##  4 MK-1775:Gemcitabine     WEE1, PLK1               Pyrimidine antimetabo~    46
##  5 MK-1775:Trametinib      WEE1, PLK1               MEK1, MEK2                46
##  6 MK-2206:Gemcitabine     AKT1, AKT2               Pyrimidine antimetabo~    46
##  7 SCH772984:Trametinib    ERK1, ERK2               MEK1, MEK2                46
##  8 Sorafenib:Trametinib    PDGFR, KIT, VEGFR, RAF   MEK1, MEK2                46
##  9 Taselisib:Gemcitabine   PI3K (beta sparing)      Pyrimidine antimetabo~    46
## 10 Taselisib:Trametinib    PI3K (beta sparing)      MEK1, MEK2                46
## 11 Trametinib:Gemcitabine  MEK1, MEK2               Pyrimidine antimetabo~    46
```

```
print(paste("There are", length(unique(combi$combination)),
            "unique drug combinations."))
```

```
## [1] "There are 11 unique drug combinations."
```

Median response metrics

The median value for each response metric will be used to assess synergy.

```
combi_med = combi %>%
  group_by(manuscript_id, combination) %>%
  summarise(median_DELTA_AUC = median(SYNERGY_DELTA_AUC),
            median_DELTA_fAUC = median(SYNERGY_DELTA_fAUC),
            median_ZSCORE = median(SYNERGY_ZSCORE),
            median_DELTA_EMAX = median(SYNERGY_DELTA_EMAX),
            median_XMID = median(SYNERGY_XMID))

head(combi_med) %>%
  select(manuscript_id, combination, contains("median"), everything())
```

```
## # A tibble: 6 x 7
##   manuscript_id combination median_DELTA_AUC median_DELTA_fA~ median_ZSCORE
##   <chr>         <chr>                  <dbl>            <dbl>         <dbl>
## 1 T1            5-Fluorour~         -0.0443          -0.0158         -2.29
## 2 T1            Lapatinib:~         -0.00766         -0.00515        -0.597
## 3 T1            Linsitinib~         -0.0511          -0.0286         -3.17
## 4 T1            MK-1775:Ge~         -0.0141           0.0111          0.179
## 5 T1            MK-1775:Tr~         -0.0443          -0.00852        -0.462
## 6 T1            MK-2206:Ge~         -0.0428          -0.0361         -3.15
## # ... with 2 more variables: median_DELTA_EMAX <dbl>, median_XMID <dbl>
```

Matrices for downstream input (elastic net)

```
median_DELTA_AUC = combi_med %>%
  select(manuscript_id, combination, median_DELTA_AUC) %>%
  spread(key = combination, value=median_DELTA_AUC) %>%
  column_to_rownames("manuscript_id") %>% as.matrix()

median_DELTA_fAUC = combi_med %>%
  select(manuscript_id, combination, median_DELTA_fAUC) %>%
  spread(key = combination, value=median_DELTA_fAUC) %>%
  column_to_rownames("manuscript_id") %>% as.matrix()

median_ZSCORE = combi_med %>%
  select(manuscript_id, combination, median_ZSCORE) %>%
  spread(key = combination, value=median_ZSCORE) %>%
  column_to_rownames("manuscript_id") %>% as.matrix()

median_DELTA_EMAX = combi_med %>%
  select(manuscript_id, combination, median_DELTA_EMAX) %>%
  spread(key = combination, value=median_DELTA_EMAX) %>%
  column_to_rownames("manuscript_id") %>% as.matrix()

#There should be no NAs
stopifnot(Reduce("+",
                 lapply(
```

```
                    list(median_DELTA_AUC, median_DELTA_fAUC,
                         median_ZSCORE, median_DELTA_EMAX),
                    function(x) sum(rowSums(is.na(x))) == 0)
```

Density plots of response variables

- delta AUC (expected AUC - observed AUC)
- delta fAUC: same as above but the AUC is computed from a curve fit to the data
- z-score: confidence in the shift in the curve in the horizontal direction (IC50 shift)
- delta E-max: change in total cell kill at maximum concentration

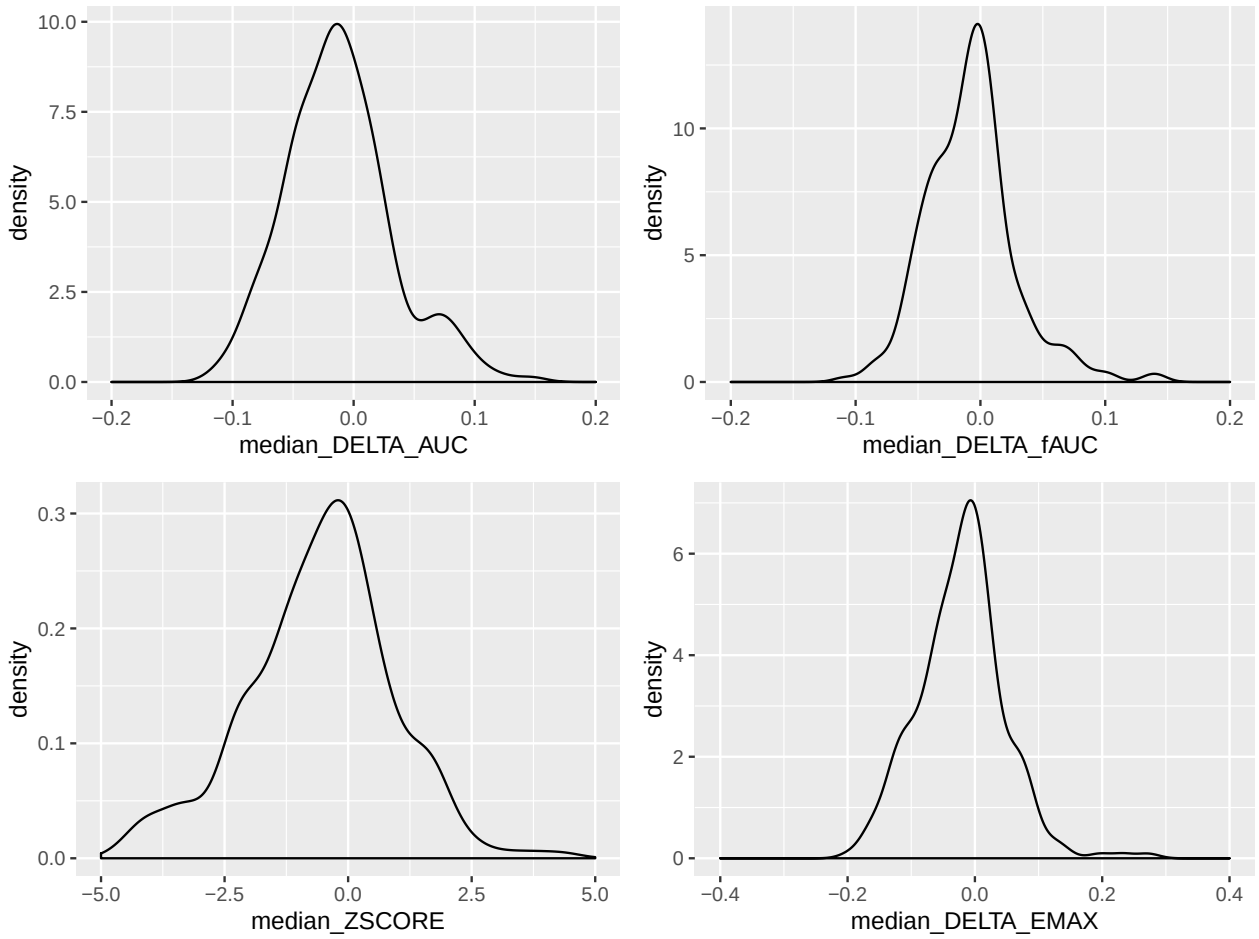
Note: There are a surprising amount of negtative values here, indicating antagonism rather than synergy.

```
ggarrange(combi_med %>% ggplot(aes(x=median_DELTA_AUC)) +
            geom_density() +  xlim(-0.2,0.2),
          combi_med %>% ggplot(aes(x=median_DELTA_fAUC)) +
            geom_density() +  xlim(-0.2,0.2),
          combi_med %>% ggplot(aes(x=median_ZSCORE)) +
            geom_density() +  xlim(-5,5) ,
          combi_med %>% ggplot(aes(x=median_DELTA_EMAX)) +
            geom_density()  + xlim(-0.4,0.4))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```

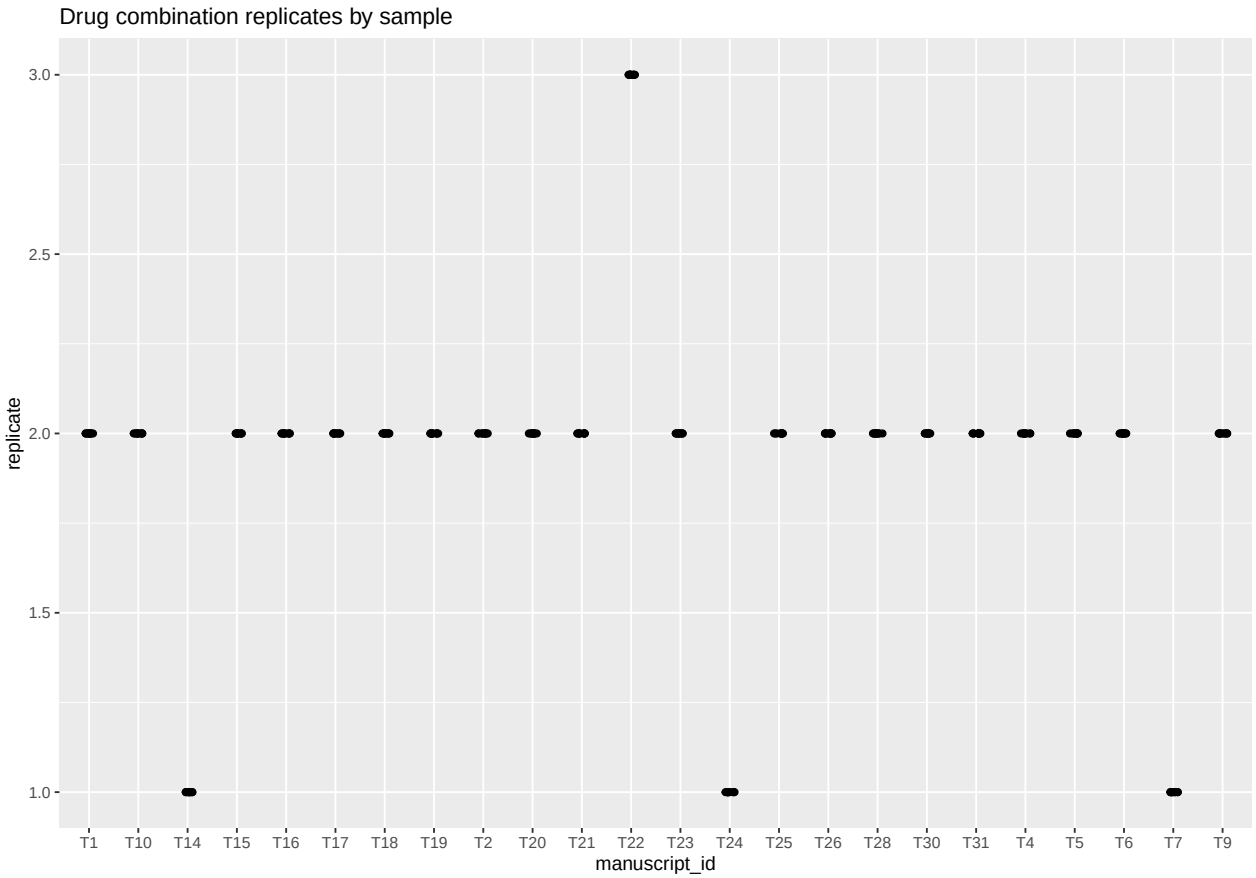
```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```



Replicates combinations x sample

Two replicates of each combination per sample, except for T14/T24/T7, which only has one replicate, and T22, which has three replicates per sample.

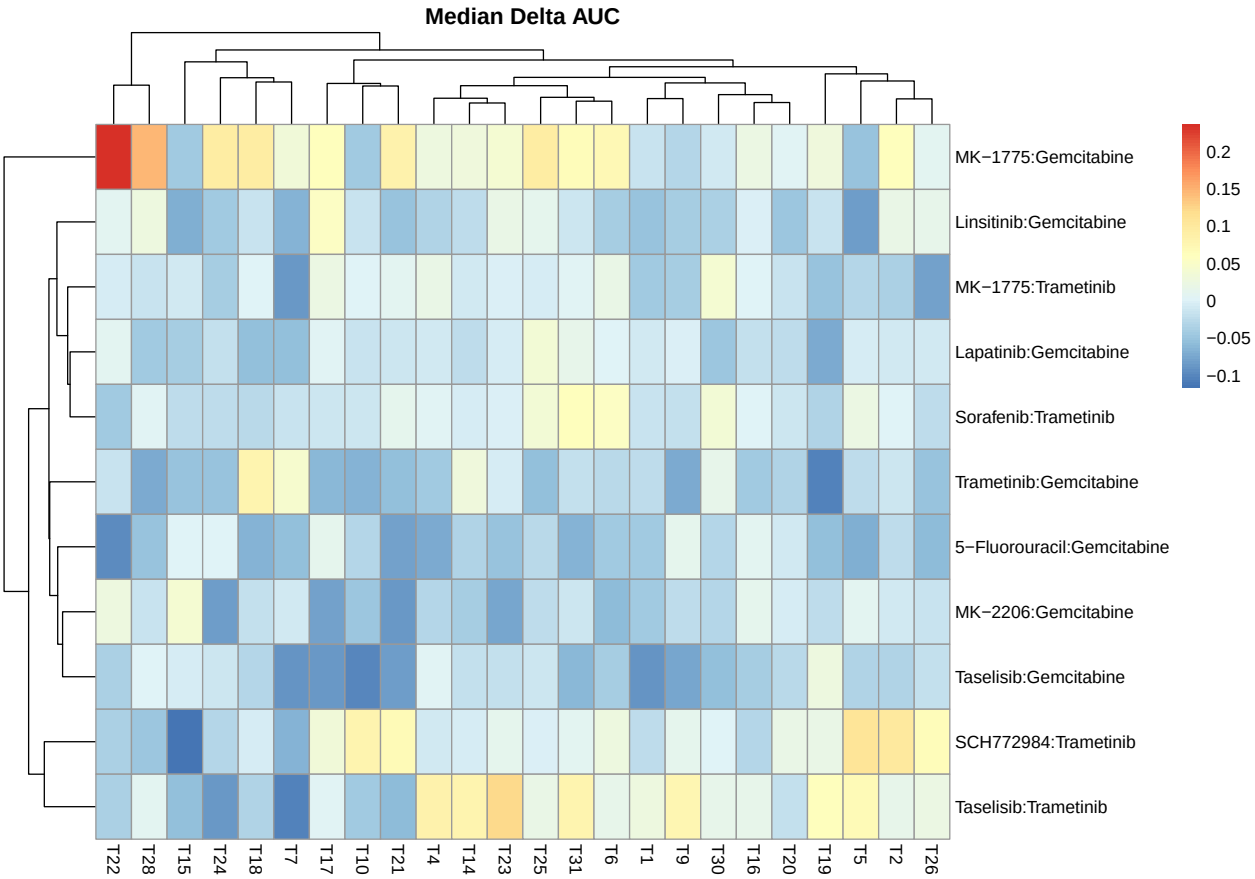
```
combi %>% group_by(manuscript_id, combination) %>% summarise(replicate=n()) %>%
  arrange(replicate) %>%
  ggplot(aes(x=manuscript_id, y=replicate)) +
  geom_jitter(height = 0, width = 0.1) +
  ggtitle("Drug combination replicates by sample")
```



Heatmaps by response metric

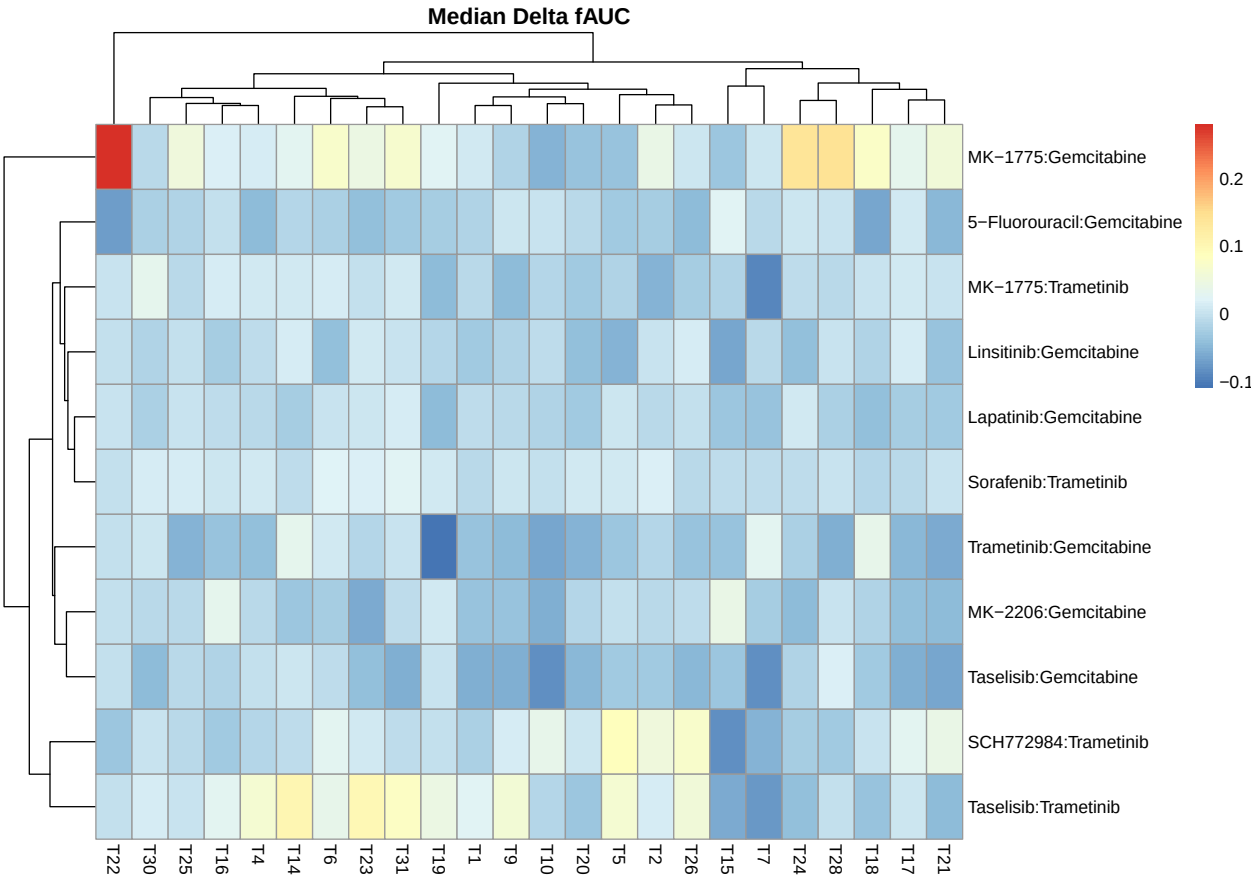
Median delta AUC

```
combi %>% group_by(manuscript_id, combination) %>%
  summarise(median_delta_auc = median(SYNERGY_DELTA_AUC)) %>%
  spread(key=manuscript_id, value=median_delta_auc) %>%
  column_to_rownames("combination") %>%
  pheatmap(., main = "Median Delta AUC")
```



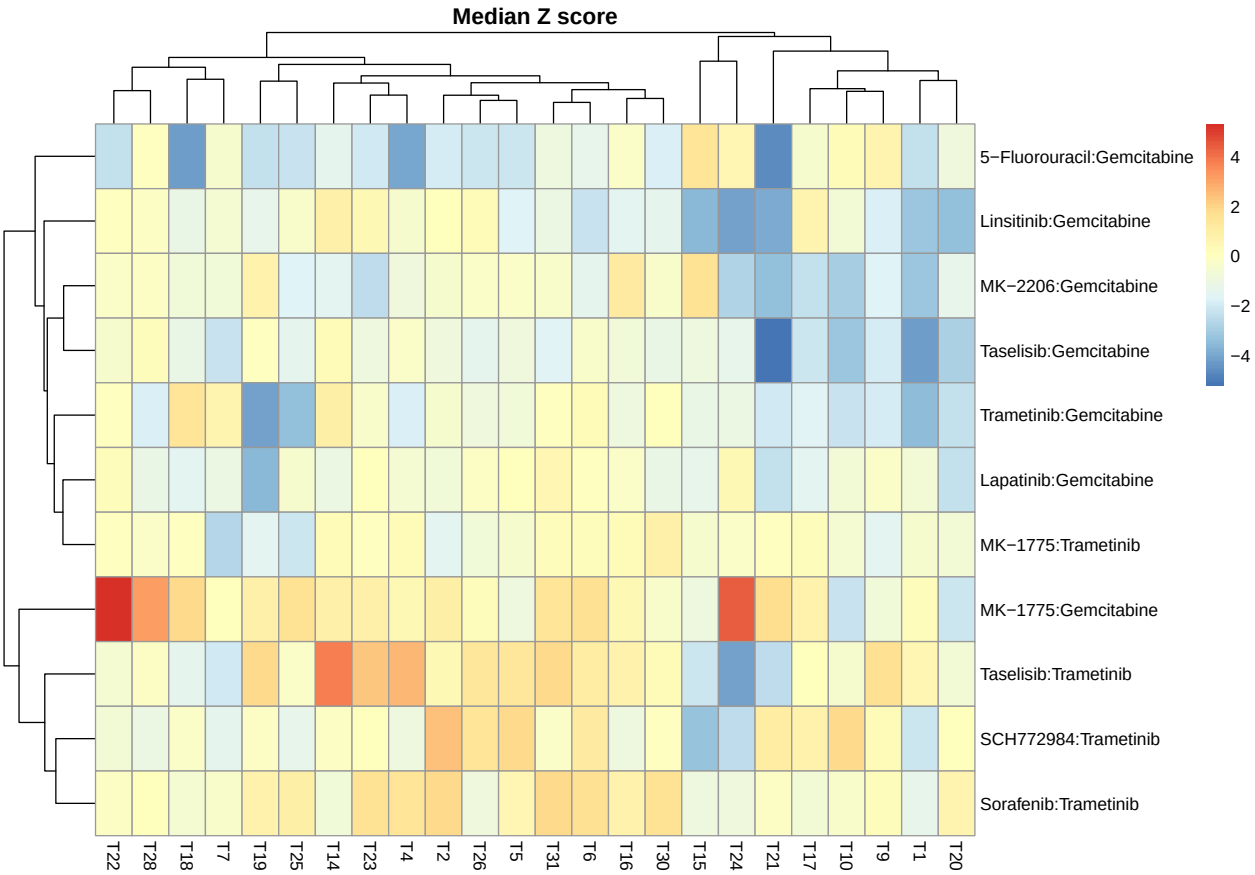
Median Delta fAUC

```
combi %>% group_by(manuscript_id, combination) %>%
  summarise(median_delta_fauc = median(SYNERGY_DELTA_fAUC)) %>%
  spread(key=manuscript_id, value=median_delta_fauc) %>%
  column_to_rownames("combination") %>%
  pheatmap(., main = "Median Delta fAUC")
```



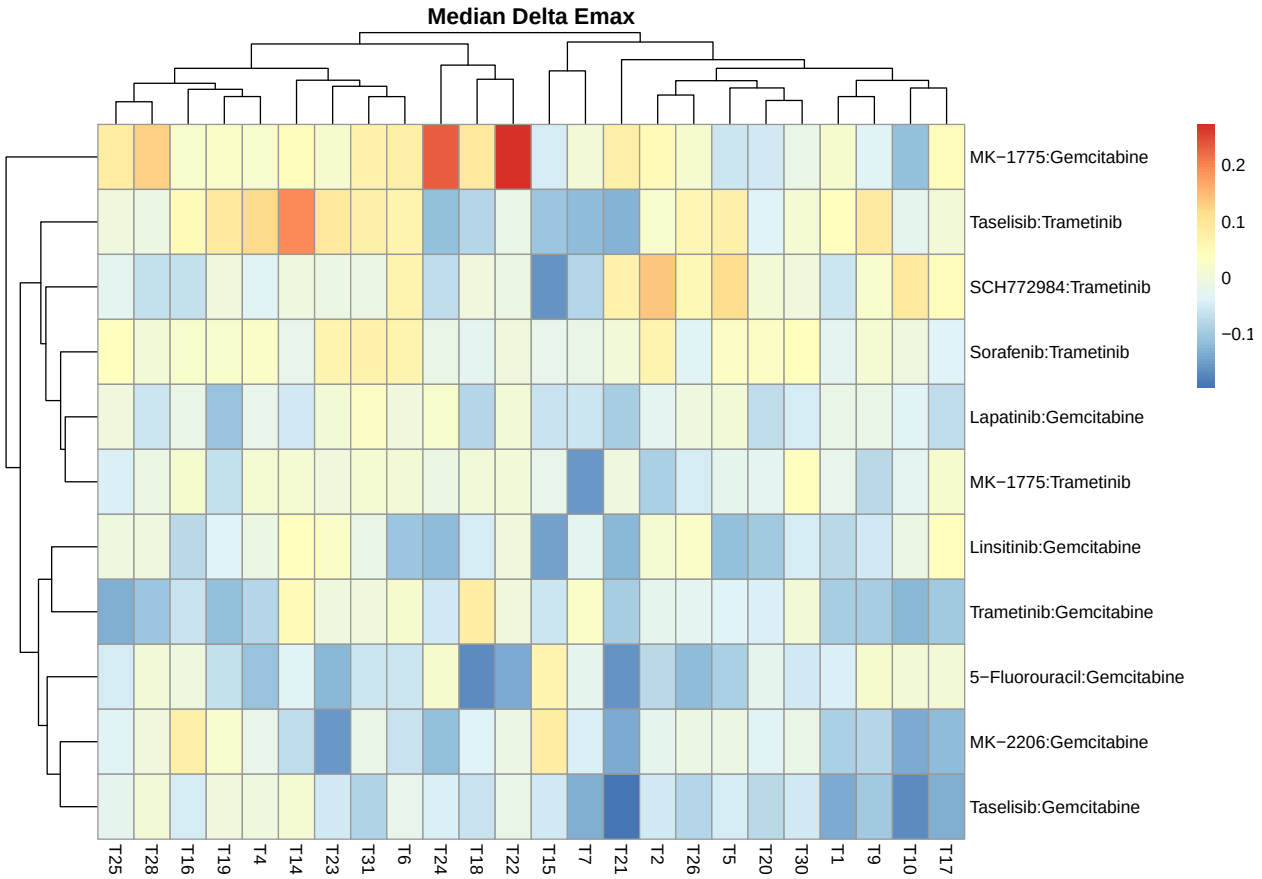
Median Z score

```
combi %>% group_by(manuscript_id, combination) %>%
summarise(median_z = median(SYNERGY_ZSCORE)) %>%
spread(key=manuscript_id, value=median_z) %>%
column_to_rownames("combination") %>%
pheatmap(., main = "Median Z score")
```



Median Delta Emax

```
combi %>% group_by(manuscript_id, combination) %>%
summarise(median_emax = median(SYNERGY_DELTA_EMAX)) %>%
spread(key=manuscript_id, value=median_emax) %>%
column_to_rownames("combination") %>%
pheatmap(., main = "Median Delta Emax")
```



Coefficient of variation plots

The coefficient of variation is the ratio of the standard deviation to the mean.

Will be NA for those samples with only one replicate.

```
coefVar_combi =
combi %>% group_by(manuscript_id, combination) %>%
  summarise(coefVar_DELTA_AUC = sd(SYNERGY_DELTA_AUC)/mean(SYNERGY_DELTA_AUC),
            coefVar_DELTA_fAUC = sd(SYNERGY_DELTA_fAUC)/
              mean(SYNERGY_DELTA_fAUC),
            coefVar_ZSCORE = sd(SYNERGY_ZSCORE)/mean(SYNERGY_ZSCORE),
            coefVar_DELTA_EMAX = sd(SYNERGY_DELTA_EMAX)/
              mean(SYNERGY_DELTA_EMAX)) %>%
  na.omit()

head(coefVar_combi)
```

```
## # A tibble: 6 x 6
##   manuscript_id combination coefVar_DELTA_A~ coefVar_DELTA_f~ coefVar_ZSCORE
##   <chr>         <chr>                  <dbl>            <dbl>          <dbl>
## 1 T1            5-Fluorour~          -0.0517          -0.0373        -0.0569
## 2 T1            Lapatinib:~          -0.814           -3.06          -2.26
## 3 T1            Linsitinib~          -0.325           -0.193         -0.285
## 4 T1            MK-1775:Ge~          -3.59             4.09          12.6
## 5 T1            MK-1775:Tr~          -0.182           -0.978         -1.06
```

```
## 6 T1            MK-2206:Ge~          -0.512           -0.201         -0.340
## # ... with 1 more variable: coefVar_DELTA_EMAX <dbl>
```

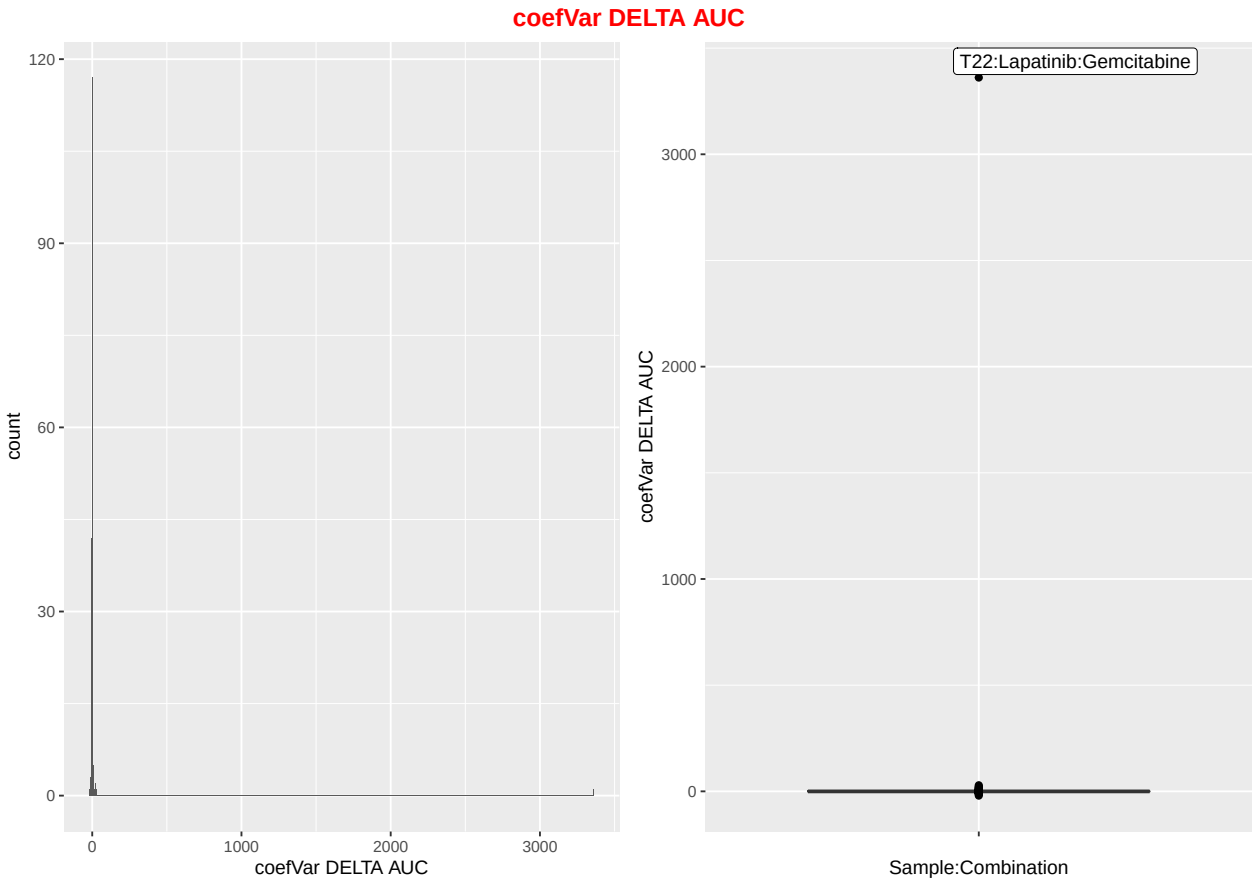
Plotting function

```
plot_coefvar = function(coefVar_combi, coefVar, label_threshold = 5){

  vec = coefVar_combi[,coefVar]
  coefVar_combi$label = if_else(abs(vec) > label_threshold,
                                paste(coefVar_combi$manuscript_id,
                                      coefVar_combi$combination, sep=":"), NULL)

  plots = ggarrange(coefVar_combi %>%
                      ggplot(aes(x=get(coefVar))) +
                      geom_histogram(binwidth = 2) +
                      xlab(gsub("_"," ", coefVar)),

                    coefVar_combi %>%
                      ggplot(aes(x="", y = get(coefVar), label=label)) +
                      geom_boxplot(alpha=0) + geom_point() +
                      ggrepel::geom_label_repel(na.rm=T) +
                      xlab("Sample:Combination") + ylab(gsub("_"," ", coefVar)),

                    ncol=2)

  plots = annotate_figure(plots, top = text_grob( gsub("_"," ", coefVar),
                                                  color = "red", face = "bold",
                                                  size = 14))

  return(plots)
}
```

Delta AUC

Demonstrates low variance, except for a single sample outlier.

```
plot_coefvar(coefVar_combi, coefVar = "coefVar_DELTA_AUC",
             label_threshold = 1000)
```



Examine outlier T22:Lapatinib:Gemcitabine

```
combi %>%
  filter(manuscript_id == "T22" & combination == "Lapatinib:Gemcitabine") %>%
  select(SYNERGY_DELTA_AUC, manuscript_id, combination)
```

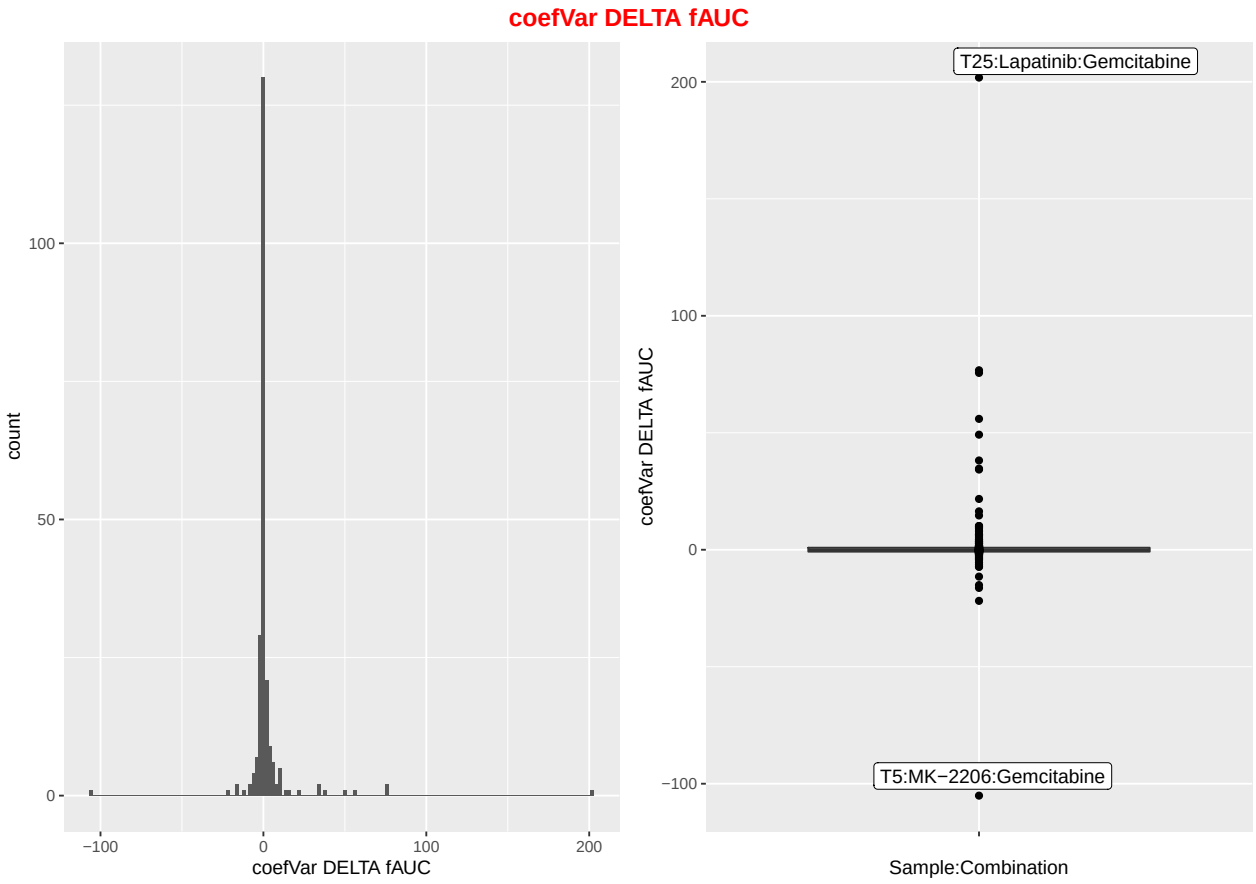
```
## # A tibble: 3 x 3
##   SYNERGY_DELTA_AUC manuscript_id combination
##               <dbl> <chr>         <chr>
## 1           0.00806 T22           Lapatinib:Gemcitabine
## 2          -0.0397  T22           Lapatinib:Gemcitabine
## 3           0.0317  T22           Lapatinib:Gemcitabine
```

Given the variance in DELTA AUC, the coefficient of variance is plausibly very high.

Delta fitted AUC

Two extreme outliers, but high coefVars more common throughout.

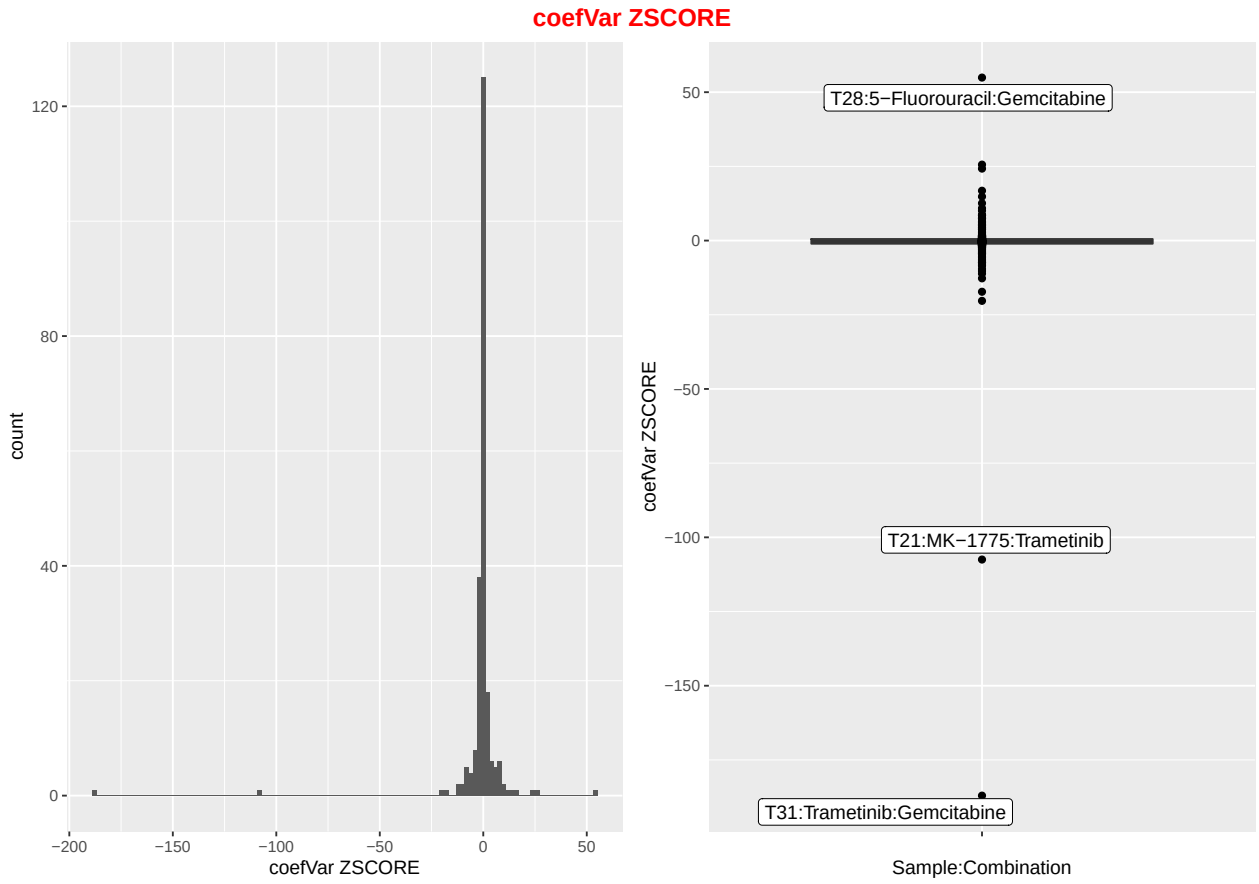
```
plot_coefvar(coefVar_combi,
             coefVar = "coefVar_DELTA_fAUC", label_threshold = 100)
```



Z score

Similar to fAUC.

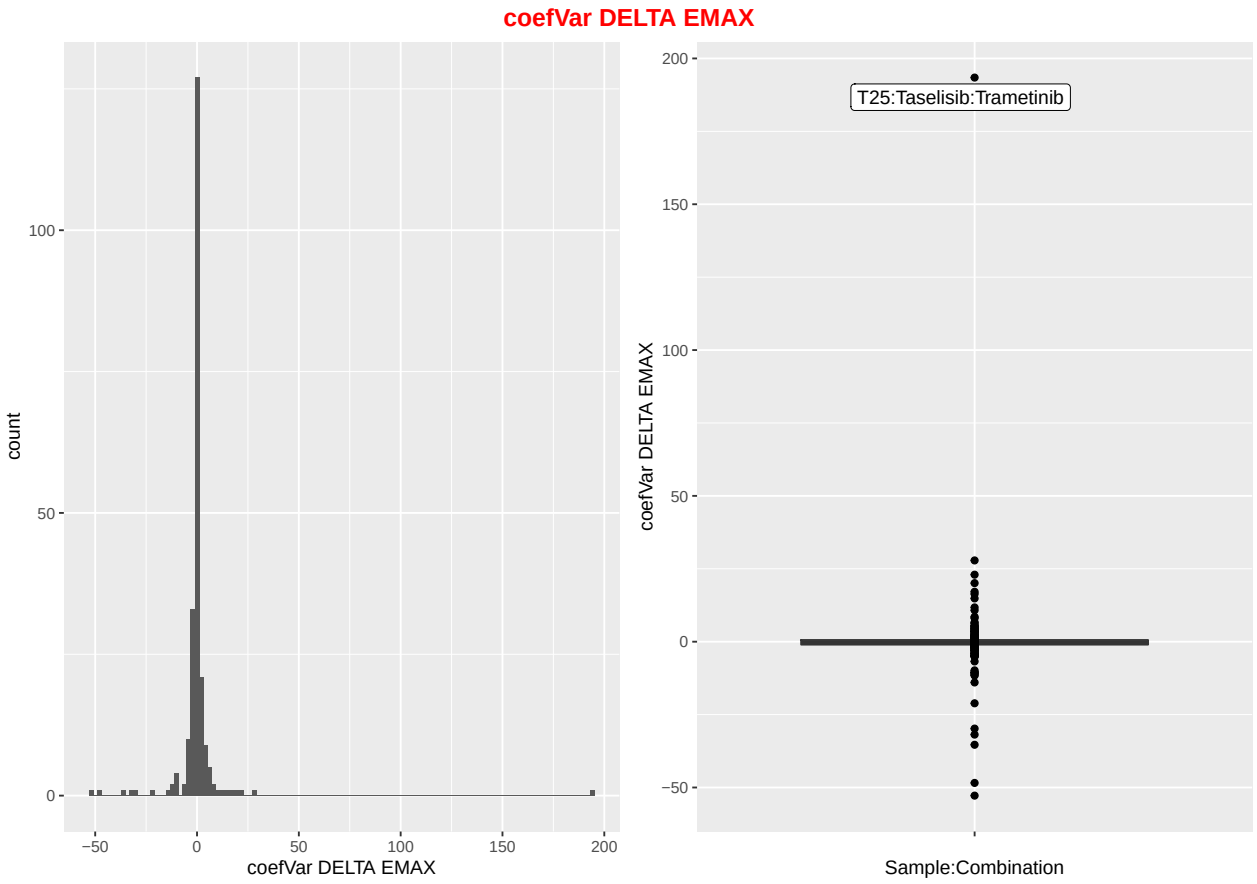
```
plot_coefvar(coefVar_combi, coefVar = "coefVar_ZSCORE", label_threshold = 50)
```



Delta Emax

A single extreme outlier.

