Agentic AI for Adaptive Pharmacogenomic Biomarker Discovery in Cancer

Case 1:

Immunotherapy and immuno-oncology (IO) datasets provide rich molecular and clinical information across many cohorts treated with immune checkpoint blockade, including transcriptomes, genomic alterations, response labels, and survival outcomes. These data are ideal for benchmarking and preliminary discovery of IO biomarkers such as checkpoint genes and immune signatures, but heterogeneity in endpoints, metadata, and data structures makes analyses difficult to scale and reproduce. An agentic framework equipped with standardized tools for cohort selection, endpoint harmonization, signature scoring, survival evaluation, and reporting can accelerate ICB biomarker analyses while producing auditable, reproducible execution traces.

Prompt to Agent:

Hypothesis: In ICB-treated cohorts, established IO markers and immune activity programs will show measurable concordance with IO outcomes (e.g., overall survival).

Goal: Quantify how well individual immune checkpoint genes or immune signatures rank IO outcomes (e.g., overall survival) in an ICB-treated cohort, using the concordance index (C-index) computed from continuous expression and a median high/low split.

ICB dataset loaded under `./data/rawdata/`

Example Agent Output:

Step 1: Load ICB data

```r
# load in csv file
filepath <- list.files(here("./data/rawdata"), pattern = "\\.csv$", full.names = TRUE)
dat_icb <- read.csv(filepath, check.names = FALSE)

# define which columns are expression features clinical/metadata
# identified the first 10 columns as clinical/metadata
genes <- colnames(dat_icb)[-c(1:10)]

# perform basic checks on dataset
stopifnot(all(c("survival_time_os", "event_occurred_os") %in% colnames(dat_icb)))
stopifnot(is.numeric(dat_icb$survival_time_os))
stopifnot(all(dat_icb$event_occurred_os %in% c(0, 1, NA)))
```

Step 2: Compute C-index using continuous markers (recommended)

C-index here assesses how well a marker ranks patients by risk. For expression, use the continuous values for best use of information.

```r
ci_cont <- lapply(1:length(genes), function(k){

# concordance index
ci <- concordance.index(
  x = dat_icb[, genes[k]],
  surv.time = dat_icb$survival_time_os,
  surv.event = dat_icb$event_occurred_os,
  method = "noether"
)

data.frame(
  gene = genes[k],
  c.idex = ci$c.index,
  ci.lower = ci$lower,
  ci.upper = ci$upper,
  pval = ci$p.value,
  cancer_type = unique(dat_icb$cancer_type),
  treatment = unique(dat_icb$treatment))
})

ci_cont <- do.call(rbind, ci_cont)
head(ci_cont)
```

```
##      gene    c.idex  ci.lower  ci.upper        pval cancer_type  treatment
## 1   ITGAL 0.3464448 0.2183384 0.4745511 0.018807770    Melanoma PD-1/PD-L1
## 2   RGPD5 0.4478064 0.3077270 0.5878857 0.465216732    Melanoma PD-1/PD-L1
## 3 HERPUD1 0.3540091 0.2433087 0.4647095 0.009743928    Melanoma PD-1/PD-L1
## 4   USP36 0.4372163 0.3104883 0.5639444 0.331545256    Melanoma PD-1/PD-L1
## 5  SLC2A3 0.4311649 0.3028393 0.5594905 0.293100800    Melanoma PD-1/PD-L1
## 6   DDX3Y 0.5000000 0.3724432 0.6275568 1.000000000    Melanoma PD-1/PD-L1
```

```r
write.csv(ci_cont, file = here("./data/results/ci_cont.csv"))
```

Step 3: Compute C-index using binary markers (High vs Low, median split)

This is mainly useful for demonstration/visualization (KM curves), but it discards information compared with continuous markers.

```r
ci_bin <- lapply(1:length(genes), function(k){

cut <- median( dat_icb[, genes[k]], na.rm = TRUE)
g <- as.integer( dat_icb[, genes[k]] >= cut)  # High=1, Low=0

ci <- concordance.index(x = g,
                        surv.time = dat_icb$survival_time_os,
                        surv.event = dat_icb$event_occurred_os,
                        method="noether")


data.frame(gene = genes[k],
           c.idex = ci$c.index,
           ci.lower = ci$lower,
           ci.upper = ci$upper,
```

```
            pval = ci$p.value,
            cancer_type = unique(dat_icb$cancer_type),
            treatment = unique(dat_icb$treatment))


})

ci_bin <- do.call(rbind, ci_bin)
head(ci_bin)
```

