2024-10-18-G9-HTG-Suppl.R

t

2024-10-18

```
# Version 2024-10-18  Analyses of G9-results and validation cohorts
#
#
#
```

SETUP

```
Sys.setenv(lang = "en_US")
```

Install required packages if missing ——————————————–

```
# Package names
packages <- c("dplyr", "readxl", "ggplot2", "tidyr", "stringr",
              "fmsb", "ggVennDiagram", "ggvenn", "ggrepel")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
##
## Attaching package: 'ggVennDiagram'
```

```
## The following object is masked from 'package:tidyr':
##
##     unite
```

```
## Loading required package: grid
```

```
#
```

Data import ——————————————————————————-

```
# Import of Excel file with data for all 2549 HTG genes
#
# 23 total columns
#
# Recode "NaN" to NA values



suppressWarnings(
  import.data <- read_excel(
    "input/G9_combined_results_all_genes_24JUL2023.xlsx",
    na = c("", "NaN"),
    col_types = c("text", "text", "numeric",
                  "text", "numeric","text", "numeric",
                  "text", "numeric", "text", "numeric",
                  "text", "numeric", "numeric", "numeric",
                  "numeric", "numeric", "numeric",
                  "numeric", "numeric", "numeric",
                  "numeric", "numeric", "numeric",
                  "numeric", "text", "text")
    ),
  classes = "warning"
  )



#
```

Select variables and change names ——————————————————

```
data <- import.data %>%  # select variables of interest
  select(
    c(Genes = variable,
      OR_pCR_durva,
      p_log_pCR_durva,
      OR_pCR_plac,
      p_log_pCR_plac,
      HR_durva,
      p_cox_DDFS_durva,
      HR_plac,
      p_cox_DDFS_plac,
      meanlog2Diff_AB_durva,
      p_AB_durva,
      meanlog2Diff_AB_plac,
      p_AB_plac,
      meanlog2Diff_AC_durva,
      p_AC_durva,
      meanlog2Diff_AC_plac,
      p_AC_plac,
      GeoMx.epi.vs.stromal = `GeoMx PanCK-_PanCK+_A_mean_log2_difference`,
```

```
      p_strom_vs_epith = `GeoMx PanCK-_PanCK+_A_p`,
      `pathway (HTG)`,
      `pathway (hallmark)`
    )
  ) %>%  # split text strings of OR/HR and 95%CIs
  separate_wider_delim(
    cols = c(OR_pCR_durva, OR_pCR_plac, HR_durva, HR_plac),
    delim = " ",
    names_sep = "_",
    too_few = "align_start"
  ) %>%  # extract numeric OR/HR values:
  mutate(
    OR_pCR_durva = as.numeric(OR_pCR_durva_1),
    OR_pCR_plac = as.numeric(OR_pCR_plac_1),
    HR_durva = as.numeric(HR_durva_1),
    HR_plac = as.numeric(HR_plac_1)
    ) %>%  # rename CI variables:
  mutate(
    OR_pCR_durva_CI = OR_pCR_durva_2,
    OR_pCR_plac_CI = OR_pCR_plac_2,
    HR_durva_CI = HR_durva_2,
    HR_plac_CI = HR_plac_2
    ) %>%  # remove renamed duplicate variables:
  select(
    !c(OR_pCR_durva_1, OR_pCR_durva_2,
       OR_pCR_plac_1, OR_pCR_plac_2,
       HR_durva_1, HR_durva_2,
       HR_plac_1, HR_plac_2)
    )




# * ------------------------------------------------------------------------
```

Definitions of gene sets for all analyses ——————————

```
# Geneset containing all genes:

g0.all <- data %>%
  select(Genes) %>% pull()


# Genesets according to pathways:
#
# Pathway information for genes are avaialble from two sources:
#  HTG-Molecular pathway information on HTG-panel in data$`pathway (HTG)`
#  Hallmark pathways in data$`pathway (hallmark)`


g1.immune <- data %>%
  filter(
    str_detect(`pathway (HTG)`, "immuno-oncology")
    | str_detect(
```

```
      `pathway (hallmark)`,
      "allograft rejection|interferon alpha response|interferon gamma response"
    )
  ) %>%
  select(Genes) %>% pull()


g2.proliferation <- data %>%
  filter(
    str_detect(`pathway (HTG)`, "cell cycle")
    | str_detect(
      `pathway (hallmark)`,
      "E2F targets|G2M checkpoint|mitotic spindle"
    )
  ) %>%
  select(Genes) %>% pull()


g3.stromalEMT <- data %>%
  filter(
    str_detect(`pathway (HTG)`, "angiogenesis")
    | str_detect(
      `pathway (hallmark)`,
      "angiogenesis|coagulation|epithelial mesenchymal transition|fatty acid
metabolism|myogenesis"
    )
  ) %>%
  select(Genes) %>% pull()


g4.stromal.NonImmune <- g3.stromalEMT[!(g3.stromalEMT %in% g1.immune)]


g5.DNArepair <- data %>%
  filter(
    str_detect(`pathway (HTG)`, "DNA repair")
    | str_detect(
      `pathway (hallmark)`, "DNA repair"
    )
  ) %>%
  select(Genes) %>% pull()


g6.stemcell <- data %>%
  filter(
    str_detect(`pathway (HTG)`, "stem cells")
  ) %>%
  select(Genes) %>% pull()

# * ------------------------------------------------------------------------
```

Gene assignment to unique class ————————————————————

```
# Ranking of unique assignments based on membership in the above genesets:
# a) immune
```

```
# b) proliferation
# c) stromal-EMT
# d) DNA repair
# e) stem cell
# f) other (not in any of the above genesets)
#
# These unique assignments are used for color coding in scatter plots

gene.class <- data %>% select(Genes) %>%
  mutate(gene.class = "other") %>%
  mutate(gene.class = if_else(Genes %in% g6.stemcell, "stemcell", gene.class)) %>%
  mutate(gene.class = if_else(Genes %in% g5.DNArepair, "DNArepair", gene.class)) %>%
  mutate(gene.class = if_else(Genes %in% g3.stromalEMT, "stromalEMT", gene.class)) %>%
  mutate(gene.class = if_else(Genes %in% g2.proliferation, "proliferation", gene.class))
%>%
  mutate(gene.class = if_else(Genes %in% g1.immune, "immune", gene.class))
```

Gene list for scatter plots ————————————————————

```
# Filter 126 genelist used for G9-pCR-DFS-Scatter plot --------------------
#
# Stringent Selection:
# Select genes with any p-value <0.01
# (either pCR or DDFS in either arm)


g9.pCR.DDFS.scatter <- data %>% left_join(gene.class, by="Genes") %>%
  filter_at(vars(p_log_pCR_durva, p_log_pCR_plac,
                 p_cox_DDFS_durva, p_cox_DDFS_plac),
            any_vars(. <0.01)) %>%
  select(Genes, gene.class)

g9.pCR.DDFS.scatter.genes <- g9.pCR.DDFS.scatter$Genes

# *-----------------------------------------------------------------------
# *-------------------------------------------------------------------------
#
```