```
plot_coefvar(coefVar_combi,
             coefVar = "coefVar_DELTA_EMAX", label_threshold = 100)
```



Note: Different samples appear as outliers depending on which metric is used to calculate coefVar. To be expected with relatively few variants.

Anchor plots

```
myPairsPlot = function(L, whichAnchor = '', whichLibrary = ''){

    anchorName = whichAnchor
    libraryName = whichLibrary

    par(mfrow = c(2,3), oma = c(0,0,2,0))

    if (nchar(whichAnchor)==0){
      whichLibrary = L[, 'LIBRARY_NAME'] == whichLibrary
      whichOne = whichLibrary

    } else if (nchar(whichLibrary)==0){
      whichAnchor = L[, 'ANCHOR_NAME'] == whichAnchor
      whichOne = whichAnchor
    }

    x = 'SYNERGY_DELTA_AUC'
    y = 'SYNERGY_DELTA_fAUC'
```

```
    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'delta AUC', ylab = 'delta fAUC')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    x = 'SYNERGY_DELTA_AUC'
    y = 'SYNERGY_ZSCORE'

    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'delta AUC', ylab = 'Z-score')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    x = 'SYNERGY_DELTA_AUC'
    y = 'SYNERGY_DELTA_EMAX'

    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'delta AUC', ylab = 'delta E-max')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    x = 'SYNERGY_DELTA_fAUC'
    y = 'SYNERGY_ZSCORE'

    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'delta fAUC', ylab = 'Z-score')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    x = 'SYNERGY_DELTA_fAUC'
    y = 'SYNERGY_DELTA_EMAX'

    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'delta fAUC', ylab = 'delta Emax')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    x = 'SYNERGY_ZSCORE'
    y = 'SYNERGY_DELTA_EMAX'

    plot(L[,x], L[,y], pch = 19, cex = 0.8, col = 'gray',
         xlab = 'Z-score', ylab = 'delta E-max')
    points(L[whichOne,x], L[whichOne,y], col = 'orange', pch = 19, cex = 1 )

    if (nchar(whichAnchor)==0){
      title(libraryName, outer = TRUE)

    } else if (nchar(whichLibrary)==0){
      title(anchorName, outer = TRUE)
    }

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

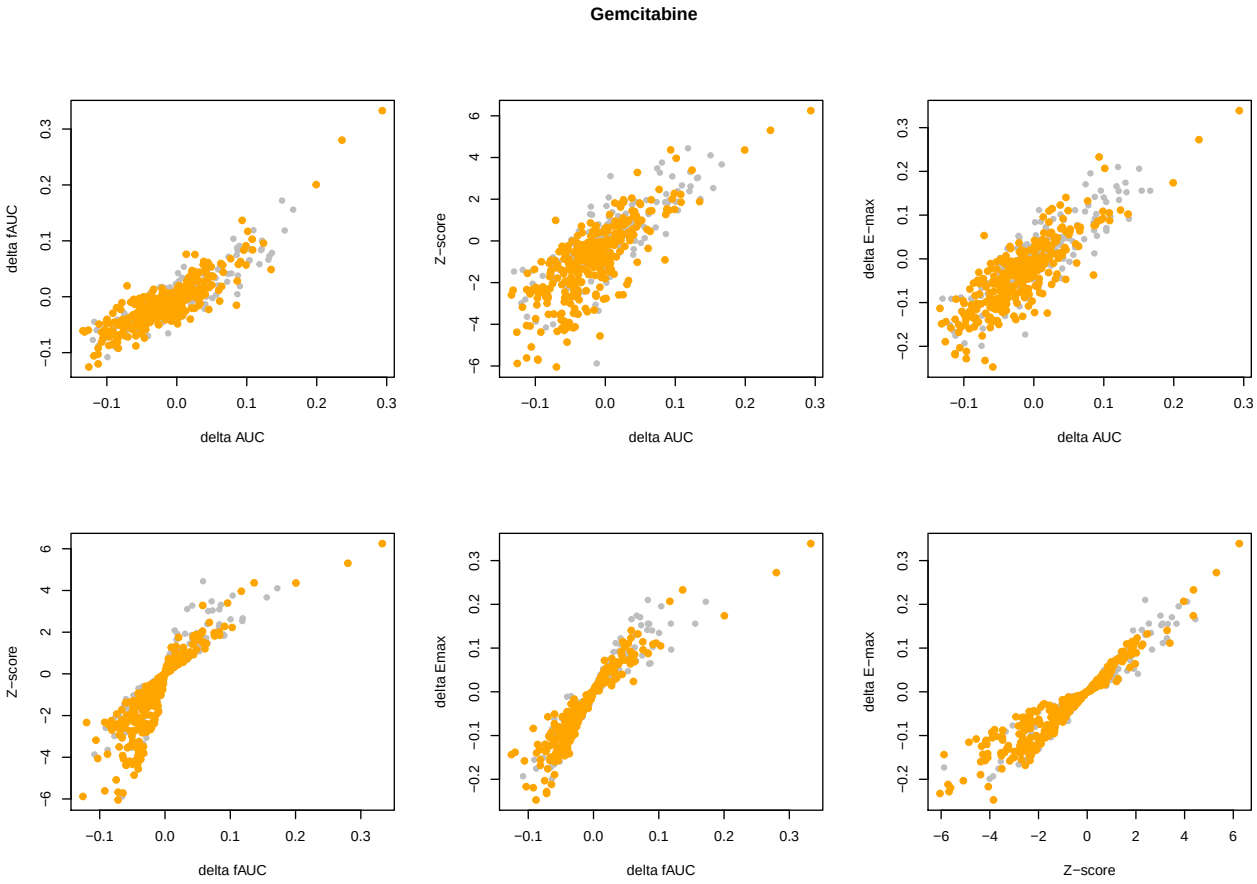
```
combi %>% select(ANCHOR_NAME, ANCHOR_TARGET, CONC_anch) %>% distinct()
```

```
## # A tibble: 4 x 3
##   ANCHOR_NAME ANCHOR_TARGET             CONC_anch
##   <chr>       <chr>                         <dbl>
```

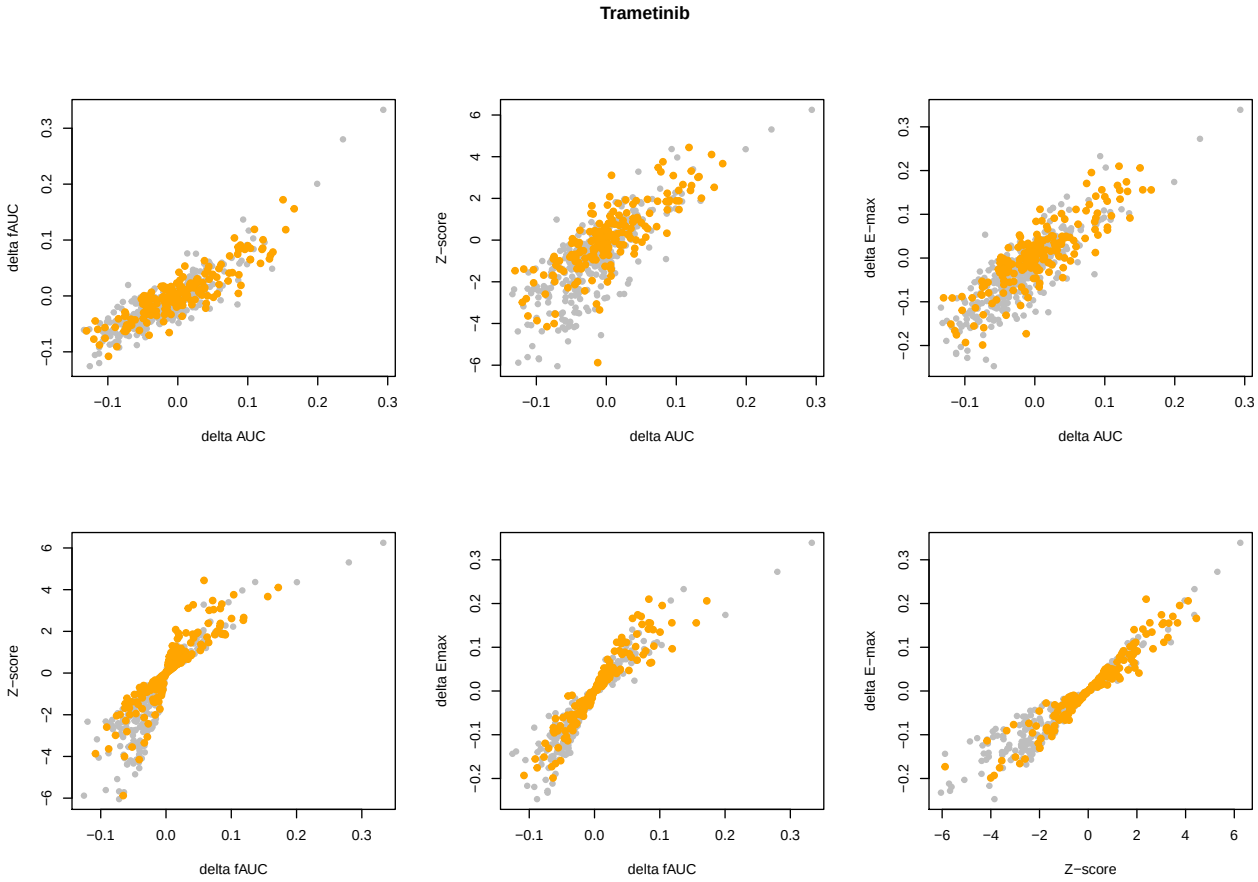
```
## 1 Gemcitabine Pyrimidine antimetabolite   0.00667
## 2 Trametinib  MEK1, MEK2                  0.00667
## 3 Gemcitabine Pyrimidine antimetabolite   0.01
## 4 Trametinib  MEK1, MEK2                  0.01
```

```
for (i in 1:length(unique(combi$ANCHOR_NAME))){
  myPairsPlot(as.data.frame(combi), whichAnchor = unique(combi$ANCHOR_NAME)[i])
}
```

```
## Warning in if (nchar(whichAnchor) == 0) {: the condition has length > 1 and only
## the first element will be used
```




```
## Warning in if (nchar(whichAnchor) == 0) {: the condition has length > 1 and only
## the first element will be used
```



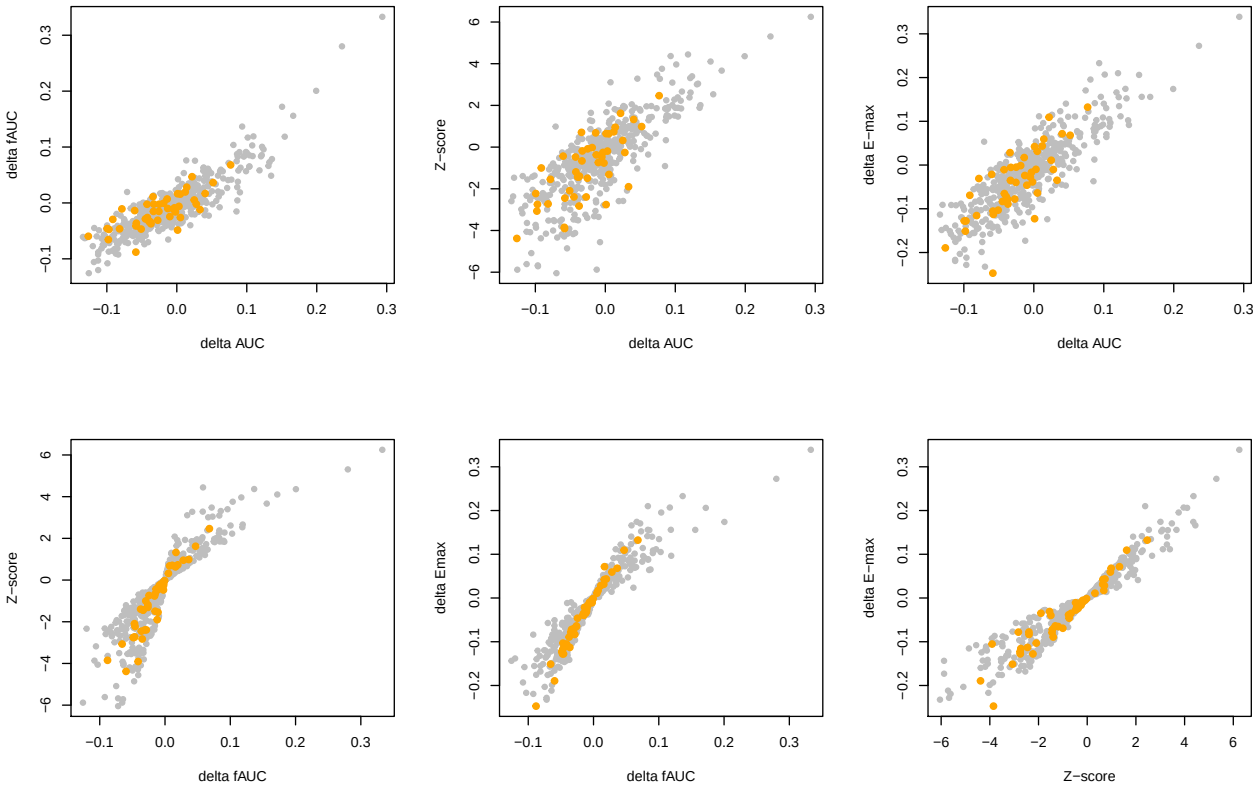
Library plots

```
unique(combi$LIBRARY_NAME)
```

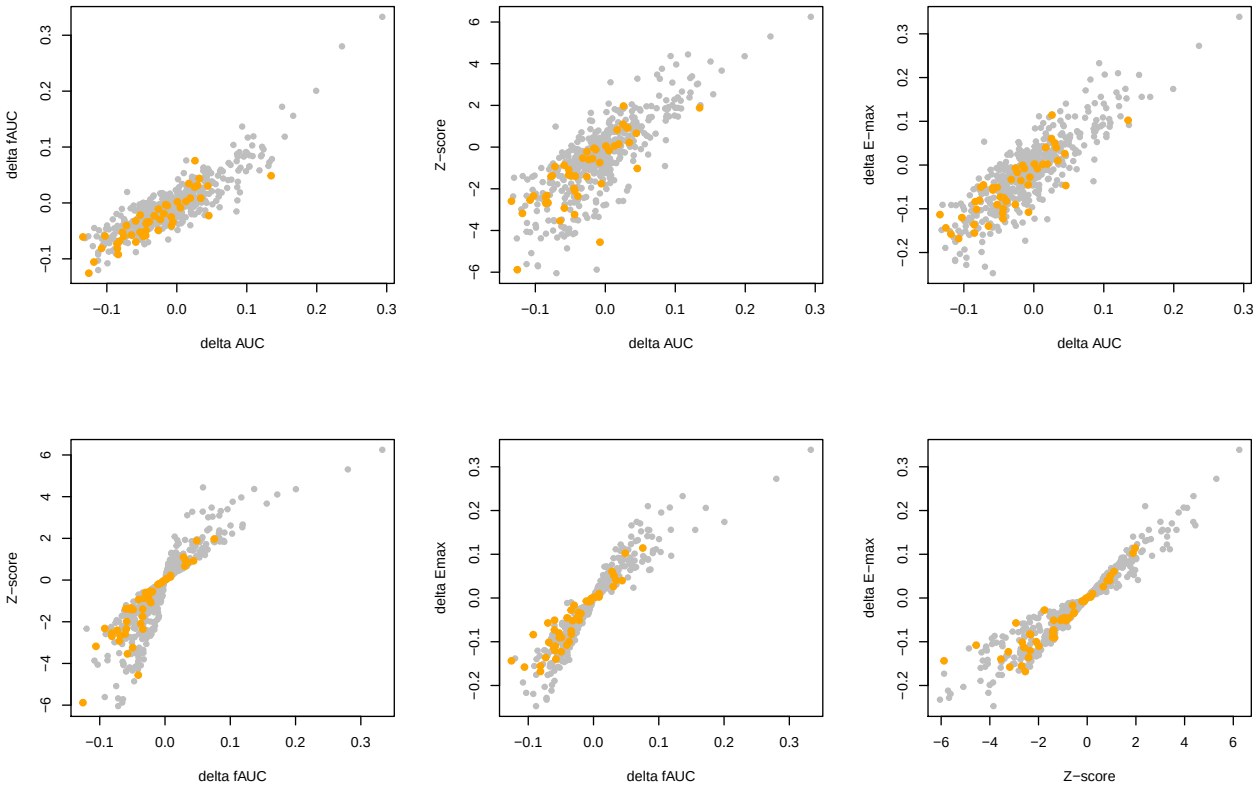
```
## [1] "MK-2206"        "Trametinib"     "Linsitinib"     "Lapatinib"
## [5] "MK-1775"        "Taselisib"      "5-Fluorouracil" "Sorafenib"
## [9] "SCH772984"
```

```
for (i in 1:length(unique(combi$LIBRARY_NAME))){
  myPairsPlot(as.data.frame(combi), whichLibrary = unique(combi$LIBRARY_NAME)[i])
}
```

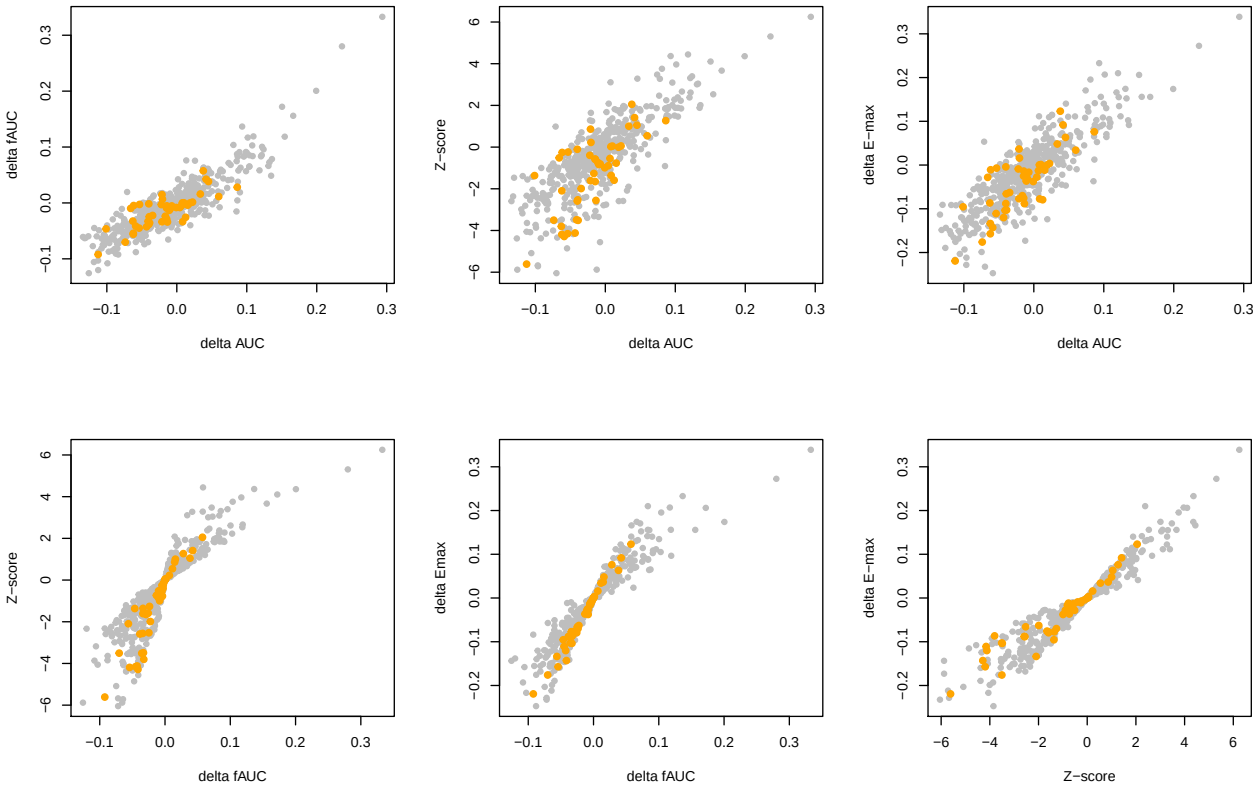
MK−2206


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

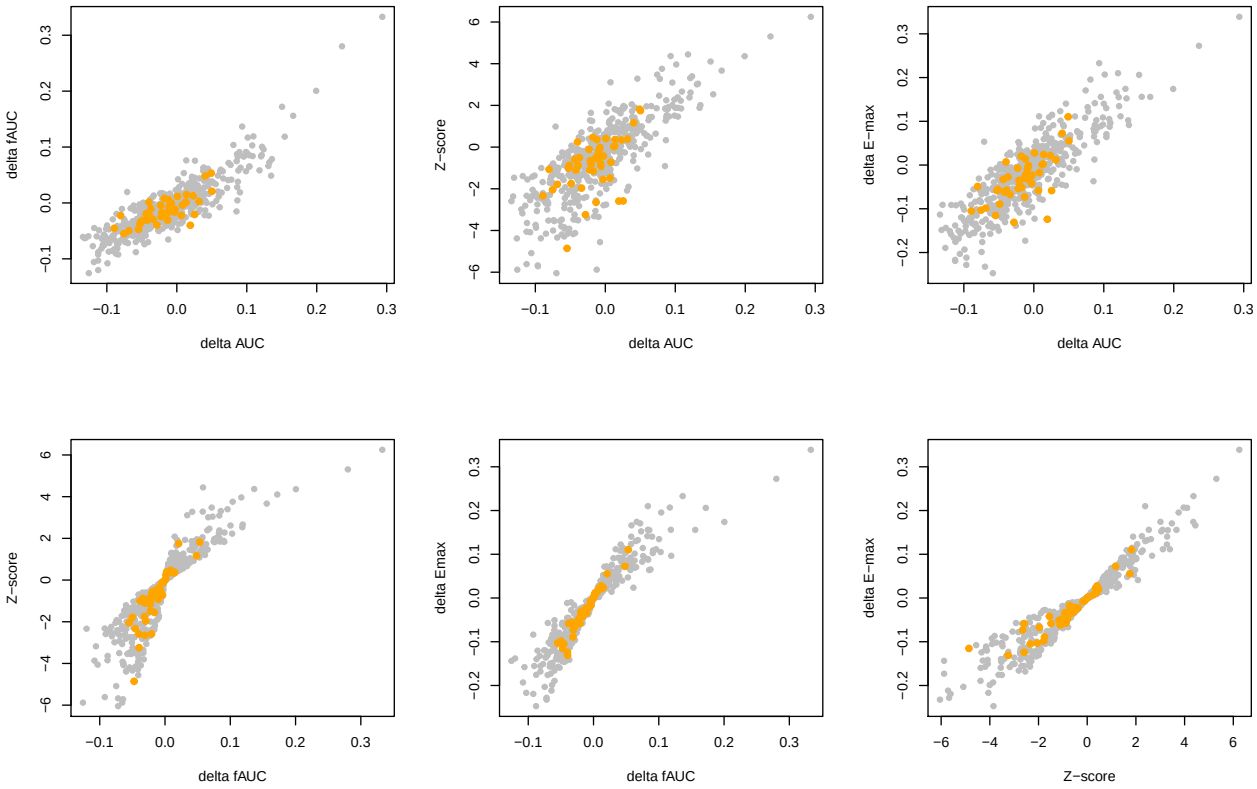
Trametinib


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

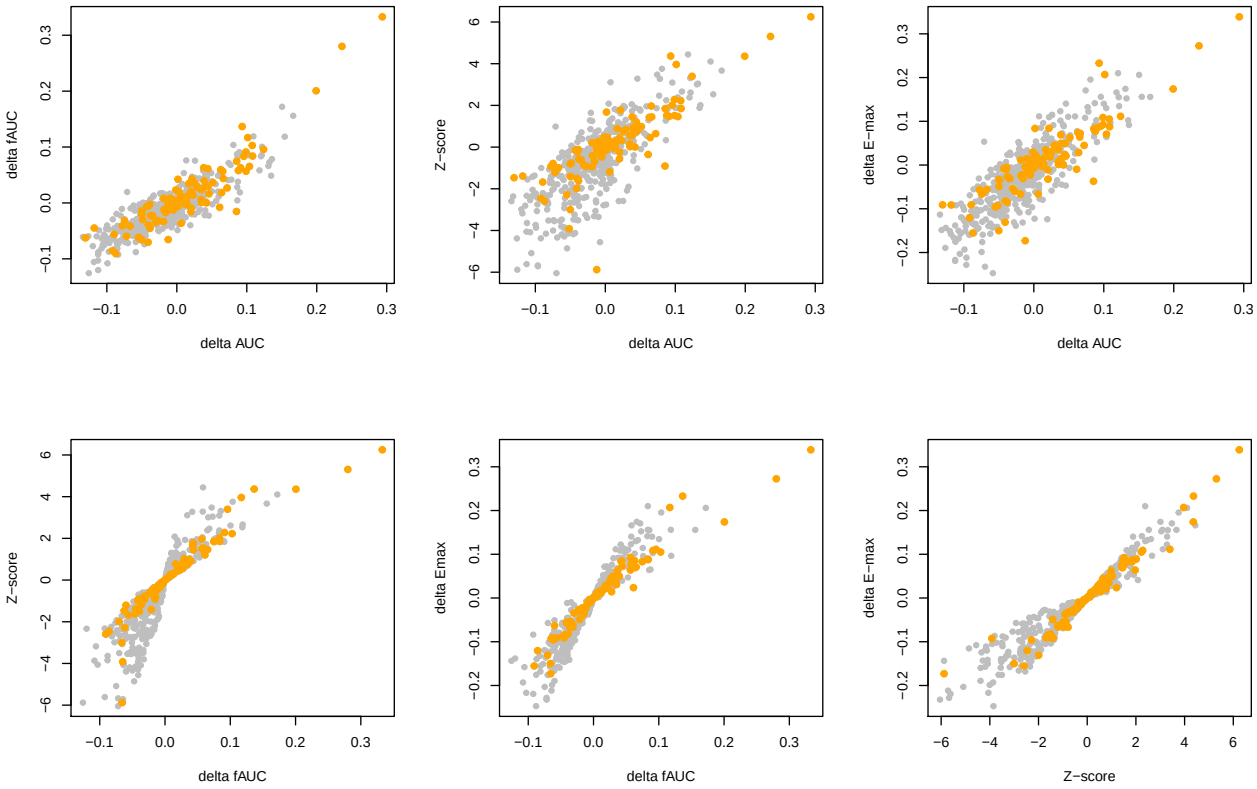
Linsitinib


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

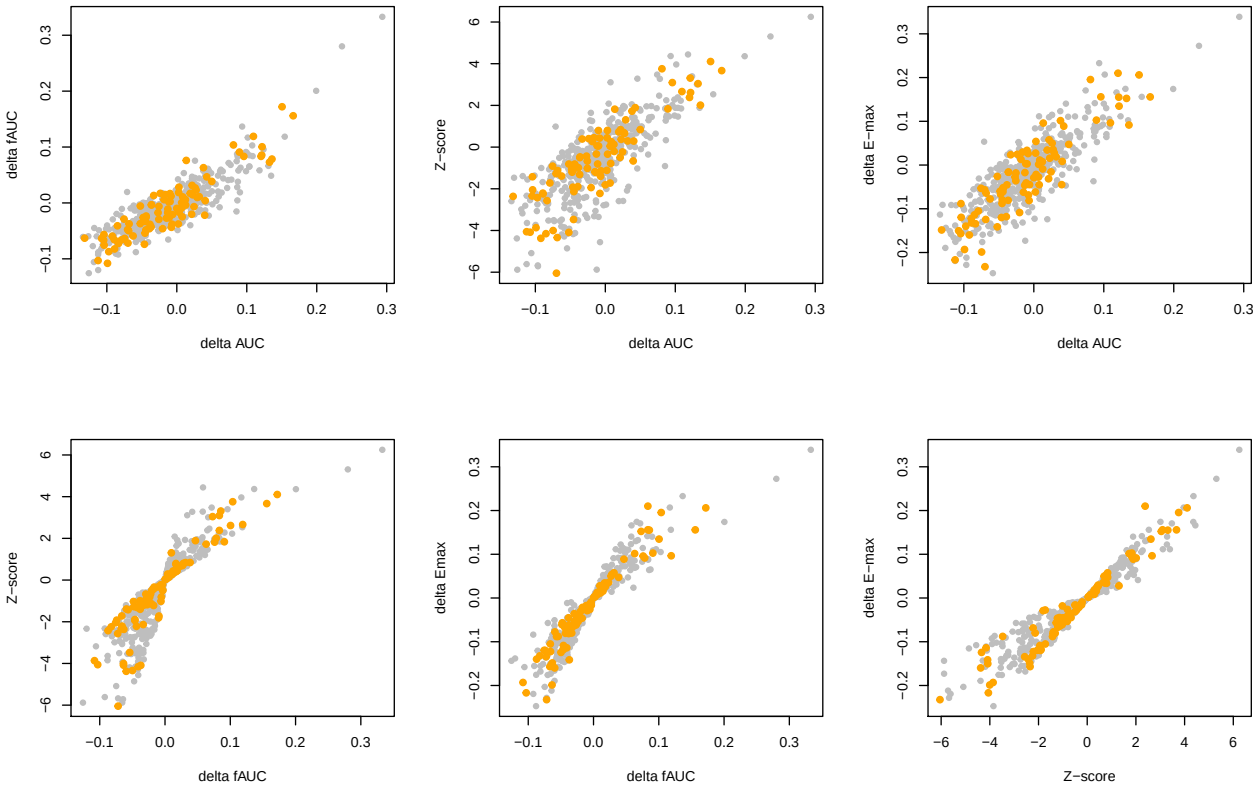
Lapatinib


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

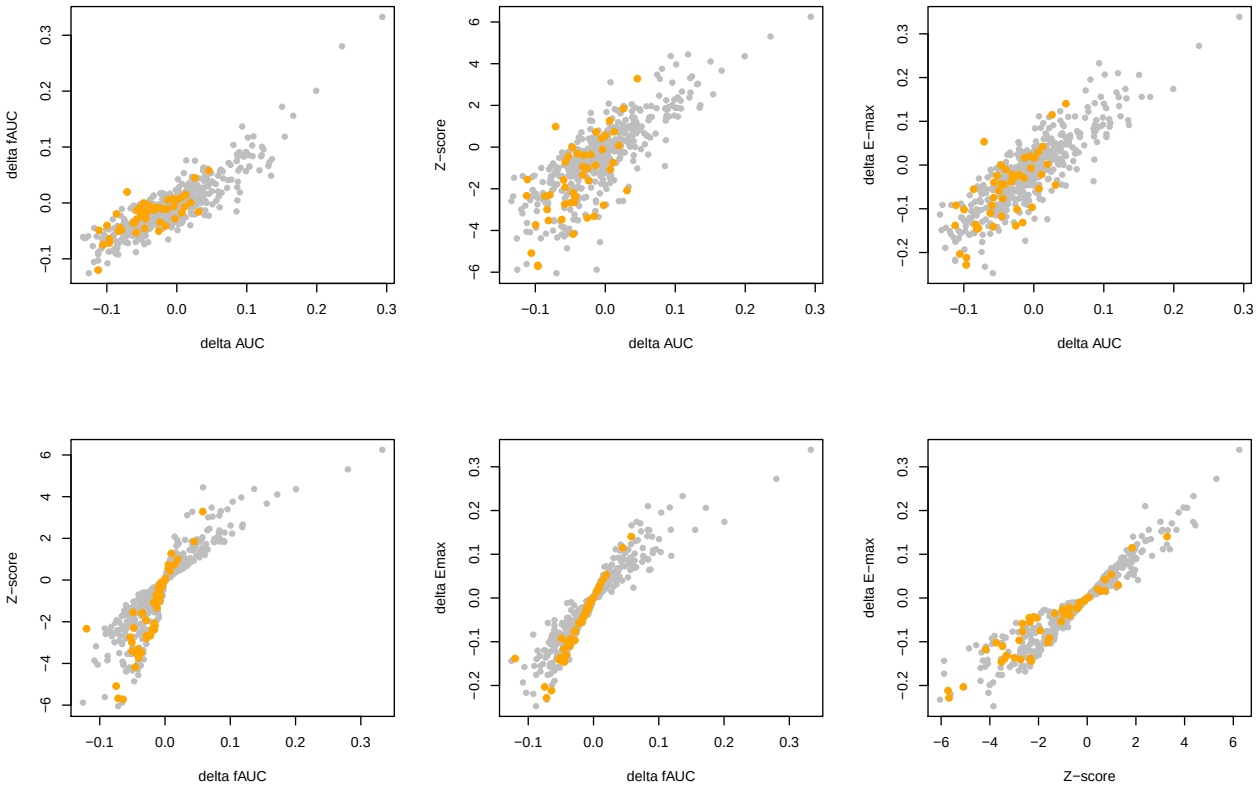
MK−1775


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

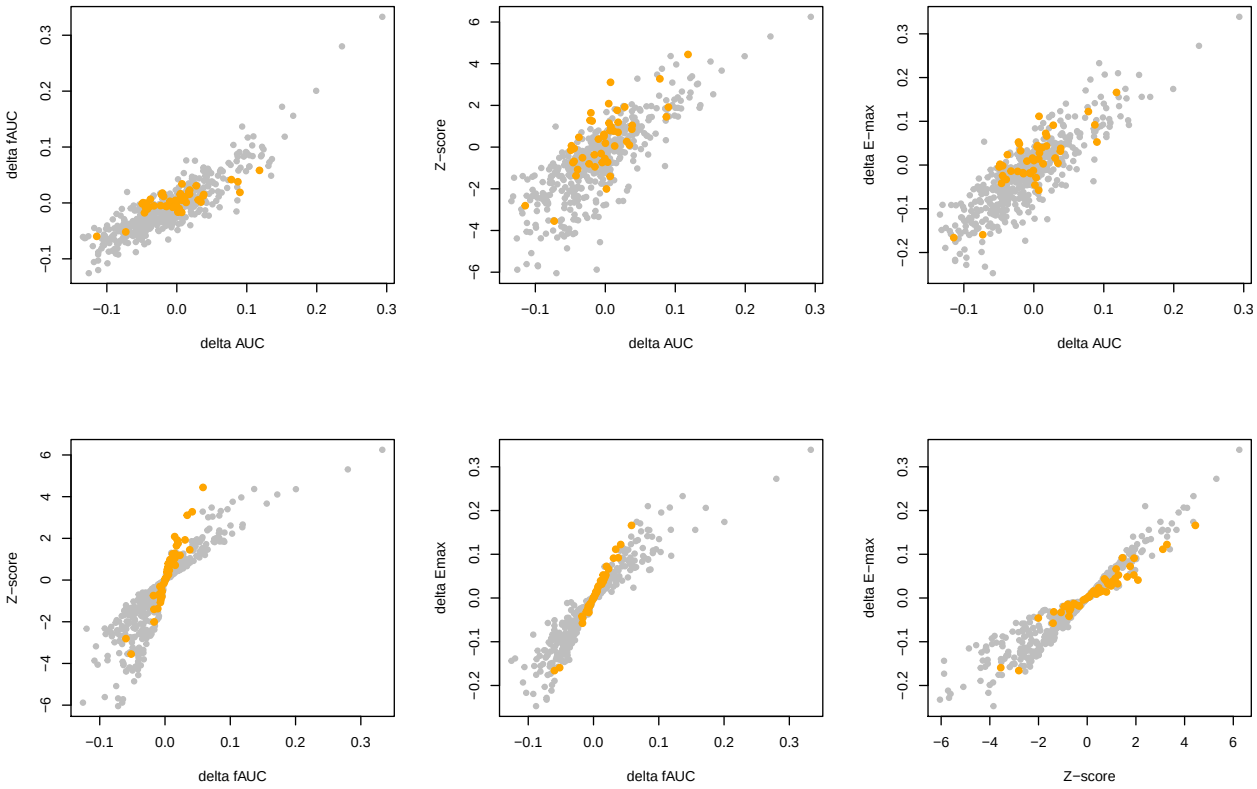
Taselisib

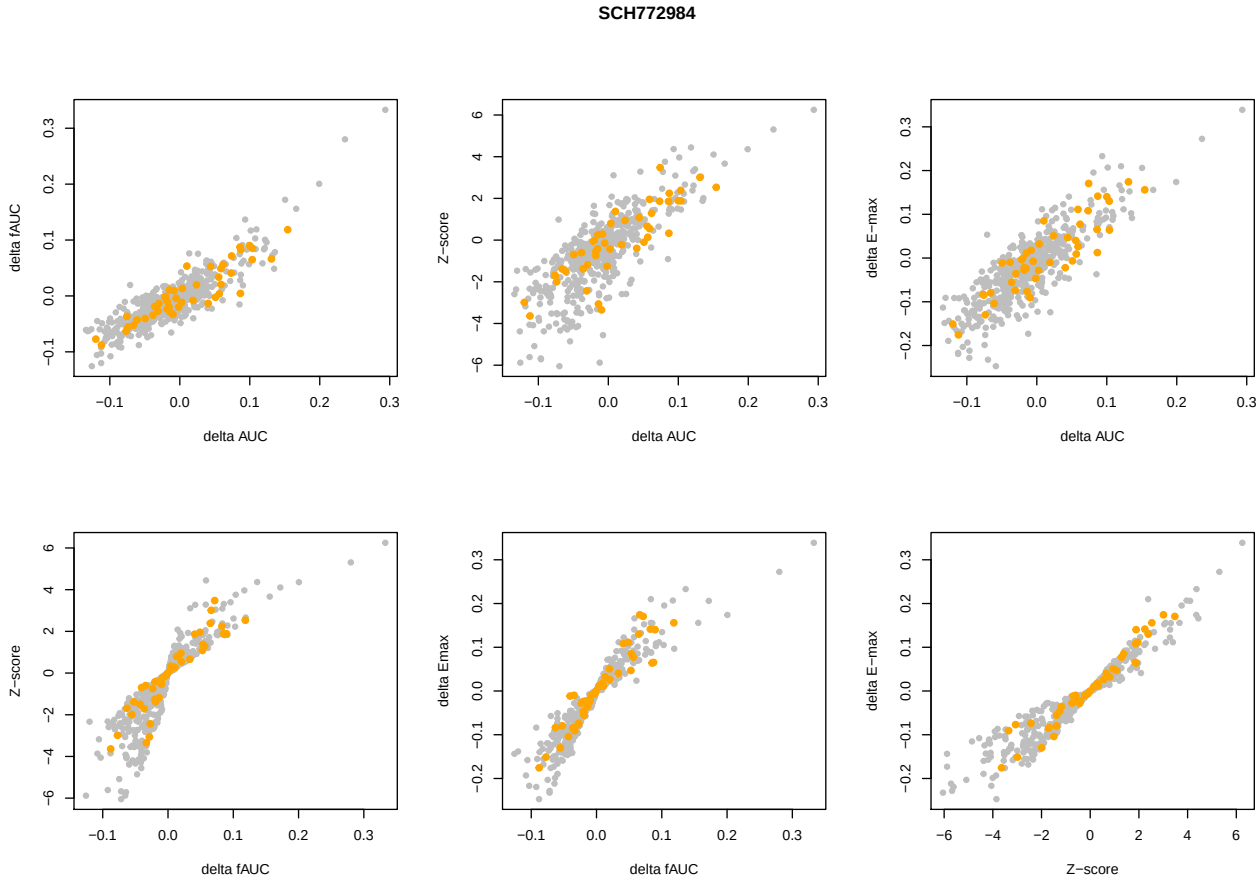

delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score

5−Fluorouracil



Sorafenib


delta fAUC
delta AUC
Z−score
delta AUC
delta E−max
delta AUC
Z−score
delta fAUC
delta Emax
delta fAUC
delta E−max
Z−score



Overview data type by sample

Most expansive list of manuscript ids:

```
overview = tibble(manuscript_id = sampleMap$manuscript_id %>% unique())
overview = overview %>%
  mutate(has_mutdata = overview$manuscript_id %in% sv$manuscript_id,
         has_cndata = overview$manuscript_id %in% cancer_cnv$manuscript_id,
         has_rnaseq = overview$manuscript_id %in% colnames(rna_raw),
         has_single_drug = overview$manuscript_id %in% single_drug$manuscript_id,
         has_combi_drug = overview$manuscript_id %in% combi$manuscript_id)

overview = overview %>%
  mutate(has_drug = (has_single_drug == T | has_combi_drug == T))

overview = overview %>%
  mutate(has_wgs_rna_atleastonedrug = (has_mutdata==T & has_cndata==T &
                                       has_rnaseq==T & has_drug == T))
overview = overview %>%
  mutate(has_all_data = rowSums(overview[,-1])==ncol(overview[,-1]))

overview
```

```
## # A tibble: 31 x 9
##    manuscript_id has_mutdata has_cndata has_rnaseq has_single_drug
```

```
##    <chr>         <lgl>       <lgl>      <lgl>      <lgl>
##  1 T1            FALSE       FALSE      TRUE       TRUE
##  2 T2            FALSE       FALSE      TRUE       TRUE
##  3 T3            TRUE        TRUE       TRUE       FALSE
##  4 T4            TRUE        TRUE       TRUE       TRUE
##  5 T5            TRUE        TRUE       TRUE       TRUE
##  6 T6            TRUE        TRUE       TRUE       TRUE
##  7 T7            TRUE        TRUE       TRUE       TRUE
##  8 T8            TRUE        TRUE       TRUE       FALSE
##  9 T9            TRUE        TRUE       TRUE       TRUE
## 10 T10           TRUE        TRUE       TRUE       TRUE
## # ... with 21 more rows, and 4 more variables: has_combi_drug <lgl>,
## #   has_drug <lgl>, has_wgs_rna_atleastonedrug <lgl>, has_all_data <lgl>
```

```
colSums(overview[,-1]) %>% enframe("data_type", "total_samples") %>%
  mutate(total_samples = paste(as.character(total_samples), "31", sep="/"))
```

```
## # A tibble: 8 x 2
##   data_type                  total_samples
##   <chr>                      <chr>
## 1 has_mutdata                26/31
## 2 has_cndata                 26/31
## 3 has_rnaseq                 31/31
## 4 has_single_drug            24/31
## 5 has_combi_drug             24/31
## 6 has_drug                   24/31
## 7 has_wgs_rna_atleastonedrug 20/31
## 8 has_all_data               20/31
```

Samples for which all data types are present:

```
overview %>% filter(has_all_data==T) %>% pull(manuscript_id)
```

```
##  [1] "T4"  "T5"  "T6"  "T7"  "T9"  "T10" "T14" "T15" "T16" "T17" "T18" "T19"
## [13] "T20" "T21" "T22" "T23" "T24" "T25" "T26" "T28"
```

```
write_csv(overview, here("drug_screen_analysis",
                         "data_type_overview_cngenes.csv"))
```

Drug response prediction with conventional elastic net

In this section, we apply an elastic net model to predicting single drug response from mutations, copy numbers and mRNA expression to identify biomarkers. We also seek to identify biomarkers for synergy in combination drug screening. In a 100-fold loop, the elastic net will attempt to predict a drug response value (yhat) based on the molecular data for the samples at hand. It will then assess the accuracy of the prediction via the mean cross validated error, on the basis of which the best performing model can be chosen.