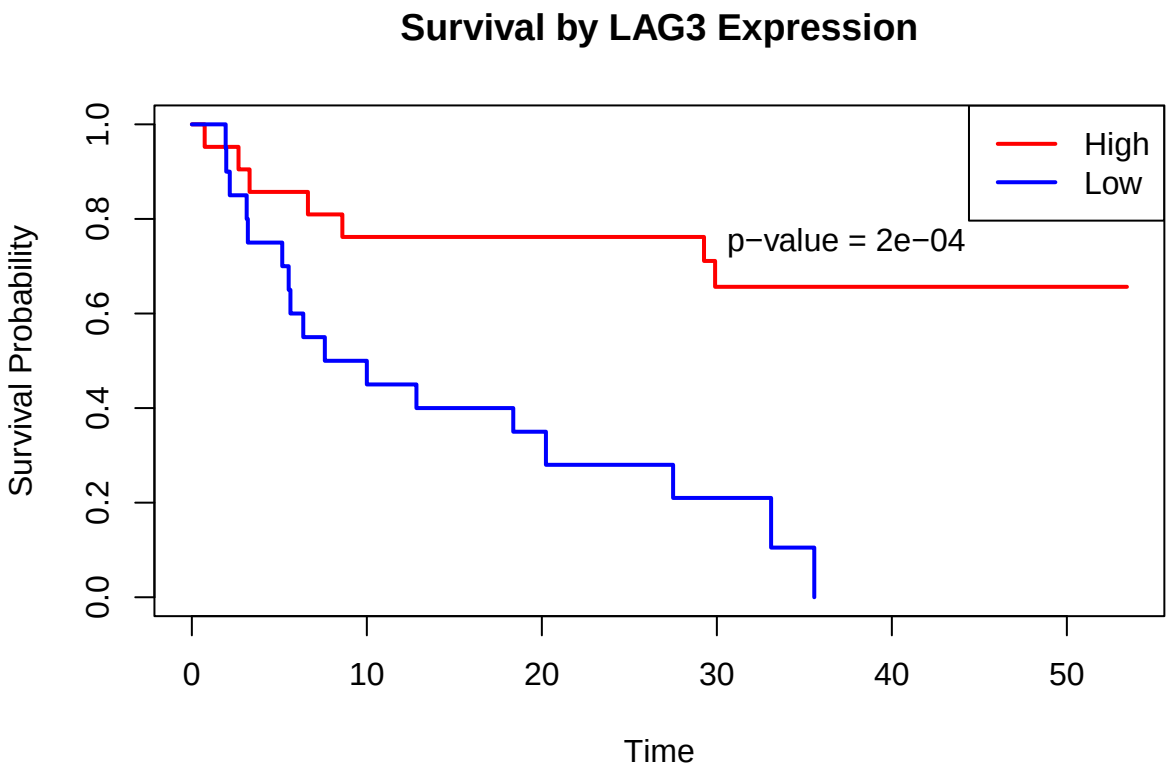
```
##      gene    c.idex  ci.lower  ci.upper       pval cancer_type  treatment
## 1    ITGAL 0.3314286 0.1452747 0.5175825 0.07592380    Melanoma PD-1/PD-L1
## 2    RGPD5 0.5178571 0.3182441 0.7174702 0.86081559    Melanoma PD-1/PD-L1
## 3 HERPUD1 0.2910663 0.1185903 0.4635422 0.01758423    Melanoma PD-1/PD-L1
## 4    USP36 0.4434524 0.2438137 0.6430911 0.57878566    Melanoma PD-1/PD-L1
## 5   SLC2A3 0.3768546 0.1882731 0.5654361 0.20058952    Melanoma PD-1/PD-L1
## 6    DDX3Y 0.5769231 0.3766904 0.7771558 0.45147617    Melanoma PD-1/PD-L1
```

```
write.csv(ci_bin, file = here("./data/results/ci_bin.csv"))
```