ANALYSIS

Gene expression, therapy response and survival ———————

```
#
```

Assemble pCR data for different genelists g0/g1/g2/g3/g4 ———

```
########################################

# Define genelist: g0.all

glist <- g0.all

########################
# Genelist analysis:   #
########################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
resp <- vector(mode="list")

# Total set of genes
resp$all <- data.glist$Genes

# Genes predictive for pCR
resp$durva.good <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva > 1) %>%
  select(Genes) %>% pull()
resp$durva.poor <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva < 1) %>%
  select(Genes) %>% pull()
resp$plac.good <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac > 1) %>%
  select(Genes) %>% pull()
resp$plac.poor <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac < 1) %>%
  select(Genes) %>% pull()
resp$unique <- unique(
  c(resp$durva.good, resp$durva.poor, resp$plac.good, resp$plac.poor)
)

####################

# Save results for genelist

resp.g0.all <- resp

########################################
```

```r
# Define genelist: g1.immune

glist <- g1.immune

########################
# Genelist analysis:   #
########################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
resp <- vector(mode="list")

# Total set of genes
resp$all <- data.glist$Genes

# Genes predictive for pCR
resp$durva.good <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva > 1) %>%
  select(Genes) %>% pull()
resp$durva.poor <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva < 1) %>%
  select(Genes) %>% pull()
resp$plac.good <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac > 1) %>%
  select(Genes) %>% pull()
resp$plac.poor <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac < 1) %>%
  select(Genes) %>% pull()
resp$unique <- unique(
  c(resp$durva.good, resp$durva.poor, resp$plac.good, resp$plac.poor)
)

####################

# Save results for genelist

resp.g1.immune <- resp

########################################


# Define genelist: g2.proliferation

glist <- g2.proliferation

########################
```

```r
# Genelist analysis:   #
#########################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
resp <- vector(mode="list")

# Total set of genes
resp$all <- data.glist$Genes

# Genes predictive for pCR
resp$durva.good <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva > 1) %>%
  select(Genes) %>% pull()
resp$durva.poor <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva < 1) %>%
  select(Genes) %>% pull()
resp$plac.good <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac > 1) %>%
  select(Genes) %>% pull()
resp$plac.poor <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac < 1) %>%
  select(Genes) %>% pull()
resp$unique <- unique(
  c(resp$durva.good, resp$durva.poor, resp$plac.good, resp$plac.poor)
)

####################

# Save results for genelist

resp.g2.proliferation <- resp

########################################


# Define genelist: g3.stromalEMT

glist <- g3.stromalEMT

#########################
# Genelist analysis:   #
#########################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
```

```r
resp <- vector(mode="list")

# Total set of genes
resp$all <- data.glist$Genes

# Genes predictive for pCR
resp$durva.good <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva > 1) %>%
  select(Genes) %>% pull()
resp$durva.poor <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva < 1) %>%
  select(Genes) %>% pull()
resp$plac.good <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac > 1) %>%
  select(Genes) %>% pull()
resp$plac.poor <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac < 1) %>%
  select(Genes) %>% pull()
resp$unique <- unique(
  c(resp$durva.good, resp$durva.poor, resp$plac.good, resp$plac.poor)
)

####################

# Save results for genelist

resp.g3.stromalEMT <- resp

########################################



# Define genelist: g4.stromal.NonImmune

glist <- g4.stromal.NonImmune

########################
# Genelist analysis:   #
########################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
resp <- vector(mode="list")

# Total set of genes
resp$all <- data.glist$Genes

# Genes predictive for pCR
resp$durva.good <- data.glist %>%
```

```
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva > 1) %>%
  select(Genes) %>% pull()
resp$durva.poor <- data.glist %>%
  filter(p_log_pCR_durva <= 0.05) %>%
  filter(OR_pCR_durva < 1) %>%
  select(Genes) %>% pull()
resp$plac.good <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac > 1) %>%
  select(Genes) %>% pull()
resp$plac.poor <- data.glist %>%
  filter(p_log_pCR_plac <= 0.05) %>%
  filter(OR_pCR_plac < 1) %>%
  select(Genes) %>% pull()
resp$unique <- unique(
  c(resp$durva.good, resp$durva.poor, resp$plac.good, resp$plac.poor)
)

####################

# Save results for genelist

resp.g4.stromal.NonImmune <- resp

########################################



# Summarize the numbers of genes predictive for  pCR ---------------------

resp.sum <- cbind(
  summary(resp.g0.all)[,1],
  summary(resp.g1.immune)[,1],
  summary(resp.g2.proliferation)[,1],
  summary(resp.g3.stromalEMT)[,1],
  summary(resp.g4.stromal.NonImmune)[,1]
)
colnames(resp.sum) <- c("g0.all", "g1.immune", "g2.proliferation",
                        "g3.stromalEMT", "g4.stromal.NonImmune")
rn <- rownames(resp.sum)
resp.sum <- apply(resp.sum, 2, as.numeric)
rownames(resp.sum) <- rn

# Numbers of genes predictive for pCR in each geneset:
resp.sum
```

```
##            g0.all g1.immune g2.proliferation g3.stromalEMT g4.stromal.NonImmune
## all          2549       431              275           331                   264
## durva.good    225        74               72            13                     5
## durva.poor     92         7                4            50                    48
## plac.good     134        36               39             8                     3
## plac.poor      14         2                0             8                     6
## unique        422       114               91            74                    57
```

```
# Calculate as percentage
resp.sum.perc <- round(resp.sum/resp.sum[1,] * 100, 2)
# Percentage of genes predictive for pCR in each geneset:
resp.sum.perc
```

```
##            g0.all g1.immune g2.proliferation g3.stromalEMT g4.stromal.NonImmune
## all        100.00    100.00           100.00        100.00                100.00
## durva.good  52.20     26.91            21.75          4.92                 0.20
## durva.poor  33.45      2.11             1.52          1.96                11.14
## plac.good   40.48     13.64             1.53          1.86                 1.09
## plac.poor    5.30      0.08             0.00          2.91                 1.81
## unique      16.56     26.45            33.09         22.36                21.59
```

```
write.csv2(resp.sum, file="out/resp_summary.csv")
write.csv2(resp.sum.perc, file="out/resp_summary_perc.csv")

# Venn diagrams for pCR ------------------------------------------------

library("ggvenn")  # color by category
library("ggVennDiagram")  # color by number of genes



#
```

Figure 3a (and Fig 4a)

```
#

# g0.all genes

genes <- resp.g0.all
title <- "Genes predictive for pCR among all genes\n"

# Venn diagram colored by category
#  (exclude first category of "all" genes)

fn <- "out/Fig3a_Venn-pCR.pdf"
pdf(file = fn)
ggvenn(
  genes[c(3, 2, 4, 5)],
  fill_color = c("#0073C2FF", "#EFC000FF", "#868686FF", "#CD534CFF"),
  stroke_size = 0.5,
  set_name_size = 4
) +
  ggtitle(title)
dev.off()
```

```
## png
##   2
```

```
####################

#  Venn diagrams for pCR Separately for Genesets
#
```

Figure 4a and Fig 4c

```r
#

# g1.immune genes

genes <- resp.g1.immune
title <- "Genes predictive for pCR among g1.immune genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4a-immune_Venn-pCR.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

####################



# g2.proliferation genes

genes <- resp.g2.proliferation
title <- "Genes predictive for pCR among g2.proliferation genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4a-prolif_Venn-pCR.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

####################



# g3.stromalEMT genes

genes <- resp.g3.stromalEMT
title <- "Genes predictive for pCR among g3.stromalEMT genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4a-stromal_Venn-pCR.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
```

```
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

####################
```

```
##############################################################################
```