To visualize the contribution of various types of features towards the model performance, a heatmap depicting the explained variance (defined as 1-mean cross validated error) is drawn. When the explained variance is less than 0, it will be reported as 0, thereby providing a threshold below which the contribution of that feature type can be considered trivial.

Drug response tools:

```
glmnetLoop = function(
  y, x, whichSamples, foldid = c(), alpha = seq(0,1,by=0.25),
```

```
family = 'gaussian', standardize=FALSE, whichDrug='', plot = FALSE
){

if (length(foldid)==0){
  # fix the folds .................
  # not allowed to compare alphas if folds are not fixed
  foldid = c(rep(1,nrow(y)/3), rep(2,nrow(y)/3), rep(3,nrow(y)/3))
  foldid = append(foldid, sample(1:3, size = nrow(y)-length(foldid),
                                 replace=TRUE))
  library(gtools, quietly=TRUE)
  foldid = permute(foldid)
}


library(glmnet, quietly = TRUE)

if (grepl(family,'gaussian')){
  y = scale(y)
  y = y[whichSamples,]
} else if (grepl(family,'binomial')){
  y = as.factor(y[whichSamples,1])
} else if (family=='mgaussian'){

  stopifnot(nchar(whichDrug)>0)
  stopifnot(whichDrug %in% colnames(y))
  # do nothing, because standardize.response=TRUE
  #y = matrix( scale(y[whichSamples,,drop=FALSE], scale=TRUE) )
  y = y[whichSamples,]
}

if (plot){
  # initialize plot
  plot(0,bty='n',pch='',ylab='',xlab='', xlim = c(-6, 6), ylim = c(0,2))
}

res = vector("list", length(alpha))
names(res) = alpha
# all genes, to initialize the intersection over all alphas
features_1se = colnames(x)

cnt = 1
for (a in alpha){

  fit = cv.glmnet(x[whichSamples,], y, foldid=foldid,
                  standardize=standardize, alpha = a,
                  family=family, standardize.response=TRUE)

  jBest_1se = which(fit$lambda==fit$lambda.1se)
  jBest_min = which(fit$lambda==fit$lambda.min)

  cnt = cnt + 1
  if (plot){
    points(log(fit$lambda),fit$cvm, pch=19, col=cnt, cex = 0.1, lwd = 0.5)
```

```
  }

  if (grepl('mgaussian',family)){
      jSel_1se = which(abs(fit$glmnet.fit$beta[[whichDrug]][,jBest_1se])>0)
      jSel_min = which(abs(fit$glmnet.fit$beta[[whichDrug]][,jBest_min])>0)

      res[[as.character(a)]]=list(
        'var'=1-min(fit$cvm),
        'min'=fit$glmnet.fit$beta[[whichDrug]][jSel_min,jBest_min,drop=FALSE],
        '1se'=fit$glmnet.fit$beta[[whichDrug]][jSel_1se,jBest_1se,drop=FALSE],
        'fit' = fit)
      features_1se = intersect(
        features_1se,
        names(fit$glmnet.fit$beta[[whichDrug]][jSel_1se,jBest_1se,drop=FALSE])
        )

  } else {
      jSel_1se = which(abs(fit$glmnet.fit$beta[,jBest_1se])>0)
      jSel_min = which(abs(fit$glmnet.fit$beta[,jBest_min])>0)

      res[[as.character(a)]]=list(
        'var'=1-min(fit$cvm),
        'min'=fit$glmnet.fit$beta[jSel_min,jBest_min,drop=FALSE],
        '1se'=fit$glmnet.fit$beta[jSel_1se,jBest_1se,drop=FALSE],
        'fit' = fit)
      features_1se = intersect(
        features_1se,
        names(fit$glmnet.fit$beta[jSel_1se,jBest_1se,drop=FALSE])
        )
  }

  #titlestring = paste(c('alpha: ', a,
  #                      '; perc var expl: ',
  #                      as.character(round(1-min(fit$cvm),3))), collapse='')
  #pdf(paste(c('figure/', drug,'_fit_AUC_mutations.pdf'), collapse=''))
  #plot(fit, main = titlestring, cex.main=1)
}

maxVar = c()
for (r in names(res)){ maxVar = append(maxVar, res[[r]]$'var') }
names(maxVar) = names(res)
maxAlpha = names(maxVar)[maxVar == max(maxVar)]

#print(paste(c('The max-"fraction of variance/deviance explained""
#(',max(maxVar),') alpha is: ', maxAlpha, ' with ',
#length(res[[maxAlpha]]$'1se'),
#' features at 1se.'), collapse=''))

#print('The intersection of 1se features across all alphas is')
#print(features_1se)

rm(list = setdiff(ls(), c('res','maxAlpha','features_1se')))
return(list(res=res, alpha=maxAlpha, features_1se = features_1se))
```

```
}
```

Single response (IC50)

```
Y = as.matrix(IC50)
```

```
start = Sys.time()

#Set up some empty variables
LvarExpl = list()
LpredCor = list()

features = list()
features$mutations = list()
for (drug in colnames(Y)){ features$mutations[[drug]]=c() }


features$cnNumber = features$mutations
features$geneEx = features$mutations

for (iter in 1:100){

# initialize a drugs by data type variance explained matrix
# Unique drug IDs by row, mutations, copy number and gene expression by column

varExplMatrix = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(varExplMatrix) = colnames(Y)
colnames(varExplMatrix) = c('mutations', 'copy number', 'gene ex')

#Mutation data

#Extract manuscript ids that contain mutation/cnv (WGS) and single drug data
int = overview %>% filter(has_mutdata == T & has_single_drug == T)

#Filter gene x sample mutation summary for those samples containing
#the necessary data
x = mutMatrix[,colnames(mutMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]


# Initialize correlation matrix
predictionCor = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(predictionCor) = colnames(Y)
colnames(predictionCor) = c('mutations', 'copy number', 'gene ex')


for (drug in colnames(Y)){
  #drug = colnames(Y)[1] #For testing
  #print(drug)

  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
```

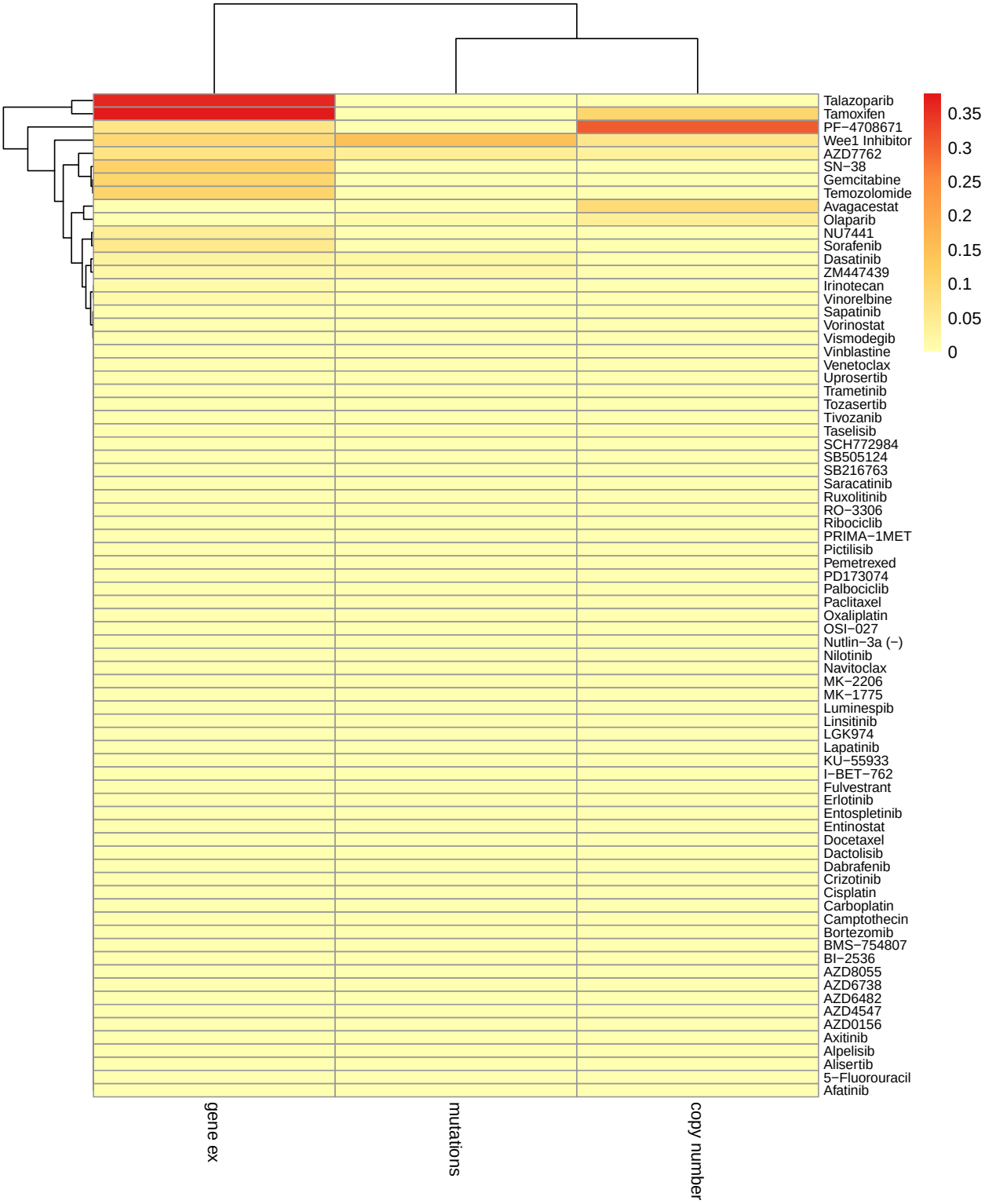
```
  res = glmnetLoop(y[whichSamples,,drop=FALSE],
                   as.matrix(t(x[,whichSamples])), whichSamples,
                   alpha = 0.5, family='gaussian', standardize=FALSE)
  fit = res$res$`0.5`$fit
  yhat = glmnet::predict.cv.glmnet(fit, newx = as.matrix(t(x[,whichSamples])),
                                   s = "lambda.min")

  features$mutations[[drug]] = append(features$mutations[[drug]],
                                      rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'mutations'] = res$res$`0.5`$var
  predictionCor[drug, 'mutations'] = cor(y[whichSamples,],yhat)
}

# Now copy number.
int = overview %>% filter(has_cndata == T & has_single_drug == T)

x = cnMatrix[,colnames(cnMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$cnNumber[[drug]] = append(features$cnNumber[[drug]],
                                     rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'copy number'] = res$res$`0.5`$var
}

# finally gene expression
int = overview %>% filter(has_rnaseq == T & has_single_drug == T)

x = geneEx[,colnames(geneEx) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$geneEx[[drug]] = append(features$geneEx[[drug]],
                                   rownames(res$res$`0.5`$`1se`))

  varExplMatrix[drug, 'gene ex'] = res$res$`0.5`$var
}
```

```
LvarExpl[[iter]] = varExplMatrix
LpredCor[[iter]] = predictionCor
}


# summarize features

for (drug in names(features$mutations)){
  features$mutations[[drug]] = sort(table(features$mutations[[drug]]),
                                    decreasing=TRUE)
}
for (drug in names(features$cnNumber)){
  features$cnNumber[[drug]] = sort(table(features$cnNumber[[drug]]),
                                   decreasing=TRUE)
}
for (drug in names(features$geneEx)){
  features$geneEx[[drug]] = sort(table(features$geneEx[[drug]]),
                                 decreasing=TRUE)
}


# average over all iterations
ave_varExplMatrix = LvarExpl[[1]]
for (xx in 2:length(LvarExpl)){
  ave_varExplMatrix = ave_varExplMatrix + LvarExpl[[xx]]
}
ave_varExplMatrix = ave_varExplMatrix/length(LvarExpl)

#Sets variance explained to 0 if mean cross validated error greater than 1
ave_varExplMatrix[ave_varExplMatrix < 0] = 0

saveRDS(list('features'=features, 'LvarExpl' = LvarExpl,
             'ave_varExplMatrix' = ave_varExplMatrix),
        file = here("Rds",'singleResponseAssociations_cngene.Rds'))

end = Sys.time()
#end-start
```

Variance explained single response

Variance explained = 1-min(fit$cvm), where cvm = mean cross validated error. When var is less than 0, var is replaced by 0.

```
single.enet = readRDS(here("Rds", "singleResponseAssociations_cngene.Rds"))
pheatmap(single.enet$ave_varExplMatrix,
         color = colorRampPalette(brewer.pal(n = 4, name ="YlOrRd"))(100),
         fontsize_row = 8,
         main="Variance explained in single response (elastic net model)")
```


Variance explained in single response (elastic net model)
Talazoparib
Tamoxifen
PF−4708671
Wee1 Inhibitor
AZD7762
SN−38
Gemcitabine
Temozolomide
Avagacestat
Olaparib
NU7441
Sorafenib
Dasatinib
ZM447439
Irinotecan
Vinorelbine
Sapatinib
Vorinostat
Vismodegib
Vinblastine
Venetoclax
Uprosertib
Trametinib
Tozasertib
Tivozanib
Taselisib
SCH772984
SB505124
SB216763
Saracatinib
Ruxolitinib
RO−3306
Ribociclib
PRIMA−1MET
Pictilisib
Pemetrexed
PD173074
Palbociclib
Paclitaxel
Oxaliplatin
OSI−027
Nutlin−3a (−)
Nilotinib
Navitoclax
MK−2206
MK−1775
Luminespib
Linsitinib
LGK974
Lapatinib
KU−55933
I−BET−762
Fulvestrant
Erlotinib
Entospletinib
Entinostat
Docetaxel
Dactolisib
Dabrafenib
Crizotinib
Cisplatin
Carboplatin
Camptothecin
Bortezomib
BMS−754807
BI−2536
AZD8055
AZD6738
AZD6482
AZD4547
AZD0156
Axitinib
Alpelisib
Alisertib
5−Fluorouracil
Afatinib
gene ex
mutations
copy number
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0

Delta AUC

```
Y = median_DELTA_AUC
```

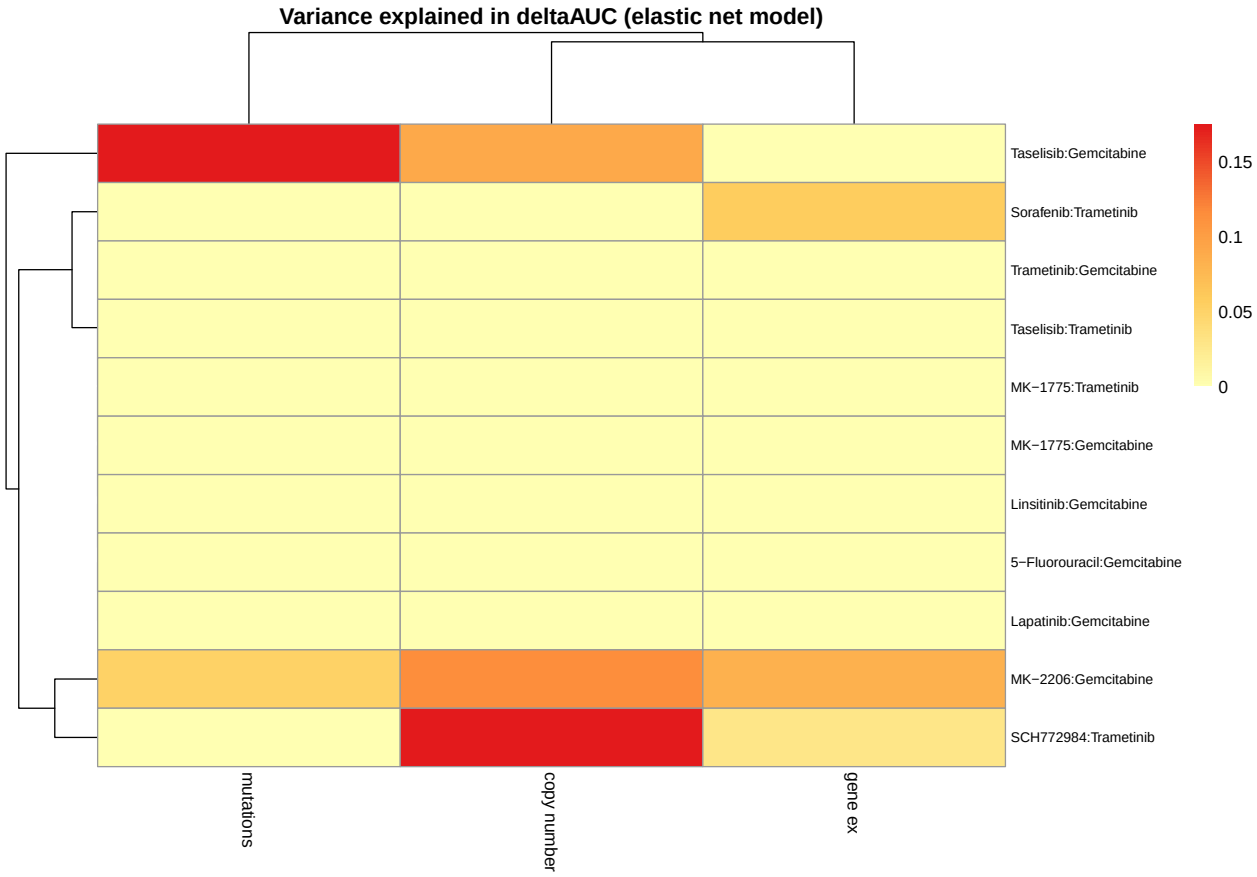
```
start = Sys.time()

# randomized folds give different results, so run this 100 times
LvarExpl = list()

features = list()
features$mutations = list()
for (drug in colnames(Y)){ features$mutations[[drug]]=c() }

features$cnNumber = features$mutations
features$geneEx = features$mutations


for (iter in 1:100){

# initialize a drugs by data type variance explained matrix
varExplMatrix = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(varExplMatrix) = colnames(Y)
colnames(varExplMatrix) = c('mutations', 'copy number', 'gene ex')

int = overview %>% filter(has_mutdata == T & has_combi_drug == T)

#First mutations
x = mutMatrix[,colnames(mutMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]


for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples,
                   alpha = 0.5, family='gaussian', standardize=FALSE)

  features$mutations[[drug]] = append(features$mutations[[drug]],
                                      rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'mutations'] = res$res$`0.5`$var
}

# now copy number
int = overview %>% filter(has_cndata == T & has_combi_drug == T)

x = cnMatrix[, colnames(cnMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
```

```
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$cnNumber[[drug]] = append(features$cnNumber[[drug]],
                                     rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'copy number'] = res$res$`0.5`$var
}

# finally gene expression
int = overview %>% filter(has_rnaseq == T & has_combi_drug == T)

x = geneEx[,colnames(geneEx) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$geneEx[[drug]] = append(features$geneEx[[drug]],
                                   rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'gene ex'] = res$res$`0.5`$var
}

LvarExpl[[iter]] = varExplMatrix

}


# summarize features
for (drug in names(features$mutations)){
  features$mutations[[drug]] = sort(table(features$mutations[[drug]]),
                                    decreasing=TRUE)
}
for (drug in names(features$cnNumber)){
  features$cnNumber[[drug]] = sort(table(features$cnNumber[[drug]]),
                                   decreasing=TRUE)
}
for (drug in names(features$geneEx)){
  features$geneEx[[drug]] = sort(table(features$geneEx[[drug]]),
                                 decreasing=TRUE)
}
```

```
# average over all iterations
ave_varExplMatrix = LvarExpl[[1]]
for (xx in 2:length(LvarExpl)){
  ave_varExplMatrix = ave_varExplMatrix + LvarExpl[[xx]]
}
ave_varExplMatrix = ave_varExplMatrix/length(LvarExpl)

ave_varExplMatrix[ave_varExplMatrix < 0] = 0


saveRDS(list('features'=features, 'LvarExpl' = LvarExpl,
             'ave_varExplMatrix' = ave_varExplMatrix),
        file = here("Rds",'combiResponseAssociations_AUC_cngene.Rds'))

end = Sys.time()

#end-start
```

Variance explained deltaAUC

```
L = readRDS(here("Rds", 'combiResponseAssociations_AUC_cngene.Rds'))

pheatmap(L$ave_varExplMatrix,
         color = colorRampPalette(brewer.pal(n = 4, name ="YlOrRd"))(100),
         fontsize_row = 8,
         main="Variance explained in deltaAUC (elastic net model)")
```



Delta fitted AUC

```
Y = median_DELTA_fAUC
```

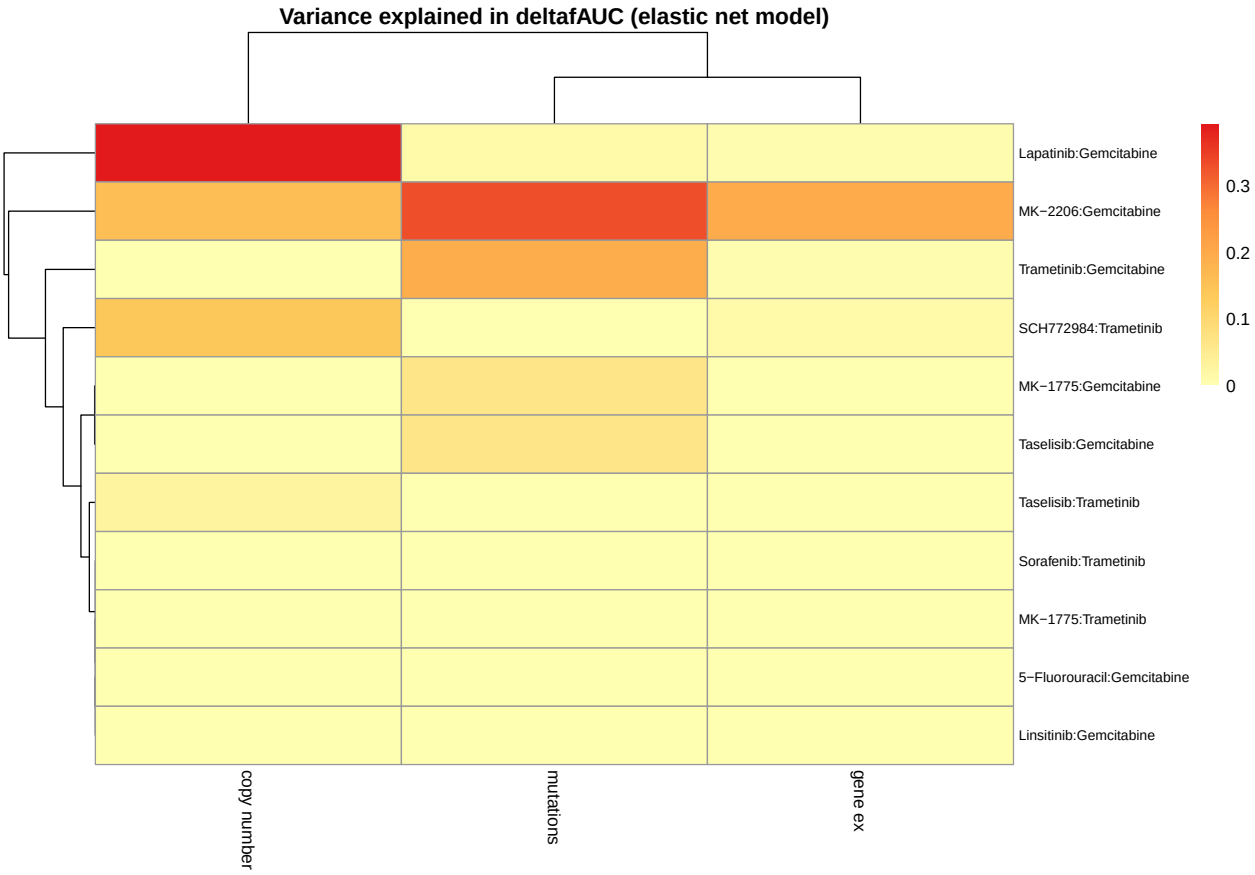
```
start = Sys.time()

# randomized folds give different results, so run this 100 times
LvarExpl = list()

features = list()
features$mutations = list()
for (drug in colnames(Y)){ features$mutations[[drug]]=c() }

features$cnNumber = features$mutations
features$geneEx = features$mutations


for (iter in 1:100){

# initialize a drugs by data type variance explained matrix
varExplMatrix = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(varExplMatrix) = colnames(Y)
colnames(varExplMatrix) = c('mutations', 'copy number', 'gene ex')

int = overview %>% filter(has_mutdata == T & has_combi_drug == T)
```

```
#First mutations
x = mutMatrix[,colnames(mutMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]


for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples,
                   alpha = 0.5, family='gaussian', standardize=FALSE)

  features$mutations[[drug]] = append(features$mutations[[drug]],
                                      rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'mutations'] = res$res$`0.5`$var
}

# now copy number
int = overview %>% filter(has_cndata == T & has_combi_drug == T)

x = cnMatrix[, colnames(cnMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$cnNumber[[drug]] = append(features$cnNumber[[drug]],
                                     rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'copy number'] = res$res$`0.5`$var
}

# finally gene expression
int = overview %>% filter(has_rnaseq == T & has_combi_drug == T)

x = geneEx[,colnames(geneEx) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
```

```
                    whichSamples, alpha = 0.5, family='gaussian',
                    standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$geneEx[[drug]] = append(features$geneEx[[drug]],
                                   rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'gene ex'] = res$res$`0.5`$var
}

LvarExpl[[iter]] = varExplMatrix

}

# summarize features
for (drug in names(features$mutations)){
  features$mutations[[drug]] = sort(table(features$mutations[[drug]]),
                                    decreasing=TRUE)
}
for (drug in names(features$cnNumber)){
  features$cnNumber[[drug]] = sort(table(features$cnNumber[[drug]]),
                                   decreasing=TRUE)
}
for (drug in names(features$geneEx)){
  features$geneEx[[drug]] = sort(table(features$geneEx[[drug]]),
                                 decreasing=TRUE)
}


# average over all iterations
ave_varExplMatrix = LvarExpl[[1]]
for (xx in 2:length(LvarExpl)){
  ave_varExplMatrix = ave_varExplMatrix + LvarExpl[[xx]]
}
ave_varExplMatrix = ave_varExplMatrix/length(LvarExpl)

ave_varExplMatrix[ave_varExplMatrix < 0] = 0

saveRDS(list('features'=features, 'LvarExpl' = LvarExpl,
             'ave_varExplMatrix' = ave_varExplMatrix),
        file = here('Rds','combiResponseAssociations_fAUC_cngene.Rds'))

end = Sys.time()
#end-start
```

Variance explained fAUC

```
L = readRDS(here("Rds", 'combiResponseAssociations_fAUC_cngene.Rds'))
pheatmap(L$ave_varExplMatrix,
         color = colorRampPalette(brewer.pal(n = 4, name ="YlOrRd"))(100),
         fontsize_row = 8,
         main="Variance explained in deltafAUC (elastic net model)")
```



Z-score

```
Y = median_ZSCORE
```

```
start = Sys.time()

# randomized folds give different results, so run this 100 times
LvarExpl = list()

features = list()
features$mutations = list()
for (drug in colnames(Y)){ features$mutations[[drug]]=c() }

features$cnNumber = features$mutations
features$geneEx = features$mutations


for (iter in 1:100){

# initialize a drugs by data type variance explained matrix
varExplMatrix = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(varExplMatrix) = colnames(Y)
colnames(varExplMatrix) = c('mutations', 'copy number', 'gene ex')
```

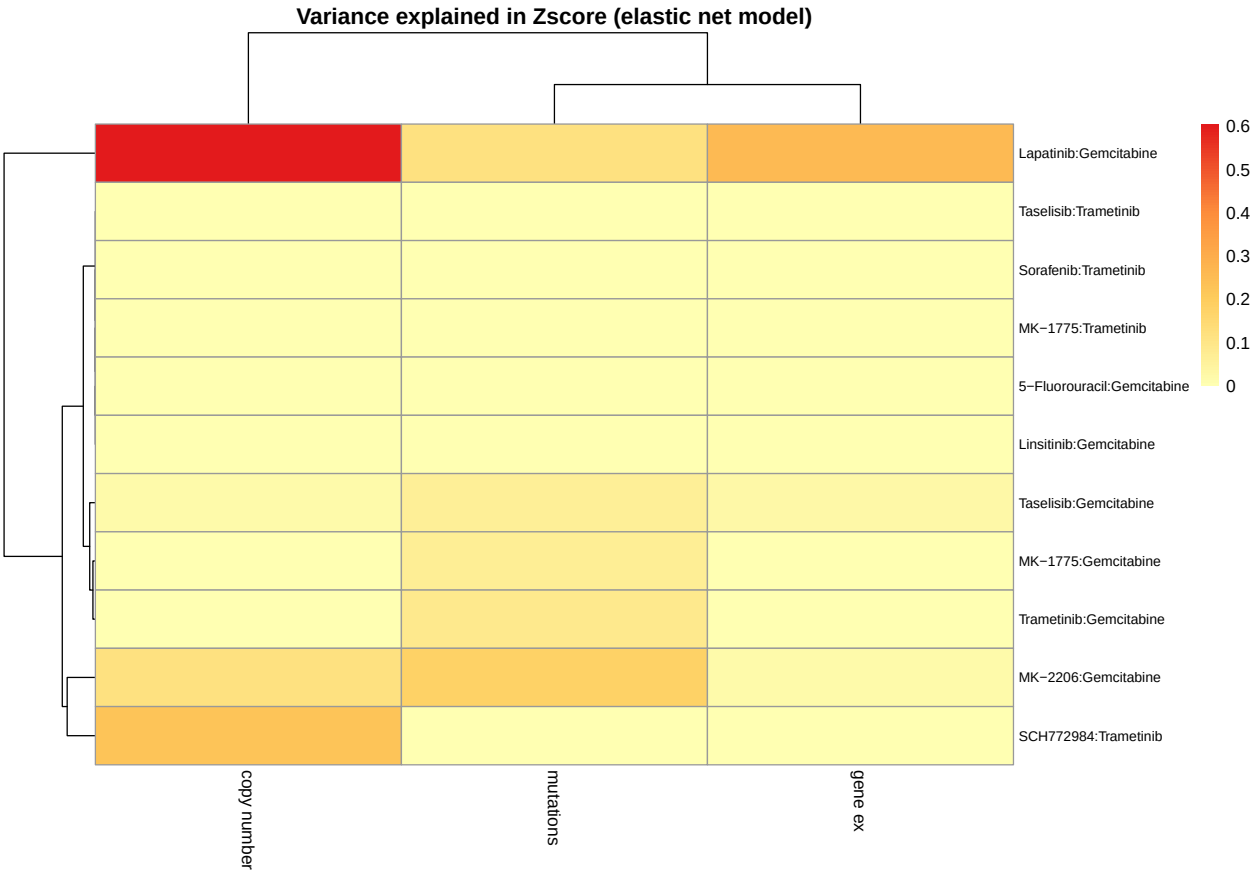
```
int = overview %>% filter(has_mutdata == T & has_combi_drug == T)

#First mutations
x = mutMatrix[,colnames(mutMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]


for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE],
                   as.matrix(t(x[,whichSamples])), whichSamples,
                   alpha = 0.5, family='gaussian', standardize=FALSE)

  features$mutations[[drug]] = append(features$mutations[[drug]],
                                      rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'mutations'] = res$res$`0.5`$var
}

# now copy number
int = overview %>% filter(has_cndata == T & has_combi_drug == T)

x = cnMatrix[, colnames(cnMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$cnNumber[[drug]] = append(features$cnNumber[[drug]],
                                     rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'copy number'] = res$res$`0.5`$var
}

# finally gene expression
int = overview %>% filter(has_rnaseq == T & has_combi_drug == T)

x = geneEx[,colnames(geneEx) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
```

```
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$geneEx[[drug]] = append(features$geneEx[[drug]],
                                   rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'gene ex'] = res$res$`0.5`$var
}

LvarExpl[[iter]] = varExplMatrix

}

# summarize features
for (drug in names(features$mutations)){
  features$mutations[[drug]] = sort(table(features$mutations[[drug]]),
                                    decreasing=TRUE)
}
for (drug in names(features$cnNumber)){
  features$cnNumber[[drug]] = sort(table(features$cnNumber[[drug]]),
                                   decreasing=TRUE)
}
for (drug in names(features$geneEx)){
  features$geneEx[[drug]] = sort(table(features$geneEx[[drug]]),
                                 decreasing=TRUE)
}


# average over all iterations
ave_varExplMatrix = LvarExpl[[1]]
for (xx in 2:length(LvarExpl)){
  ave_varExplMatrix = ave_varExplMatrix + LvarExpl[[xx]]

}
ave_varExplMatrix = ave_varExplMatrix/length(LvarExpl)

ave_varExplMatrix[ave_varExplMatrix < 0] = 0

saveRDS(list('features'=features, 'LvarExpl' = LvarExpl,
             'ave_varExplMatrix' = ave_varExplMatrix),
        file = here('Rds','combiResponseAssociations_Zscore_cngene.Rds'))

end = Sys.time()
#end-start
```

Variance explained Zscore

```
L = readRDS(here("Rds", 'combiResponseAssociations_Zscore_cngene.Rds'))

pheatmap(L$ave_varExplMatrix,
```

```
        color = colorRampPalette(brewer.pal(n = 4, name ="YlOrRd"))(100),
        fontsize_row = 8,
        main="Variance explained in Zscore (elastic net model)")
```



E-max

```
Y = median_DELTA_EMAX
```

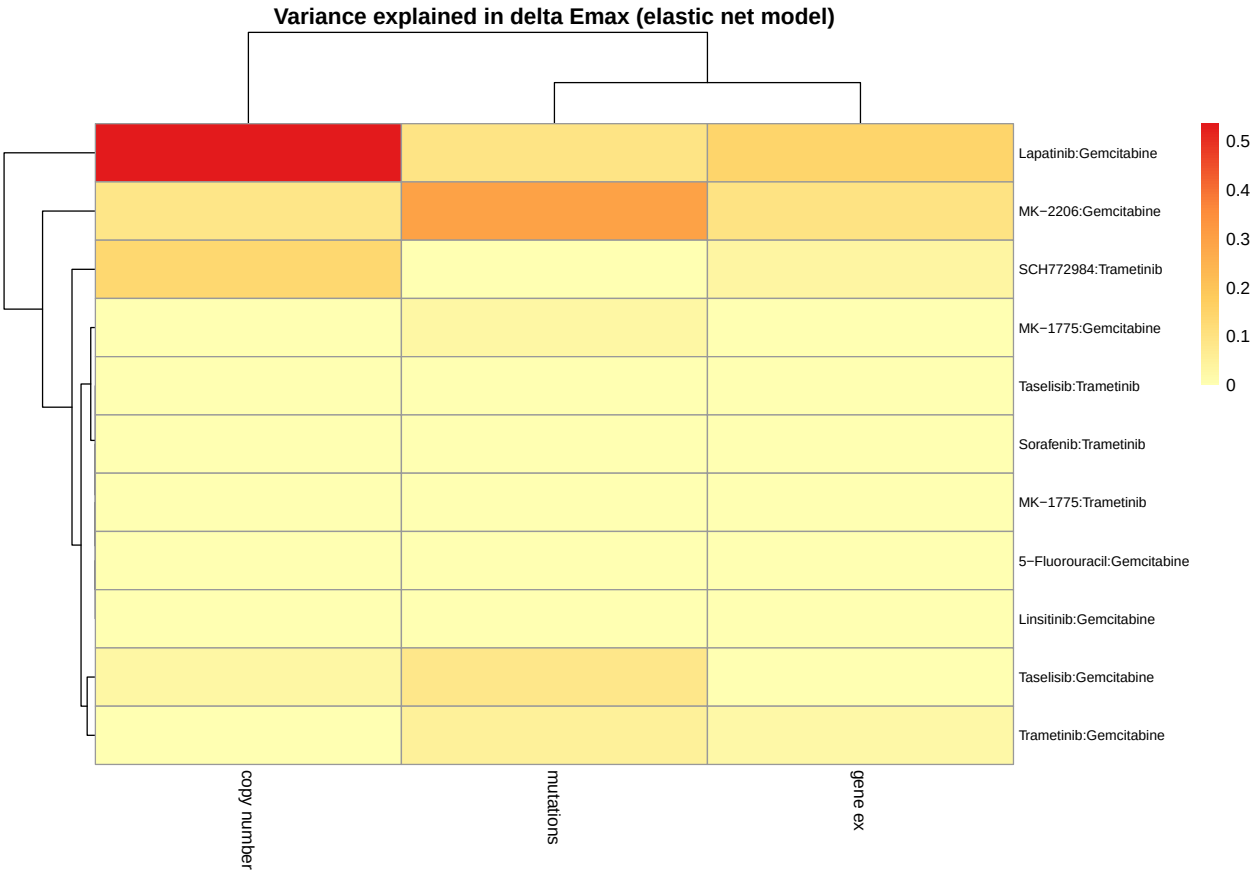
```
start = Sys.time()

# randomized folds give different results, so run this 100 times
LvarExpl = list()

features = list()
features$mutations = list()
for (drug in colnames(Y)){ features$mutations[[drug]]=c() }

features$cnNumber = features$mutations
features$geneEx = features$mutations


for (iter in 1:100){

# initialize a drugs by data type variance explained matrix
```

```
varExplMatrix = matrix(NA, nrow = ncol(Y), ncol = 3)
rownames(varExplMatrix) = colnames(Y)
colnames(varExplMatrix) = c('mutations', 'copy number', 'gene ex')

int = overview %>% filter(has_mutdata == T & has_combi_drug == T)

#First mutations
x = mutMatrix[,colnames(mutMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]


for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples,
                   alpha = 0.5, family='gaussian', standardize=FALSE)

  features$mutations[[drug]] = append(features$mutations[[drug]],
                                      rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'mutations'] = res$res$`0.5`$var
}

# now copy number

int = overview %>% filter(has_cndata == T & has_combi_drug == T)

x = cnMatrix[, colnames(cnMatrix) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]

for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE],
                   as.matrix(t(x[,whichSamples])), whichSamples, alpha = 0.5,
                   family='gaussian', standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$cnNumber[[drug]] = append(features$cnNumber[[drug]],
                                     rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'copy number'] = res$res$`0.5`$var
}

# finally gene expression
int = overview %>% filter(has_rnaseq == T & has_combi_drug == T)
x = geneEx[,colnames(geneEx) %in% int$manuscript_id]
x = x[rowSums(x)!=0,]
```

```
for (drug in colnames(Y)){
  #print(drug)
  #y = as.matrix(Y[int[,'allHUBids'],drug,drop=FALSE])
  y = as.matrix(Y[rownames(Y) %in% int$manuscript_id,drug,drop=FALSE])
  whichSamples = rownames(y)[!is.na(y)]
  res = glmnetLoop(y[whichSamples,,drop=FALSE], as.matrix(t(x[,whichSamples])),
                   whichSamples, alpha = 0.5, family='gaussian',
                   standardize=FALSE)
  #print(res$res$`0.5`$`1se`)
  features$geneEx[[drug]] = append(features$geneEx[[drug]],
                                   rownames( res$res$`0.5`$`1se` ))

  varExplMatrix[drug, 'gene ex'] = res$res$`0.5`$var
}

LvarExpl[[iter]] = varExplMatrix

}

# summarize features
for (drug in names(features$mutations)){
  features$mutations[[drug]] = sort(table(features$mutations[[drug]]),
                                    decreasing=TRUE)
}
for (drug in names(features$cnNumber)){
  features$cnNumber[[drug]] = sort(table(features$cnNumber[[drug]]),
                                   decreasing=TRUE)
}
for (drug in names(features$geneEx)){
  features$geneEx[[drug]] = sort(table(features$geneEx[[drug]]),
                                 decreasing=TRUE)
}


# average over all iterations
ave_varExplMatrix = LvarExpl[[1]]
for (xx in 2:length(LvarExpl)){
  ave_varExplMatrix = ave_varExplMatrix + LvarExpl[[xx]]
}
ave_varExplMatrix = ave_varExplMatrix/length(LvarExpl)

ave_varExplMatrix[ave_varExplMatrix < 0] = 0

saveRDS(list('features'=features, 'LvarExpl' = LvarExpl,
             'ave_varExplMatrix' = ave_varExplMatrix),
        file = here('Rds','combiResponseAssociations_Emax_cngene.Rds'))

end = Sys.time()
#end - start
```