Notes / next steps

- Known ICB-related markers often discussed include: LAG3, TIGIT, CD274.
- If you want a simple KM plot for one marker:

```
# fit survival model
cut <- median(dat_icb$LAG3, na.rm=TRUE)
dat_icb$LAG3_group <- ifelse(dat_icb$LAG3 >= cut, "High", "Low")
fit <- survfit(Surv(survival_time_os, event_occurred_os) ~ LAG3_group, data=dat_icb)

# calculate p-value with log-rank test
survdiff_result <- survdiff(Surv(survival_time_os, event_occurred_os) ~ LAG3_group,
                            data=dat_icb)
pval <- pchisq(survdiff_result$chisq, df=1, lower.tail=FALSE)

# plot kaplan-meier curves
plot(fit, col=c("red", "blue"), lwd=2,
     xlab="Time", ylab="Survival Probability",
     main="Survival by LAG3 Expression")
legend("topright", legend=c("High", "Low"), col=c("red", "blue"), lwd=2)
text(x=max(dat_icb$survival_time_os, na.rm=TRUE)*0.7,
     y=0.75,
     labels=paste("p-value =", round(pval, 4)))
```



- If you want to use an ORCESTRA-derived MultiAssayExperiment (MAE) instead of CSV: you can load the MAE, extract a `SummarizedExperiment`, then build a clean data.frame with survival + markers for the same workflow above.

```
library(PredictioR)

files <- list.files(file.path(dir_input, 'rds'))
dat <- readRDS(file.path(dir_input, 'rds', files))
dat_icb <- createSE(dat)
expr <- assay(dat_icb)
clin <- colData(expr)

# From here, you'd create dat_icb-like object:
dat_icb <- data.frame(
  survival_time_os  = clin$survival_time_os,
  event_occurred_os = clin$event_occurred_os,
  t(expr[c("LAG3","CD274"), ]) |> t()  # or compute signature scores first
)
```

Case 2A:

Pharmacogenomic datasets provide a wealth of biological information across various experiments and sequencing technologies for a large collection of preclinical samples. While this makes for an ideal data source for preliminary biomarker discovery, the data container structure can be difficult to navigate. An agentic

framework, provided with tools from the PharmacoGx package, can expedite bioinformatic analyses using these datasets.

Prompt to Agent:

Gene expression-based subtypes of breast cancer are widely studied to be associated with response to treatment. I want a plot to show that PAM50-based Her2 subtypes are associated with HER2-targeted therapies.

Use the GRAY dataset loaded under `./data/rawdata/`

Example Agent output:

Step 1: Load in PharmacoSet

The following dataset was downloaded from orcestra.ca. The dataset is stored as a `PharmacoSet` object from the PharmacoGx package. The dataset contains 74 breast cancer samples with 9756 drug response experiments.

```
gray <- readRDS(here("./data/rawdata/PSet_GRAY2017.rds")) |> updateObject()
gray
```

```
## <PharmacoSet>
## Name: GRAY
## Date Created: Thu Apr  6 16:50:15 2023
## Number of samples:  74
## Molecular profiles:
## Kallisto_0.46.1.rnaseq :
##   Dim: 60662, 64
## Kallisto_0.46.1.rnaseq.counts :
##   Dim: 60662, 64
## Kallisto_0.46.1.isoforms :
##   Dim: 227912, 64
## Kallisto_0.46.1.isoforms.counts :
##   Dim: 227912, 64
## Treatment response: Drug pertubation:
##    Please look at pertNumber(cSet) to determine number of experiments  for each drug-sample combinati
## Drug sensitivity:
##    Number of Experiments:  9756
##    Please look at sensNumber(cSet) to determine number of  experiments for each drug-sample combinati
```