Assemble DDFS data for different genelists g0/g1/g2/g3/g4 ———-

```
########################################

# Define genelist: g0.all

glist <- g0.all

##############################
# Genelist DDFS analysis:   #
##############################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
ddfs <- vector(mode="list")

# Total set of genes
ddfs$all <- data.glist$Genes

# Genes predictive for improved DDFS
ddfs$durva.good <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva < 1)  %>%
  select(Genes) %>% pull()
ddfs$durva.poor <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva > 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.good <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac < 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.poor <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac > 1)  %>%
  select(Genes) %>% pull()
ddfs$unique <- unique(
  c(ddfs$durva.good, ddfs$durva.poor, ddfs$plac.good, ddfs$plac.poor)
)

####################

# Save results for genelist

ddfs.g0.all <- ddfs

########################################

# Define genelist: g1.immune

glist <- g1.immune
```

```r
##############################
# Genelist DDFS analysis:   #
##############################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
ddfs <- vector(mode="list")

# Total set of genes
ddfs$all <- data.glist$Genes

# Genes predictive for improved DDFS
ddfs$durva.good <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva < 1)  %>%
  select(Genes) %>% pull()
ddfs$durva.poor <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva > 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.good <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac < 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.poor <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac > 1)  %>%
  select(Genes) %>% pull()
ddfs$unique <- unique(
  c(ddfs$durva.good, ddfs$durva.poor, ddfs$plac.good, ddfs$plac.poor)
)

####################

# Save results for genelist

ddfs.g1.immune <- ddfs

########################################

# Define genelist: g2.proliferation

glist <- g2.proliferation

##############################
# Genelist DDFS analysis:   #
##############################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)
```

```r
# Prepare a list object for results:
ddfs <- vector(mode="list")

# Total set of genes
ddfs$all <- data.glist$Genes

# Genes predictive for improved DDFS
ddfs$durva.good <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva < 1)  %>%
  select(Genes) %>% pull()
ddfs$durva.poor <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva > 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.good <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac < 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.poor <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac > 1)  %>%
  select(Genes) %>% pull()
ddfs$unique <- unique(
  c(ddfs$durva.good, ddfs$durva.poor, ddfs$plac.good, ddfs$plac.poor)
)

####################

# Save results for genelist

ddfs.g2.proliferation <- ddfs

########################################

# Define genelist: g3.stromalEMT

glist <- g3.stromalEMT

##############################
# Genelist DDFS analysis:   #
##############################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
ddfs <- vector(mode="list")

# Total set of genes
ddfs$all <- data.glist$Genes

# Genes predictive for improved DDFS
ddfs$durva.good <- data.glist %>%
```

```r
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva < 1)  %>%
  select(Genes) %>% pull()
ddfs$durva.poor <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva > 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.good <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac < 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.poor <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac > 1)  %>%
  select(Genes) %>% pull()
ddfs$unique <- unique(
  c(ddfs$durva.good, ddfs$durva.poor, ddfs$plac.good, ddfs$plac.poor)
)

####################

# Save results for genelist

ddfs.g3.stromalEMT <- ddfs

########################################

# Define genelist: g4.stromal.NonImmune

glist <- g4.stromal.NonImmune

##############################
# Genelist DDFS analysis:   #
##############################

# Filter data for genelist
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
ddfs <- vector(mode="list")

# Total set of genes
ddfs$all <- data.glist$Genes

# Genes predictive for improved DDFS
ddfs$durva.good <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva < 1)  %>%
  select(Genes) %>% pull()
ddfs$durva.poor <- data.glist %>%
  filter(p_cox_DDFS_durva <= 0.05) %>%
  filter(HR_durva > 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.good <- data.glist %>%
```

```r
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac < 1)  %>%
  select(Genes) %>% pull()
ddfs$plac.poor <- data.glist %>%
  filter(p_cox_DDFS_plac <= 0.05) %>%
  filter(HR_plac > 1)  %>%
  select(Genes) %>% pull()
ddfs$unique <- unique(
  c(ddfs$durva.good, ddfs$durva.poor, ddfs$plac.good, ddfs$plac.poor)
)

####################

# Save results for genelist

ddfs.g4.stromal.NonImmune <- ddfs

########################################



# Summarize results on DDFS ----------------------------------------------

ddfs.sum <- cbind(
  summary(ddfs.g0.all)[,1],
  summary(ddfs.g1.immune)[,1],
  summary(ddfs.g2.proliferation)[,1],
  summary(ddfs.g3.stromalEMT)[,1],
  summary(ddfs.g4.stromal.NonImmune)[,1]
)
colnames(ddfs.sum) <- c("g0.all", "g1.immune", "g2.proliferation",
                        "g3.stromalEMT", "g4.stromal.NonImmune")
rn <- rownames(ddfs.sum)
ddfs.sum <- apply(ddfs.sum, 2, as.numeric)
rownames(ddfs.sum) <- rn

ddfs.sum
```

```
##            g0.all g1.immune g2.proliferation g3.stromalEMT g4.stromal.NonImmune
## all          2549       431              275           331                   264
## durva.good    169        45               18            21                    13
## durva.poor     12         0                2             2                     2
## plac.good      15         4                4             2                     0
## plac.poor      69         7                8            11                     9
## unique        262        56               31            36                    24
```

```r
# Calculate as percentage
ddfs.sum.perc <- round(ddfs.sum/ddfs.sum[1,] * 100, 2)
# Percentage of genes prognostic for DDFS in each geneset:
ddfs.sum.perc
```

```
##            g0.all g1.immune g2.proliferation g3.stromalEMT g4.stromal.NonImmune
## all        100.00    100.00           100.00        100.00                100.00
## durva.good  39.21     16.36             5.44          7.95                  0.51
## durva.poor   4.36      0.00             0.76          0.08                  0.46
```

```
## plac.good    4.53      1.52             0.16          0.46                 0.00
## plac.poor   26.14      0.27             1.86          4.00                 2.72
## unique      10.28     12.99            11.27         10.88                 9.09
```

```
write.csv2(ddfs.sum, file="out/ddfs_summary.csv")
write.csv2(ddfs.sum.perc, file="out/ddfs_summary_perc.csv")
```

```
# Venn diagrams for DDFS ------------------------------------------------

library("ggvenn")  # color by category
library("ggVennDiagram")  # color by number of genes


# g0.all genes

#
```

Figure 3b (and Fig 4c)

```
#

genes <- ddfs.g0.all
title <- "Genes prognostic for DDFS among all genes\n"

# Venn diagram colored by category
#  (exclude first category of "all" genes)

fn <- "out/Fig3b_Venn-DDFS.pdf"
pdf(file = fn)
ggvenn(
  genes[c(3, 2, 4, 5)],
  fill_color = c("#0073C2FF", "#EFC000FF", "#868686FF", "#CD534CFF"),
  stroke_size = 0.5,
  set_name_size = 4
) +
  ggtitle(title)
dev.off()

## png
##   2

####################


#  Venn diagrams for DDFS Separately for Genesets
#
```

Figure 4c

```
#

# g1.immune genes
```

```r
genes <- ddfs.g1.immune
title <- "Genes prognostic for DDFS among g1.immune genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4c-immune_Venn-DDFS.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

####################


# g2.proliferation genes

genes <- ddfs.g2.proliferation
title <- "Genes prognostic for DDFS among g2.proliferation genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4c-prolif_Venn-DDFS.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

####################


# g3.stromalEMT genes

genes <- ddfs.g3.stromalEMT
title <- "Genes prognostic for DDFS among g3.stromalEMT genes\n"

# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig4c-stromal_Venn-DDFS.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 2, 4, 5)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()
```

```
## png
##   2

####################
```

```
# Venn diagrams  pCR vs DDFS ----------------------------------------------

library("ggvenn")  # color by category
library("ggVennDiagram")  # color by number of genes


##################

# g0.all genes

resp <- resp.g0.all
ddfs <- ddfs.g0.all
title <-
  "Genes predictive for pCR and prognostic for DDFS (among all genes)\n"


# Prepare a list object with all signif genes combined (good+poor):
resp.ddfs <- vector(mode = "list")
resp.ddfs$resp.durva <- c(resp$durva.good, resp$durva.poor)
resp.ddfs$resp.plac <- c(resp$plac.good, resp$plac.poor)
resp.ddfs$ddfs.durva <- c(ddfs$durva.good, ddfs$durva.poor)
resp.ddfs$ddfs.plac <- c(ddfs$plac.good, ddfs$plac.poor)

genes <- resp.ddfs




#
```

Figure 3d

```
#
# Venn diagram colored by number of genes
#  (exclude first category of "all" genes)

fn <- "out/Fig3d_Venn-pCR-DDFS.pdf"
pdf(file = fn)
ggVennDiagram(genes[c(3, 1, 2, 4)], label_alpha = 0) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle(title)
dev.off()

## png
##   2

################################################################################
```

```
#
```

Figure 6a and Fig 6b:

Scatter plot figures pCR vs DDFS

```
#



library(dplyr)
library(ggplot2)
library(ggrepel)

# Scatter plot for genes with any p-value <0.01
#   (either pCR or DDFS in either arm)
#
# These genes are assembled in geneset "g9.pCR.DDFS.scatter.genes"
#  (see section "Data-Import and data table generation" above)
#
# Color genes by gene.class  (unique assignment, see first section)
#




# Select geneset:

gene.set.name <- "g9.pCR.DDFS.scatter.genes"

gene.set <- get(gene.set.name)


plotdat <- data %>% left_join(gene.class, by="Genes") %>%
  filter(Genes %in% gene.set) %>%
  select(Genes, gene.class, OR_pCR_durva, HR_durva,
         p_log_pCR_durva, p_cox_DDFS_durva,
         OR_pCR_plac, HR_plac,
         p_log_pCR_plac, p_cox_DDFS_plac) %>%
  filter(Genes != "PIK3C2A")  # exclude outlier


OR.min <- min(c(plotdat$OR_pCR_durva, plotdat$OR_pCR_plac))
OR.max <- max(c(plotdat$OR_pCR_durva, plotdat$OR_pCR_plac))
HR.min <- min(c(plotdat$HR_durva, plotdat$HR_plac))
HR.max <- max(c(plotdat$HR_durva, plotdat$HR_plac))


# Define color palette ("Dark2" from RColorBrewer):

# 6 groups: DNArepair, immune, other, proliferation, stemcell, stromalEMT
colpal <- c("#1B9E77" ,"#7570B3", "#666666",
            "#E7298A" ,"#66A61E", "#A6761D")
```

```r
# Plot to pdf files


# Durvalumab arm  pCR vs DDFS
#
fn <- "out/Fig6a_Scatter-pCR-DDFS-Durva.pdf"
pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_pCR_durva, 1/HR_durva, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 0.8) +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("Durvalumab arm") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
dev.off()

## png
##   2

# Placebo arm  pCR vs DDFS
#
fn <- "out/Fig6b_Scatter-pCR-DDFS-Plac.pdf"
pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_pCR_plac, 1/HR_plac, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 1) +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("Placebo arm") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")

## Warning: ggrepel: 2 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

```r
dev.off()

## png
##   2

###############

###  Additional figures including p-value information:
###  Filled circles size based on log10(p-value for pCR)
###  Open circles  size based on log10(p-value for DDFS)


# Durvalumab arm  pCR vs DDFS
#
fn <- "out/Suppl_Scatter-pCR-DDFS-Durva-pVal.pdf"
pdf(file = fn, width = 11, height = 8.5)
ggplot(plotdat) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(aes(x = OR_pCR_durva, y = 1/HR_durva, color = gene.class,
                 size = -log10(p_log_pCR_durva)), alpha = 0.5) +
  geom_point(aes(x = OR_pCR_durva, y = 1/HR_durva, color = gene.class,
                 size = -log10(p_cox_DDFS_durva)), shape = 1) +
  scale_size(range = c(.1, 14), name = "-log10(P-value) [pCR, DDFS]") +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(x = OR_pCR_durva, y = 1/HR_durva,
                      label = Genes, color = gene.class),
                  size = 3,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("Durvalumab arm \n(filled circles: pCR-p-Value / open circles: DDFS-p-Value") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
dev.off()