Variance explained Delta Emax

```
L = readRDS(here("Rds", 'combiResponseAssociations_Emax_cngene.Rds'))

pheatmap(L$ave_varExplMatrix,
         color = colorRampPalette(brewer.pal(n = 4, name ="YlOrRd"))(100),
         fontsize_row = 8,
         main="Variance explained in delta Emax (elastic net model)")
```



Visualizing strongest associations

Single drug response

Plotting tools:

```
getColors = function(n){

  fewColors = c('black', 'red', 'blue', 'green',
                'cyan', 'magenta', 'coral', 'gray', 'orange')
  if (n <= length(fewColors)){ return(fewColors[1:n]) }



  colors = colors()[!grepl('grey',colors())]
```

```
  colors = colors[!grepl('gray',colors)]
  colors = colors[!grepl('white',colors)]
  colors = colors[!grepl('aliceblue',colors)]
  colors = colors[!grepl('gainsboro',colors)]
  colors = colors[!grepl('ivory1',colors)]
  colors = colors[!grepl('lavenderblush1',colors)]

  rem = length(colors) %% n

  k = (length(colors)-rem) / n  # by

  ind = seq(1,length(colors)-rem, by = k)
  stopifnot(length(ind)==n)

  return(colors[ind])
}


#' Plot the distributions of two sets of plots. Jitter around (1) and (2).
#'
#' Boxplot may not work here, because the two sets might not have the same size.
#'
#' @param set1 - set of points
#' @param set2 - second set of points
#' @param title - desired title for the plot; default is ''
#' @export
#' @examples
#' ...

plotTwoSets = function(L, title ='', xlabel = '', ylabel = '', titlecex = 1,
  color1 = 'black', color2 = 'red', ylim = c(), sampleNames = FALSE, cex = 1){

    set1 = L[[1]]
    set2 = L[[2]]

    lbound = min(c(set1,set2))
    ubound = max(c(set1,set2))

    if (length(ylim)==0){ ylim = c(lbound, ubound) }

    if (length(set1)<=1 | length(set2)<=1){
      pval = NA
    } else {
      res = t.test(set1,set2)
      pval = res$p.value
    }

    title = paste(title, ':', '\n', 'p-value ', format(pval, digits = 3),sep='')

    library(beeswarm, quietly=TRUE)
    #L = list()
    #L[[1]] = set1
    #L[[2]] = set2
```

```
    # removed yaxt = 'n'
    plottingInfo = beeswarm(L, method='swarm', cex=cex,
                            col = c(color1,color2), pch=19,
                            ylab=ylabel, main = title,
                            cex.main = titlecex, xlab = xlabel, ylim = ylim)
    xcoords = plottingInfo$'x'
    ycoords = plottingInfo$'y'

    if (sampleNames){
      cnt = 0
      for (coords in xcoords){
          cnt = cnt + 1
          if (cnt%%2==0){
            xcoords[cnt] = xcoords[cnt] - 0.1
          } else {
            xcoords[cnt] = xcoords[cnt] + 0.1
          }
      }

      text(xcoords, ycoords, c(names(set1), names(set2)))

    }
    #plot(c(0.5,2.5),c(lbound,ubound),type='n',
    #main = title, xaxt='n', xlab = xlabel, ylab = ylabel)

    #j = jitter(rep(1,length(set1)), factor = 4)
    #points(j,set1, col = color1, pch = 19, cex = 0.7)
    ##j = jitter(rep(1.2,length(set1)), factor = 3.5)
    #text(j+0.07, set1-0.05, labels = names(set1), cex = 0.7, col = color1)

    #j = jitter(rep(2,length(set2)), factor = 4)
    #points(j,set2, col = color2, pch = 19, cex = 0.7)
    ##j = jitter(rep(2.2,length(set2)), factor = 3.5)
    #text(j+0.07, set2-0.05, labels = names(set2), cex = 0.7, col = color2)

}

dplot = function(drug, top, dataType, varExpl, response, ylab){

  topFour = top[1:min(4,length(top))]

  #stopifnot('DrugID' == substr(drugID, 1, 6))
  #stopifnot(length(topFour) == 4)
  stopifnot(class(topFour) == 'character')
  stopifnot( dataType %in% c("mutations", "cnNumber", "geneEx"))

  # response can be {IC50, AUC, median_DELTA_AUC,
  # median_DELTA_fAUC, median_ZSCORE, median_EMAX}
  response = response[, drug, drop=FALSE]
  samples = rownames(response)

  stopifnot(class(varExpl) == 'numeric')
  stopifnot(length(varExpl) == 1)
```

```
if (dataType == 'mutations'){

  whichSamples = colnames(mutMatrix)
  whichSamples = intersect(whichSamples, samples)
  x = mutMatrix[topFour, whichSamples, drop=F] #Added drop=F to fix bug
  y = response[whichSamples,,drop=FALSE]

  par(mfrow=c(2,2))

  for (topHowMany in 1:length(topFour)){

    wt = colnames(x)[ x[topFour[topHowMany], ] == 0 ]
    mt = colnames(x)[ x[topFour[topHowMany], ] != 0 ]
    plotTwoSets(list('mt'=y[mt,], 'wt'=y[wt,]),
                color1 = 'darkred', color2 = 'darkgreen',
                ylabel = ylab,
                title = paste0(drug, ' and ', topFour[topHowMany]))

  }

  par(mfrow=c(1,1))

} else if (dataType == 'cnNumber'){

  whichSamples = colnames(cnMatrix)

  whichSamples = intersect(whichSamples, samples)
  x = cnMatrix[topFour, whichSamples ]
  y = response[whichSamples,,drop=FALSE]

  par(mfrow=c(2,2))

  for (topHowMany in 1:length(topFour)){

    wt = colnames(x)[ x[topFour[topHowMany], ] == 0 ]
    mt = colnames(x)[ x[topFour[topHowMany], ] != 0 ]
    plotTwoSets(list('mt'=y[mt,], 'wt'=y[wt,]),
                color1 = 'darkred', color2 = 'darkgreen',
                ylabel = ylab,
                title = paste0(drug, ' and ', topFour[topHowMany]))

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

} else if (dataType == 'geneEx'){

  whichSamples = colnames(geneEx)
  whichSamples = intersect(whichSamples, samples)
  y = response[whichSamples,]

  temp = geneEx[top,whichSamples]
  cols = getColors(nrow(temp))
```

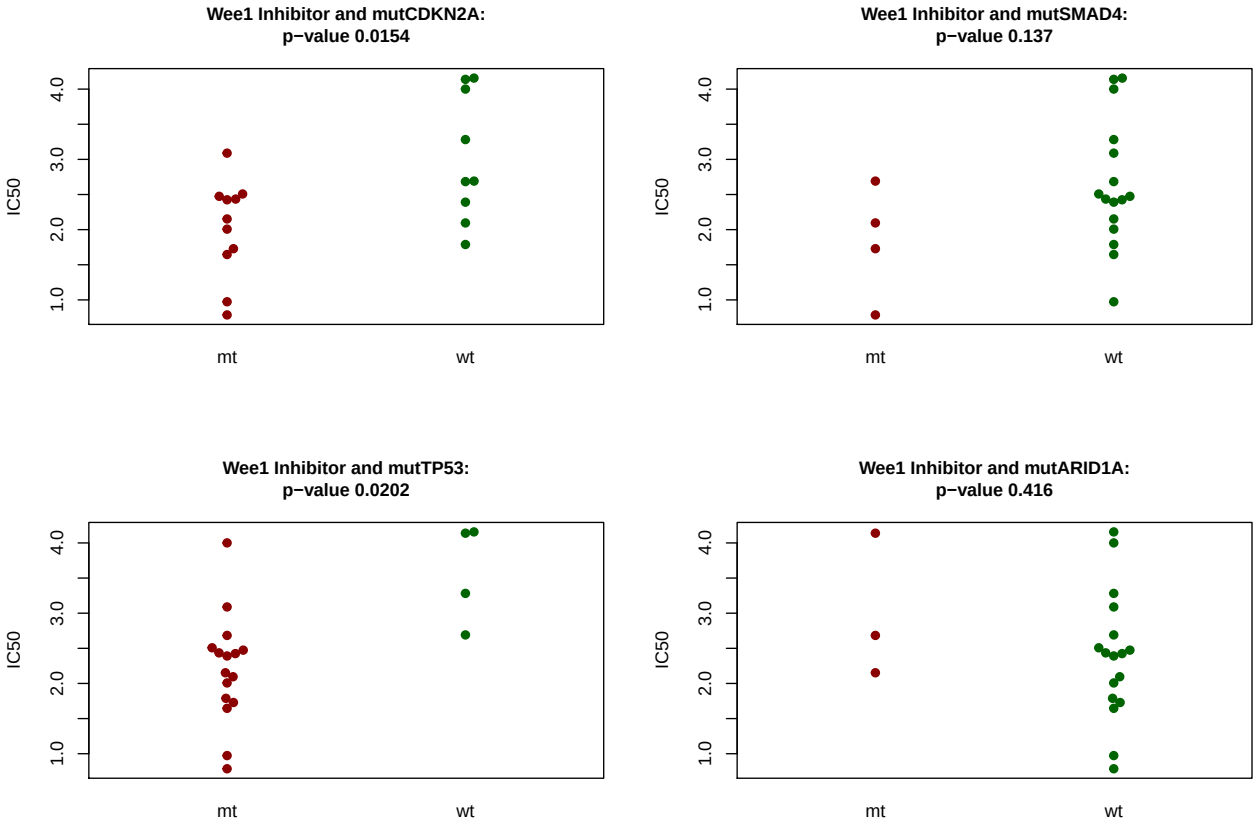
```
    #print(temp)
    #print(top)

    par(mfrow=c(2,2))
    plot(y, temp[top[1],], pch = 19, cex = 1, col = cols[1],
         xlab = ylab, ylab = top[1],
         main = paste0(drug, ' response against expression of ', top[1]),
         cex.main = 0.8)
    plot(y, temp[top[2],], pch = 19, cex = 1, col = cols[2],
         xlab = ylab, ylab = top[2],
         main = paste0(drug, ' response against expression of ', top[2]),
         cex.main = 0.8)

    if (length(top) > 2){
    plot(y, temp[top[3],], pch = 19, cex = 1, col = cols[3],
         xlab = ylab, ylab = top[3],
         main = paste0(drug, ' response against expression of ', top[3]),
         cex.main = 0.8)
    }
    if (length(top) > 3){
    plot(y, temp[top[4],], pch = 19, cex = 1, col = cols[4],
         xlab = ylab, ylab=top[4],
         main = paste0(drug, ' response against expression of ',
                       top[4]), cex.main = 0.8)
    }
    par(mfrow=c(1,1))

  }

}
```

IC50

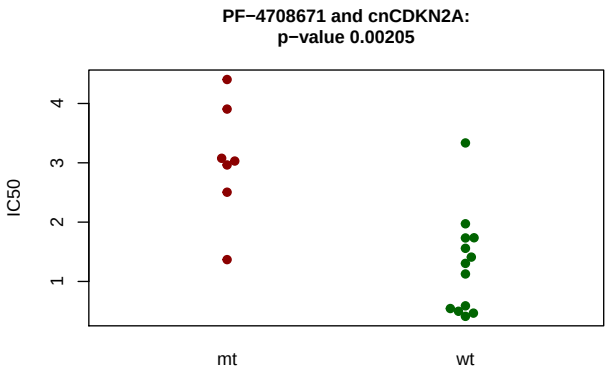
Note: A filter of 20 may have been appropriate for CRC but returns no hits in this analysis. Todo: Discuss a more appropriate filter.

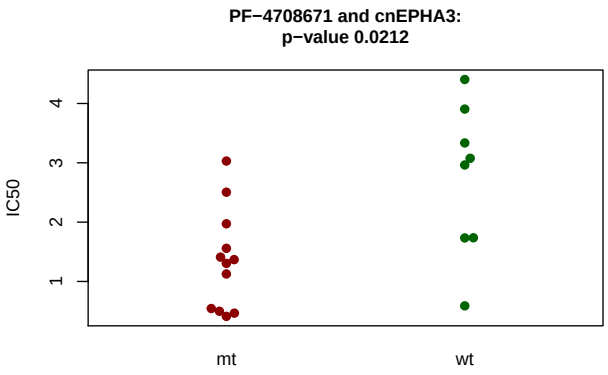
```
L = readRDS(here('Rds','singleResponseAssociations_cngene.Rds'))
colnames(L$ave_varExplMatrix) = names(L$features)

for (dataType in colnames(L$ave_varExplMatrix)){
  #dataType = colnames(L$ave_varExplMatrix)[1] #testing
  posDrugs = rownames(L$ave_varExplMatrix)[L$ave_varExplMatrix[, dataType] > 0.1]

  #print(c(dataType, posDrugs))
  for (drug in posDrugs){

    top = L$features[[dataType]][[drug]][ L$features[[dataType]][[drug]] > 20 ]
    if (length(top)==0){ next }
    if (length(top) == 1){ top = L$features[[dataType]][[drug]][1:2]  }

    # dotplot for these three features
    plot = dplot(drug, names(top[1:min(4,length(top))]), dataType,
                 L$ave_varExplMatrix[drug, dataType], IC50, "IC50")
```

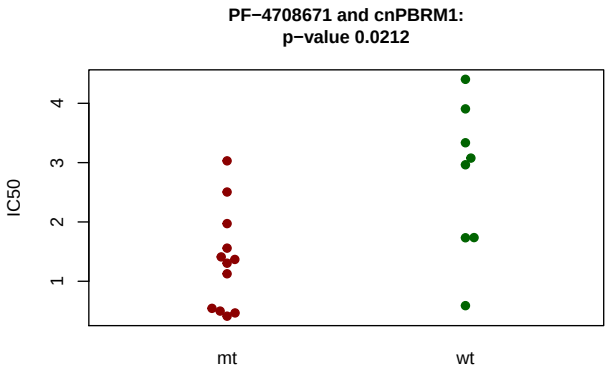
```
    print(plot)
  }
}
```

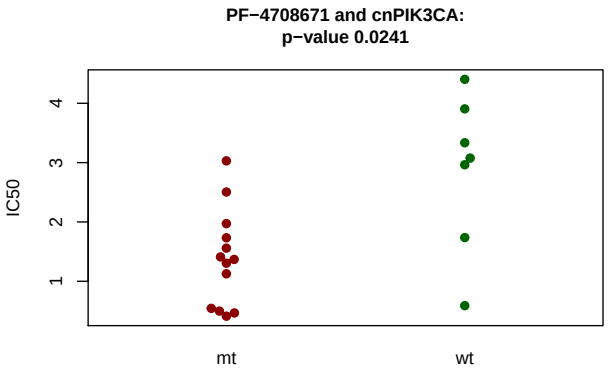



```
## $mfrow
## [1] 2 2
```

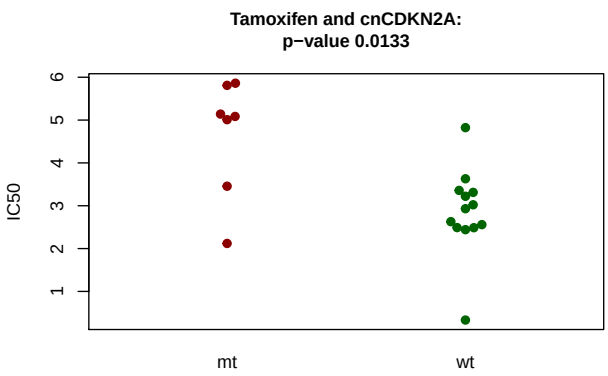

PF−4708671 and cnCDKN2A:
p−value 0.00205
IC50
mt
wt

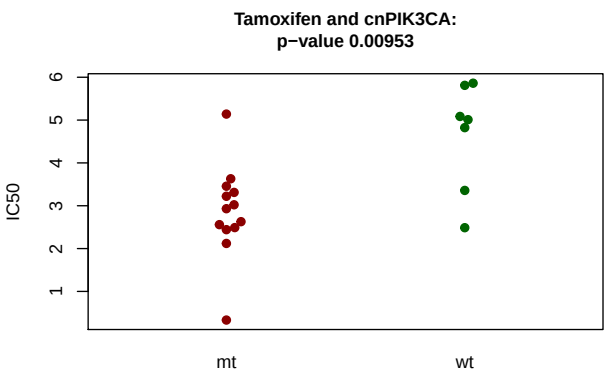

PF−4708671 and cnEPHA3:
p−value 0.0212
IC50
mt
wt

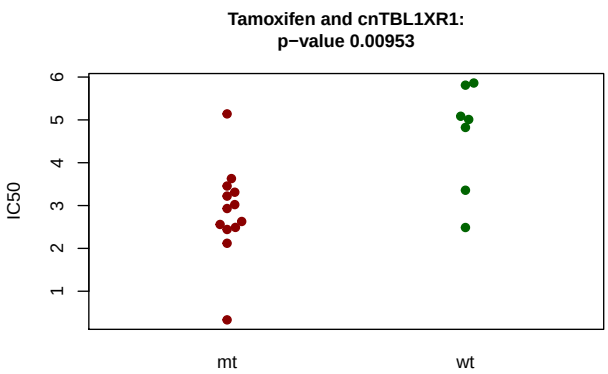

PF−4708671 and cnPBRM1:
p−value 0.0212
IC50
mt
wt


PF−4708671 and cnPIK3CA:
p−value 0.0241
IC50
mt
wt

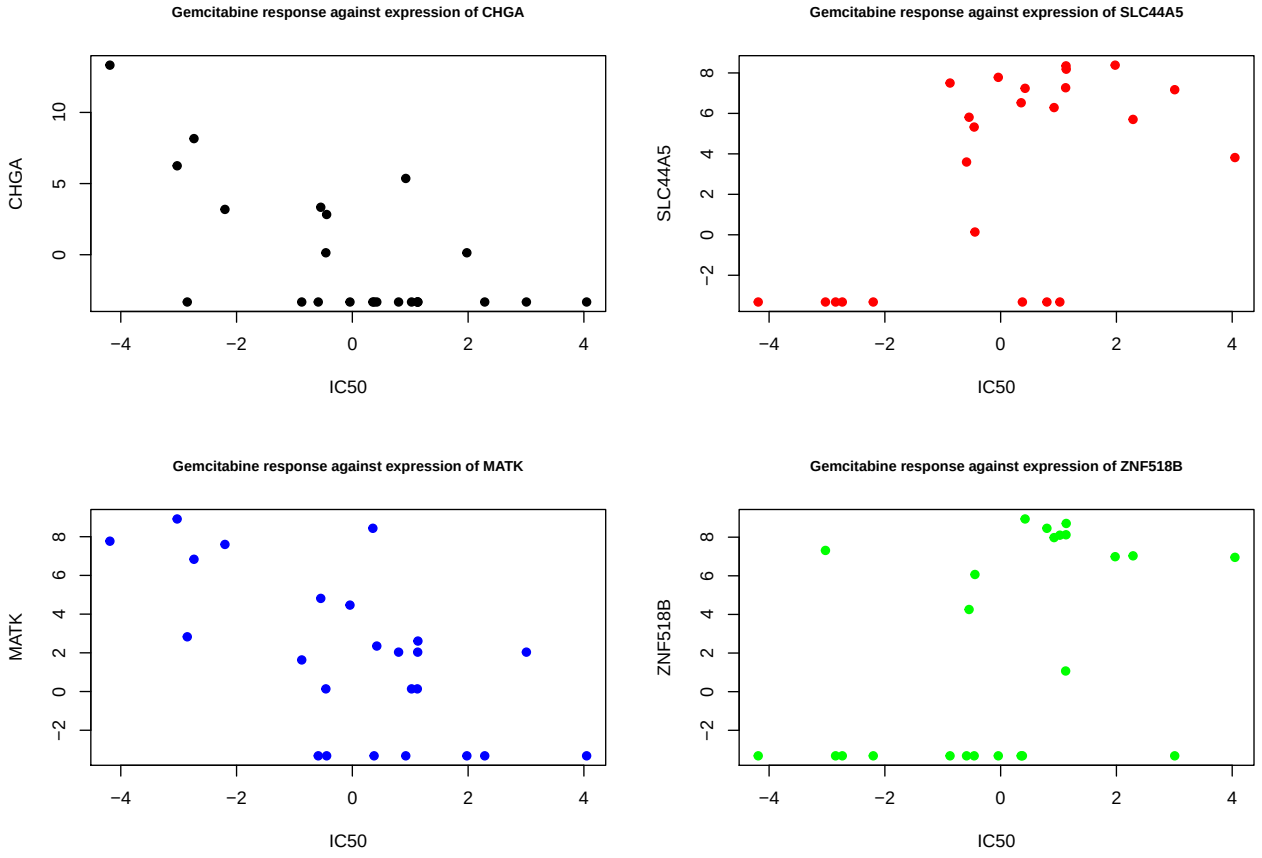
```
## $mfrow
## [1] 2 2
```


Tamoxifen and cnCDKN2A:
p−value 0.0133
IC50
mt
wt

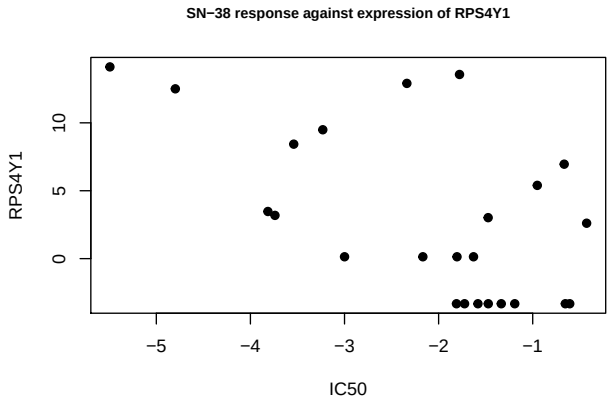

Tamoxifen and cnPIK3CA:
p−value 0.00953
IC50
mt
wt

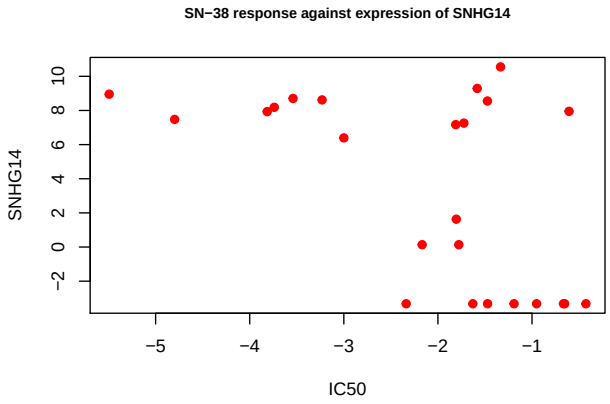

Tamoxifen and cnTBL1XR1:
p−value 0.00953
IC50
mt
wt

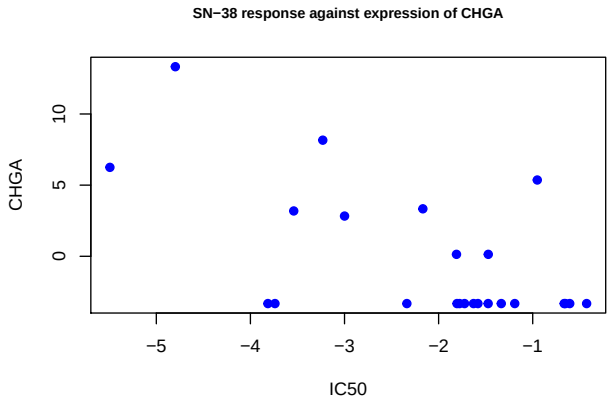
```
## $mfrow
## [1] 2 2
```

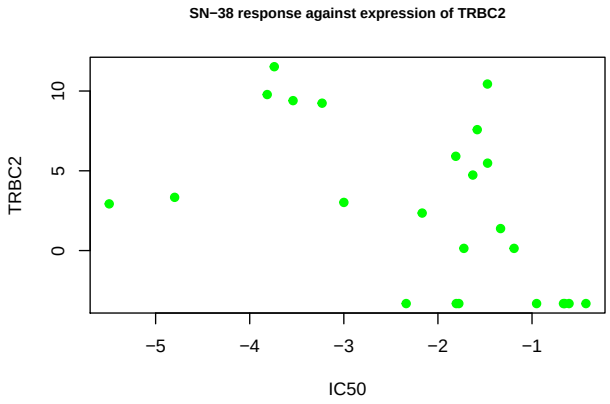



```
## $mfrow
## [1] 2 2
```

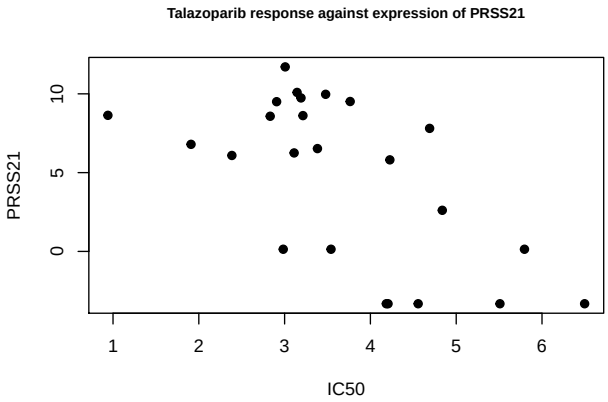

SN−38 response against expression of RPS4Y1
RPS4Y1
0
5
10
−5
−4
−3
−2
−1
IC50

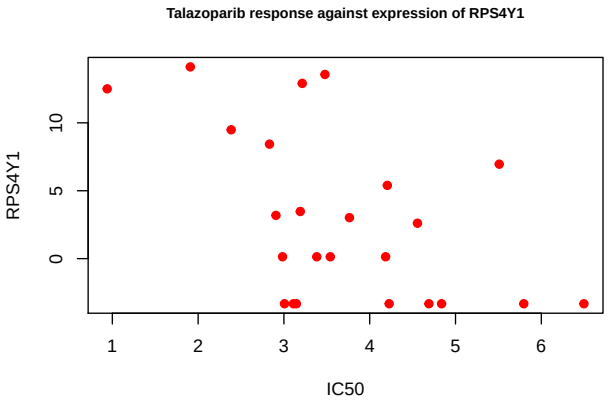

SN−38 response against expression of SNHG14
SNHG14
−2
0
2
4
6
8
10
−5
−4
−3
−2
−1
IC50

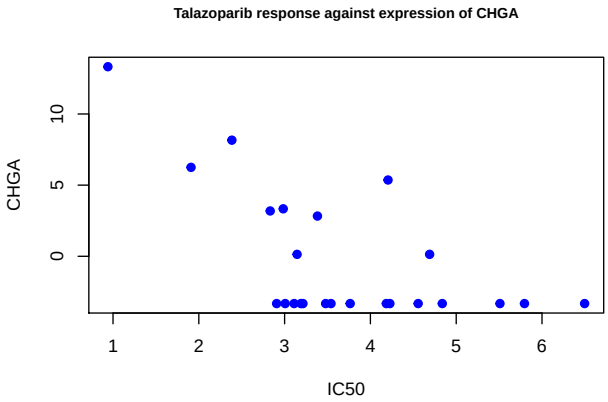

SN−38 response against expression of CHGA
CHGA
0
5
10
−5
−4
−3
−2
−1
IC50

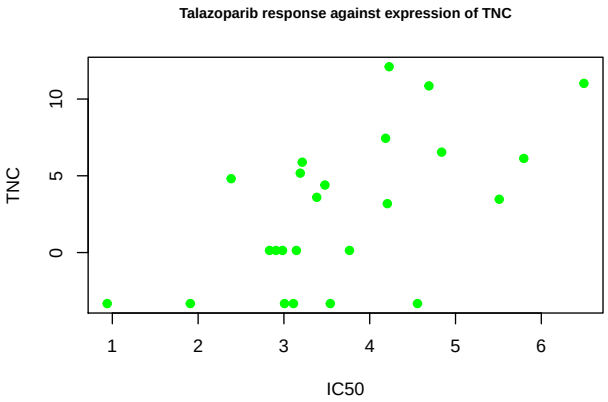

SN−38 response against expression of TRBC2
TRBC2
0
5
10
−5
−4
−3
−2
−1
IC50

```
## $mfrow
## [1] 2 2
```

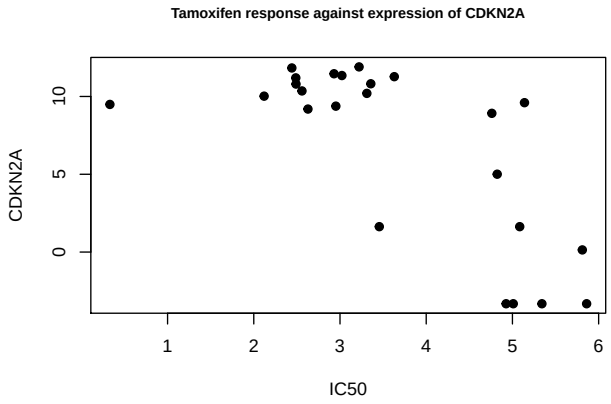

Talazoparib response against expression of PRSS21
PRSS21
IC50

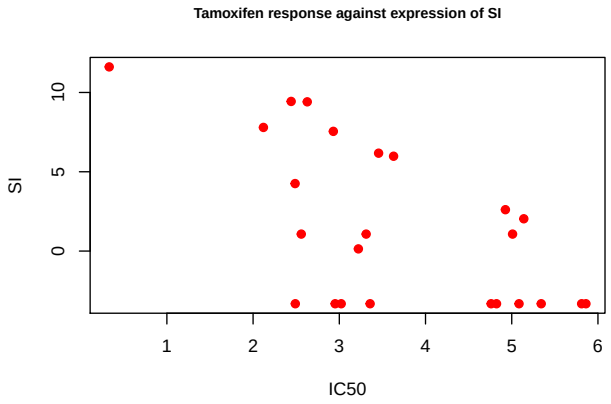

Talazoparib response against expression of RPS4Y1
RPS4Y1
IC50

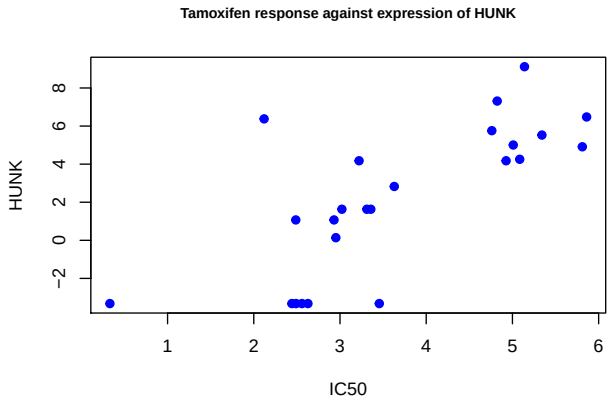

Talazoparib response against expression of CHGA
CHGA
IC50

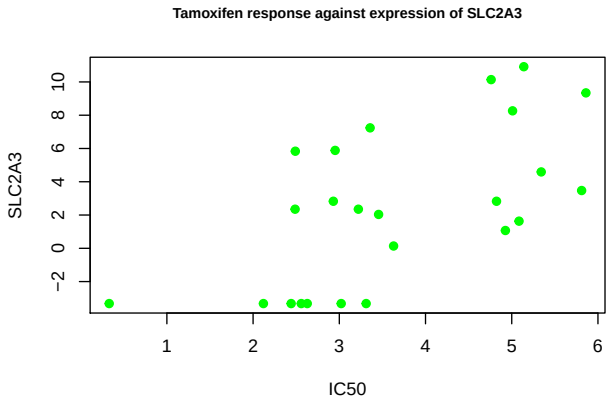

Talazoparib response against expression of TNC
TNC
IC50

```
## $mfrow
## [1] 2 2
```

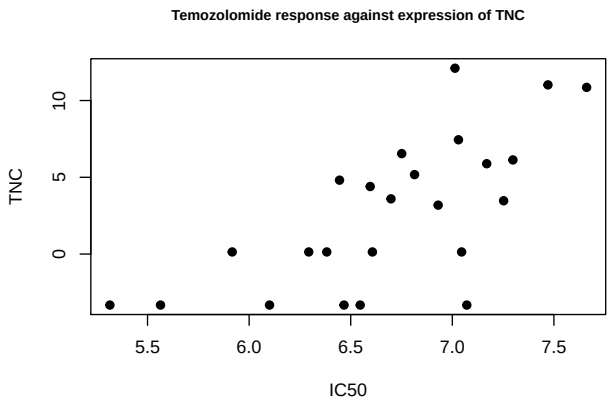


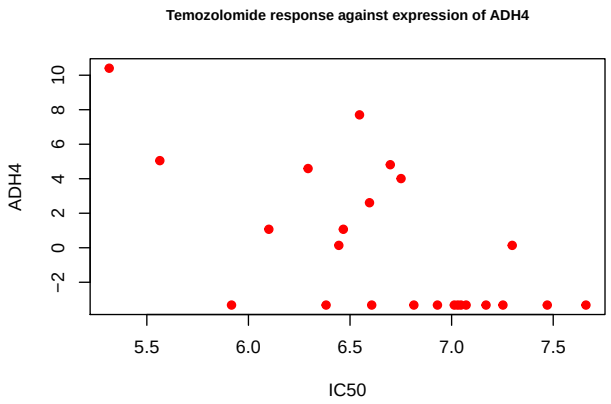







```
## $mfrow
## [1] 2 2
```






```
## $mfrow
## [1] 2 2
```

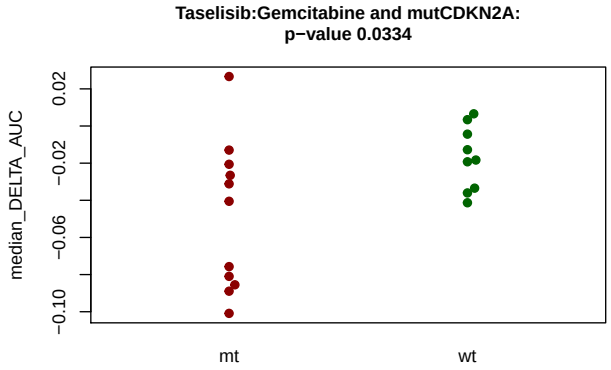
Combination response

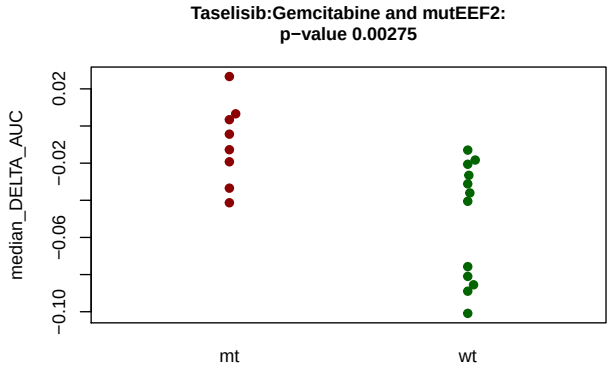
Delta AUC

```
L = readRDS(here('Rds','combiResponseAssociations_AUC_cngene.Rds'))
colnames(L$ave_varExplMatrix) = names(L$features)
```

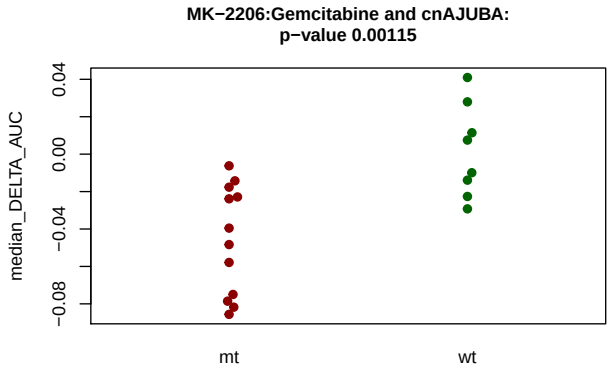
```
for (dataType in colnames(L$ave_varExplMatrix)){
  posDrugs = rownames(L$ave_varExplMatrix)[L$ave_varExplMatrix[, dataType] > 0.1]

  #print(c(dataType, posDrugs))
  for (drug in posDrugs){

    #drug = posDrugs[1] #testing
    #print(c(dataType, drug, L$ave_varExplMatrix[drug, dataType]))
    #print( L$features[[dataType]][[drug]] )

    top = L$features[[dataType]][[drug]][ L$features[[dataType]][[drug]] > 20 ]
    print(top)
    if (length(top)==0){ next }
    if (length(top) == 1){ top = L$features[[dataType]][[drug]][1:2]  }

    # dotplot for these three features
    dplot(drug, names(top[1:min(4,length(top))]), dataType,
          L$ave_varExplMatrix[drug, dataType],
          median_DELTA_AUC, "median_DELTA_AUC")
  }
}
```

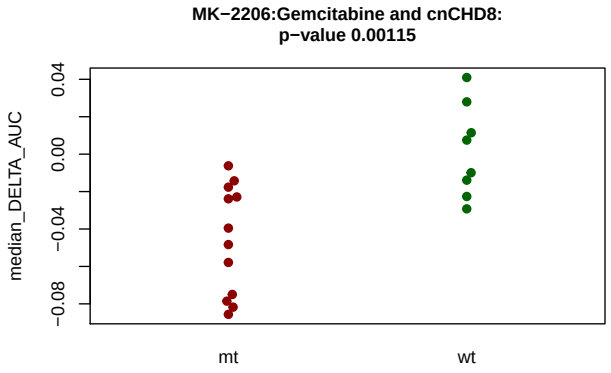
```
##
## mutCDKN2A   mutEEF2
##        57        39
```

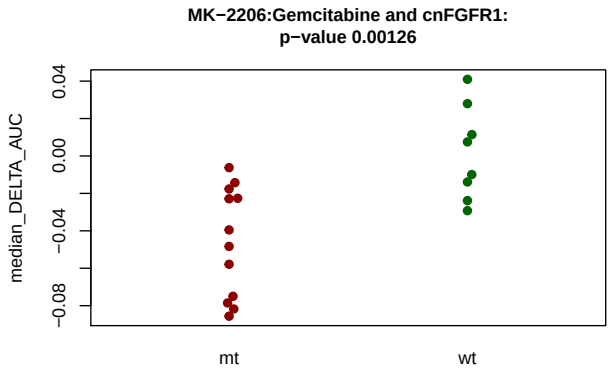


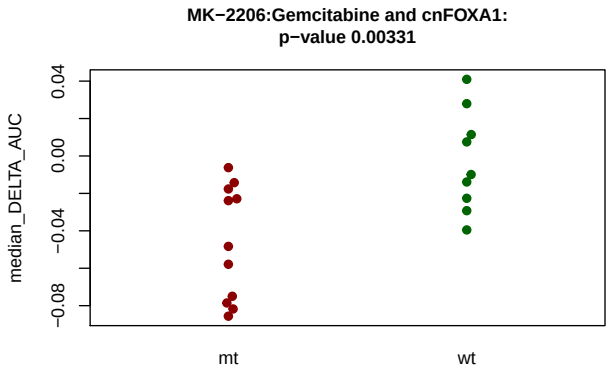



```
##
##  cnAJUBA   cnCHD8  cnFGFR1  cnFOXA1 cnCDKN2A
##       42       42       42       42       29
```