Step 2: Use genefu package to compute PAM50 subtypes

```
# load in genefu PAM50 subtyping model
data(pam50.robust)

# get gene expression
gray_rna <- summarizeMolecularProfiles(
  gray,
  mDataType = "Kallisto_0.46.1.rnaseq"
) |> suppressMessages()
```

```
##   |                                                                      |
```

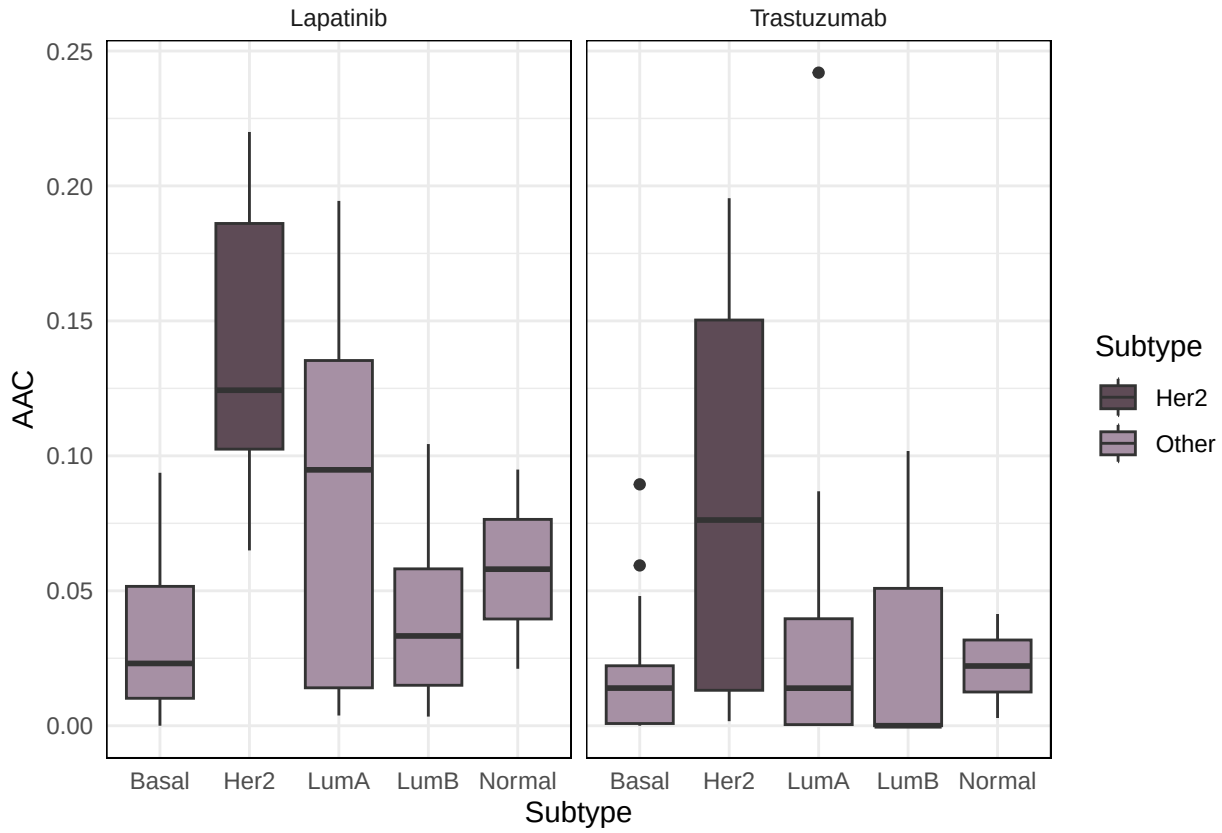
```r
rna <- t(assay(gray_rna)) |> as.data.frame()

# create metadata file
meta <- gray_rna@elementMetadata[,c("gene_id", "gene_name")] |> as.data.frame()
colnames(meta) <- c("Ensembl", "Gene.Symbol")
colnames(rna) <- meta$Gene.Symbol[match(colnames(rna), meta$Ensembl)]

# get subtype predictions for PAM50
pam50 <- molecular.subtyping(
  sbt.model = "pam50",
  data = rna,
  annot = meta,
  do.mapping = FALSE
)
gray_pam50 <- cbind(
  Subtype = as.character(pam50$subtype),
  as.data.frame(pam50$subtype.proba)
)
head(gray_pam50)
```

```
##        Subtype     Basal      Her2      LumA      LumB    Normal
## 184A1    Basal 0.7365824 0.0000000 0.0000000 0.0000000 0.2634176
## 184B5   Normal 0.2359258 0.0000000 0.1468017 0.0000000 0.6172726
## 21MT-1    Her2 0.2914219 0.7085781 0.0000000 0.0000000 0.0000000
## 21MT-2    Her2 0.0000000 0.8524511 0.0000000 0.1475489 0.0000000
## 21NT      Her2 0.3817810 0.6182190 0.0000000 0.0000000 0.0000000
## 21PT      Her2 0.3871703 0.6128297 0.0000000 0.0000000 0.0000000
```