## png
##   2

# Placebo arm  pCR vs DDFS
#
fn <- "out/Suppl_Scatter-pCR-DDFS-Plac-pval.pdf"
pdf(file = fn, width = 11, height = 8.5)
ggplot(plotdat) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(aes(x = OR_pCR_plac, y = 1/HR_plac, color = gene.class,
                 size = -log10(p_log_pCR_plac)), alpha = 0.5) +
  geom_point(aes(x = OR_pCR_plac, y = 1/HR_plac, color = gene.class,
                 size = -log10(p_cox_DDFS_plac)), shape = 1) +
  scale_size(range = c(.1, 14), name = "-log10(P-value) [pCR, DDFS]") +
```

```
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(x = OR_pCR_plac, y = 1/HR_plac,
                      label = Genes, color = gene.class),
                  size = 3,
                  box.padding   = 0.3,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color ='darkgrey', show.legend = F) +
  ggtitle("Placebo arm \n(filled circles: pCR-p-Value / open circles: DDFS-p-Value") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
dev.off()

## png
##   2

################################################################################
```

```
#
```

Alteration of stromal and epithelial gene expression by durvalumab

```
#

# Analysis expression changes during window (A vs. B)
#  for stromal and epithelial genes


# Define genelist: Select genes from GeoMx significantly enriched
#  in epithelial compartment
glist <- data %>% filter(p_strom_vs_epith <= 0.05) %>%
  filter(GeoMx.epi.vs.stromal > 0) %>% select(Genes) %>% pull()

##########################
# Window-AB analysis:    #
##########################

# Filter data using genelist:
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
longAB <- vector(mode="list", length=5 )
names(longAB) = c("all", "durva.up", "durva.down",
                  "plac.up", "plac.down")

# Total set of genes
longAB$all <- data.glist %>% select(Genes) %>% pull()
# Durvalumab arm AB changes
longAB$durva.up <- data.glist %>%
  filter(p_AB_durva <= 0.05 & meanlog2Diff_AB_durva > 0) %>%
  select(Genes) %>% pull()
longAB$durva.down <- data.glist %>%
  filter(p_AB_durva <= 0.05 & meanlog2Diff_AB_durva < 0) %>%
  select(Genes) %>% pull()
# Placebo arm AB changes
longAB$plac.up <- data.glist %>%
  filter(p_AB_plac <= 0.05 & meanlog2Diff_AB_plac > 0) %>%
  select(Genes) %>% pull()
longAB$plac.down <- data.glist %>%
  filter(p_AB_plac <= 0.05 & meanlog2Diff_AB_plac < 0) %>%
  select(Genes) %>% pull()

#############

# Save results for epi:

longAB.GeoMx.epi <- longAB

#############



# Define genelist: Select genes from GeoMx significantly enriched
```

```r
#  in stromal compartment
glist <- data %>% filter(p_strom_vs_epith <= 0.05) %>%
  filter(GeoMx.epi.vs.stromal < 0) %>%
  select(Genes) %>% pull()

##########################
# Window-AB analysis:    #
##########################

# Filter data using genelist:
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
longAB <- vector(mode="list", length=5 )
names(longAB) = c("all", "durva.up", "durva.down",
                  "plac.up", "plac.down")

# Total set of genes
longAB$all <- data.glist %>% select(Genes) %>% pull()
# Durvalumab arm AB changes
longAB$durva.up <- data.glist %>%
  filter(p_AB_durva <= 0.05 & meanlog2Diff_AB_durva > 0) %>%
  select(Genes) %>% pull()
longAB$durva.down <- data.glist %>%
  filter(p_AB_durva <= 0.05 & meanlog2Diff_AB_durva < 0) %>%
  select(Genes) %>% pull()
# Placebo arm AB changes
longAB$plac.up <- data.glist %>%
  filter(p_AB_plac <= 0.05 & meanlog2Diff_AB_plac > 0) %>%
  select(Genes) %>% pull()
longAB$plac.down <- data.glist %>%
  filter(p_AB_plac <= 0.05 & meanlog2Diff_AB_plac < 0) %>%
  select(Genes) %>% pull()

##############

# Save results for stroma:

longAB.GeoMx.stroma <- longAB

##############



# Summarize the numbers of genes changed:

change.AB <- tibble(
  Genes = c("up", "down", "unchanged"),
  Durva.epi = c(
    length(longAB.GeoMx.epi$durva.up),
    length(longAB.GeoMx.epi$durva.down),
    length(longAB.GeoMx.epi$all)
    - length(longAB.GeoMx.epi$durva.up)
    - length(longAB.GeoMx.epi$durva.down)
  ),
```

```
  Plac.epi = c(
    length(longAB.GeoMx.epi$plac.up),
    length(longAB.GeoMx.epi$plac.down),
    length(longAB.GeoMx.epi$all)
    - length(longAB.GeoMx.epi$plac.up)
    - length(longAB.GeoMx.epi$plac.down)
  ),
  Durva.stromal =  c(
    length(longAB.GeoMx.stroma$durva.up),
    length(longAB.GeoMx.stroma$durva.down),
    length(longAB.GeoMx.stroma$all)
    - length(longAB.GeoMx.stroma$durva.up)
    - length(longAB.GeoMx.stroma$durva.down)
  ),
  Plac.stromal = c(
    length(longAB.GeoMx.stroma$plac.up),
    length(longAB.GeoMx.stroma$plac.down),
    length(longAB.GeoMx.stroma$all)
    - length(longAB.GeoMx.stroma$plac.up)
    - length(longAB.GeoMx.stroma$plac.down)
  )
)

change.AB
```

```
## # A tibble: 3 × 5
##   Genes     Durva.epi Plac.epi Durva.stromal Plac.stromal
##   <chr>         <int>    <int>         <int>        <int>
## 1 up                0        8            54            1
## 2 down             47        1             2           10
## 3 unchanged       116      154           158          203
```

```
# Plot Change_A_vs_B_stromal_epithelial.pdf ---------------------------------

library(tidyr)
df <- pivot_longer(change.AB, cols = -1)
colnames(df) <- c("Genes", "Treatment arm - compartment", "value")


fn <- "out/Change_A_vs_B_stromal_epithelial.pdf"

pdf(file = fn, width = 5.5, height = 6)
ggplot(df,
       aes(
         x = Genes,
         y = `Treatment arm - compartment`,
         colour = Genes,
         size = value
       )) +
  geom_point(shape = 19, stroke = 0) +
  geom_text(aes(label = value),
            colour = "white",
            size = 4) +
  scale_x_discrete(position = "top") +
  scale_size_continuous(range = c(0, 60)) + #adjusted to observ-1 (below)
```

```
  scale_color_manual(values = c("#79AF97FF", "#80796BFF", "#B24745FF")) +
  labs(x = NULL, y = NULL) +
  theme(
    legend.position = "none",
    panel.background = element_blank(),
    panel.grid = element_blank(),
    axis.ticks = element_blank()
  ) +
  ggtitle("Gene expression changes \nduring window (A vs. B) \nfor stromal and epithelial
genes\n")
dev.off()

## png
##   2

# *------------------------------------------------------------------
```

```r
# Analysis of expression changes during chemotherapy (A vs. C)
#  for stromal and epithelial genes


# Define genelist: Select genes from GeoMx significantly enriched
#  in epithelial compartment
glist <- data %>% filter(p_strom_vs_epith <= 0.05) %>%
  filter(GeoMx.epi.vs.stromal > 0) %>%
  select(Genes) %>% pull()

############################
# Chemo-AC analysis:    #
############################

# Filter data using genelist:
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
longAC <- vector(mode = "list", length = 5)
names(longAC) = c("all", "durva.up", "durva.down",
                  "plac.up", "plac.down")

# Total set of genes
longAC$all <- data.glist %>% select(Genes) %>% pull()
# Durvalumab arm AC changes
longAC$durva.up <- data.glist %>%
  filter(p_AC_durva <= 0.05 & meanlog2Diff_AC_durva > 0) %>%
  select(Genes) %>% pull()
longAC$durva.down <- data.glist %>%
  filter(p_AC_durva <= 0.05 & meanlog2Diff_AC_durva < 0) %>%
  select(Genes) %>% pull()
# Placebo arm stromal genes longACudinal changes
longAC$plac.up <- data.glist %>%
  filter(p_AC_plac <= 0.05 & meanlog2Diff_AC_plac > 0) %>%
  select(Genes) %>% pull()
longAC$plac.down <- data.glist %>%
  filter(p_AC_plac <= 0.05 & meanlog2Diff_AC_plac < 0) %>%
  select(Genes) %>% pull()