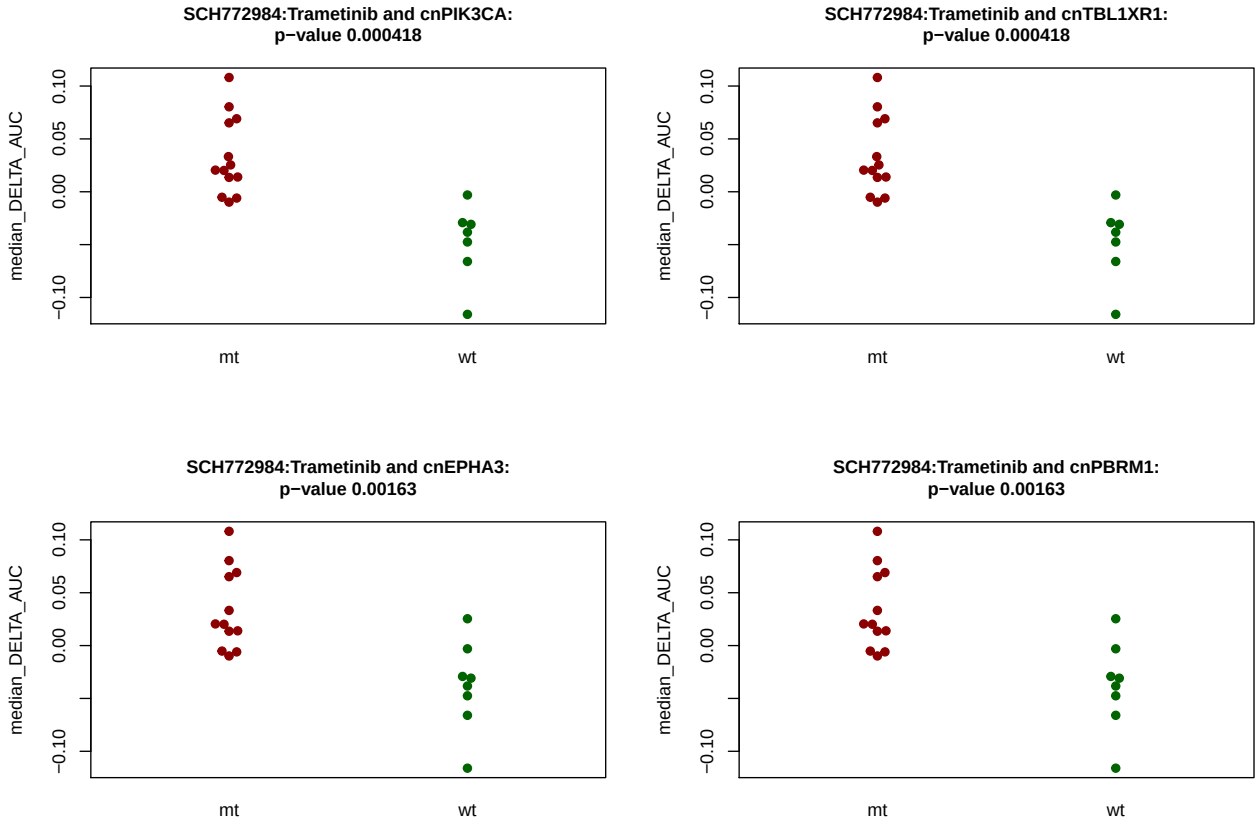






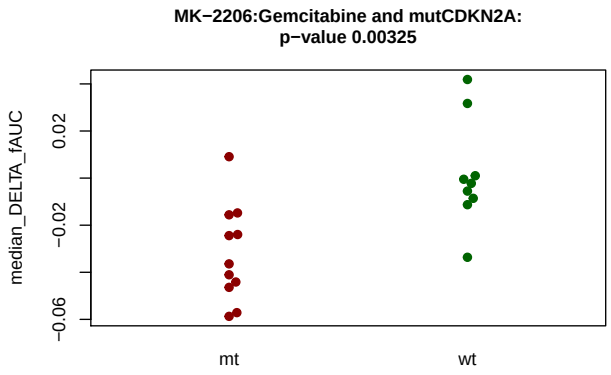


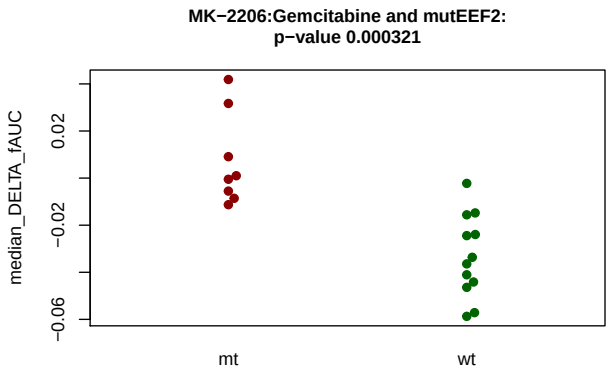

```
##
##  cnPIK3CA cnTBL1XR1   cnEPHA3   cnPBRM1
##        25        25        21        21
```

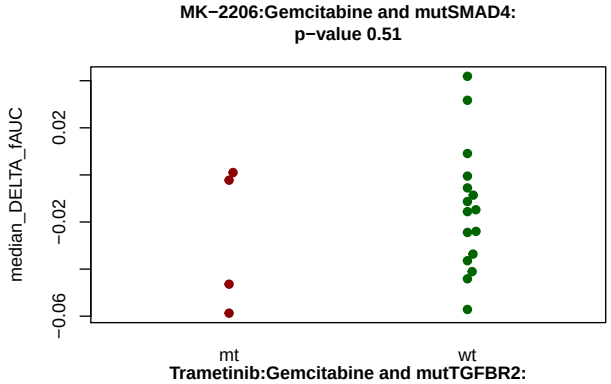


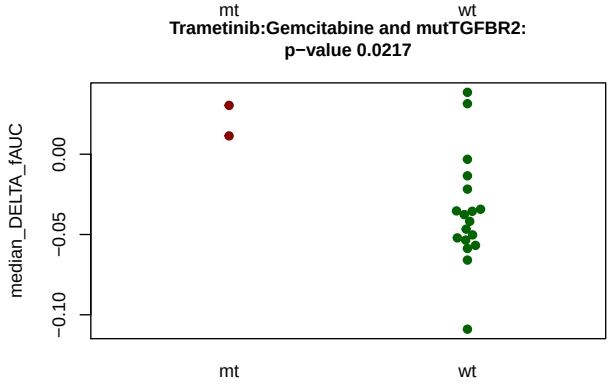
Delta fitted AUC

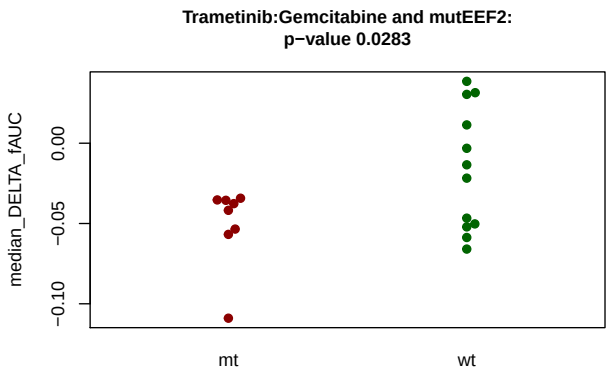
```
L = readRDS(here('Rds','combiResponseAssociations_fAUC_cngene.Rds'))
colnames(L$ave_varExplMatrix) = names(L$features)

for (dataType in colnames(L$ave_varExplMatrix)){
  posDrugs = rownames(L$ave_varExplMatrix)[L$ave_varExplMatrix[, dataType] > 0.1]

  for (drug in posDrugs){
    top = L$features[[dataType]][[drug]][ L$features[[dataType]][[drug]] > 20 ]
    if (length(top)==0){ next }
    if (length(top) == 1){ top = L$features[[dataType]][[drug]][1:2]  }

    # dotplot for these three features
    dplot(drug, names(top[1:min(4,length(top))]), dataType,
          L$ave_varExplMatrix[drug, dataType],
          median_DELTA_fAUC, "median_DELTA_fAUC")
  }
}
```

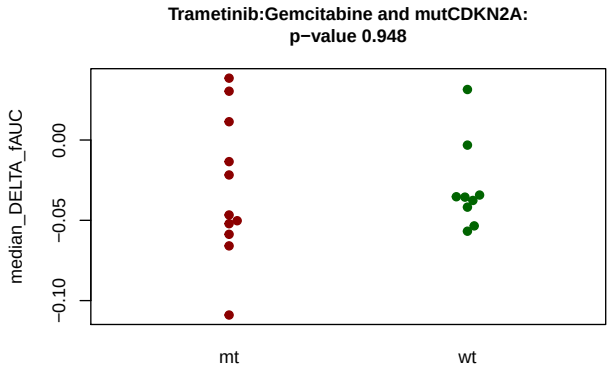

MK−2206:Gemcitabine and mutCDKN2A:
p−value 0.00325
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

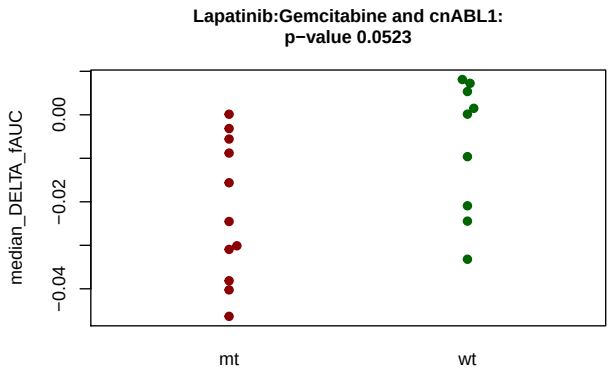

MK−2206:Gemcitabine and mutEEF2:
p−value 0.000321
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

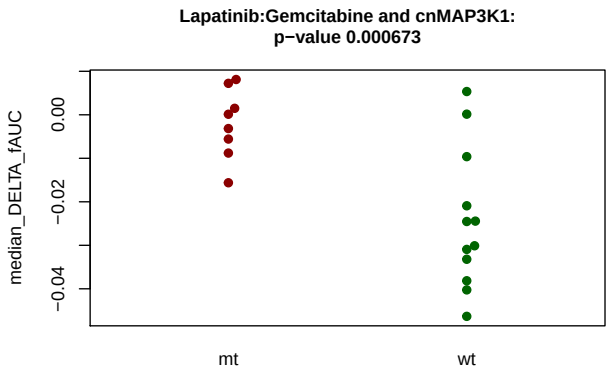

MK−2206:Gemcitabine and mutSMAD4:
p−value 0.51
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

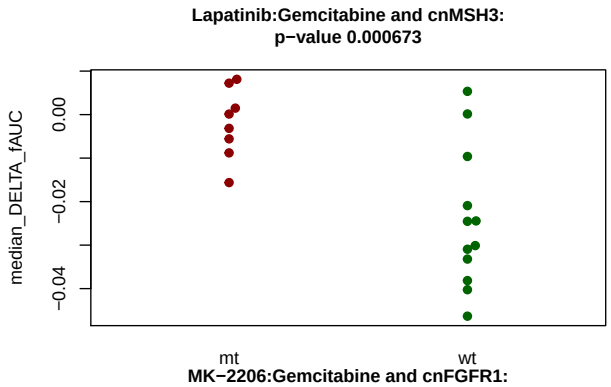

Trametinib:Gemcitabine and mutTGFBR2:
p−value 0.0217
median_DELTA_fAUC
0.00
−0.05
−0.10
mt
wt

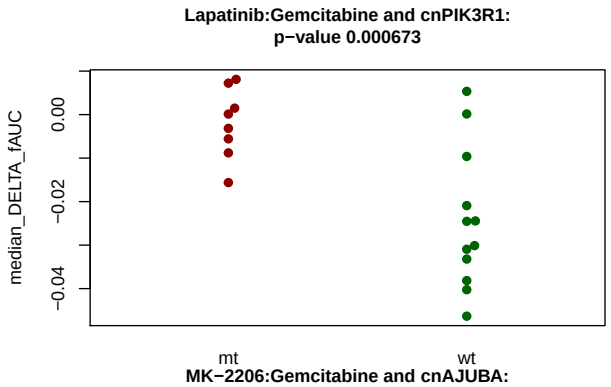

Trametinib:Gemcitabine and mutEEF2:
p−value 0.0283
median_DELTA_fAUC
0.00
−0.05
−0.10
mt
wt

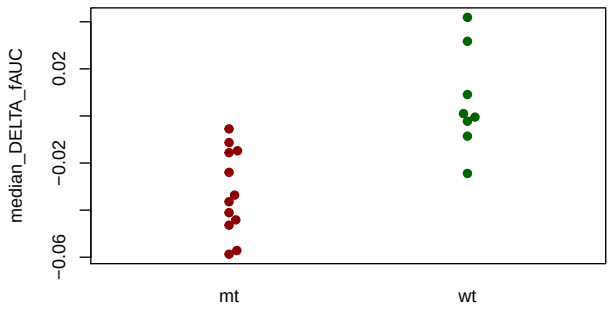

Trametinib:Gemcitabine and mutCDKN2A:
p−value 0.948
median_DELTA_fAUC
0.00
−0.05
−0.10
mt
wt

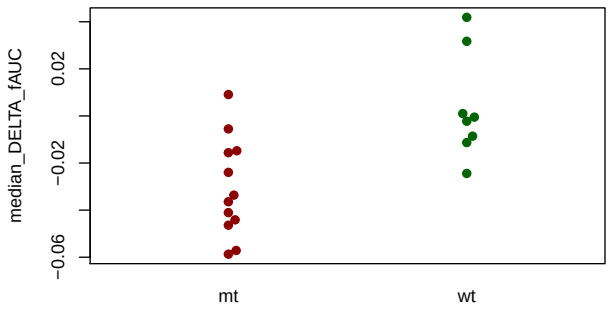

Lapatinib:Gemcitabine and cnABL1:
p−value 0.0523
median_DELTA_fAUC
0.00
−0.02
−0.04
mt
wt

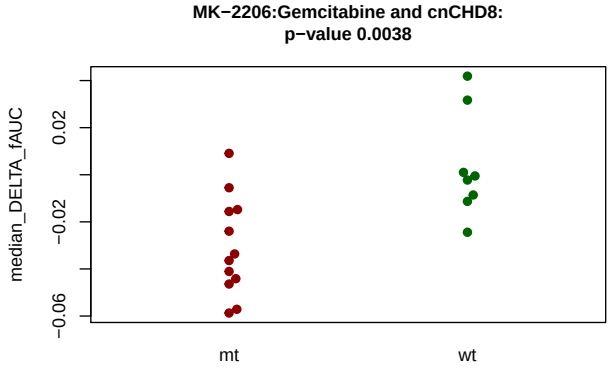

Lapatinib:Gemcitabine and cnMAP3K1:
p−value 0.000673
median_DELTA_fAUC
0.00
−0.02
−0.04
mt
wt

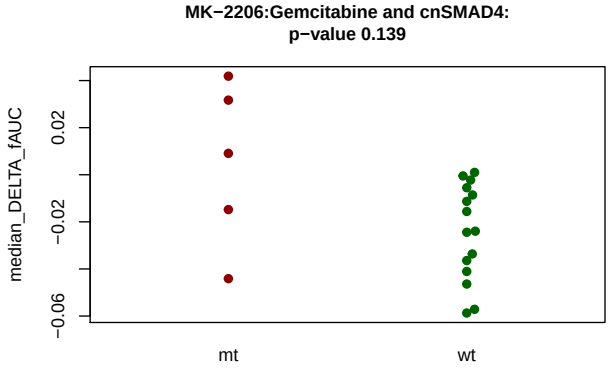

Lapatinib:Gemcitabine and cnMSH3:
p−value 0.000673
median_DELTA_fAUC
0.00
−0.02
−0.04
mt
wt

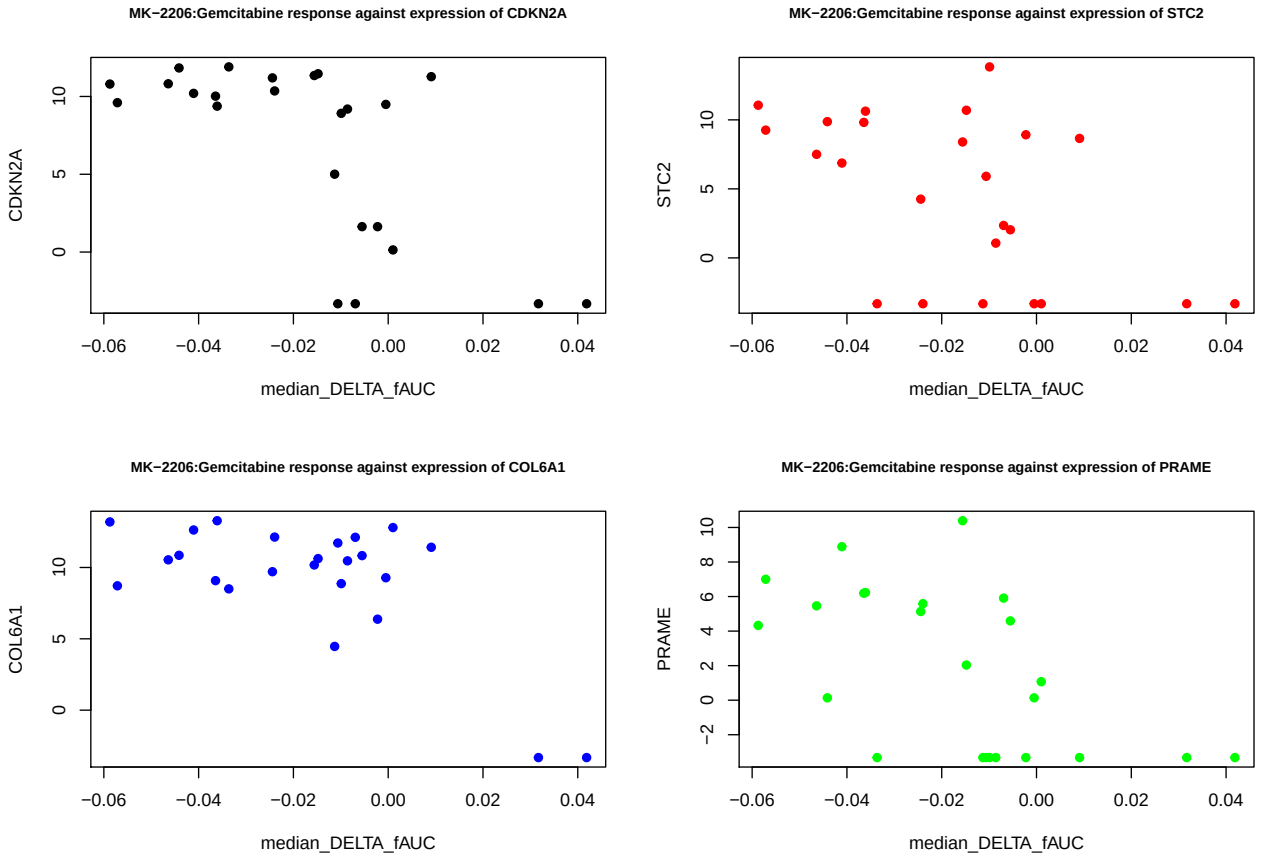

Lapatinib:Gemcitabine and cnPIK3R1:
p−value 0.000673
median_DELTA_fAUC
0.00
−0.02
−0.04
mt
wt


MK−2206:Gemcitabine and cnFGFR1:
p−value 0.00106
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt


MK−2206:Gemcitabine and cnAJUBA:
p−value 0.0038
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

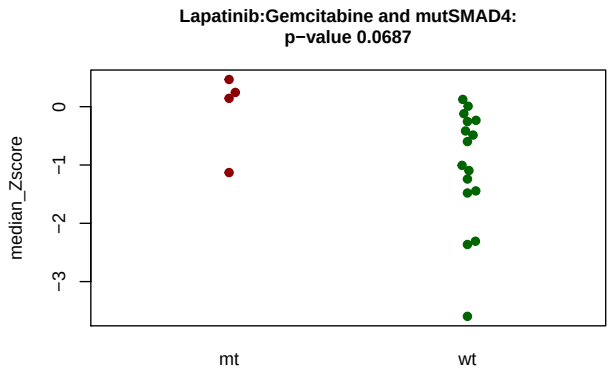

MK−2206:Gemcitabine and cnCHD8:
p−value 0.0038
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

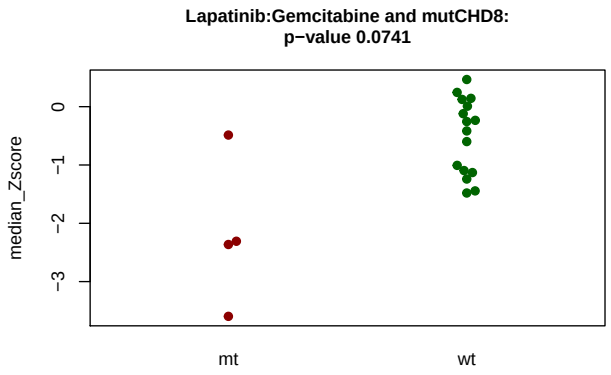

MK−2206:Gemcitabine and cnSMAD4:
p−value 0.139
median_DELTA_fAUC
0.02
−0.02
−0.06
mt
wt

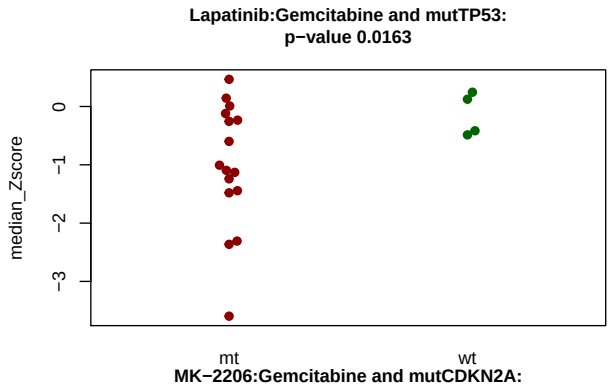


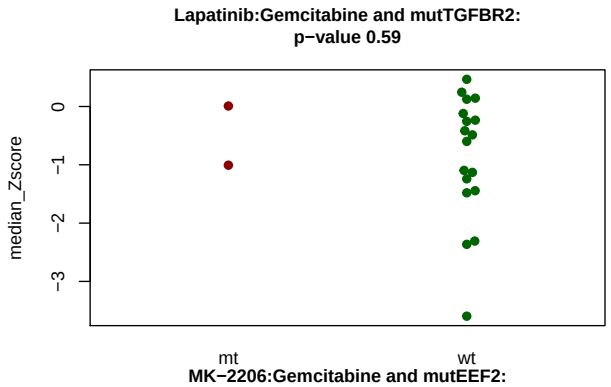
Z score

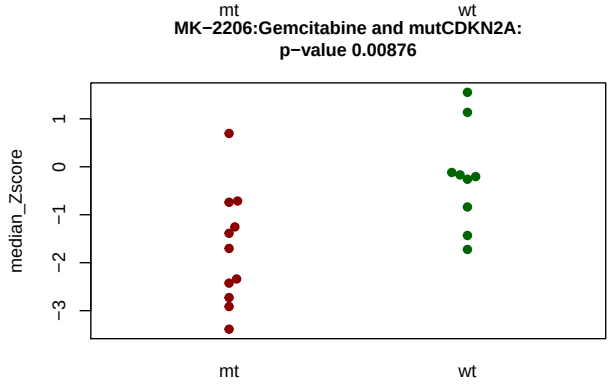
```
L = readRDS(here('Rds','combiResponseAssociations_Zscore_cngene.Rds'))
colnames(L$ave_varExplMatrix) = names(L$features)

for (dataType in colnames(L$ave_varExplMatrix)){
  posDrugs = rownames(L$ave_varExplMatrix)[L$ave_varExplMatrix[, dataType] > 0.1]

  #print(c(dataType, posDrugs))
  for (drug in posDrugs){

    top = L$features[[dataType]][[drug]][ L$features[[dataType]][[drug]] > 20 ]
    if (length(top)==0){ next }
    if (length(top) == 1){ top = L$features[[dataType]][[drug]][1:2]  }

    # dotplot for these three features
    dplot(drug, names(top[1:min(4,length(top))]), dataType,
          L$ave_varExplMatrix[drug, dataType], median_ZSCORE, "median_Zscore")
  }
}
```

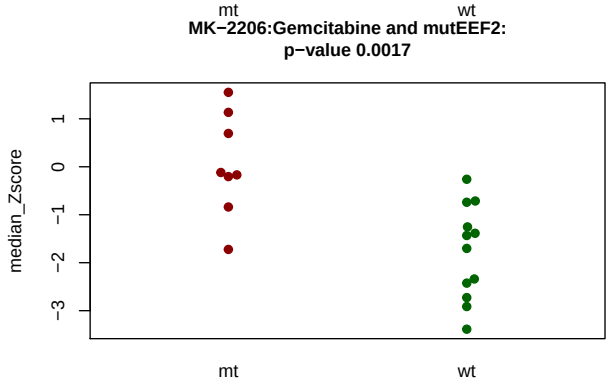

Lapatinib:Gemcitabine and mutSMAD4:
p−value 0.0687
median_Zscore
0
−1
−2
−3
mt
wt

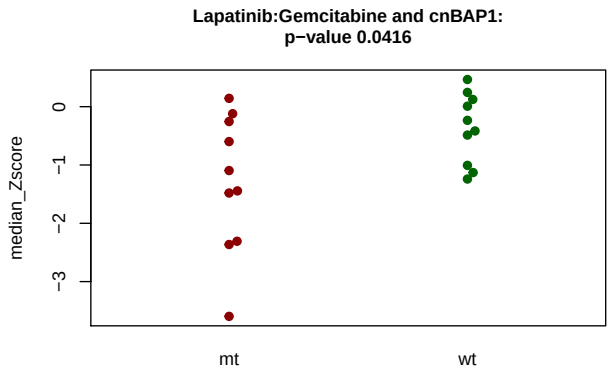

Lapatinib:Gemcitabine and mutCHD8:
p−value 0.0741
median_Zscore
0
−1
−2
−3
mt
wt

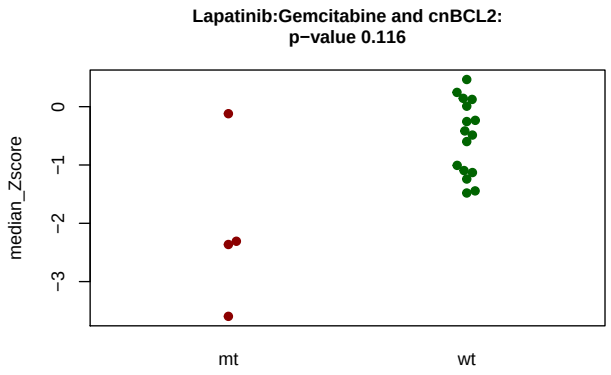

Lapatinib:Gemcitabine and mutTP53:
p−value 0.0163
median_Zscore
0
−1
−2
−3
mt
wt

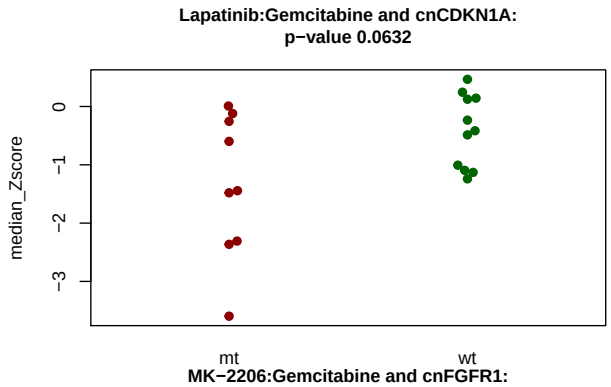

Lapatinib:Gemcitabine and mutTGFBR2:
p−value 0.59
median_Zscore
0
−1
−2
−3
mt
wt

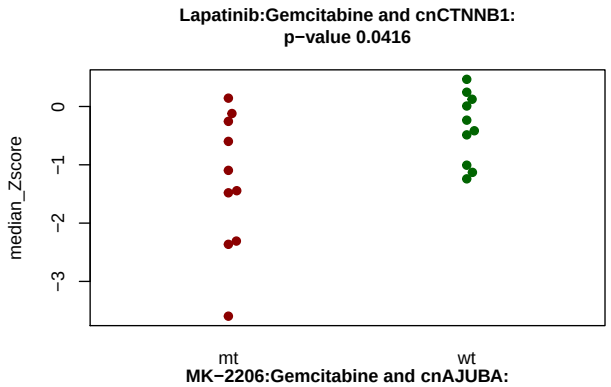

MK−2206:Gemcitabine and mutCDKN2A:
p−value 0.00876
median_Zscore
1
0
−1
−2
−3
mt
wt

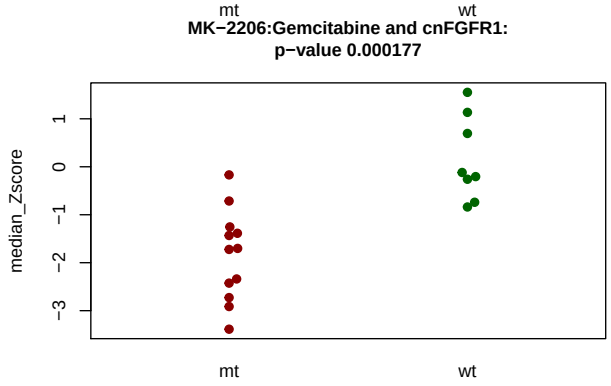

MK−2206:Gemcitabine and mutEEF2:
p−value 0.0017
median_Zscore
1
0
−1
−2
−3
mt
wt

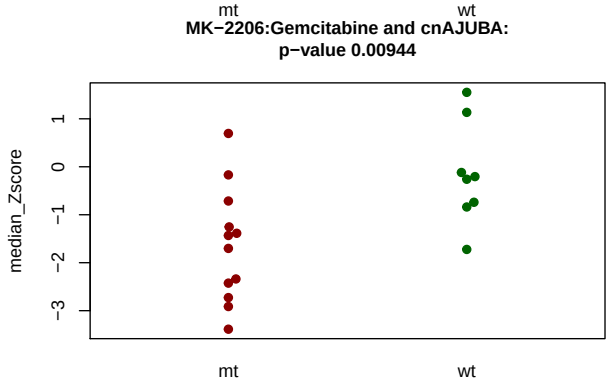

Lapatinib:Gemcitabine and cnBAP1:
p−value 0.0416
median_Zscore
0
−1
−2
−3
mt
wt

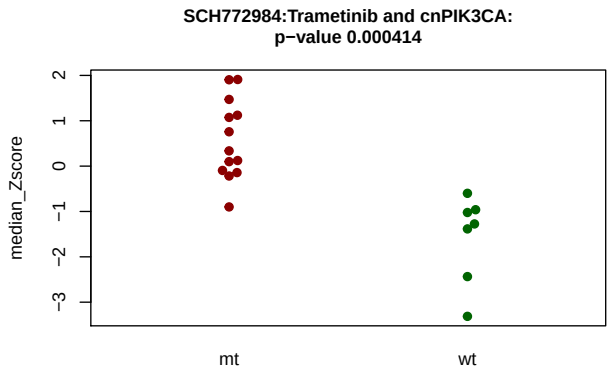

Lapatinib:Gemcitabine and cnBCL2:
p−value 0.116
median_Zscore
0
−1
−2
−3
mt
wt

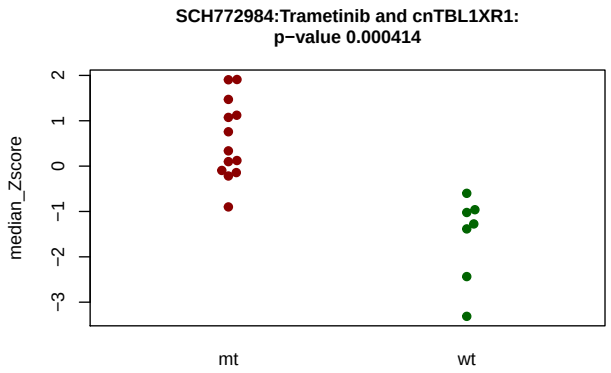

Lapatinib:Gemcitabine and cnCDKN1A:
p−value 0.0632
median_Zscore
0
−1
−2
−3
mt
wt

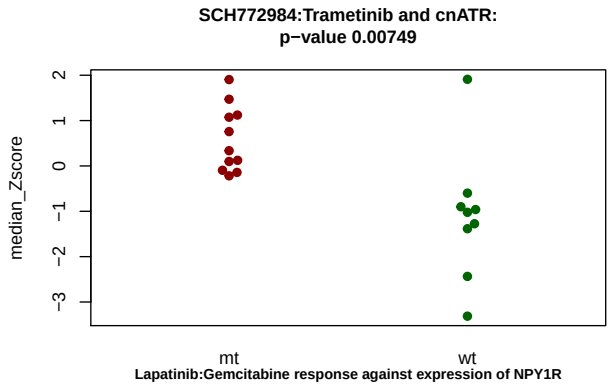

Lapatinib:Gemcitabine and cnCTNNB1:
p−value 0.0416
median_Zscore
0
−1
−2
−3
mt
wt

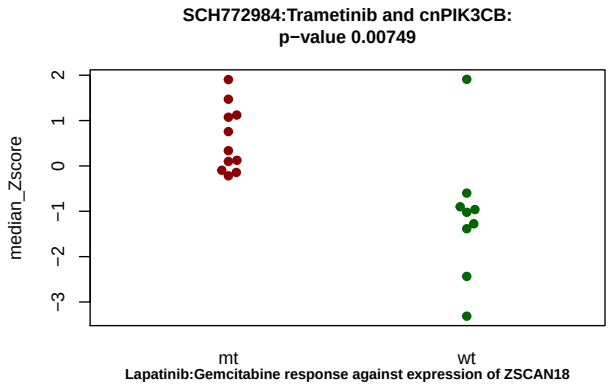

MK−2206:Gemcitabine and cnFGFR1:
p−value 0.000177
median_Zscore
1
0
−1
−2
−3
mt
wt

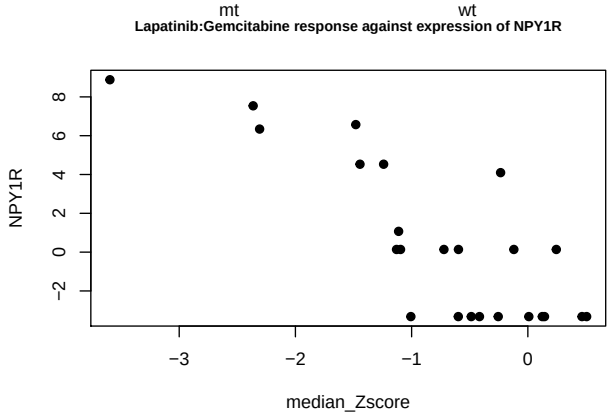

MK−2206:Gemcitabine and cnAJUBA:
p−value 0.00944
median_Zscore
1
0
−1
−2
−3
mt
wt

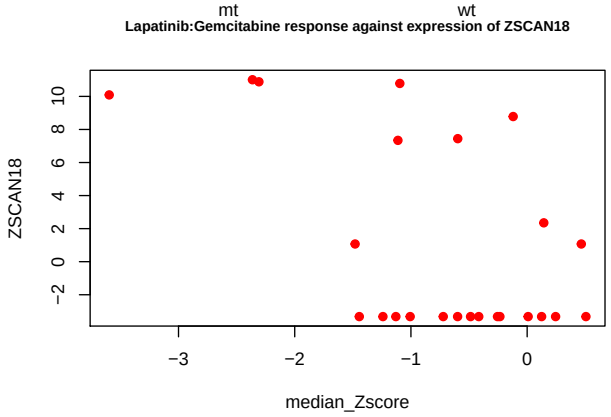

SCH772984:Trametinib and cnPIK3CA:
p−value 0.000414
median_Zscore
mt
wt

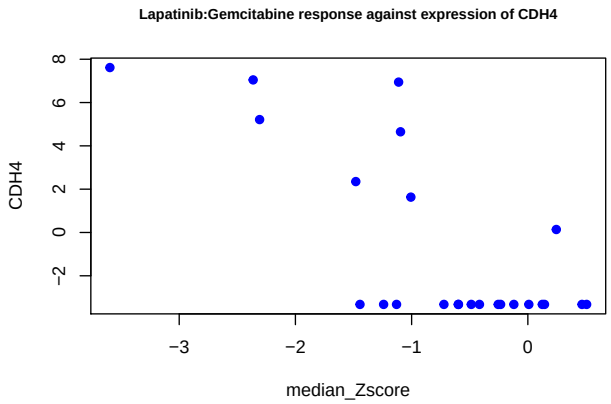

SCH772984:Trametinib and cnTBL1XR1:
p−value 0.000414
median_Zscore
mt
wt

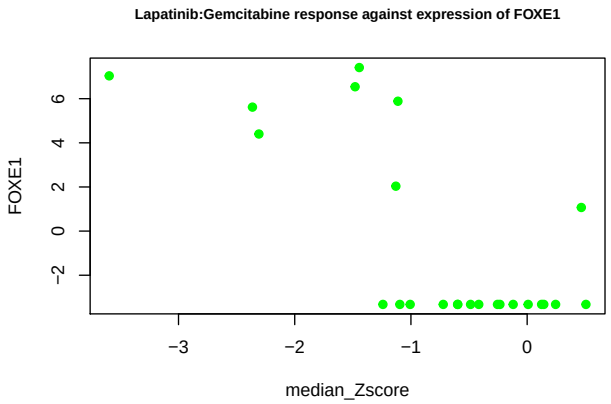

SCH772984:Trametinib and cnATR:
p−value 0.00749
median_Zscore
mt
wt


SCH772984:Trametinib and cnPIK3CB:
p−value 0.00749
median_Zscore
mt
wt


Lapatinib:Gemcitabine response against expression of NPY1R
NPY1R
median_Zscore


Lapatinib:Gemcitabine response against expression of ZSCAN18
ZSCAN18
median_Zscore

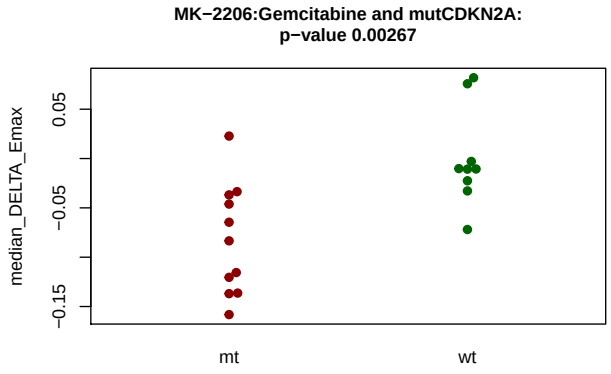

Lapatinib:Gemcitabine response against expression of CDH4
CDH4
median_Zscore

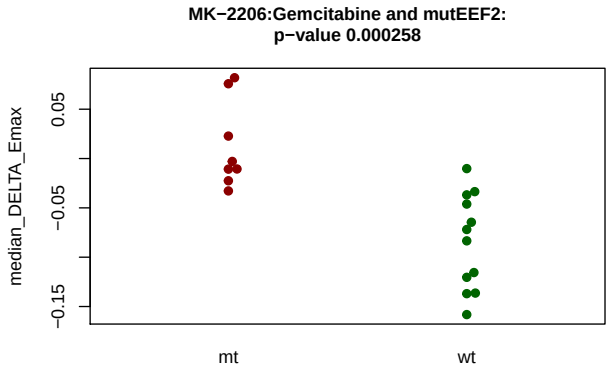

Lapatinib:Gemcitabine response against expression of FOXE1
FOXE1
median_Zscore

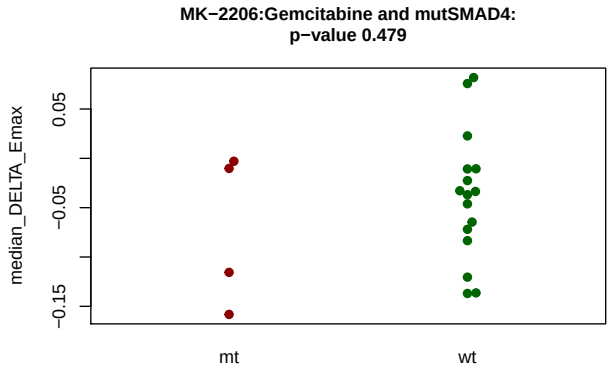
Delta Emax

```
L = readRDS(here('Rds','combiResponseAssociations_Emax_cngene.Rds'))
colnames(L$ave_varExplMatrix) = names(L$features)

for (dataType in colnames(L$ave_varExplMatrix)){
  posDrugs = rownames(L$ave_varExplMatrix)[L$ave_varExplMatrix[, dataType] > 0.1]

  #print(c(dataType, posDrugs))
  for (drug in posDrugs){

    top = L$features[[dataType]][[drug]][ L$features[[dataType]][[drug]] > 20 ]
    if (length(top)==0){ next }
    print(top)

    # dotplot for these three features
    dplot(drug, names(top[1:min(4,length(top))]), dataType,
          L$ave_varExplMatrix[drug, dataType],
          median_DELTA_EMAX, "median_DELTA_Emax")
  }
}
```

```
##
## mutCDKN2A   mutEEF2  mutSMAD4
##        78        59        59
```

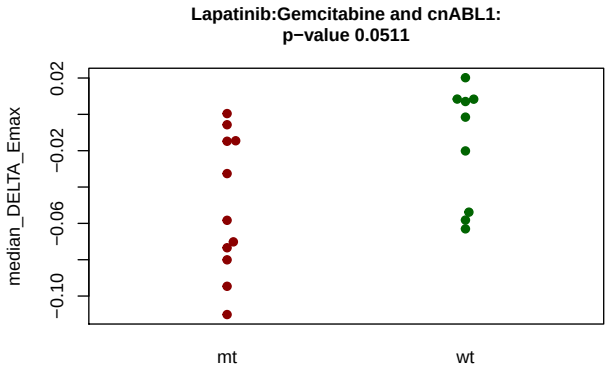


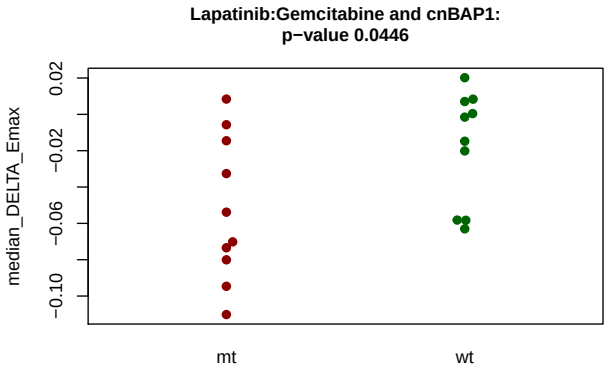


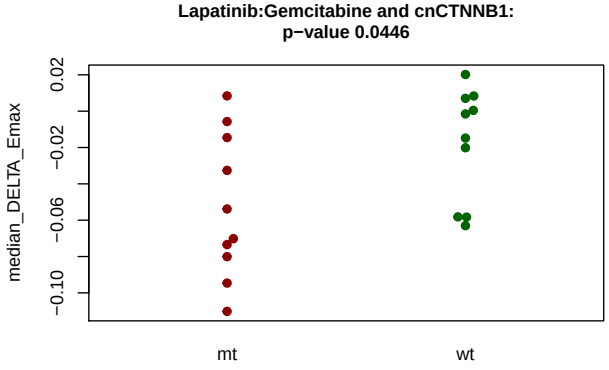


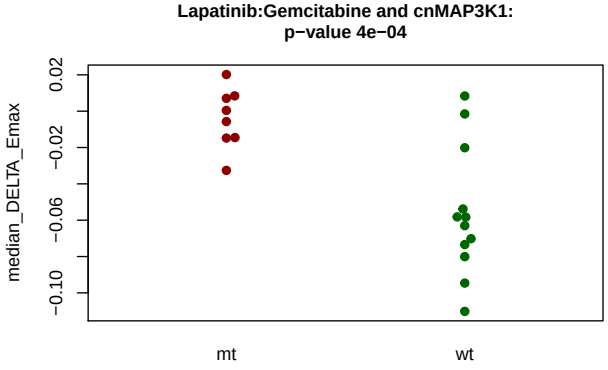

```
##
##    cnABL1    cnBAP1   cnCTNNB1   cnMAP3K1    cnMLH1    cnMSH3   cnPIK3R1    cnPTCH1
```

```
##        99        99        99        99        99        99        99        99
##    cnRAF1    cnRHOA   cnSETD2   cnSMAD4    cnTLR4    cnTSC1     cnVHL    cnEGR3
##        99        99        99        99        99        99        99        98
##   cnFUBP1   cnMACF1  cnCDKN2C   cnEPHA2    cnJAK1    cnMTOR   cnRPL22  cnTHRAP3
##        97        97        96        96        96        96        96        96
## cnZC3H12A     cnAPC   cnRASA1  cnMAP2K4    cnTP53    cnPAX5  cnCDKN1A   cnHLA-A
##        96        87        87        72        72        61        60        60
##   cnHLA-B   cnLEMD2    cnGNAS  cnTGFBR2   cnBRD7    cnCBFB  cnCREBBP    cnCTCF
##        60        60        38        26        24        24        24        24
```

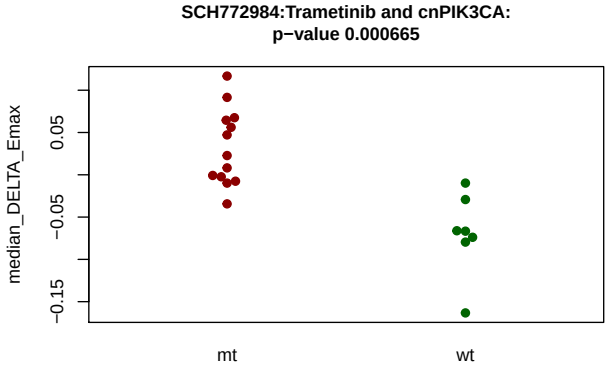


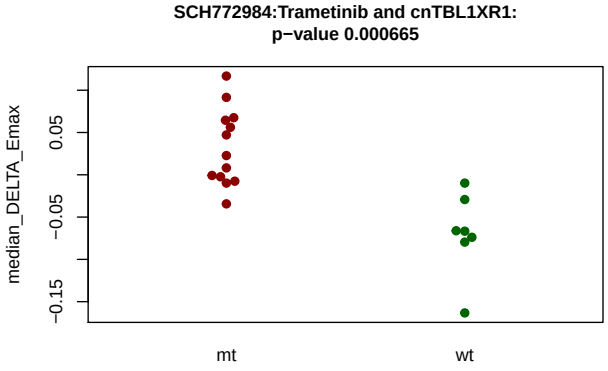




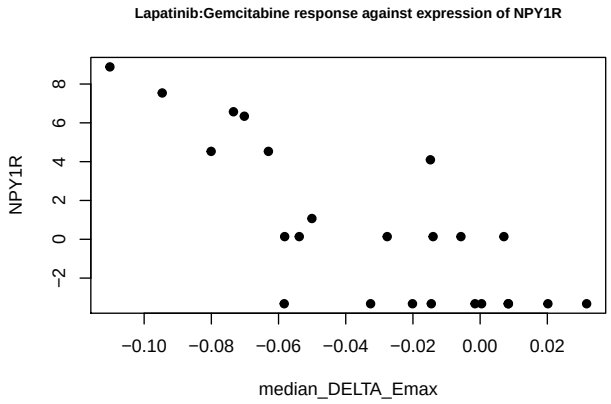


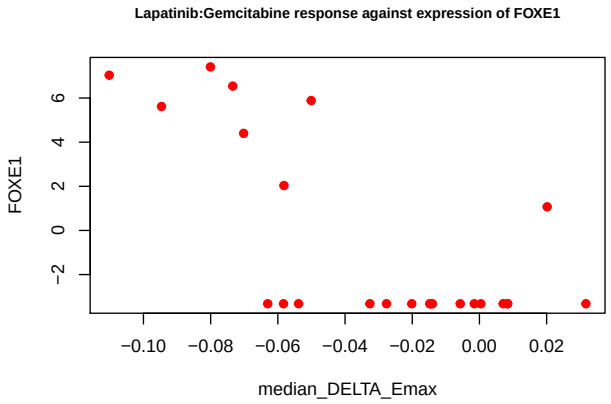

```
##
##  cnPIK3CA cnTBL1XR1
##        26        26
```

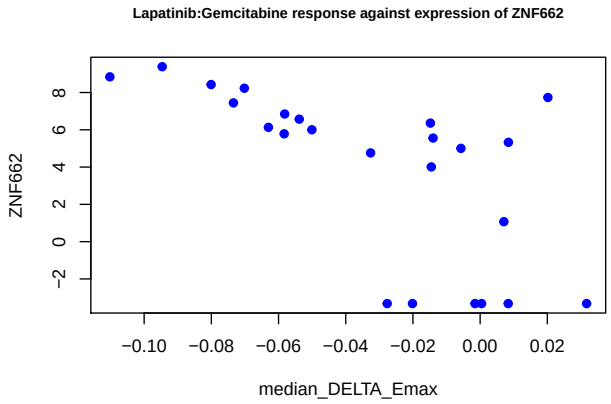


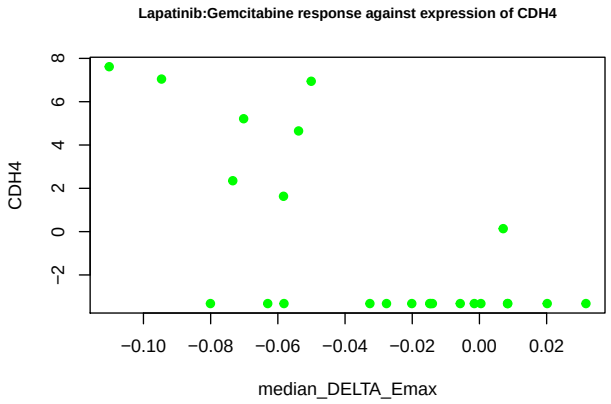



```
##
##  NPY1R  FOXE1 ZNF662   CDH4
##     57     41     38     21
```









Integrating all molecular data with TANDEM

The idea behind Tandem is to utilize a two stage approach that first fits an elastic net to predict drug response based on “upstream” (copy number, mutations) data, then fit a second elastic net to use 'downstream' (gene expression) to predict the residuals in the first model. This results in a model which is easier to interpret and less reliant on gene expression data.