##############

# Save results for epi:

longAC.GeoMx.epi <- longAC

##############


# Define genelist: Select genes from GeoMx significantly enriched
#  in stromal compartment
glist <- data %>% filter(p_strom_vs_epith <= 0.05) %>%
  filter(GeoMx.epi.vs.stromal < 0) %>%
  select(Genes) %>% pull()
```

```r
##########################
# Chemo-AC analysis:    #
##########################

# Filter data using genelist:
data.glist <- data %>% filter(Genes %in% glist)

# Prepare a list object for results:
longAC <- vector(mode = "list", length = 5)
names(longAC) = c("all", "durva.up", "durva.down",
                  "plac.up", "plac.down")

# Total set of genes
longAC$all <- data.glist %>% select(Genes) %>% pull()
# Durvalumab arm AC changes
longAC$durva.up <- data.glist %>%
  filter(p_AC_durva <= 0.05 & meanlog2Diff_AC_durva > 0) %>%
  select(Genes) %>% pull()
longAC$durva.down <- data.glist %>%
  filter(p_AC_durva <= 0.05 & meanlog2Diff_AC_durva < 0) %>%
  select(Genes) %>% pull()
# Placebo arm stromal genes longACudinal changes
longAC$plac.up <- data.glist %>%
  filter(p_AC_plac <= 0.05 & meanlog2Diff_AC_plac > 0) %>%
  select(Genes) %>% pull()
longAC$plac.down <- data.glist %>%
  filter(p_AC_plac <= 0.05 & meanlog2Diff_AC_plac < 0) %>%
  select(Genes) %>% pull()

##############

# Save results for stroma:

longAC.GeoMx.stroma <- longAC

##############

# Summarize the numbers of genes changed:

change.AC <- tibble(
  Genes = c("up", "down", "unchanged"),
  Durva.epi = c(
    length(longAC.GeoMx.epi$durva.up),
    length(longAC.GeoMx.epi$durva.down),
    length(longAC.GeoMx.epi$all)
    - length(longAC.GeoMx.epi$durva.up)
    - length(longAC.GeoMx.epi$durva.down)
  ),
  Plac.epi = c(
    length(longAC.GeoMx.epi$plac.up),
    length(longAC.GeoMx.epi$plac.down),
    length(longAC.GeoMx.epi$all)
    - length(longAC.GeoMx.epi$plac.up)
    - length(longAC.GeoMx.epi$plac.down)
  ),
```

```
  Durva.stromal =  c(
    length(longAC.GeoMx.stroma$durva.up),
    length(longAC.GeoMx.stroma$durva.down),
    length(longAC.GeoMx.stroma$all)
    - length(longAC.GeoMx.stroma$durva.up)
    - length(longAC.GeoMx.stroma$durva.down)
  ),
  Plac.stromal = c(
    length(longAC.GeoMx.stroma$plac.up),
    length(longAC.GeoMx.stroma$plac.down),
    length(longAC.GeoMx.stroma$all)
    - length(longAC.GeoMx.stroma$plac.up)
    - length(longAC.GeoMx.stroma$plac.down)
  )
)

change.AC
```

```
## # A tibble: 3 × 5
##   Genes     Durva.epi Plac.epi Durva.stromal Plac.stromal
##   <chr>         <int>    <int>         <int>        <int>
## 1 up                1        2            35           22
## 2 down             64       44            18           32
## 3 unchanged        98      117           161          160
```

```
# Plot Change_A_vs_C_stromal_epithelial.pdf ---------------------------------

library(tidyr)
df <- pivot_longer(change.AC, cols = -1)
colnames(df) <- c("Genes", "Treatment arm - compartment", "value")


fn <- "out/Change_A_vs_C_stromal_epithelial.pdf"

pdf(file = fn, width = 5.5, height = 6)
ggplot(df,
       aes(
         x = Genes,
         y = `Treatment arm - compartment`,
         colour = Genes,
         size = value
       )) +
  geom_point(shape = 19, stroke = 1) +
  geom_text(aes(label = value),
            colour = "white",
            size = 4) +
  scale_x_discrete(position = "top") +
  scale_size_continuous(range = c(3, 55)) + # Adjusted to sizes observation-1 !
  scale_color_manual(values = c("#79AF97FF", "#80796BFF", "#B24745FF")) +
  labs(x = NULL, y = NULL) +
  theme(
    legend.position = "none",
    panel.background = element_blank(),
    panel.grid = element_blank(),
    axis.ticks = element_blank()
```

```
  ) +
  ggtitle(
    "Gene expression changes during chemotherapy (A vs. C) \nfor stromal and epithelial
genes\n"
  )
dev.off()

## png
##   2

# *------------------------------------------------------------------
```

VALIDATION ANALYSIS

Assemble validation RNA-Seq datasets

```
# Import log2 RNAseq data for available HTG genes of 3 validation datasets

load("input/valid_datasets_HTG.RData")


# MEDI (NCT02489448):
#  medi.htg
#  medi.htg.cli.dat
# Source:
# 55 samples with pCR information
# RNAseq data for 2508 of 2549 HTG genes


# BRTN (NCT02032277):
#  brtn.htg
#  brtn.htg.cli.dat
# Source: GSE164458_BrighTNess_RNAseq_log2_Processed_ASTOR.txt
# 482 samples with pCR information
# RNAseq data for 2505 of 2549 HTG genes


# SCANB (NCT02306096) 326 TNBC:
#  scanb.htg
#  scanb.htg.cli.dat
# Source:
# 326 samples with survival information
# RNAseq data for 2540 of 2549 HTG genes



# Add pCR status as T/F to cli.dat:
medi.htg.cli.dat$pCR <- medi.htg.cli.dat$pathologic.complete.response == "Yes"
brtn.htg.cli.dat$pCR <- brtn.htg.cli.dat$pathologic_complete_response == "pCR"
```

Assemble pCR-ORs and DFS-HRs for validation datasets ————————————————

```
############################
###     pCR-datasets:    ###
############################


#########################################################################
# Calculate Odds Ratios for all 2549 HTG genes in validation datasets  ######
#########################################################################

val.genes <- g0.all  # all 2549 genes from HTG panel

###  Dataset medi.htg.Rseq

# Define dataset and genes
```

```r
coh <- "medi"
rseq <- medi.htg.Rseq
cli.dat <- medi.htg.cli.dat
cli.dat$resp <-
  as.numeric(cli.dat$pCR) # pCR=1, RD=0
gen.set <- val.genes


# Select data (SCALED)
dat <- rseq[row.names(rseq) %in% gen.set,
            colnames(rseq) %in% rownames(cli.dat)]
dat <- dat[order(rownames(dat)), ] # order by gene name
dat <- na.omit(t(scale(t(dat))))   # scale RNA-Seq for each gene
pCR <- cli.dat$resp

# define dataframe for results
ORtab <- data.frame(gene=NA, OR=NA, CI.lo=NA, CI.up=NA, p.value=NA)

# loop for odds ratios of pCR for all genes
stopifnot(colnames(dat) == rownames(cli.dat))
suppressWarnings(suppressMessages(
  for (i in 1:nrow(dat)) {
    mod <-
      glm(pCR ~ dat[i, ],
          family = binomial(link = "logit"),
          na.action = na.pass)   # logit model  of pCR
    ORtab[i, 1] <- rownames(dat)[i]  # gene name
    ORtab[i, 2] <- exp(mod$coefficients)[2]  # odds ratio
    ORtab[i, 3:4] <-  exp(confint(mod))[2, ] # CI
    ORtab[i, 5] <- summary(mod)$coefficients[2, 'Pr(>|z|)'] # pval
  }
))

# Save the result as named variable  (SCALED)
assign(paste0("ORtab.",coh,".scaled"),ORtab)

#####################################################

###  Dataset brtn.htg.Rseq

# Define dataset and genes
coh <- "brtn"
rseq <- brtn.htg.Rseq
cli.dat <- brtn.htg.cli.dat
cli.dat$resp <- as.numeric(cli.dat$pCR)  # code pCR=1, RD=0
gen.set <- val.genes

# Select data (SCALED)
dat <- rseq[row.names(rseq) %in% gen.set,
            colnames(rseq) %in% rownames(cli.dat)]
dat <- dat[order(rownames(dat)), ] # order by gene name
dat <- na.omit(t(scale(t(dat))))   # scale RNA-Seq for each gene
pCR <- cli.dat$resp
```

```r
# define dataframe for results
ORtab <- data.frame(gene=NA, OR=NA, CI.lo=NA, CI.up=NA, p.value=NA)

# loop for odds ratios of pCR for all genes
stopifnot(colnames(dat) == rownames(cli.dat))
suppressWarnings(suppressMessages(
  for (i in 1:nrow(dat)) {
    mod <-
      glm(pCR ~ dat[i, ],
          family = binomial(link = "logit"),
          na.action = na.pass)   # logit model  of pCR
    ORtab[i, 1] <- rownames(dat)[i]  # gene name
    ORtab[i, 2] <- exp(mod$coefficients)[2]  # odds ratio
    ORtab[i, 3:4] <-  exp(confint(mod))[2, ] # CI
    ORtab[i, 5] <- summary(mod)$coefficients[2, 'Pr(>|z|)'] # pval
  }
))

# Save the result as named variable  (SCALED)
assign(paste0("ORtab.",coh,".scaled"),ORtab)


##############################################################################
#    Assemble combined pCR-OR-tables for cohorts                      #######
##############################################################################

# Assemble OR for G9

ORtab.g9 <- data %>%
  select(
    Genes,
    OR_pCR_durva,
    OR_pCR_durva_CI,
    p_log_pCR_durva,
    OR_pCR_plac,
    OR_pCR_plac_CI,
    p_log_pCR_plac
  )

####  Combine OR tables for datasets
####  by left-joining on  ORtab.g9  the data
####  of the respective genes from
####  ORtab.medi.scaled  and  ORtab.brtn.scaled

ORtab.g9.medi.brtn.scaled <- left_join(ORtab.g9, ORtab.medi.scaled,
                                       join_by(Genes == gene)) %>%
  mutate(
    OR_medi = OR,
    OR_medi_CI = paste(as.character(round(CI.lo, digits = 2)),
                       as.character(round(CI.up, digits = 2)),
                       sep = "-"),
    p_medi = p.value,
    OR = NULL,
    CI.lo = NULL,
    CI.up = NULL,
```

```
    p.value = NULL
  ) %>%
  left_join(ORtab.brtn.scaled,
            join_by(Genes == gene)) %>%
  mutate(
    OR_brtn = OR,
    OR_brtn_CI = paste(as.character(round(CI.lo, digits = 2)),
                       as.character(round(CI.up, digits = 2)),
                       sep = "-"),
    p_brtn = p.value,
    OR = NULL,
    CI.lo = NULL,
    CI.up = NULL,
    p.value = NULL
  )









############################
###   Survival-Dataset   ###
############################

library("survival")


#############################################################################
# Calculate Hazard Ratios for all 2549 HTG genes in validation datasets #####
#############################################################################

val.genes <- g0.all  # all 2549 genes from HTG panel


#################################
###  Dataset scanb.htg.Rseq  ###
#################################

# Define dataset and genes
coh <- "scanb"
rseq <- scanb.htg.Rseq
cli.dat <- scanb.htg.cli.dat

gen.set <- val.genes

# Select data (scaled):
dat <- rseq[row.names(rseq) %in% gen.set,
            colnames(rseq) %in% rownames(cli.dat)]
dat <- dat[order(rownames(dat)),]  # order by gene name
dat <- dat[,order(colnames(dat))] # order by sample names
dat <- na.omit(t(scale(t(dat))))   # scale RNA-Seq for each gene
cli.dat <- cli.dat[order(rownames(cli.dat)),]
cli.dat$time <- cli.dat$OS_months
```

```r
cli.dat$status <- cli.dat$OS_event


# define dataframe for results
HRtab <- data.frame(gene=NA, HR=NA, CI.lo=NA, CI.up=NA, p.value=NA)

# loop for hazard ratios of survival for all genes
stopifnot(colnames(dat)==rownames(cli.dat))

suppressWarnings(suppressMessages(
  for(i in 1:nrow(dat)){
    mod <- summary(coxph(Surv(cli.dat$time, cli.dat$status) ~ dat[i,]))
    HRtab[i,1] <- rownames(dat)[i]  # gene name
    HRtab[i,2] <- mod$conf.int[1]  # HR = exp(coef)
    HRtab[i,3:4] <- mod$conf.int[3:4]     # CI of HR
    HRtab[i,5] <- mod$logtest[3]  # p-value
  }
))


# Save the result as named variable
assign(paste0("HRtab.",coh,".scaled"),HRtab)

####################


##############################################################################
#      Assemble combined HR-tables for cohorts                        #######
##############################################################################

# Assemble HR info for G9

HRtab.g9 <- data %>%
  select(Genes,
         HR_durva, HR_durva_CI, p_cox_DDFS_durva,
         HR_plac, HR_plac_CI, p_cox_DDFS_plac
  )

HRtab.g9.scanb.scaled <- left_join(HRtab.g9, HRtab.scanb.scaled,
                                   join_by(Genes == gene)) %>%
  mutate(
    HR_scanb = HR,
    HR_scanb_CI = paste(as.character(round(CI.lo, digits = 2)),
                        as.character(round(CI.up, digits = 2)),
                        sep = "-"),
    p_scanb = p.value,
    HR = NULL,
    CI.lo = NULL,
    CI.up = NULL,
    p.value = NULL
  )


##############################################################################
```

```
#
```

Figure 7a, 7b:

Scatter plot figures pCR-Durva vs pCR-Placebo

```
#
```

FINDING-cohort G9

```
val.genes.set <- "g9.pCR.DDFS.scatter.genes"
val.genes <- get(val.genes.set)


plotdat <- gene.class %>%
  filter(Genes %in% val.genes) %>%
  left_join(ORtab.g9.medi.brtn.scaled,
            join_by(Genes)) %>%
  select(Genes, gene.class,
         OR_pCR_durva,
         p_log_pCR_durva,
         OR_pCR_plac,
         p_log_pCR_plac,
         OR_medi,
         p_medi,
         OR_brtn,
         p_brtn) %>%
  filter(Genes != "IL9")  # exclude outlier


# Set axis limits for comparison of plots

Durva.min <- min(c(plotdat$OR_pCR_durva), na.rm = T)
Durva.max <- max(c(plotdat$OR_pCR_durva), na.rm = T)
NonDurva.min <- min(c(plotdat$OR_pCR_plac), na.rm = T)
NonDurva.max <- max(c(plotdat$OR_pCR_plac), na.rm = T)


# Define color palette:

# 6 groups: DNArepair, immune, other, proliferation, stemcell, stromalEMT
colpal <- c("#1B9E77" ,"#7570B3", "#666666",
            "#E7298A" ,"#66A61E", "#A6761D")


# Finding cohort:  pCR-g9-durva vs pCR-g9-placebo

fn <- "out/Fig7a_Scatter-pCR-Durva-Plac-G9.pdf"

pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_pCR_durva, OR_pCR_plac, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 0.8) +
  scale_x_continuous(trans='log2', limits = c(Durva.min, Durva.max)) +
```

```
  scale_y_continuous(trans='log2', limits = c(NonDurva.min, NonDurva.max)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("pCR-G9-Durva vs pCR-G9-Placebo") +
  xlab("OR for pCR Durva") +
  ylab("OR for pCR Placebo")
dev.off()

## png
##   2

#
```

VALIDATION-cohorts: pCR-MEDI-Durva vs. pCR-BrighTNess-NonDurva

```
# Set axis limits for comparison of plots

Durva.min <- min(c(plotdat$OR_medi), na.rm = T)
Durva.max <- max(c(plotdat$OR_medi), na.rm = T)
NonDurva.min <- min(c(plotdat$OR_brtn), na.rm = T)
NonDurva.max <- max(c(plotdat$OR_brtn), na.rm = T)



# pCR-MEDI-durva vs pCR-BrighTNess-placebo

fn <- "out/Fig7b_Scatter-pCR-Durva-MEDI-NonDurva-Brightn.pdf"

pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_medi, OR_brtn, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 0.8) +
  scale_x_continuous(trans='log2', limits = c(Durva.min, Durva.max)) +
  scale_y_continuous(trans='log2', limits = c(NonDurva.min, NonDurva.max)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("pCR-MEDI-Durva vs pCR-BrighTNess-NonDurva") +
  xlab("OR for pCR Durva") +
  ylab("OR for pCR NonDurva")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).

dev.off()

## png
##   2

##############################################################################
```

```
#
```

Scatter plot figures pCR vs DDFS

Fig. 7c 7d

```
#

val.genes.set <- "g9.pCR.DDFS.scatter.genes"
val.genes <- get(val.genes.set)

# Scatter plot for genes with any p-value <0.01 in G9
#   (either pCR or DDFS in either arm)
# color genes by gene.class, Palette "Dark2" colors from RColorBrewer
#
# These 126 genes are in genelist  "g9.pCR.DDFS.scatter.genes"
#
# The OR and HR data for these 126 genes for the different validation
#   datasets will be extracted from the following tables:
#      ORtab.g9.medi.brtn.scaled
#      HRtab.g9.scanb.scaled
#
# Combine all data needed for scatter plots in one table (plotdat):

plotdat <- gene.class %>%
  filter(Genes %in% val.genes) %>%
  left_join(ORtab.g9.medi.brtn.scaled,
            join_by(Genes)
            ) %>%
  left_join(HRtab.g9.scanb.scaled,
            join_by(Genes)
            ) %>%
  select(Genes, gene.class,
         OR_medi, p_medi,
         OR_brtn, p_brtn,
         HR_scanb, p_scanb
         ) %>%
  filter(Genes != "IL9")  # exclude outlier in SCANB