Note: Already in the “conventional” elastic net, gene expression (RNAseq) data contributed far, far less to the model than is expected/is normal.

Tandem models

Single response (IC50)

```
# install.packages('TANDEM')
#library(TANDEM)
# x - molecular data "cbound" and transposed (for samples by features)
# y - response variable, 1x(num samples)
# upstream - logical 1x(num features)
whichSamples = colnames(mutMatrix)
whichSamples = intersect(whichSamples, colnames(cnMatrix))
#whichSamples = tumorHubID[whichSamples]

# save for mut and cn matrices
```

```
#whichSamplesPD = names(whichSamples)[whichSamples %in% intersect(colnames(geneEx),
#rownames(AUC))] #names(whichsSamples) returns NULL

whichSamplesPD = whichSamples[whichSamples %in% intersect(colnames(geneEx),
                                                          rownames(AUC))]
whichSamples = intersect(whichSamples, colnames(geneEx))
whichSamples = intersect(whichSamples, rownames(AUC))

gEx = scale(t( geneEx[, whichSamples]))
#NaN is returned by scale if all rows in geneEx were 0 except for one sample
gEx = gEx[,colSums(is.na(gEx)) == 0]

x = cbind(t(mutMatrix[, whichSamplesPD]), t(cnMatrix[, whichSamplesPD]), gEx)

#Since we removed some columns from gEx, we need to ensure that
#the number of features in x and upstream match
#upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, nrow(geneEx)))
upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, ncol(gEx)))

dataTypes = rep('downstream', length(upstream))
dataTypes[upstream] = 'upstream'

#y = as.numeric(scale(IC50[whichSamples,1]))

singleResponseStats = list()
start = Sys.time()
# loop over different "y", IC50s, delta_AUC, delta_fAUC, Zscore, delta_Emax
# Misbehaves when encountering NAs
for (drug in colnames(IC50)[colSums(is.na(IC50))==0]){

  #print(drug)

  #We already have the drug names as col names
  #singleResponseStats[[singleDrugInfo[drug,'DRUG_NAME']]] = list()
  singleResponseStats[[drug]] = list()

  y = as.numeric( scale(IC50[whichSamples, drug]) ) #Causes crash with NAs

  fit = tandem(x, y, upstream, alpha=0.5)
  beta = coef(fit)
  singleResponseStats[[drug]]$betas = beta[beta[,1]!=0,]

  contr_tandem = relative.contributions(fit, x, dataTypes)
  singleResponseStats[[drug]]$contr = contr_tandem

  out = nested.cv(x, y, upstream, method='tandem', alpha=0.5)

  singleResponseStats[[drug]]$mse = out$mse
  singleResponseStats[[drug]]$y = y
  singleResponseStats[[drug]]$y_hat = out$y_hat
  singleResponseStats[[drug]]$cor = cor(y, out$y_hat)
}
end = Sys.time()
```

```
#end-start
saveRDS(singleResponseStats, here("Rds", "singleResponseStats_cngene.Rds"))
```

```
L = readRDS(here("Rds", "singleResponseStats_cngene.Rds"))

for (drug in names(L)){
  if (length(L[[drug]]$betas) > 1 & L[[drug]]$cor > 0.1){
    print(drug)
    print(L[[drug]]$betas)
    print(L[[drug]]$contr)
    print(L[[drug]]$cor)
  }
}
```

```
## [1] "PF-4708671"
##  (Intercept)      mutABL1    mutACVR2A    mutARID1A       mutATR      mutATRX
##  0.370206872 -0.019687301 -0.019798717  0.081839851 -0.018739350  0.029472756
##       mutB2M      mutBCOR      mutBRAF     mutBRCA1       mutCIC      mutEGFR
## -0.018058591 -0.008713462 -0.016969668 -0.016654570 -0.253848094 -0.076634112
##     mutFBXW7     mutH3F3A      mutIDH1      mutJAK1      mutJAK3    mutKANSL1
## -0.018501260 -0.018651581 -0.019104184 -0.019864849 -0.020812017 -0.021737161
##     mutKMT2C     mutMACF1    mutMAP2K1       mutMGA      mutMSH3       mutNF1
## -0.075824188 -0.255501076 -0.021851815 -0.022436638 -0.022369713 -0.021951368
##      mutPAX5     mutPTCH1    mutRNF111     mutRPL22     mutSCAF4       cnABL1
## -0.021297514 -0.020511591 -0.019677459 -0.018850972  0.008438927 -0.041934744
##      cnASXL1     cnATF7IP       cnBAP1     cnCDKN1B     cnCDKN2A       cnCHD4
##  0.017896589  0.049154665 -0.017297277  0.049629184  0.811205854  0.049793153
##     cnCTNNB1      cnEPHA3       cnKRAS     cnMAP3K1       cnMLH1       cnMSH3
## -0.016289688 -0.190623108  0.049700099  0.005466250 -0.016338364  0.005722623
##     cnNOTCH1      cnPBRM1     cnPIK3CA     cnPIK3R1      cnPTCH1       cnRAF1
## -0.167326783 -0.186437498 -0.209640088  0.006747012 -0.042306020 -0.015933664
##       cnRHOA      cnSETD2    cnTBL1XR1       cnTLR4       cnTSC1        cnVHL
## -0.015901739 -0.016714892 -0.211222834 -0.042257804 -0.041969562 -0.018060983
##   upstream downstream
##          1          0
## [1] 0.405296
## [1] "Wee1 Inhibitor"
##  (Intercept)       mutAPC      mutCDH1    mutCTNNB1     mutKMT2A    mutPIK3CA
##  0.487177832  0.042269953 -0.068142497  0.399661737  0.042088473  0.096053699
##       mutRB1     mutRNF43     mutSMAD4      mutTP53     cnARID1A        cnB2M
##  0.163494334  0.081861294 -0.038147403 -0.200361980  0.144181698 -0.054313561
##       cnCDH1      cnCHEK2      cnEP300       cnIDH2      cnKEAP1     cnMAP2K1
## -0.174568724  0.084562838  0.086030848 -0.053891316 -0.037600069 -0.053818905
##      cnMAPK1        cnMGA        cnNF2     cnPLXNB2     cnRNF111      cnSIN3A
##  0.004286885 -0.054038940  0.084446370  0.021818486 -0.053913541 -0.053868805
##    cnSMARCA4      cnTCF12
## -0.038221617 -0.053907633
##   upstream downstream
##          1          0
## [1] 0.3355781
```

Delta AUC

```
stopifnot(colSums(is.na(median_DELTA_AUC))==0)
```

```
# install.packages('TANDEM')
library(TANDEM)
# x - molecular data "cbound" and transposed (for samples by features)
# y - response variable, 1x(num samples)
# upstream - logical 1x(num features)
whichSamples = colnames(mutMatrix)
whichSamples = intersect(whichSamples, colnames(cnMatrix))
#whichSamples = tumorHubID[whichSamples]

# save for mut and cn matrices
whichSamplesPD = whichSamples[whichSamples %in% intersect(colnames(geneEx),
                                                          rownames(AUC))]
whichSamples = intersect(whichSamples, colnames(geneEx))
whichSamples = intersect(whichSamples, rownames(AUC))

gEx = scale(t( geneEx[, whichSamples]) )
#NaN is returned by scale if all rows in geneEx were 0 except for one sample
gEx = gEx[,colSums(is.na(gEx)) == 0]

x = cbind(t(mutMatrix[, whichSamplesPD]), t(cnMatrix[, whichSamplesPD]), gEx)

upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, ncol(gEx)))

dataTypes = rep('downstream', length(upstream))
dataTypes[upstream] = 'upstream'

#y = as.numeric(scale(IC50[whichSamples,1]))
combiResponseStats = list()

start = Sys.time()

# loop over different "y", IC50s, delta_AUC, delta_fAUC, Zscore, delta_Emax
for (drug in colnames(median_DELTA_AUC)){

  #combiResponseStats[[names(combinations)[combinations==drug]]] = list()
  combiResponseStats[[drug]] = list()

  y = as.numeric( scale(median_DELTA_AUC[whichSamples, drug]) )

  fit = tandem(x, y, upstream, alpha=0.5)
  beta = coef(fit)
  combiResponseStats[[drug]]$betas = beta[beta[,1]!=0,]

  contr_tandem = relative.contributions(fit, x, dataTypes)
  combiResponseStats[[drug]]$contr = contr_tandem

  out = nested.cv(x, y, upstream, method='tandem', alpha=0.5)

  combiResponseStats[[drug]]$mse = out$mse
```

```
  combiResponseStats[[drug]]$y = y
  combiResponseStats[[drug]]$y_hat = out$y_hat
  combiResponseStats[[drug]]$cor = cor(y, out$y_hat)
}

end = Sys.time()
#end - start
for (drug in names(combiResponseStats)){
  if (length(combiResponseStats[[drug]]$betas) > 1 &
      combiResponseStats[[drug]]$cor > 0.1){
    print(drug)
    print(combiResponseStats[[drug]]$betas)
    print(combiResponseStats[[drug]]$contr)
    print(combiResponseStats[[drug]]$cor)
  }
}

combiResponseStats_AUC = combiResponseStats
saveRDS(combiResponseStats_AUC,
        file = here("Rds", "combiResponseStats_AUC_cngene.Rds"))
```

```
L = readRDS(here("Rds", "combiResponseStats_AUC_cngene.Rds"))

for (drug in names(L)){
  if (length(L[[drug]]$betas) > 1 & L[[drug]]$cor > 0.1){
    print(drug)
    print(L[[drug]]$betas)
    print(L[[drug]]$contr)
    print(L[[drug]]$cor)
  }
}
```

No items are printed here because are no drugs for which the correlation between the expected and predicted AUC was greater than one.

Delta fAUC

```
stopifnot(colSums(is.na(median_DELTA_fAUC))==0)
```

```
# install.packages('TANDEM')
library(TANDEM)
# x - molecular data "cbound" and transposed (for samples by features)
# y - response variable, 1x(num samples)
# upstream - logical 1x(num features)
whichSamples = colnames(mutMatrix)
whichSamples = intersect(whichSamples, colnames(cnMatrix))

whichSamplesPD = whichSamples[whichSamples %in% intersect(colnames(geneEx),
                                                          rownames(AUC))]
whichSamples = intersect(whichSamples, colnames(geneEx))
whichSamples = intersect(whichSamples, rownames(AUC))

gEx = scale(t( geneEx[, whichSamples]) )
```

```
#NaN is returned by scale if all rows in geneEx were 0 except for one sample
gEx = gEx[,colSums(is.na(gEx)) == 0]

x = cbind(t(mutMatrix[, whichSamplesPD]), t(cnMatrix[, whichSamplesPD]), gEx)

upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, ncol(gEx)))

dataTypes = rep('downstream', length(upstream))
dataTypes[upstream] = 'upstream'

#y = as.numeric(scale(IC50[whichSamples,1]))
combiResponseStats = list()

start = Sys.time()

# loop over different "y", IC50s, delta_AUC, delta_fAUC, Zscore, delta_Emax
for (drug in colnames(median_DELTA_fAUC)){

  combiResponseStats[[drug]] = list()

  y = as.numeric( scale(median_DELTA_fAUC[whichSamples, drug]) )

  fit = tandem(x, y, upstream, alpha=0.5)
  beta = coef(fit)
  combiResponseStats[[drug]]$betas = beta[beta[,1]!=0,]

  contr_tandem = relative.contributions(fit, x, dataTypes)
  combiResponseStats[[drug]]$contr = contr_tandem

  out = nested.cv(x, y, upstream, method='tandem', alpha=0.5)

  combiResponseStats[[drug]]$mse = out$mse
  combiResponseStats[[drug]]$y = y
  combiResponseStats[[drug]]$y_hat = out$y_hat
  combiResponseStats[[drug]]$cor = cor(y, out$y_hat)
}

end = Sys.time()
#end - start

for (drug in names(combiResponseStats)){
  if (length(combiResponseStats[[drug]]$betas) > 1 &
      combiResponseStats[[drug]]$cor > 0.1){
    print(drug)
    print(combiResponseStats[[drug]]$betas)
    print(combiResponseStats[[drug]]$contr)
    print(combiResponseStats[[drug]]$cor)
  }
}

combiResponseStats_fAUC = combiResponseStats
saveRDS(combiResponseStats_fAUC, file = here("Rds", "combiResponseStats_fAUC_cngene.Rds"))
```

```
L = readRDS(here("Rds", "combiResponseStats_fAUC_cngene.Rds"))

for (drug in names(L)){
  if (length(L[[drug]]$betas) > 1 & L[[drug]]$cor > 0.1){
    print(drug)
    print(L[[drug]]$betas)
    print(L[[drug]]$contr)
    print(L[[drug]]$cor)
  }
}
```

```
## [1] "Lapatinib:Gemcitabine"
## (Intercept)    mutASXL2      mutCIC   mutCTNNB1    mutMACF1     mutTP53
## -0.03636523  0.02708472  0.21059975  0.18048225  0.21006150 -0.17095320
##      cnABL1       cnAPC   cnCACNA1A    cnDAZAP1      cnEEF2      cnEGR3
## -0.08348866  0.01971670  0.06927173  0.07039121  0.05441856  0.02605503
##     cnGNA11      cnGNAS      cnJAK3    cnMAP3K1      cnMSH3    cnPIK3R1
##  0.06935785 -0.01662017  0.06984047  0.27826369  0.27650190  0.27331224
##    cnPIK3R2     cnPTCH1     cnRASA1     cnSMAD4     cnSTK11      cnTLR4
##  0.06948500 -0.08333414  0.01919472 -0.15275024  0.06938174 -0.08288432
##      cnTSC1
## -0.08214372
##   upstream downstream
##          1          0
## [1] 0.7915868
## [1] "MK-2206:Gemcitabine"
##   (Intercept)       mutCDH1     mutCDKN2A       mutEEF2       cnAJUBA
##  0.4590892816 -0.1695173182 -0.1447458170  0.0029201771 -0.0631613891
##      cnCDKN2A        cnCHD8       cnFGFR1       cnSMAD4          ANO4
##  0.0209279404 -0.0623766094 -0.4358003709  0.2926964526 -0.0028907006
##       CLSTN2       CSNK2A1      CYB561A3     DPP10-AS1         EFNA3
##  0.0062195624 -0.0009241835  0.0175794943  0.0105344948 -0.0042969913
##         FAT3          GAB3        KLHL32          MAEA          MDP1
##  0.0034643275  0.0182526198  0.0690713676  0.0020702818  0.0097222309
##       MIR3145          PAOX         RBCK1          ROS1      RPL7AP34
##  0.0329398801  0.0006935296 -0.0102550557  0.0078023050  0.0065863040
##       SEPT7P3       SLC39A1       SNORA34      SNORA80B         SUSD5
##  0.0098185129  0.0439925845  0.0021780158  0.0081749377  0.0187708171
##          SYT7       TMEM129          TTPA        ZNF277        ZSWIM3
## -0.0099177823  0.0200095606  0.0113084344 -0.0360282022  0.0006297397
##   upstream downstream
##  0.6551494  0.3448506
## [1] 0.3769489
```

Z-score

```
stopifnot(colSums(is.na(median_ZSCORE)) == 0)
```

```
# install.packages('TANDEM')
library(TANDEM)
# x - molecular data "cbound" and transposed (for samples by features)
# y - response variable, 1x(num samples)
# upstream - logical 1x(num features)
```

```
whichSamples = colnames(mutMatrix)
whichSamples = intersect(whichSamples, colnames(cnMatrix))


whichSamplesPD = whichSamples[whichSamples %in% intersect(colnames(geneEx),
                                                          rownames(AUC))]
whichSamples = intersect(whichSamples, colnames(geneEx))
whichSamples = intersect(whichSamples, rownames(AUC))

gEx = scale(t( geneEx[, whichSamples]) )
#NaN is returned by scale if all rows in geneEx were 0 except for one sample
gEx = gEx[,colSums(is.na(gEx)) == 0]

x = cbind(t(mutMatrix[, whichSamplesPD]), t(cnMatrix[, whichSamplesPD]), gEx)

upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, ncol(gEx)))

dataTypes = rep('downstream', length(upstream))
dataTypes[upstream] = 'upstream'

combiResponseStats = list()

# loop over different "y", IC50s, delta_AUC, delta_fAUC, Zscore, delta_Emax
start = Sys.time()
for (drug in colnames(median_ZSCORE)){

  combiResponseStats[[drug]] = list()

  y = as.numeric( scale(median_ZSCORE[whichSamples, drug]) )

  fit = tandem(x, y, upstream, alpha=0.5)
  beta = coef(fit)
  combiResponseStats[[drug]]$betas = beta[beta[,1]!=0,]

  contr_tandem = relative.contributions(fit, x, dataTypes)
  combiResponseStats[[drug]]$contr = contr_tandem

  out = nested.cv(x, y, upstream, method='tandem', alpha=0.5)

  combiResponseStats[[drug]]$mse = out$mse
  combiResponseStats[[drug]]$y = y
  combiResponseStats[[drug]]$y_hat = out$y_hat
  combiResponseStats[[drug]]$cor = cor(y, out$y_hat)
}
end = Sys.time()
end-start

for (drug in names(combiResponseStats)){
  if (length(combiResponseStats[[drug]]$betas) > 1 &
      combiResponseStats[[drug]]$cor > 0.1){
    print(drug)
    print(combiResponseStats[[drug]]$betas)
    print(combiResponseStats[[drug]]$contr)
```

```
    print(combiResponseStats[[drug]]$cor)
  }
}

combiResponseStats_Zscore = combiResponseStats
saveRDS(combiResponseStats_Zscore,
        file = here("Rds", "combiResponseStats_Zscore_cngene.Rds"))
```

Delta E-max

```
stopifnot(colSums(is.na(median_DELTA_EMAX)) == 0)
```

```
# install.packages('TANDEM')
library(TANDEM)
# x - molecular data "cbound" and transposed (for samples by features)
# y - response variable, 1x(num samples)
# upstream - logical 1x(num features)
whichSamples = colnames(mutMatrix)
whichSamples = intersect(whichSamples, colnames(cnMatrix))
#whichSamples = tumorHubID[whichSamples]

# save for mut and cn matrices
whichSamplesPD = whichSamples[whichSamples %in% intersect(colnames(geneEx),
                                                          rownames(AUC))]
whichSamples = intersect(whichSamples, colnames(geneEx))
whichSamples = intersect(whichSamples, rownames(AUC))

gEx = scale(t( geneEx[, whichSamples]) )
#NaN is returned by scale if all rows in geneEx were 0 except for one sample
gEx = gEx[,colSums(is.na(gEx)) == 0]

x = cbind(t(mutMatrix[, whichSamplesPD]), t(cnMatrix[, whichSamplesPD]), gEx)

upstream = c(rep(TRUE, nrow(mutMatrix)+nrow(cnMatrix)), rep(FALSE, ncol(gEx)))

dataTypes = rep('downstream', length(upstream))
dataTypes[upstream] = 'upstream'

combiResponseStats = list()

start = Sys.time()

# loop over different "y", IC50s, delta_AUC, delta_fAUC, Zscore, delta_Emax
for (drug in colnames(median_DELTA_EMAX)){

  combiResponseStats[[drug]] = list()

  y = as.numeric( scale(median_DELTA_fAUC[whichSamples, drug]) )

  fit = tandem(x, y, upstream, alpha=0.5)
  beta = coef(fit)
  combiResponseStats[[drug]]$betas = beta[beta[,1]!=0,]
```

```
  contr_tandem = relative.contributions(fit, x, dataTypes)
  combiResponseStats[[drug]]$contr = contr_tandem

  out = nested.cv(x, y, upstream, method='tandem', alpha=0.5)

  combiResponseStats[[drug]]$mse = out$mse
  combiResponseStats[[drug]]$y = y
  combiResponseStats[[drug]]$y_hat = out$y_hat
  combiResponseStats[[drug]]$cor = cor(y, out$y_hat)
}

end = Sys.time()
#end-start

for (drug in names(combiResponseStats)){
  if (length(combiResponseStats[[drug]]$betas) > 1 &
      combiResponseStats[[drug]]$cor > 0.1){
    print(drug)
    print(combiResponseStats[[drug]]$betas)
    print(combiResponseStats[[drug]]$contr)
    print(combiResponseStats[[drug]]$cor)
  }
}

combiResponseStats_Emax = combiResponseStats
saveRDS(combiResponseStats_Emax,
        file = here("Rds", "combiResponseStats_Emax_cngene.Rds"))
```

```
L = readRDS(here("Rds", "combiResponseStats_Emax_cngene.Rds"))

for (drug in names(L)){
  if (length(L[[drug]]$betas) > 1 & L[[drug]]$cor > 0.1){
    print(drug)
    print(L[[drug]]$betas)
    print(L[[drug]]$contr)
    print(L[[drug]]$cor)
  }
}
```

```
## [1] "Lapatinib:Gemcitabine"
##  (Intercept)     mutASXL2       mutCIC    mutCTNNB1     mutMACF1      mutTP53
## -0.040228213  0.024075794  0.205051477  0.179037570  0.206000400 -0.168703629
##       cnABL1        cnAPC    cnCACNA1A     cnDAZAP1       cnEEF2       cnEGR3
## -0.081991065  0.018789023  0.068460056  0.069384643  0.047894843  0.024261222
##      cnGNA11       cnGNAS       cnJAK3     cnMAP3K1       cnMSH3     cnPIK3R1
##  0.069528125 -0.006916468  0.070448574  0.272005723  0.271419631  0.270059366
##     cnPIK3R2      cnPTCH1      cnRASA1      cnSMAD4      cnSTK11       cnTLR4
##  0.069379847 -0.081883229  0.018278824 -0.149939350  0.069068376 -0.082153811
##       cnTSC1
## -0.081426651
##   upstream downstream
##          1          0
## [1] 0.7325596
## [1] "MK-2206:Gemcitabine"
```

```
## (Intercept)   mutCDKN2A     cnAJUBA      cnCHD8     cnFGFR1     cnSMAD4
##  0.33212876 -0.08567391 -0.02977284 -0.02910310 -0.33776429  0.06892586
##   upstream downstream
##          1          0
## [1] 0.2065276
```

Summary TANDEM results

The TANDEM model attempts to infer the relationship between the three data types and the drug metric (ex. deltaAUC) for each drug combination. To get an overall idea of model performance, we plot the differences between the model prediction (yhat) and the observed value (y) from the cross validation loop. We generally consider the cutoff for correlation to be 0.1.

Box plot TANDEM correlations

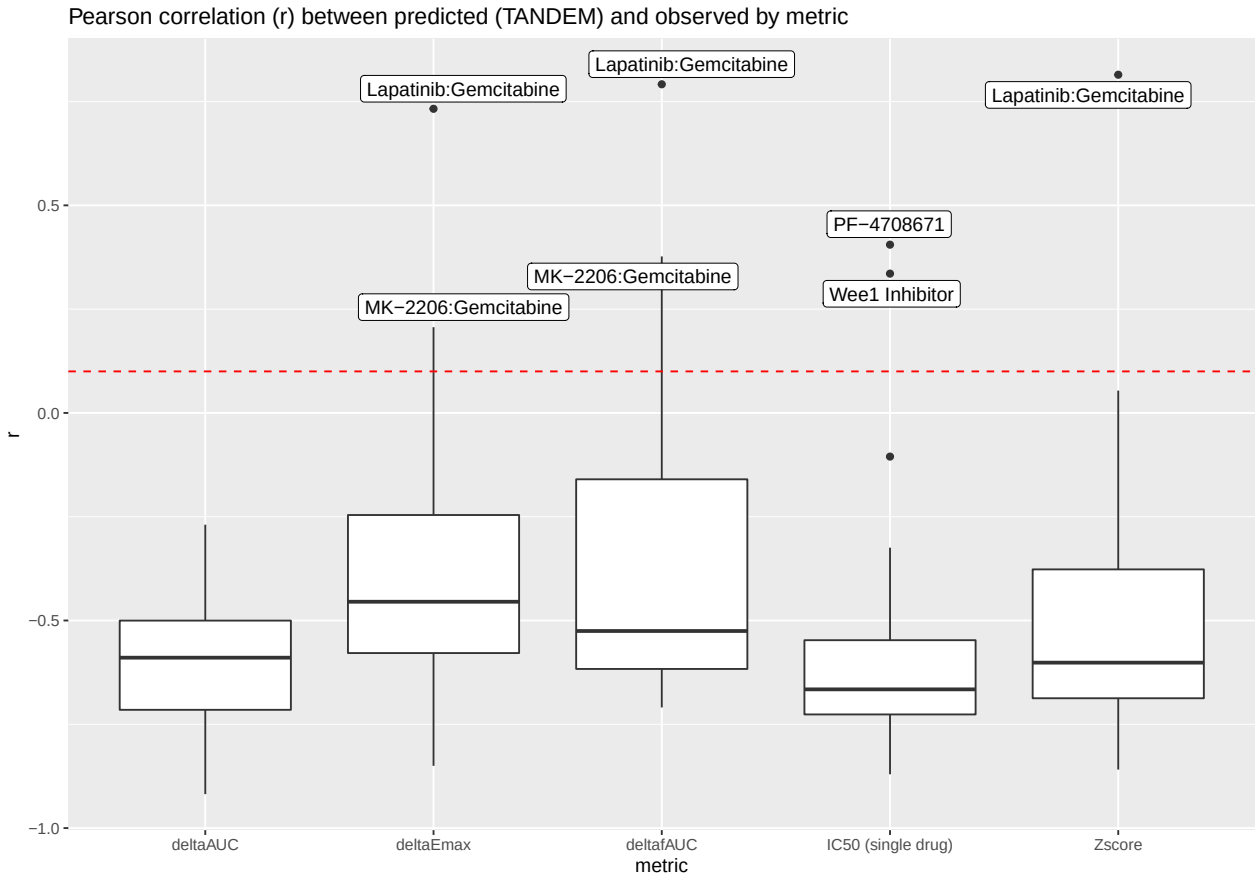
Immediately clear is that very few drug combinations exceed the 0.1 correlation threshold and that the overall performance is poor. Generally it is quite difficiult to identify biomarkers from drug synergy, so the poor performance is to be expected. Also striking is the preponderance of negative correlations for all metrics. This is an artefact of cross validation loops, particularly those that select no features, i.e., intercept-only models.

Load TANDEM results and process them into a tidy-style data frame. Plot those drugs and drug combinations which exceed the 0.1 correlatation threshold.

```
tandemfiles =
tibble(file = list.files(here("Rds"), pattern = "Stats.*cngene.Rds")) %>%
  mutate(screen = if_else(str_detect(file, "single"), "single", "combi")) %>%
  mutate(metric = case_when(
    str_detect(file, "fAUC") ~ "deltafAUC",
    str_detect(file, "AUC") ~ "deltaAUC",
    str_detect(file, "Zscore") ~ "Zscore",
    str_detect(file, "Emax") ~ "deltaEmax",
    TRUE ~ "IC50")
  )

tandemdata = lapply(here("Rds", tandemfiles$file), readRDS)
names(tandemdata) = str_remove(tandemfiles$file, "\\.Rds")

#str(tandemdata)

tandemdata =
unlist(tandemdata) %>% enframe("name", "value") %>%
  mutate(type = case_when(
    str_detect(name, "\\.cor") ~ "cor",
    str_detect(name, "\\.beta") ~ "beta",
    str_detect(name, "\\.mse") ~ "mse",
    str_detect(name, "\\.upstream") ~ "contr_upstream",
    str_detect(name, "\\.downstream") ~ "contr_downstream",
    str_detect(name, "y_hat*") ~ "yhat",
    str_detect(name, "y[0-9]*") ~ "y",
    TRUE ~ ""
  )) %>% select(type, name, everything())# %>%
  #filter(type != "y") #Actually keep this
```

```
tandemdata = tandemdata %>%
  separate(name, into=c("group", "drug", NA),
           sep="\\.", remove=F, extra = "merge") %>%
  mutate(file = paste0(group, ".Rds")) %>%
  left_join(.,tandemfiles, by = "file") %>%
  mutate(metric = str_replace(metric, "IC50", "IC50 (single drug)")) %>%
  select(value, drug, metric, screen, everything())

tandemdata %>% filter(type == "cor") %>%
  rename(cor = value) %>%
  mutate(label = if_else(cor > 0.1, drug, NULL)) %>%
  ggplot(aes(x = metric, y = cor, label=label)) +
  geom_boxplot() +
  ggtitle("Pearson correlation (r) between predicted (TANDEM) and observed by metric") +
  ylab("r") +
  geom_hline(yintercept = 0.1, color="red", linetype="dashed") +
  ggrepel::geom_label_repel()
```

```
## Warning: Removed 112 rows containing missing values (geom_label_repel).
```



Boxplot MSE (TANDEM)

A plot of the mean squared error gives us the same top hits.

```
tandemdata %>% filter(type == "mse") %>%
  rename(mse = value) %>%
  mutate(label = if_else(mse < 0.9, drug, NULL)) %>%
  ggplot(aes(x = metric, y = mse, label = label)) +
  geom_boxplot() +
  ggtitle("Mean squared error of TANDEM model by metric") +
  ylab("MSE") +
  geom_hline(yintercept = 0.9, color = "red", linetype="dashed") +
  ggrepel::geom_label_repel()
```

```
## Warning: Removed 113 rows containing missing values (geom_label_repel).
```

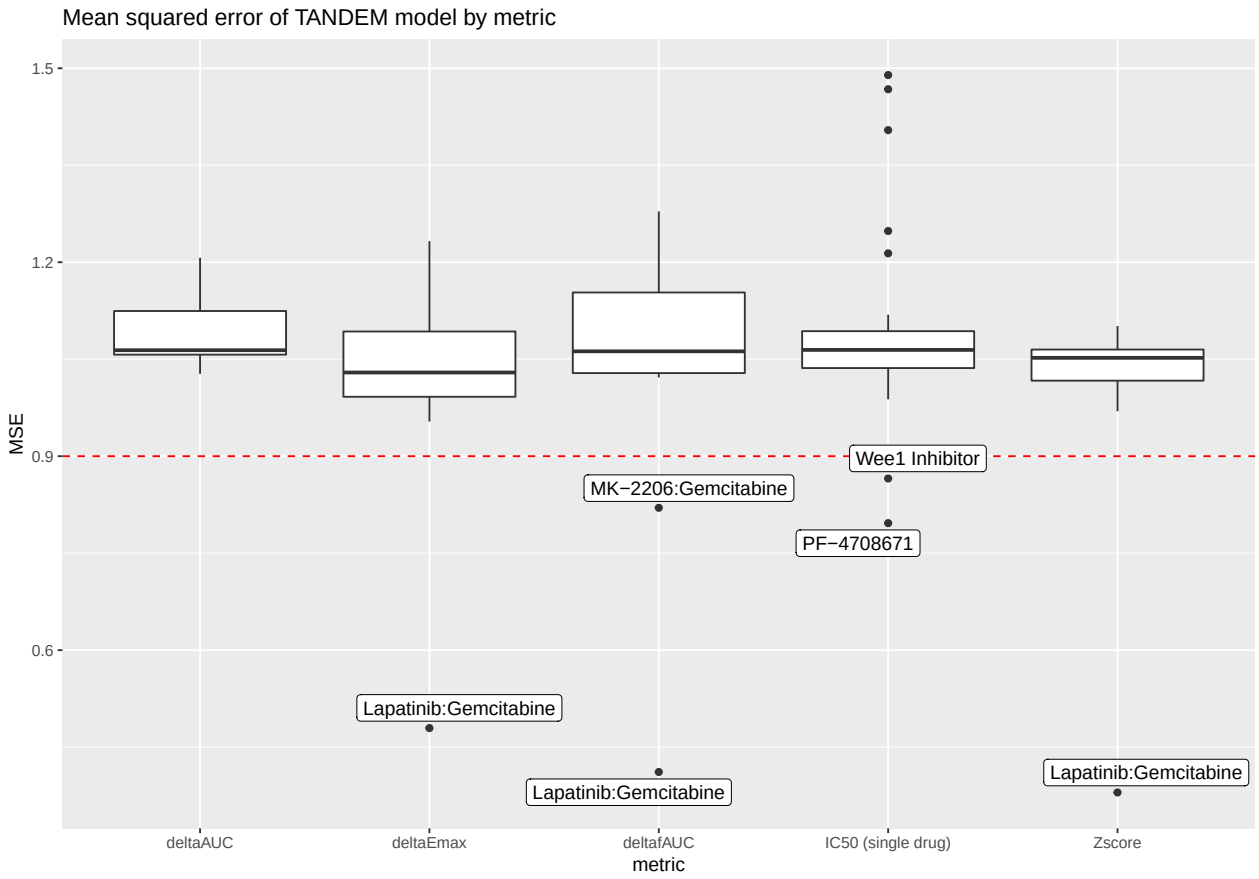


Table best performers

Summary overview of those drug (combinations) with a correlation above 0.1.

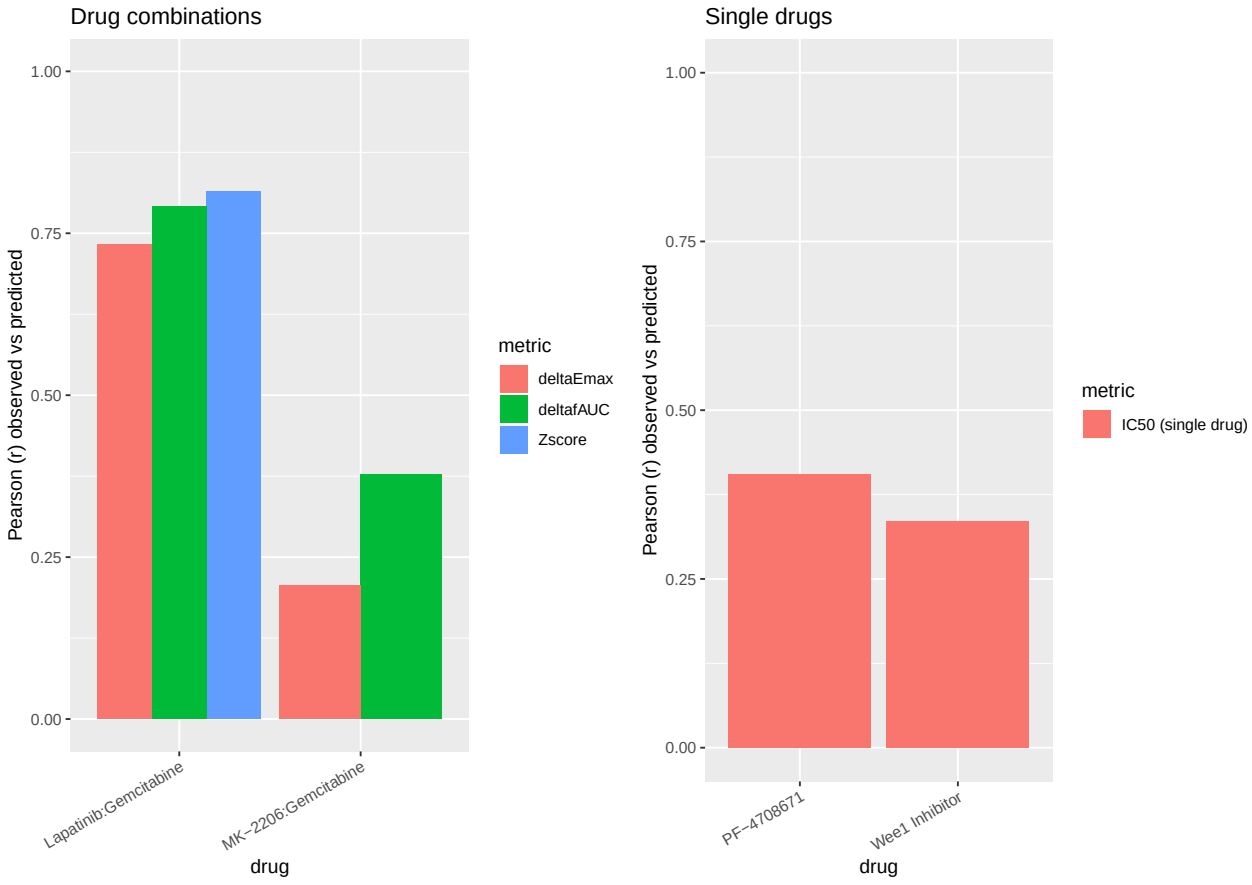
```
best_drugs = tandemdata %>% filter(type == "cor") %>% filter(value > 0.1) %>%
  rename(cor = value) %>% arrange(desc(cor)) %>%
  select(cor, drug, metric, screen)

best_drugs
```

```
## # A tibble: 7 x 4
##     cor drug                 metric            screen
##   <dbl> <chr>                <chr>             <chr>
## 1 0.815 Lapatinib:Gemcitabine Zscore            combi
```

```
## 2 0.792 Lapatinib:Gemcitabine deltafAUC          combi
## 3 0.733 Lapatinib:Gemcitabine deltaEmax          combi
## 4 0.405 PF-4708671            IC50 (single drug) single
## 5 0.377 MK-2206:Gemcitabine   deltafAUC          combi
## 6 0.336 Wee1 Inhibitor        IC50 (single drug) single
## 7 0.207 MK-2206:Gemcitabine   deltaEmax          combi
```

Plot the correlations of the best performing drugs:

```
ggarrange(best_drugs %>% filter(screen == "combi") %>%
            ggplot(aes(x=drug, y = cor, fill = metric)) +
            geom_bar(stat="identity", position = "dodge") +
            theme(axis.text.x = element_text(angle=30, hjust=1)) +
            ggtitle("Drug combinations") +
            ylab("Pearson (r) observed vs predicted") + ylim(c(0,1)),
          best_drugs %>% filter(screen == "single") %>%
            ggplot(aes(x=drug, y = cor, fill = metric)) +
            geom_bar(stat="identity", position = "dodge") +
            theme(axis.text.x = element_text(angle=30, hjust=1)) +
            ggtitle("Single drugs") +
            ylab("Pearson (r) observed vs predicted") + ylim(c(0,1)),
  ncol = 2, nrow=1)
```



By far the drug combination with the best performance is Lapatinib:Gemcitabine, in terms of correlation. However, it is worth noting that the effect size is quite small, as can be seen in the heatmaps in the section Combination response data > Heatmaps by response metric.