# Set axis limits for comparison of plots
HR.min <- min(plotdat$HR_scanb, na.rm = T)
HR.max <- max(plotdat$HR_scanb, na.rm = T)

# Define color palette:

# 6 groups: DNArepair, immune, other, proliferation, stemcell, stromalEMT
colpal <- c("#1B9E77" ,"#7570B3", "#666666",
            "#E7298A" ,"#66A61E", "#A6761D")


# Plot to pdf files

# pCR-MEDI vs DFS-SCANB
```

```r
OR.min <- min(c(plotdat$OR_medi), na.rm = T)
OR.max <- max(c(plotdat$OR_medi), na.rm = T)

fn <- "out/Fig7c_Scatter-pCR-DDFS-MEDI-SCANB.pdf"

pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_medi, 1/HR_scanb, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 0.8) +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("pCR-MEDI vs DFS-SCANB") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
dev.off()

## png
##   2

# pCR-BrighTNess vs DFS-SCANB

OR.min <- min(c( plotdat$OR_brtn), na.rm = T)
OR.max <- max(c(plotdat$OR_brtn), na.rm = T)

fn <- "out/Fig7d_Scatter-pCR-DDFS-BRTN-SCANB.pdf"
pdf(file = fn, width = 11, height = 8.5)
plotdat %>% ggplot(aes(OR_brtn, 1/HR_scanb, color = gene.class)) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(size = 10, alpha = 1) +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(label = Genes, color = gene.class),
                  size = 4,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("pCR-BrighTNess vs DFS-SCANB") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).

dev.off()

## png
##   2

###############


###  Additional figures including p-value information:
###  Filled circles size based on log10(p-value for pCR)
###  Open circles  size based on log10(p-value for DDFS)



# pCR-MEDI vs DFS-SCANB
fn <- "out/Suppl_Scatter-pCR-DDFS-MEDI-SCANB-pVal.pdf"

OR.min <- min(c(plotdat$OR_medi), na.rm = T)
OR.max <- max(c(plotdat$OR_medi), na.rm = T)


pdf(file = fn, width = 11, height = 8.5)
ggplot(plotdat) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(aes(x = OR_medi, y = 1/HR_scanb, color = gene.class,
                 size = -log10(p_medi)), alpha = 0.5) +
  geom_point(aes(x = OR_medi, y = 1/HR_scanb, color = gene.class,
                 size = -log10(p_scanb)), shape = 1) +
  scale_size(range = c(.1, 14), name = "-log10(P-value) [pCR, DDFS]") +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(x = OR_medi, y = 1/HR_scanb,
                      label = Genes, color = gene.class),
                  size = 3,
                  box.padding   = 0.5,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color = 'darkgrey', show.legend = F) +
  ggtitle("pCR-MEDI vs DFS-SCANB \n(filled circles: pCR-p-Value / open circles: DDFS-p-
Value") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
dev.off()

## png
##   2
```

```r
# pCR-BrighTNess vs DFS-SCANB

OR.min <- min(c( plotdat$OR_brtn), na.rm = T)
OR.max <- max(c(plotdat$OR_brtn), na.rm = T)

fn <- "out/Suppl_Scatter-pCR-DDFS-BRTN-SCANB-pVal.pdf"
pdf(file = fn, width = 11, height = 8.5)
ggplot(plotdat) +
  theme_light() +
  scale_color_manual(values = colpal) +
  geom_point(aes(x = OR_brtn, y = 1/HR_scanb, color = gene.class,
                 size = -log10(p_brtn)), alpha = 0.5) +
  geom_point(aes(x = OR_brtn, y = 1/HR_scanb, color = gene.class,
                 size = -log10(p_scanb)), shape = 1) +
  scale_size(range = c(.1, 14), name = "-log10(P-value) [pCR, DDFS]") +
  scale_x_continuous(trans='log2', limits = c(OR.min, OR.max)) +
  scale_y_continuous(trans='log2', limits = c(1/HR.max, 1/HR.min)) +
  geom_hline(yintercept = 1, color = "blue", linewidth = 1) +
  geom_vline(xintercept = 1, color = "orange", linewidth = 1) +
  geom_text_repel(aes(x = OR_brtn, y = 1/HR_scanb,
                      label = Genes, color = gene.class),
                  size = 3,
                  box.padding   = 0.3,
                  point.padding = 0.3,
                  segment.size= 0.5,
                  max.overlaps = 50,
                  segment.color ='darkgrey', show.legend = F) +
  ggtitle("pCR-BrighTNess vs DFS-SCANB \n(filled circles: pCR-p-Value / open circles:
DDFS-p-Value") +
  xlab("OR for pCR") +
  ylab("1 / HR for DDFS")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).
```

```r
dev.off()
```

```
## png
##   2
```

```
#
```

Generate Signatures from genesets ——————————————

```
#

# The data frames
#   resp.g1.immune,
#   resp.g2.proliferation,
#   resp.g4.stromal.NonImmune
# contain the separate list of the genes, which are
# associated either with good or poor response (pCR) in
# each treatment arm in G9, based on the respective geneset.
#
# These dataframes will be used to select the specific genes
# for calculating three signatures based on the mean of the
# scaled expression in the code below.
#
#
# For immune and proliferation signatures
#  positively predictive (good) genes will be included (OR>1, p<=0.05)
#
# resp.g1.immune$durva.good
# resp.g2.proliferation$durva.good
#
# For the stromal signature
#  negatively predictive (poor) genes will be included (OR<1, p<=0.05)
#
# resp.g4.stromal.NonImmune$durva.poor

# Export gene lists:

write.csv2(resp.g1.immune$durva.good,
           file = "out/signature-immune-genes.csv")
write.csv2(resp.g2.proliferation$durva.good,
           file = "out/signature-proliferation-genes.csv")
write.csv2(resp.g4.stromal.NonImmune$durva.poor,
           file = "out/signature-stromal-genes.csv")



########################################
###          MEDI dataset            ###
########################################

# Define data

dat <- medi.htg.Rseq
coh <- "medi"


# The following code uses dat and coh
# and is IDENTICAL for BOTH COHORTS (medi & brtn)

##############################
```

```r
### CALCULATE SIGNATURES: ####
################################

### Only Durva signatures that are USED for Figures calculated here:

### Signature "Good pCR Durva g1.immune": ###

# calculate mean expression of SCALED genes from genelist
#  predictive for pCR in durva arm:
sig.durva.good.g1.immune.scaled <- dat[
  rownames(dat) %in% resp.g1.immune$durva.good,] %>%
  t() %>%
  scale() %>%
  t() %>%
  na.omit() %>%
  apply(2, mean)


### Signature "Good pCR Durva g2.proliferation" ###

# calculate mean expression of SCALED genes from genelist
#  predictive for pCR in durva arm:
sig.durva.good.g2.proliferation.scaled <- dat[
  rownames(dat) %in% resp.g2.proliferation$durva.good,] %>%
  t() %>%
  scale() %>%
  t() %>%
  na.omit() %>%
  apply(2, mean)


### Signature "Poor pCR Durva g4.stromal.NonImmune" ###

# calculate mean expression of SCALED genes from genelist
#  predictive for pCR in durva arm:
sig.durva.poor.g4.stromal.NonImmune.scaled <- dat[
  rownames(dat) %in% resp.g4.stromal.NonImmune$durva.poor,] %>%
  t() %>%
  scale() %>%
  t() %>%
  na.omit() %>%
  apply(2, mean)

#########################################################


### Assemble all signature results in one data.frame
### for the cohort

sig <- data.frame(
  cbind(
    sig.durva.good.g1.immune.scaled,
    sig.durva.good.g2.proliferation.scaled,
    sig.durva.poor.g4.stromal.NonImmune.scaled
```

```r
  )
)

# Add categories (based on cutoff = 0) for g1, g2, g4
# in order to calculate frequencies in Quadrants:

sig <- sig %>%
  mutate(
    cat.durva.good.g1.immune=cut(sig.durva.good.g1.immune.scaled,
                                 breaks=c(-Inf, 0, Inf),
                                 labels=c("low","high")),
    cat.durva.good.g2.proliferation=cut(sig.durva.good.g2.proliferation.scaled,
                                        breaks=c(-Inf, 0, Inf),
                                        labels=c("low","high")),
    cat.durva.poor.g4.stromal=cut(sig.durva.poor.g4.stromal.NonImmune.scaled,
                                  breaks=c(-Inf, 0, Inf),
                                  labels=c("low","high"))
  )