Targets of the best drugs:

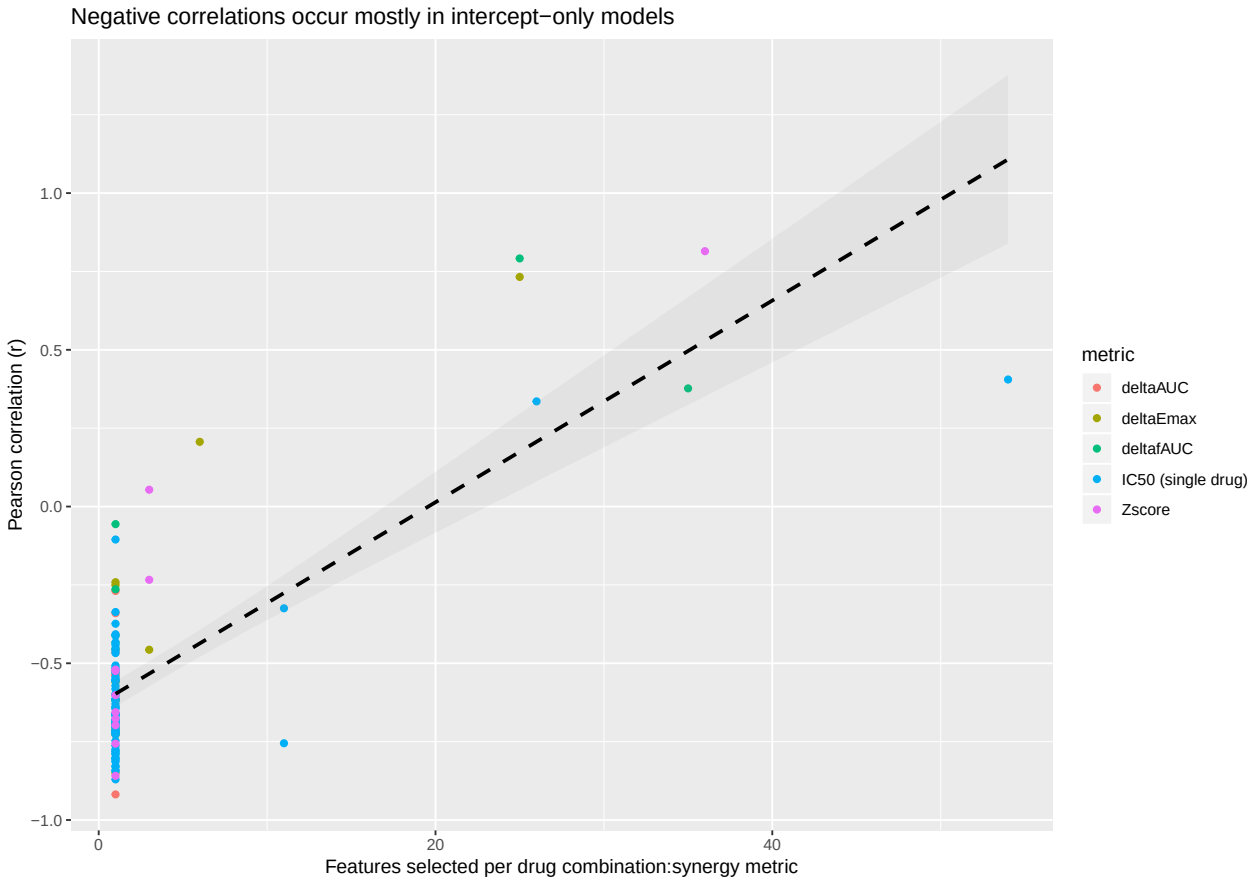
```
drug_decode %>%
  filter(DRUG_NAME %in% c("Lapatinib","Gemcitabine","MK-2206",
                          "Wee1 Inhibitor","PF-4708671")) %>%
  select(DRUG_NAME, PUTATIVE_TARGET)
```

```
## # A tibble: 5 x 2
##   DRUG_NAME      PUTATIVE_TARGET
##   <chr>          <chr>
## 1 MK-2206        AKT1, AKT2
## 2 Wee1 Inhibitor WEE1, CHEK1
## 3 PF-4708671     S6K1
## 4 Gemcitabine    Pyrimidine antimetabolite
## 5 Lapatinib      ERBB2, EGFR
```

Feature selection

Test the theory that negative coeficients are largely due to no features being selected (a feature count of 1 = intercept only). Except for a couple of oddities in the single drug IC50 data, this theory is confirmed.

```
tandemdata %>% filter(type=="beta") %>%
  separate(name, into = c(NA,NA,NA, "feature"), remove = F, sep="\\.") %>%
  select(feature, drug, metric) %>%
  mutate(key = paste(drug, metric, sep=":")) %>%
  group_by(metric, drug, key) %>% summarize(n_features=n()) %>%
  arrange(desc(n_features)) %>%
  left_join(.,(tandemdata %>% filter(type == "cor") %>%
                 mutate(key = paste(drug, metric, sep=":"),) %>%
                 select(key, cor = value)),
            by="key") %>%
  ggplot(aes(x = n_features, y = cor)) +
  geom_point(aes(color = metric)) +
  geom_smooth(method="lm", alpha = 0.1, color = "black", linetype="dashed") +
  ggtitle("Negative correlations occur mostly in intercept-only models") +
  ylab("Pearson correlation (r)") +
  xlab("Features selected per drug combination:synergy metric")
```



Feature importance

For those drug combinations which have a correlation of at least 0.1, which features contribute the most? We rank the features according to an importance metric, which we define as the coefficient scaled by variance(yhat). The feature importance may be either positively or negatively predictive of a synergy metric.

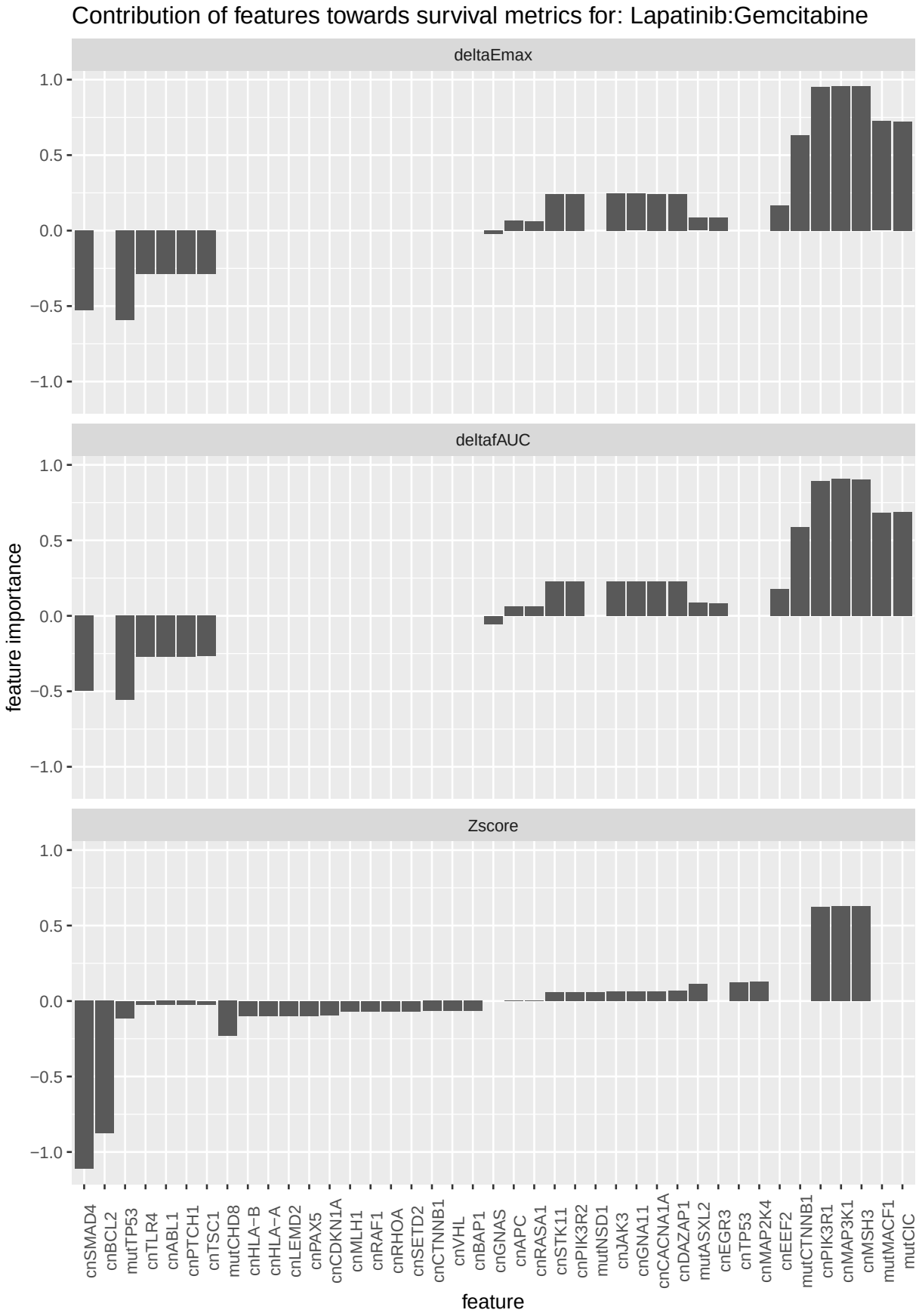
```
featuredata=
  mutate(tandemdata,key = paste(drug, metric, sep=":")) %>%
  #Add key for merging
  filter(key %in% mutate(best_drugs,key = paste(drug, metric, sep=":"))$key) %>%
  filter(type == "beta") %>%
  #extract the feature from the name
  separate(name, into = c(NA,NA,NA, "feature"), remove = F, sep="\\.") %>%
  filter(feature != "(Intercept)") %>%
  #calculate the variance and join it to the data
  left_join(., (tandemdata %>% filter(type == "yhat") %>%
                  mutate(key = paste(drug, metric, sep=":")) %>%
                  group_by(key) %>%
                  summarise(yhat_var = var(value))), by="key") %>%
  left_join(., (tandemdata %>% mutate(key = paste(drug, metric, sep=":")) %>%
                  filter(type == "cor") %>%
                  select(key, cor=value)), by="key") %>%
  #ungroup() %>%
  #mutate(feature_importance = beta/yhat_var) %>% #Complains about S4 type
  select(feature,  beta = value, yhat_var,cor, drug, metric, everything())
```

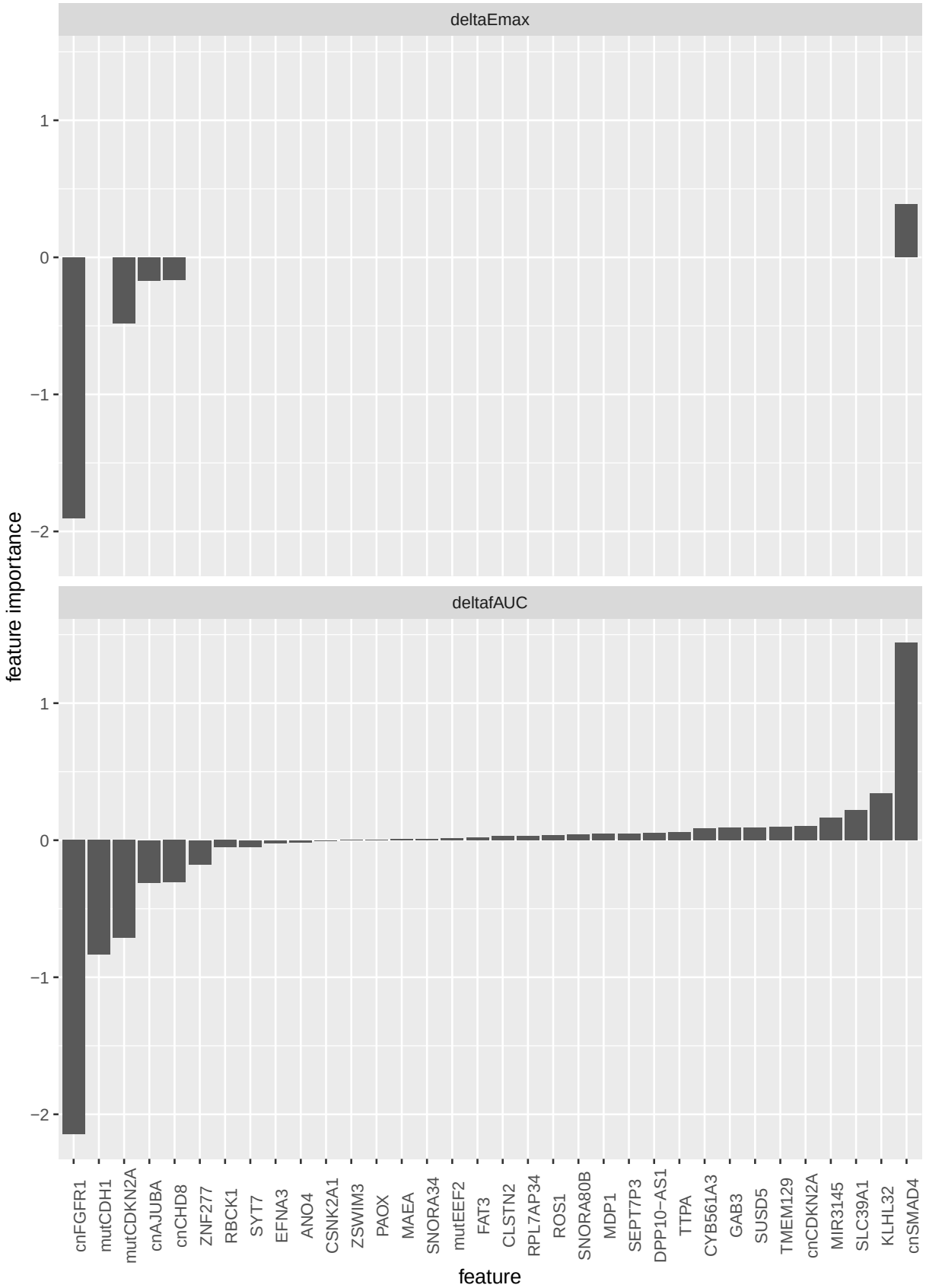
```
featuredata$feature_importance = featuredata$beta / featuredata$yhat_var

featuredata = featuredata %>% select(feature_importance, everything())
featuredata = featuredata %>% arrange(feature_importance, drug)

head(featuredata)
```

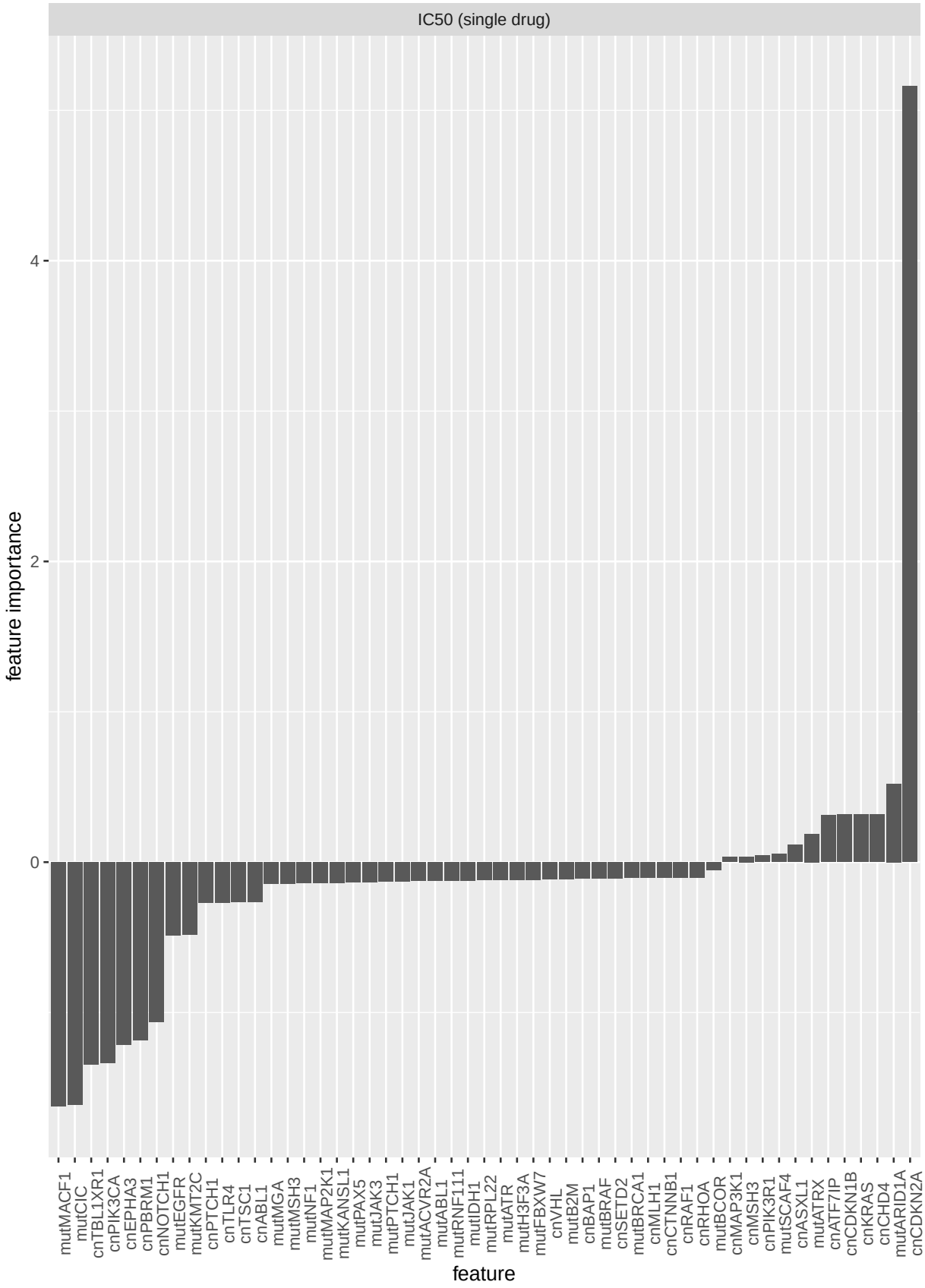
```
## # A tibble: 6 x 13
##   feature_importa~ feature    beta yhat_var   cor drug  metric screen type
##              <dbl> <chr>     <dbl>    <dbl> <dbl> <chr> <chr>  <chr>  <chr>
## 1            -5.93 mutTP53 -0.200    0.0338 0.336 Wee1~ IC50 ~ single beta
## 2            -5.16 cnCDH1  -0.175    0.0338 0.336 Wee1~ IC50 ~ single beta
## 3            -2.15 cnFGFR1 -0.436    0.203  0.377 MK-2~ delta~ combi  beta
## 4            -2.02 mutCDH1 -0.0681   0.0338 0.336 Wee1~ IC50 ~ single beta
## 5            -1.90 cnFGFR1 -0.338    0.178  0.207 MK-2~ delta~ combi  beta
## 6            -1.63 mutMAC~ -0.256    0.157  0.405 PF-4~ IC50 ~ single beta
## # ... with 4 more variables: name <chr>, group <chr>, file <chr>, key <chr>
```

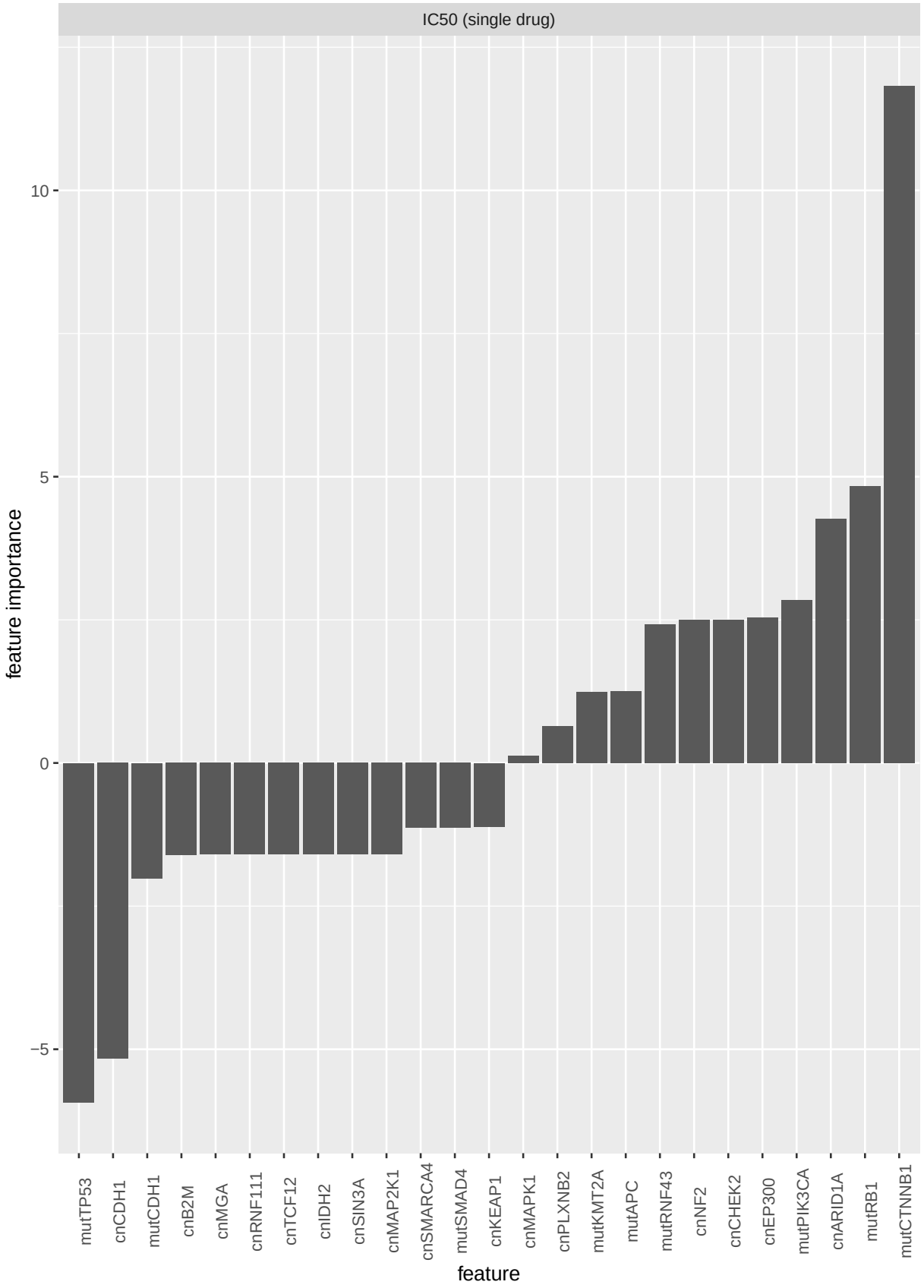
When plotted together, it is evident that features with strong importance scores in one metric generally have strong importance scores in the other metrics. There are a few exceptions, for example mutCIC has a high feature importance for deltaEmax and fAUC in Lapatinib:Gemcitabine, but was not selected as a feature in the Zscore model.

```
featuredata = featuredata %>%
  mutate(drug = factor(drug,
                       levels=c("Lapatinib:Gemcitabine","MK-2206:Gemcitabine",
                                "PF-4708671", "Wee1 Inhibitor")))

for (i in 1:length(levels(featuredata$drug))){
  plotdata = featuredata[featuredata$drug == levels(featuredata$drug)[i],]
  plotdata = plotdata[order(plotdata$feature_importance),]
  plot = plotdata %>%
    ggplot(aes(x = factor(feature, levels=unique(feature)),
               y = feature_importance)) +
    geom_bar(stat="identity") + facet_wrap(~metric, ncol = 1) +
    xlab("feature") + ylab("feature importance") +
    theme(axis.text.x = element_text(angle=90)) +
    ggtitle(paste("Contribution of features towards survival metrics for:",
                  levels(featuredata$drug)[i]))
  print(plot)
}
```


Contribution of features towards survival metrics for: Lapatinib:Gemcitabine
deltaEmax
deltafAUC
Zscore
feature importance
1.0
0.5
0.0
−0.5
−1.0
cnSMAD4
cnBCL2
mutTP53
cnTLR4
cnABL1
cnPTCH1
cnTSC1
mutCHD8
cnHLA−B
cnHLA−A
cnLEMD2
cnPAX5
cnCDKN1A
cnMLH1
cnRAF1
cnRHOA
cnSETD2
cnCTNNB1
cnVHL
cnBAP1
cnGNAS
cnAPC
cnRASA1
cnSTK11
cnPIK3R2
mutNSD1
cnJAK3
cnGNA11
cnCACNA1A
cnDAZAP1
mutASXL2
cnEGR3
cnTP53
cnMAP2K4
cnEEF2
mutCTNNB1
cnPIK3R1
cnMAP3K1
cnMSH3
mutMACF1
mutClC
feature


Contribution of features towards survival metrics for: MK−2206:Gemcitabine
deltaEmax
deltafAUC
feature importance
1
0
−1
−2
cnFGFR1
mutCDH1
mutCDKN2A
cnAJUBA
cnCHD8
ZNF277
RBCK1
SYT7
EFNA3
ANO4
CSNK2A1
ZSWIM3
PAOX
MAEA
SNORA34
mutEEF2
FAT3
CLSTN2
RPL7AP34
ROS1
SNORA80B
MDP1
SEPT7P3
DPP10−AS1
TTPA
CYB561A3
GAB3
SUSD5
TMEM129
cnCDKN2A
MIR3145
SLC39A1
KLHL32
cnSMAD4
feature


Contribution of features towards survival metrics for: PF−4708671
IC50 (single drug)
feature importance
4
2
0
mutMACF1
mutCIC
cnTBL1XR1
cnPIK3CA
cnEPHA3
cnPBRM1
cnNOTCH1
mutEGFR
mutKMT2C
cnPTCH1
cnTLR4
cnTSC1
cnABL1
mutMGA
mutMSH3
mutNF1
mutMAP2K1
mutKANSL1
mutPAX5
mutJAK3
mutPTCH1
mutJAK1
mutACVR2A
mutABL1
mutRNF111
mutIDH1
mutRPL22
mutATR
mutH3F3A
mutFBXW7
cnVHL
mutB2M
cnBAP1
mutBRAF
cnSETD2
mutBRCA1
cnMLH1
cnCTNNB1
cnRAF1
cnRHOA
mutBCOR
cnMAP3K1
cnMSH3
cnPIK3R1
mutSCAF4
cnASXL1
mutATRX
cnATF7IP
cnCDKN1B
cnKRAS
cnCHD4
mutARID1A
cnCDKN2A
feature


Contribution of features towards survival metrics for: Wee1 Inhibitor
IC50 (single drug)
feature importance
10
5
0
−5
mutTP53
cnCDH1
mutCDH1
cnB2M
cnMGA
cnRNF111
cnTCF12
cnIDH2
cnSIN3A
cnMAP2K1
cnSMARCA4
mutSMAD4
cnKEAP1
cnMAPK1
cnPLXNB2
mutKMT2A
mutAPC
mutRNF43
cnNF2
cnCHEK2
cnEP300
mutPIK3CA
cnARID1A
mutRB1
mutCTNNB1
feature

Plot top important features

This function will rank the features by abs(feature_importance) and plot the top 10 most important.

```
plot_tandem = function(featuredata, drug, metric, metinput, topn=10){

  require(tidyverse)
  require(ggbeeswarm)


  d = featuredata %>% filter(drug == !!drug & metric == !!metric) %>%
    mutate(abs_fi = abs(feature_importance)) %>% arrange(desc(abs_fi)) %>%
    select(abs_fi, feature_importance, feature, cor) %>%
    mutate(type = case_when(
      str_detect(feature, "^cn") ~ "copynumber",
      str_detect(feature, "^mut") ~ "mutation",
      TRUE ~ "gene_expression"
    )) %>% head(topn)

  plotlist = list()

  for (i in 1:nrow(d)){
    thistype = d$type[[i]]
    thisfeature = d$feature[[i]]
    thiscor = d$cor[[i]]
    thisfi = d$feature_importance[[i]]

    if (thistype == "mutation"){
      mutfeature <- mutMatrix[rownames(mutMatrix)==thisfeature, ]
      mutfeature <- mutfeature %>% enframe("sample", "mutdata") %>%
        mutate(mut = if_else(mutdata == 0, "wt", "mut")) %>%
        left_join(.,
             rownames_to_column(as.data.frame(
               metinput[,colnames(metinput)==drug, drop = F]),
               "sample"),
             by="sample") %>% na.omit() %>%
        mutate(metric = thistype, feature = thisfeature,
               cor = thiscor, importance = thisfi)
        mutplot = mutfeature %>%
          ggplot(aes(x = mut, y = get(drug), color = mut)) +
           geom_jitter(width = 0.2, height = 0)+
          #geom_beeswarm(groupOnX = T) + #By request to reduce whitespace
          ggtitle(paste0(thisfeature, ", feat.imp: ", round(thisfi, 3))) +
                    ylab(metric)+ theme(plot.title = element_text(size = 10),
                              axis.title.x = element_text(size=8),
                              axis.title.y = element_text(size=8),
                              legend.position = "none") +
          xlab("Mutation")
      plotlist[[i]] = mutplot
    }


    if (thistype == "copynumber"){
      cnfeature <- cnMatrix[rownames(cnMatrix)==thisfeature, ]
```

```
    cnfeature <- cnfeature %>% enframe("sample", "cndata") %>%
      mutate(cn_event = if_else(cndata == 0, "wt", "cn_abnormal")) %>%
      left_join(.,
           rownames_to_column(as.data.frame(
             metinput[,colnames(metinput)==drug, drop = F]),
             "sample"),
           by="sample") %>% na.omit() %>%
      mutate(metric = thistype, feature = thisfeature,
             cor = thiscor, importance = thisfi)
      cnplot = cnfeature %>%
        ggplot(aes(x = cn_event, y = get(drug), color = cn_event)) +
         geom_jitter(width = 0.2, height = 0)+
        #geom_beeswarm(groupOnX = T) + #By request to reduce whitespace
        ggtitle(paste0(thisfeature, ", feat.imp: ", round(thisfi, 3))) +
                  ylab(metric)+ theme(plot.title = element_text(size = 10),
                            axis.title.x = element_text(size=8),
                            axis.title.y = element_text(size=8),
                            legend.position = "none") +
        xlab("Copy number")

    plotlist[[i]] = cnplot
  }

  if (thistype == "gene_expression"){
    gexfeature <- geneEx[rownames(geneEx)==thisfeature, ]
    gexfeature <- gexfeature %>% enframe("sample", "gexdata") %>%
      left_join(.,
           rownames_to_column(as.data.frame(
             metinput[,colnames(metinput)==drug, drop = F]),
             "sample"),
           by="sample") %>% na.omit() %>%
      mutate(metric = thistype, feature = thisfeature,
             cor = thiscor, importance = thisfi)
      gexplot = gexfeature %>%
        ggplot(aes(x = gexdata, y = get(drug))) +
        geom_jitter(width = 0.2, height = 0)+
        #geom_beeswarm(groupOnX = T) + #By request to reduce whitespace
        ggtitle(paste0(thisfeature, ", feat.imp: ", round(thisfi, 3))) +
        ylab(metric)+ theme(plot.title = element_text(size = 10),
                            axis.title.x = element_text(size=8),
                            axis.title.y = element_text(size=8),
                            legend.position = "none") +
        xlab("Gene expression")
    plotlist[[i]] = gexplot
  }

}

arrangeplot = ggarrange(plotlist = plotlist, ncol=2, nrow=5)

arrangeplot = annotate_figure(arrangeplot,
              top = text_grob(paste0(drug, ", ",
                                     metric, ", cor: ", round(thiscor, 3)))
```

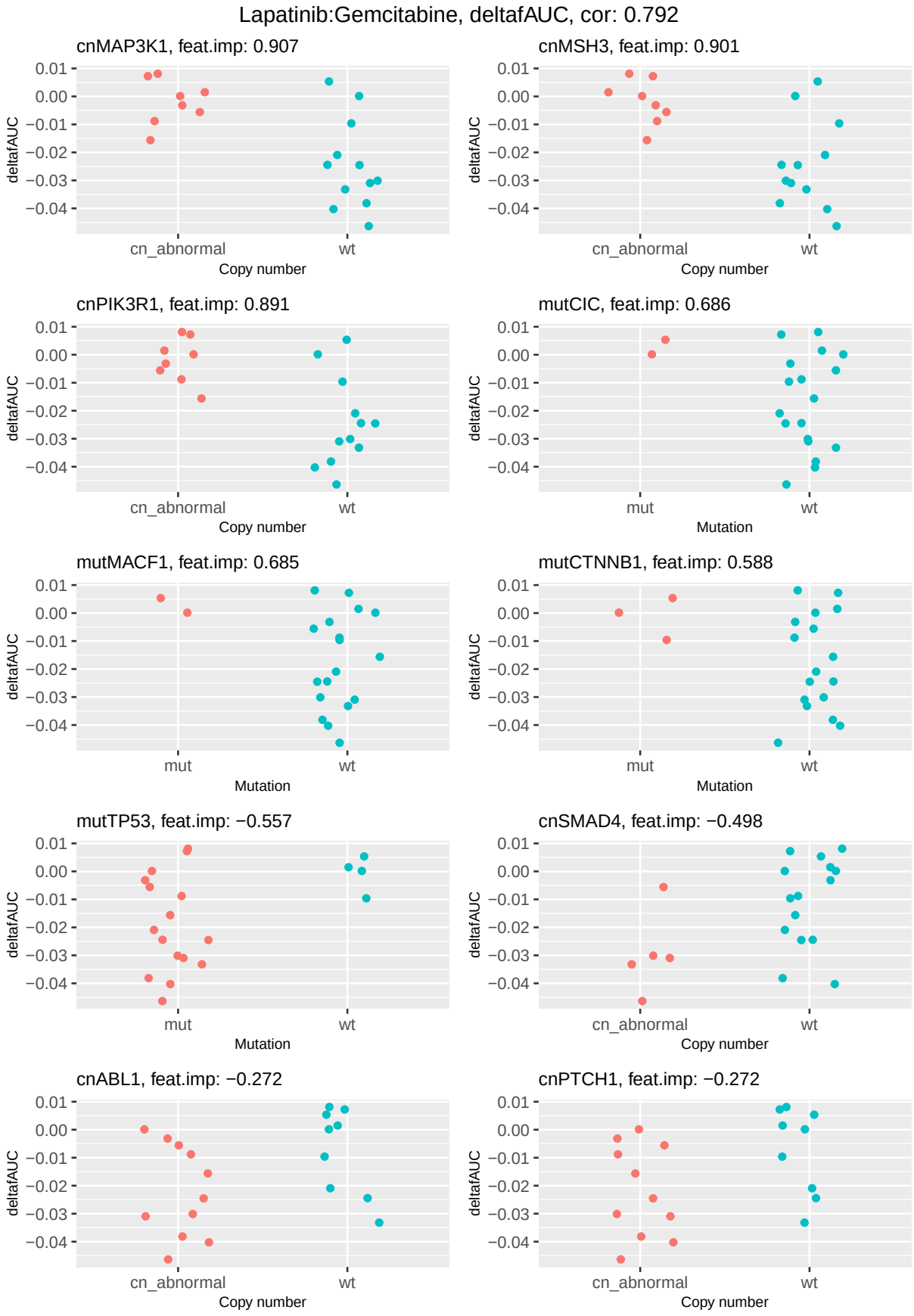
```
                )
  return(arrangeplot)
}
```

fAUC Lapatinib:Gemcitabine

Focusing on Lapatinib:Gemcitabine (best combination) deltafAUC (most intuitive metric).

```
plot_tandem(featuredata,
            drug = "Lapatinib:Gemcitabine",
            metric = "deltafAUC", metinput = median_DELTA_fAUC)
```

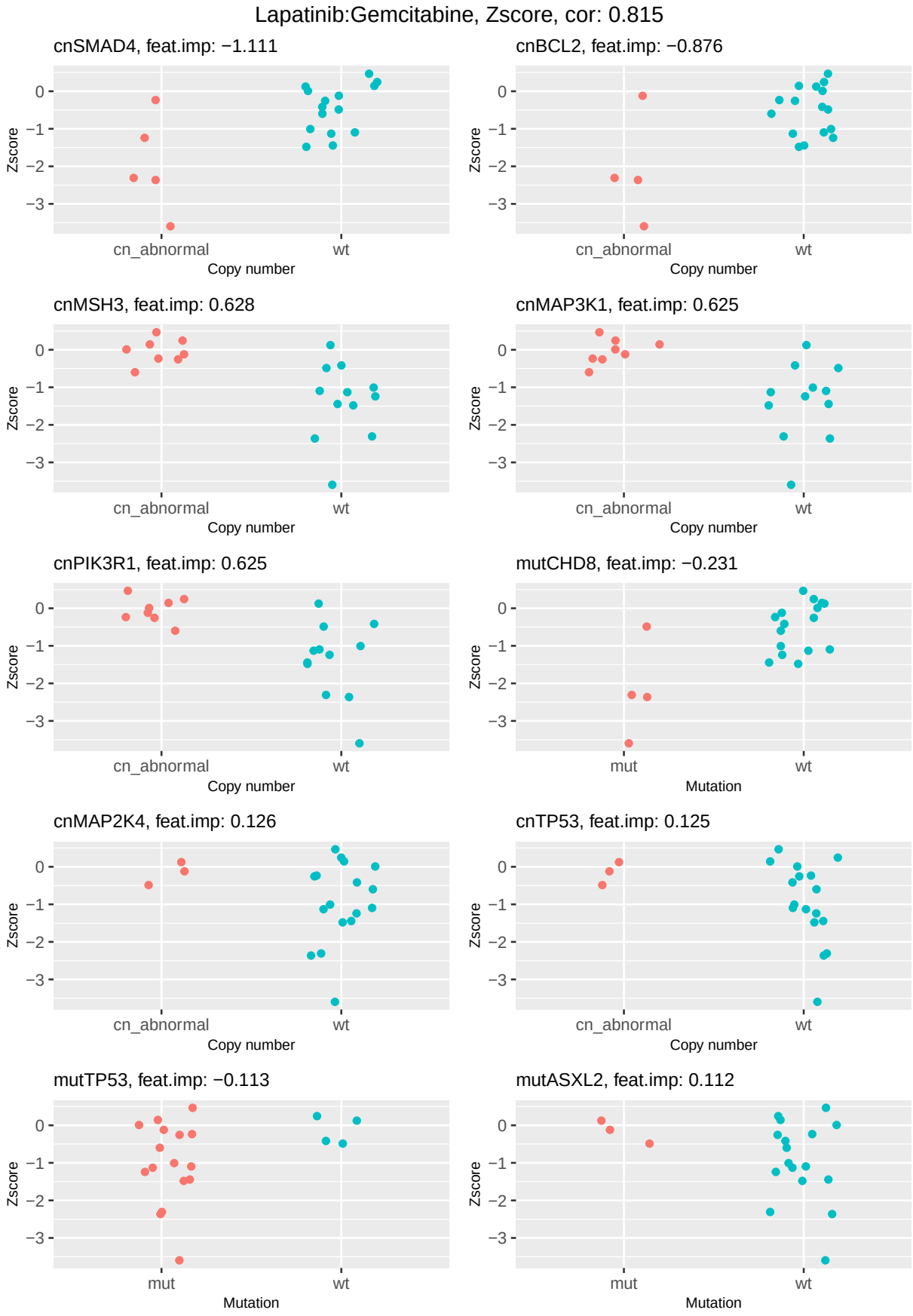
```
## Loading required package: ggbeeswarm
```


Lapatinib:Gemcitabine, deltafAUC, cor: 0.792
cnMAP3K1, feat.imp: 0.907
cnMSH3, feat.imp: 0.901
cnPIK3R1, feat.imp: 0.891
mutCIC, feat.imp: 0.686
mutMACF1, feat.imp: 0.685
mutCTNNB1, feat.imp: 0.588
mutTP53, feat.imp: −0.557
cnSMAD4, feat.imp: −0.498
cnABL1, feat.imp: −0.272
cnPTCH1, feat.imp: −0.272
deltafAUC
0.01
0.00
−0.01
−0.02
−0.03
−0.04
cn_abnormal
wt
Copy number
mut
Mutation

Zscore Lapatinib:Gemcitabine

Highest overall correlation.

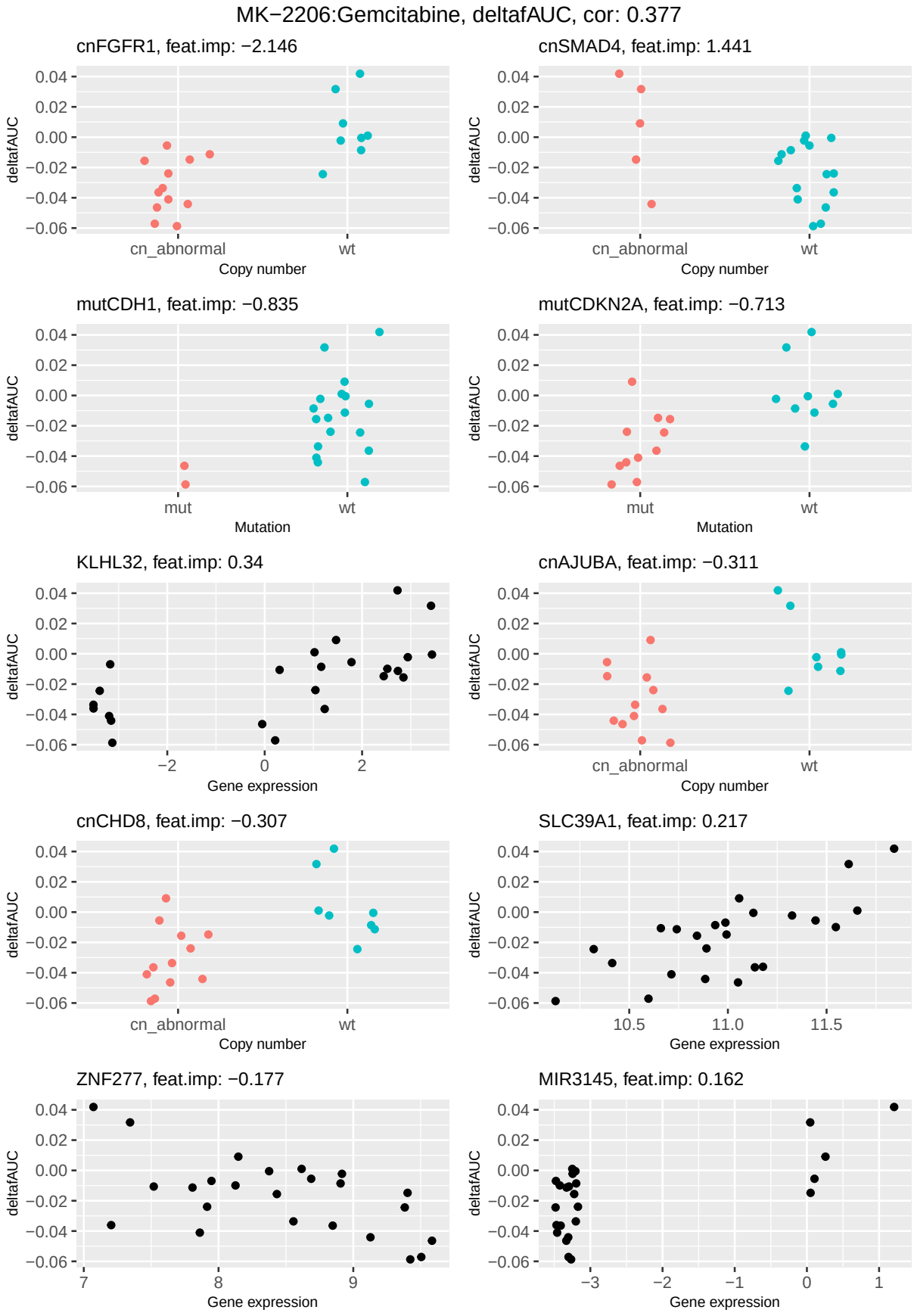
```
plot_tandem(featuredata, drug = "Lapatinib:Gemcitabine",
            metric = "Zscore", metinput = median_ZSCORE)
```


Lapatinib:Gemcitabine, Zscore, cor: 0.815
cnSMAD4, feat.imp: −1.111
cnBCL2, feat.imp: −0.876
cnMSH3, feat.imp: 0.628
cnMAP3K1, feat.imp: 0.625
cnPIK3R1, feat.imp: 0.625
mutCHD8, feat.imp: −0.231
cnMAP2K4, feat.imp: 0.126
cnTP53, feat.imp: 0.125
mutTP53, feat.imp: −0.113
mutASXL2, feat.imp: 0.112
Zscore
0
−1
−2
−3
cn_abnormal
wt
Copy number
mut
Mutation

MK-2206:Gemcitabine deltafAUC

The correlation is not as good, but the effect size is much greater.

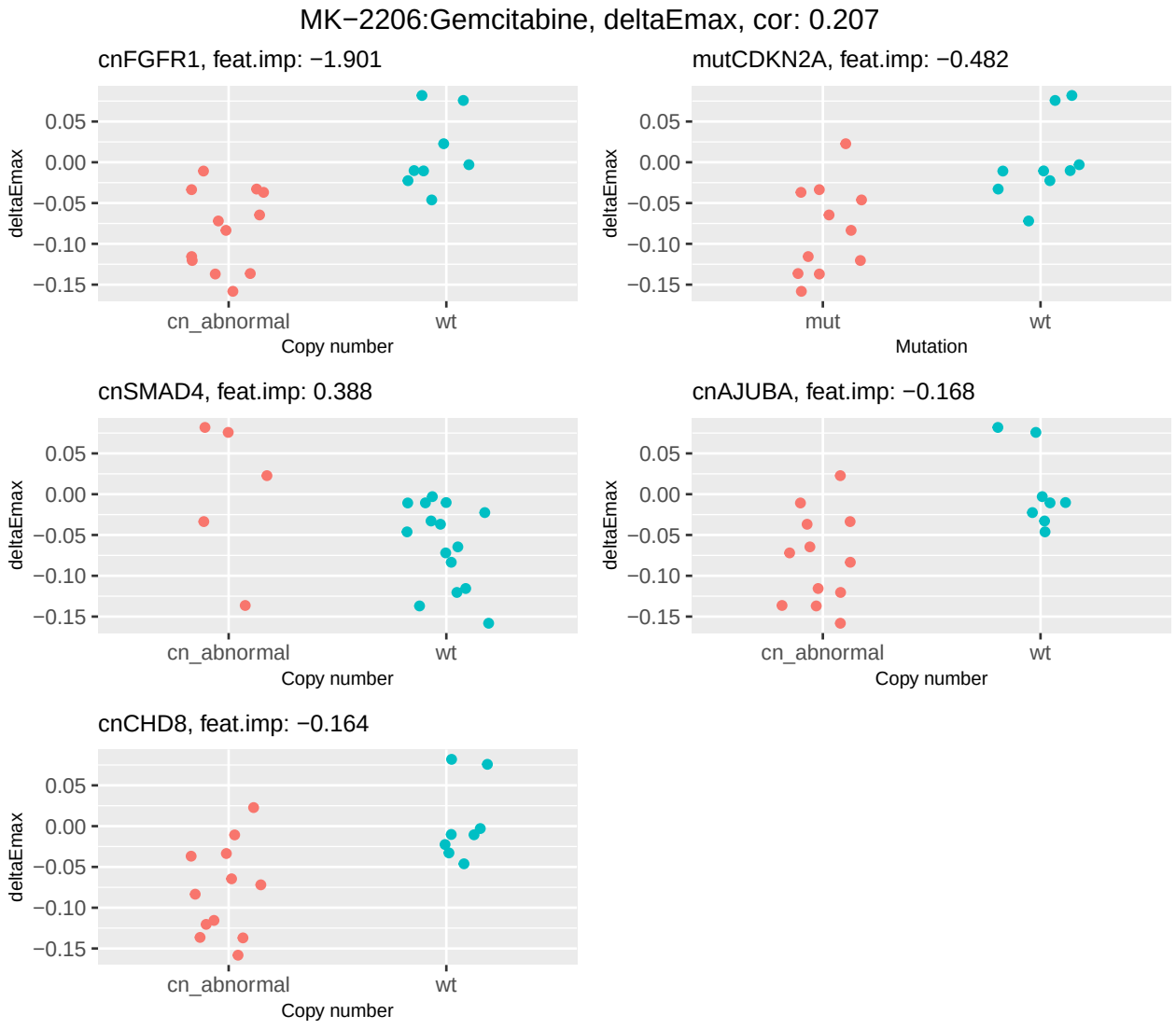
```
plot_tandem(featuredata, drug = "MK-2206:Gemcitabine",
            metric = "deltafAUC", metinput = median_DELTA_fAUC)
```


MK−2206:Gemcitabine, deltafAUC, cor: 0.377
cnFGFR1, feat.imp: −2.146
cnSMAD4, feat.imp: 1.441
mutCDH1, feat.imp: −0.835
mutCDKN2A, feat.imp: −0.713
KLHL32, feat.imp: 0.34
cnAJUBA, feat.imp: −0.311
cnCHD8, feat.imp: −0.307
SLC39A1, feat.imp: 0.217
ZNF277, feat.imp: −0.177
MIR3145, feat.imp: 0.162
deltafAUC
0.04
0.02
0.00
−0.02
−0.04
−0.06
cn_abnormal
wt
Copy number
mut
Mutation
−2
0
2
Gene expression
10.5
11.0
11.5
7
8
9
−3
−2
−1
0
1

MK-2206:Gemcitabine deltaEmax

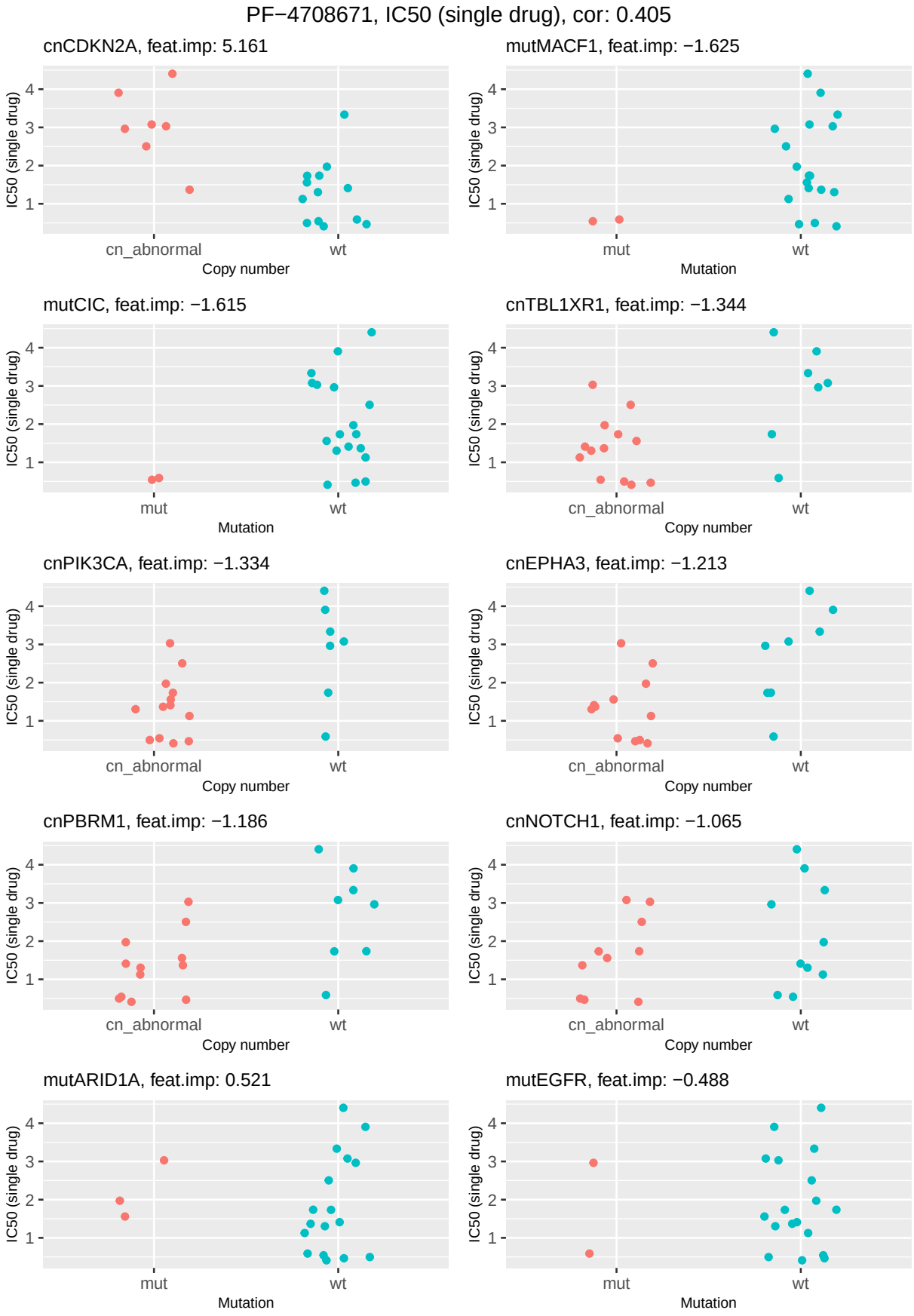
Still a greater effect size than Lapatinib:Gemcitabine deltaEmax

```
plot_tandem(featuredata, drug = "MK-2206:Gemcitabine",
            metric = "deltaEmax", metinput = median_DELTA_EMAX)
```



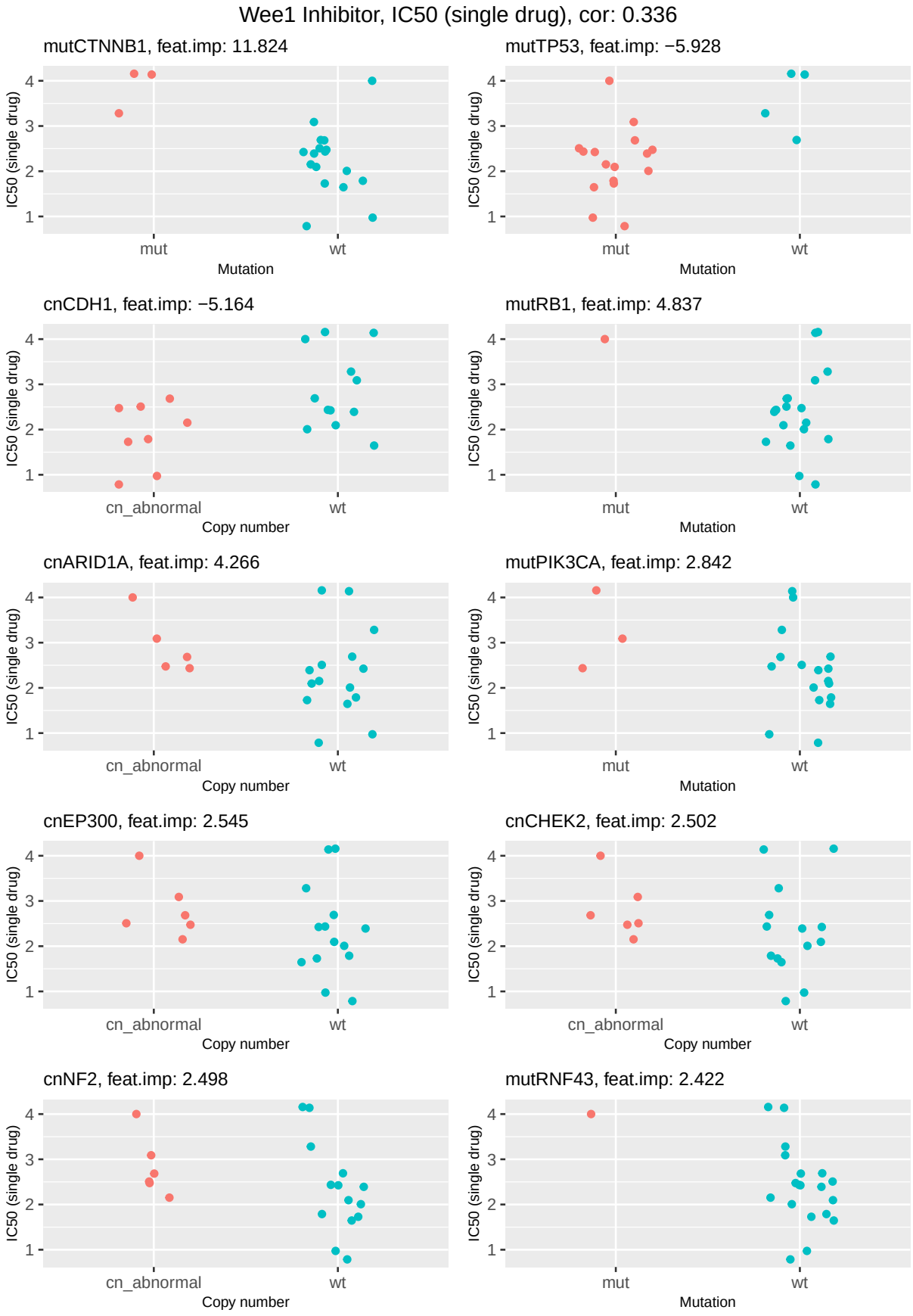
Single drug PF-4708671

```
plot_tandem(featuredata, drug = "PF-4708671",
            metric = "IC50 (single drug)", metinput = IC50)
```


PF−4708671, IC50 (single drug), cor: 0.405
cnCDKN2A, feat.imp: 5.161
mutMACF1, feat.imp: −1.625
mutCIC, feat.imp: −1.615
cnTBL1XR1, feat.imp: −1.344
cnPIK3CA, feat.imp: −1.334
cnEPHA3, feat.imp: −1.213
cnPBRM1, feat.imp: −1.186
cnNOTCH1, feat.imp: −1.065
mutARID1A, feat.imp: 0.521
mutEGFR, feat.imp: −0.488
IC50 (single drug)
1
2
3
4
cn_abnormal
wt
mut
Copy number
Mutation

Single drug Wee1 Inhibitor

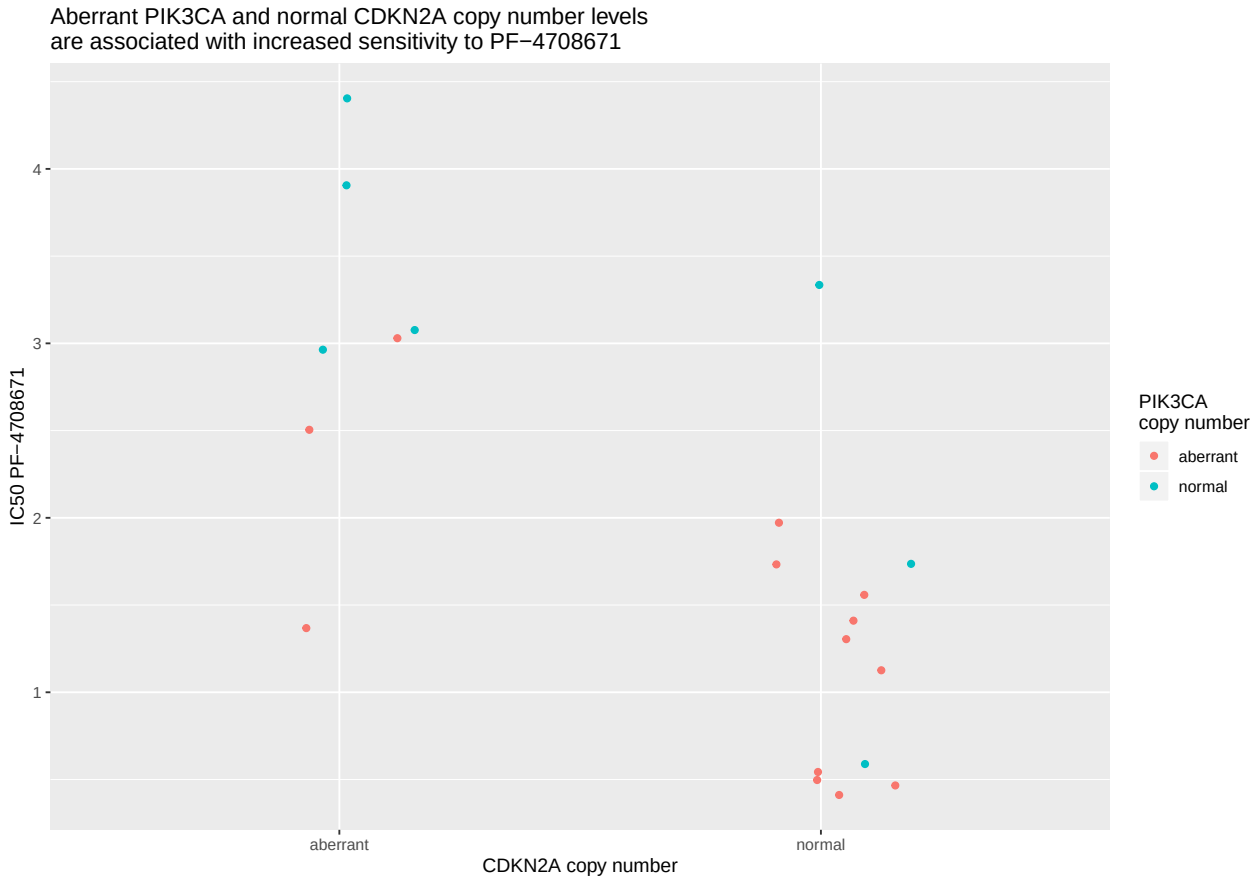
```
plot_tandem(featuredata, drug = "Wee1 Inhibitor",
            metric = "IC50 (single drug)", metinput = IC50)
```


Wee1 Inhibitor, IC50 (single drug), cor: 0.336
mutCTNNB1, feat.imp: 11.824
mutTP53, feat.imp: −5.928
cnCDH1, feat.imp: −5.164
mutRB1, feat.imp: 4.837
cnARID1A, feat.imp: 4.266
mutPIK3CA, feat.imp: 2.842
cnEP300, feat.imp: 2.545
cnCHEK2, feat.imp: 2.502
cnNF2, feat.imp: 2.498
mutRNF43, feat.imp: 2.422
IC50 (single drug)
1
2
3
4
mut
wt
Mutation
cn_abnormal
Copy number

Feature combinations

Aberrant copy number PIK3CA and normal copy number CDKN2A (pf-4708671). This combination appears especially potent.

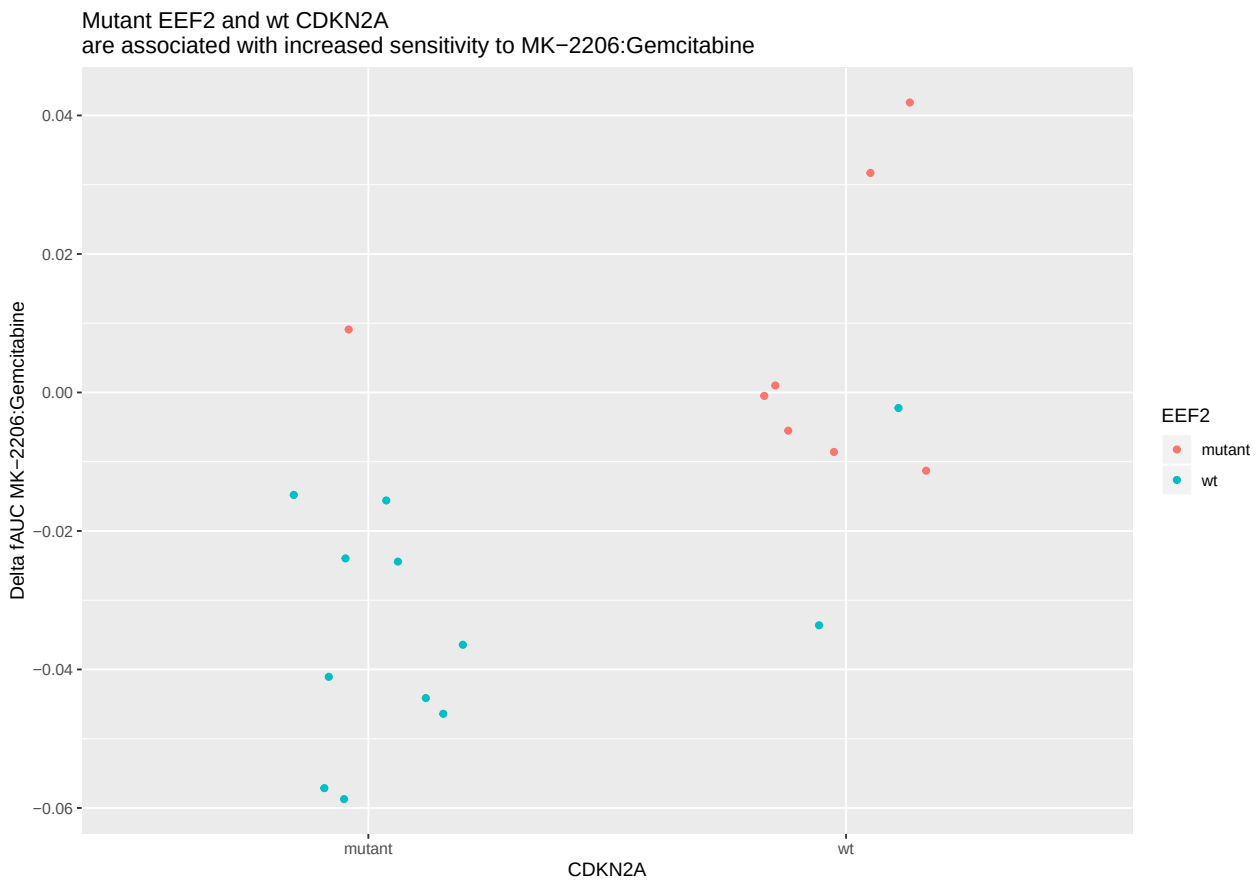
```
cnMatrix[rownames(cnMatrix) %in% c("cnPIK3CA", "cnCDKN2A"),] %>%
  as.data.frame() %>% rownames_to_column("feature") %>%
  gather(key = manuscript_id, value = copynumber, -feature) %>%
  mutate(copynumber = if_else(copynumber == 1, "aberrant", "normal")) %>%
  spread(key = feature, value = copynumber) %>%
  left_join(.,rownames_to_column(as.data.frame(IC50[,"PF-4708671", drop = F]),
                                 "manuscript_id"),
            by="manuscript_id") %>% na.omit() %>%
  select(manuscript_id, CDKN2A = cnCDKN2A, PIK3CA = cnPIK3CA,
         `IC50_PF-4708671` = `PF-4708671`) %>%
  ggplot(aes(x = CDKN2A, y = `IC50_PF-4708671`, color = PIK3CA)) +
  geom_jitter(width = 0.2, height = 0) +
  xlab("CDKN2A copy number") + ylab("IC50 PF-4708671") +
  ggtitle("Aberrant PIK3CA and normal CDKN2A copy number levels\nare associated with increased sensitivi
  labs(color = "PIK3CA\ncopy number")
```




```
ggsave(here("figs", "PF_cnPIK3CA_cnCDKN2A.pdf"), device = "pdf",
       width = 8, height = 5, units = "in")
```

Mutant EEF2 and mutant CDKN2A (fAUC MK-2206:Gemcitabine) EEF2 has a relatively low feature importance score, but is frequently mutated.

```
#featuredata %>%
#  filter(metric == "deltafAUC" & drug == "MK-2206:Gemcitabine" &
#           feature  %in% c("mutEEF2", "mutCDKN2A"))

mutMatrix[rownames(mutMatrix) %in% c("mutEEF2", "mutCDKN2A"),] %>%
  as.data.frame() %>% rownames_to_column("feature") %>%
  gather(key = manuscript_id, value = mutation, -feature) %>%
  mutate(mutation = if_else(mutation > 0, "mutant", "wt")) %>%
  spread(key = feature, value = mutation) %>%
  left_join(.,rownames_to_column(
    as.data.frame(
      median_DELTA_fAUC[,"MK-2206:Gemcitabine", drop = F]
    ), "manuscript_id"), by="manuscript_id") %>% na.omit() %>%
  select(manuscript_id, CDKN2A = mutCDKN2A, EEF2 = mutEEF2,
         `IC50_MK-2206:Gemcitabine` = `MK-2206:Gemcitabine`) %>%
  ggplot(aes(x = CDKN2A, y = `IC50_MK-2206:Gemcitabine`, color = EEF2)) +
  geom_jitter(width = 0.2, height = 0) +
  xlab("CDKN2A") + ylab("Delta fAUC MK-2206:Gemcitabine") +
  ggtitle("Mutant EEF2 and wt CDKN2A \nare associated with increased sensitivity to MK-2206:Gemcitabine
  labs(color = "EEF2")
```

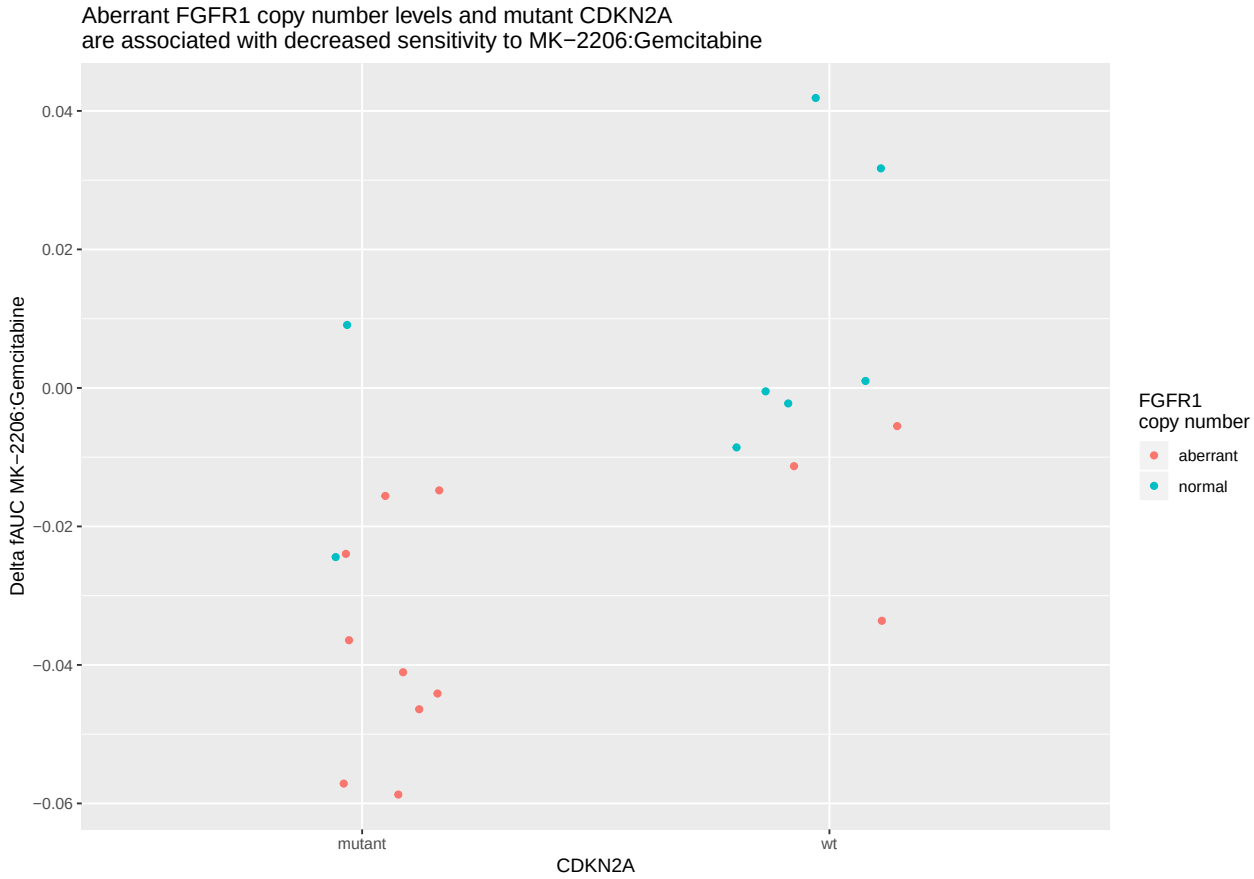



```
ggsave(here("figs", "MK2206:Gem_mutEEF2_mutCDKN2A.pdf"),
       device = "pdf", width = 8, height = 5, units = "in")
```

Copy number aberrant FGFR1 and mutant CDKN2A (fAUC MK-2206:Gemcitabine)

Strong negative coefficients.

```
cnMatrix[rownames(cnMatrix) == "cnFGFR1", ,drop = F] %>%
  as.data.frame() %>% rownames_to_column("feature") %>%
  gather(key = manuscript_id, value = copynumber, -feature) %>%
  mutate(copynumber = if_else(copynumber == 1, "aberrant", "normal")) %>%
  spread(key = feature, value = copynumber) %>%
  left_join(., (mutMatrix[rownames(mutMatrix) == "mutCDKN2A", ,drop = F] %>%
                  as.data.frame() %>% rownames_to_column("feature") %>%
                  gather(key = manuscript_id, value = mutation, -feature) %>%
                  mutate(mutation = if_else(mutation > 0, "mutant", "wt")) %>%
                  spread(key = feature, value = mutation)),
            by = "manuscript_id") %>%
  left_join(.,rownames_to_column(
    as.data.frame(median_DELTA_fAUC[,"MK-2206:Gemcitabine", drop = F]),
    "manuscript_id"),
            by="manuscript_id") %>% na.omit() %>%
  select(manuscript_id, CDKN2A = mutCDKN2A, FGFR1 = cnFGFR1,
         `fAUC_MK-2206:Gemcitabine` = `MK-2206:Gemcitabine`) %>%
  ggplot(aes(x = CDKN2A, y = `fAUC_MK-2206:Gemcitabine`, color = FGFR1)) +
  geom_jitter(width = 0.2, height = 0) +
  xlab("CDKN2A") + ylab("Delta fAUC MK-2206:Gemcitabine") +
  ggtitle("Aberrant FGFR1 copy number levels and mutant CDKN2A \nare associated with decreased sensitivi
  labs(color = "FGFR1\ncopy number")
```

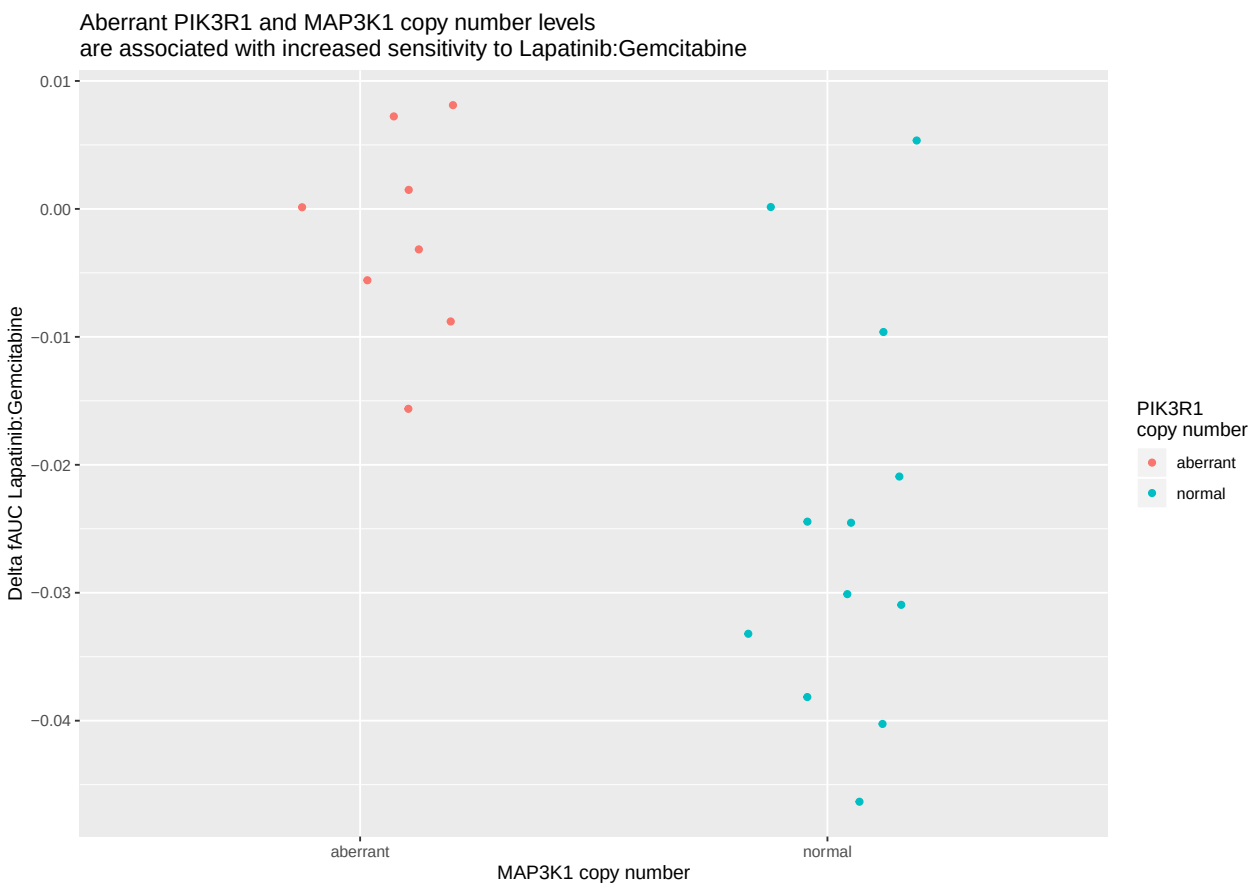



```
ggsave(here("figs", "MK2206:Gem_cnFGFR1_mutCDKN2A.pdf"), device = "pdf",
       width = 8, height = 5, units = "in")
```

Lapatinib/Gemcitabine fAUC MAPK31 and PIK3R1 cn

Complete congruity between both genes!

```
#featuredata %>% filter(metric == "deltafAUC" & drug == "Lapatinib:Gemcitabine" & feature %in% c("cnMAP

cnMatrix[rownames(cnMatrix) %in% c("cnPIK3R1", "cnMAP3K1"),] %>%
  as.data.frame() %>% rownames_to_column("feature") %>%
  gather(key = manuscript_id, value = copynumber, -feature) %>%
  mutate(copynumber = if_else(copynumber == 1, "aberrant", "normal")) %>%
  spread(key = feature, value = copynumber) %>%
  left_join(.,rownames_to_column(as.data.frame(
    median_DELTA_fAUC[,"Lapatinib:Gemcitabine", drop = F]
    ), "manuscript_id"), by="manuscript_id") %>% na.omit() %>%
  select(manuscript_id, PIK3R1 = cnPIK3R1, MAP3K1 = cnMAP3K1,
         `deltafAUC_Lapatinib:Gemcitabine` = `Lapatinib:Gemcitabine`) %>%
  ggplot(aes(x = MAP3K1,
             y = `deltafAUC_Lapatinib:Gemcitabine`, color = PIK3R1)) +
  geom_jitter(width = 0.2, height = 0) +
  xlab("MAP3K1 copy number") + ylab("Delta fAUC Lapatinib:Gemcitabine") +
  ggtitle("Aberrant PIK3R1 and MAP3K1 copy number levels\nare associated with increased sensitivity to
  labs(color = "PIK3R1\ncopy number")
```



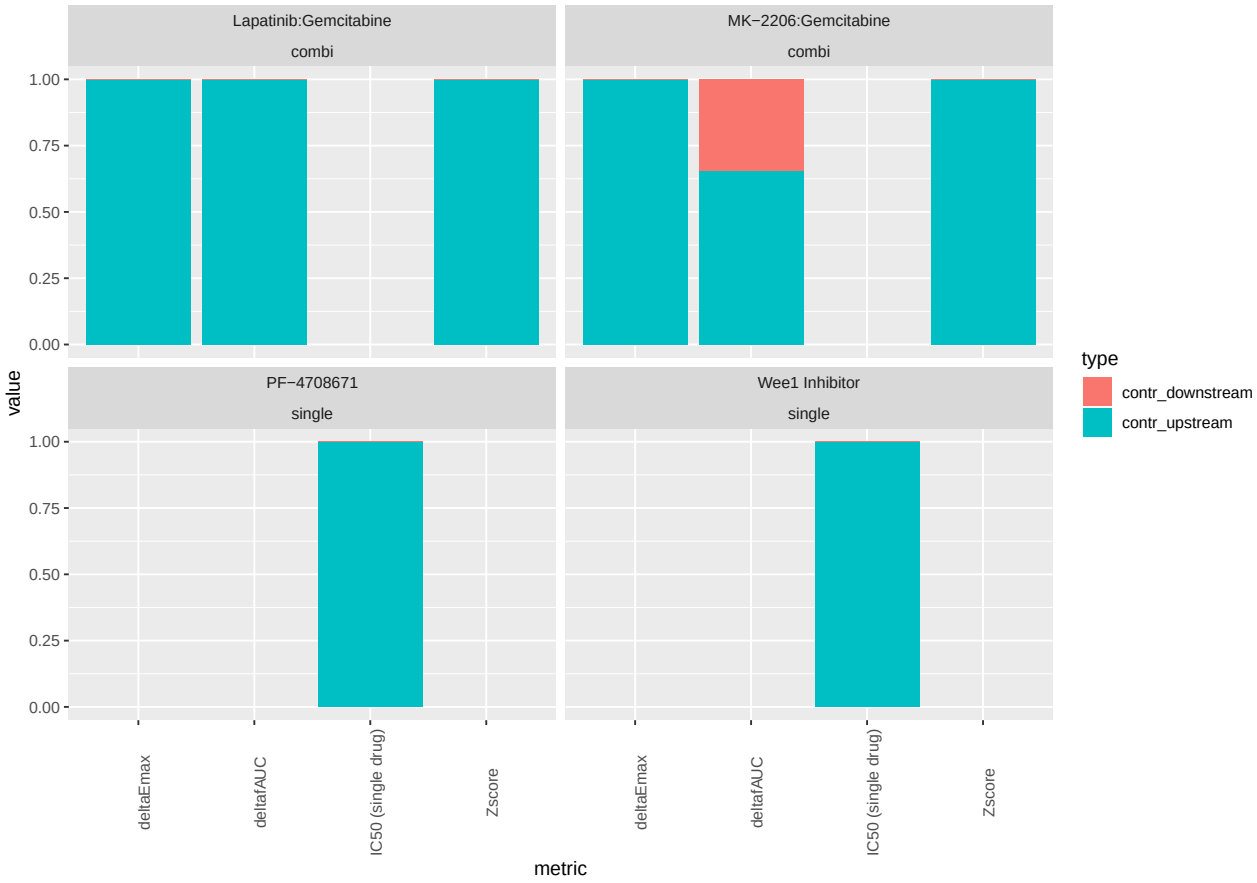

```
ggsave(here("figs", "Lap:Gem_cnPIK3R1_cnMAP3K1.pdf"),
       device = "pdf", width = 8, height = 5, units = "in")
```

Contribution of upstream vs downstream features

Upstream features = mutations, copy number Downstream features = gene expression (RNAseq)

This is only meaningful when the feature selection is not 0.

```
tandemdata %>%
  filter(type %in% c("contr_upstream", "contr_downstream")) %>%
  filter(!is.nan(value)) %>%
  filter(drug %in% best_drugs$drug) %>%
  ggplot(aes(x=metric, y = value, fill=type)) + geom_bar(stat="identity") +
  facet_wrap(~drug + screen, nrow=2) +
  theme(axis.text.x = element_text(angle= 90))
```



Save notebook data

```
save.image(here('reports', 'report_pncr_final.RData'))
```