# Get cli.dat for the selected cohort (coh):

cli <- get(paste(coh, ".htg.cli.dat", sep = ""))

# Add pCR info:
sig$pCR <- cli[rownames(sig), colnames(cli)=="pCR"]


# Calculate pCR frequencies in Quadrants based on combination of g1, g2, g4

########################################

# categories g1 vs g2:
cat1 <- "cat.durva.good.g1.immune"
cat2 <- "cat.durva.good.g2.proliferation"


### Calculate pCR frequencies and percentages and save results
quadr <- list()
quadr$low1.low2 <- sig %>%
  filter(get(cat1) == "low", get(cat2) == "low") %>% count(pCR)
quadr$high1.low2 <- sig %>%
  filter(get(cat1) == "high", get(cat2) == "low") %>% count(pCR)
quadr$low1.high2 <- sig %>%
  filter(get(cat1) == "low", get(cat2) == "high") %>% count(pCR)
quadr$high1.high2 <- sig %>%
  filter(get(cat1) == "high", get(cat2) == "high") %>% count(pCR)
# add percentages
quadr$low1.low2$perc <- round(quadr$low1.low2$n/sum(quadr$low1.low2$n)*100, 1)
quadr$high1.low2$perc <- round(quadr$high1.low2$n/sum(quadr$high1.low2$n)*100, 1)
quadr$low1.high2$perc <- round(quadr$low1.high2$n/sum(quadr$low1.high2$n)*100, 1)
quadr$high1.high2$perc <- round(quadr$high1.high2$n/sum(quadr$high1.high2$n)*100, 1)
# output the results:
quadr
```

```
## $low1.low2
##     pCR  n perc
## 1 FALSE 10 71.4
## 2  TRUE  4 28.6
##
## $high1.low2
##     pCR n perc
## 1 FALSE 6   60
## 2  TRUE 4   40
##
## $low1.high2
##     pCR  n perc
## 1 FALSE 10 76.9
## 2  TRUE  3 23.1
##
## $high1.high2
##     pCR  n perc
## 1 FALSE  4 22.2
## 2  TRUE 14 77.8
```

```
# save the variable by adding the signature names
assign(paste(coh,"quadr",cat1,cat2, sep = "_"), quadr)
#########################################################
```

```
# categories g1 vs g4:
cat1 <- "cat.durva.good.g1.immune"
cat2 <- "cat.durva.poor.g4.stromal"
```

```
### Calculate pCR frequencies and percentages and save results
quadr <- list()
quadr$low1.low2 <- sig %>%
  filter(get(cat1) == "low", get(cat2) == "low") %>% count(pCR)
quadr$high1.low2 <- sig %>%
  filter(get(cat1) == "high", get(cat2) == "low") %>% count(pCR)
quadr$low1.high2 <- sig %>%
  filter(get(cat1) == "low", get(cat2) == "high") %>% count(pCR)
quadr$high1.high2 <- sig %>%
  filter(get(cat1) == "high", get(cat2) == "high") %>% count(pCR)
# add percentages
quadr$low1.low2$perc <- round(quadr$low1.low2$n/sum(quadr$low1.low2$n)*100, 1)
quadr$high1.low2$perc <- round(quadr$high1.low2$n/sum(quadr$high1.low2$n)*100, 1)
quadr$low1.high2$perc <- round(quadr$low1.high2$n/sum(quadr$low1.high2$n)*100, 1)
quadr$high1.high2$perc <- round(quadr$high1.high2$n/sum(quadr$high1.high2$n)*100, 1)
# output the results:
quadr
```

```
## $low1.low2
##     pCR n perc
## 1 FALSE 7 58.3
## 2  TRUE 5 41.7
##
## $high1.low2
```

```
##     pCR  n perc
## 1 FALSE  4 26.7
## 2  TRUE 11 73.3
##
## $low1.high2
##     pCR  n perc
## 1 FALSE 13 86.7
## 2  TRUE  2 13.3
##
## $high1.high2
##     pCR n perc
## 1 FALSE 6 46.2
## 2  TRUE 7 53.8
```

```r
# save the variable by adding the signature names
assign(paste(coh,"quadr",cat1,cat2, sep = "_"), quadr)
#########################################################

# SAVE THE SIGNATURE DATA FORT THE COHORT (coh)

assign(paste("sig.", coh, sep = ""), sig)
#########################################################
```

```
#
```

pCR Plots in MEDI validation Cohort

*Figures 7f, 7h, and Supplementary Figures 8b,8d,8f

```
#

colpal <- c("#D55E00", "#009E73")

################
#  MEDI cohort #
################

coh <- "MEDI"

plotdat <- sig.medi



### Scatter Plots: ###

# Figures 7f, 7h

sig.pairs <- tribble(
  ~pname, ~x.ax, ~y.ax,
  "Fig7f_MEDI_g1sca_vs_g4sca", "sig.durva.good.g1.immune.scaled",
"sig.durva.poor.g4.stromal.NonImmune.scaled",
  "Fig7h_MEDI_g1sca_vs_g2sca", "sig.durva.good.g1.immune.scaled",
"sig.durva.good.g2.proliferation.scaled"
)

for (i in 1:nrow(sig.pairs)){
  pname <- sig.pairs$pname[i]
  x.ax <- sig.pairs$x.ax[i]
  y.ax <- sig.pairs$y.ax[i]
  # Generate plot
  p <- ggplot(plotdat, aes(get(x.ax),
                           get(y.ax),
                           color = pCR)) +
    theme_light() +
    geom_point(size = 10, alpha = 0.8) +
    scale_x_continuous() +
    scale_y_continuous() +
    scale_color_manual(values = colpal) +
    geom_hline(yintercept = 0, color = "black", linewidth = 1) +
    geom_vline(xintercept = 0, color = "black", linewidth = 1) +
    ggtitle(paste(coh, x.ax, "vs.", y.ax)) +
    xlab(x.ax) +
    ylab(y.ax)
  # export plot
  fn <- paste0("out/", pname, "_SigScatter", ".pdf")
  pdf(file = fn, width = 11, height = 8.5)
  print(p)
  dev.off()
```

```
}



# Distribution of pCR according signatures --------------------------------

# Supplementary Figure 8 (b,d,f)

colpal <- c("#D55E00", "#009E73")

siglist <- list("sig.durva.good.g1.immune.scaled",
                "sig.durva.good.g2.proliferation.scaled",
                "sig.durva.poor.g4.stromal.NonImmune.scaled")


# Loop through siglist and save plots as pdf files:

for(plotsig in siglist) {
  # create plot
  p <- ggplot(plotdat,
              aes(x = get(plotsig))) +
    geom_density(aes(color = pCR,
                     fill = pCR),
                 position = "identity",
                 alpha = 0.3) +
    scale_color_manual(values = colpal) +
    scale_fill_manual(values = colpal) +
    ggtitle(paste(coh, " ", plotsig)) +
    xlab(plotsig)
  # export plot
  fn <- paste0("out/SuppFig4_pCRDistr_", coh, "_", plotsig, ".pdf")
  pdf(file = fn,
      width = 11,
      height = 8.5)
  print(p)
  dev.off()
}



# end ---------------------------------------------------------------------
```

```r
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=German_Germany.utf8  LC_CTYPE=German_Germany.utf8
## [3] LC_MONETARY=German_Germany.utf8 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.utf8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] survival_3.7-0      ggrepel_0.9.6       ggvenn_0.1.10
##  [4] ggVennDiagram_1.5.2 fmsb_0.7.6          stringr_1.5.1
##  [7] tidyr_1.3.1         ggplot2_3.5.1       readxl_1.4.3
## [10] dplyr_1.1.4
##
## loaded via a namespace (and not attached):
##  [1] Matrix_1.7-0      gtable_0.3.5      compiler_4.4.1    Rcpp_1.0.13
##  [5] tidyselect_1.2.1  splines_4.4.1     scales_1.3.0      yaml_2.3.10
##  [9] fastmap_1.2.0     lattice_0.22-6    R6_2.5.1          labeling_0.4.3
## [13] generics_0.1.3    knitr_1.48        tibble_3.2.1      munsell_0.5.1
## [17] pillar_1.9.0      rlang_1.1.4       utf8_1.2.4        stringi_1.8.4
## [21] xfun_0.47         cli_3.6.3         withr_3.0.1       magrittr_2.0.3
## [25] digest_0.6.37     rstudioapi_0.16.0 lifecycle_1.0.4   vctrs_0.6.5
## [29] evaluate_1.0.0    glue_1.7.0        farver_2.1.2      cellranger_1.1.0
## [33] fansi_1.0.6       colorspace_2.1-1  rmarkdown_2.28    purrr_1.0.2
## [37] tools_4.4.1       pkgconfig_2.0.3   htmltools_0.5.8.1
```