Step 3: Create plots to show response to HER2-targeted therapies HER2 vs Other subtypes

```r
# get drug sensitivity matrix
gray_sen <- summarizeSensitivityProfiles(
    gray,
    sensitivity.measure = "aac_recomputed",
    summary.stat = "median",
    verbose = TRUE
) |> t() |> as.data.frame()

# add drugs of interest
toPlot <- cbind(gray_pam50, gray_sen[,c("Trastuzumab", "Lapatinib")])
her2_toPlot <- toPlot %>%
  dplyr::select(Subtype, Trastuzumab, Lapatinib) %>%
  pivot_longer(
    cols = c(Trastuzumab, Lapatinib),
    names_to = "Drug",
    values_to = "AAC"
  )
her2_toPlot$label <- ifelse(her2_toPlot$Subtype == "Her2", "Her2", "Other")

ggplot(her2_toPlot, aes(x = Subtype, y = AAC, fill = label)) +
  geom_boxplot() + facet_wrap(~Drug) +
  theme_minimal() + theme(panel.border = element_rect()) +
  scale_fill_manual("Subtype", values = c("#5E4B56", "#A690A4"))
```

```
## Warning: Removed 62 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



Notes / next steps

- You can consider performing an Analysis Of VAriance (ANOVA) test to quantitatively demonstrate this association

Case 2B:

There is evidence that drugs targeting EGFR can have a synthetic lethal affect on samples with BRCA mutations.

Prompt to Agent:

Using the `gCSI` PSet in `./data/rawdata/`, please first confirm that EGFR expression is associated with response to EGFR-targeted therapies, and then check if BRCA1 mutation status changes this association.

Example Agent Output:

Step 1: Load in PharmacoSet

This dataset was downloaded from orcestra.ca. The dataset is stored as a `PharmacoSet` object from the PharmacoGx package. The dataset contains 788 samples 675 cell lines with mutation and transcriptomic profiles, along with 16,688 drug response experiments.

```
gcsi <- readRDS(here("./data/rawdata/gCSI.rds")) |> updateObject()
gcsi
```

```
## <PharmacoSet>
## Name: gCSI
## Date Created: Sun May 24 23:50:46 2020
## Number of samples:  747
## Molecular profiles:
## Kallisto_0.46.1.rnaseq :
##   Dim: 60662, 675
## Kallisto_0.46.1.rnaseq.counts :
##   Dim: 60662, 675
## Kallisto_0.46.1.isoforms :
##   Dim: 227912, 675
## Kallisto_0.46.1.isoforms.counts :
##   Dim: 227912, 675
## CNV :
##   Dim: 26291, 329
## mutation :
##   Dim: 55, 329
## Treatment response: Drug pertubation:
##    Please look at pertNumber(cSet) to determine number of experiments  for each drug-sample combinati
## Drug sensitivity:
##    Number of Experiments:  6471
##    Please look at sensNumber(cSet) to determine number of  experiments for each drug-sample combinati
```

Step 2: Use PharmacoGx functions to summarize molecular and treatment profiles

```
# get RNAseq profile
gcsi_rna <- summarizeMolecularProfiles(
  gcsi,
  mDataType = "Kallisto_0.46.1.rnaseq"
) |> suppressMessages()
```

```
##   |                                                                      |
```

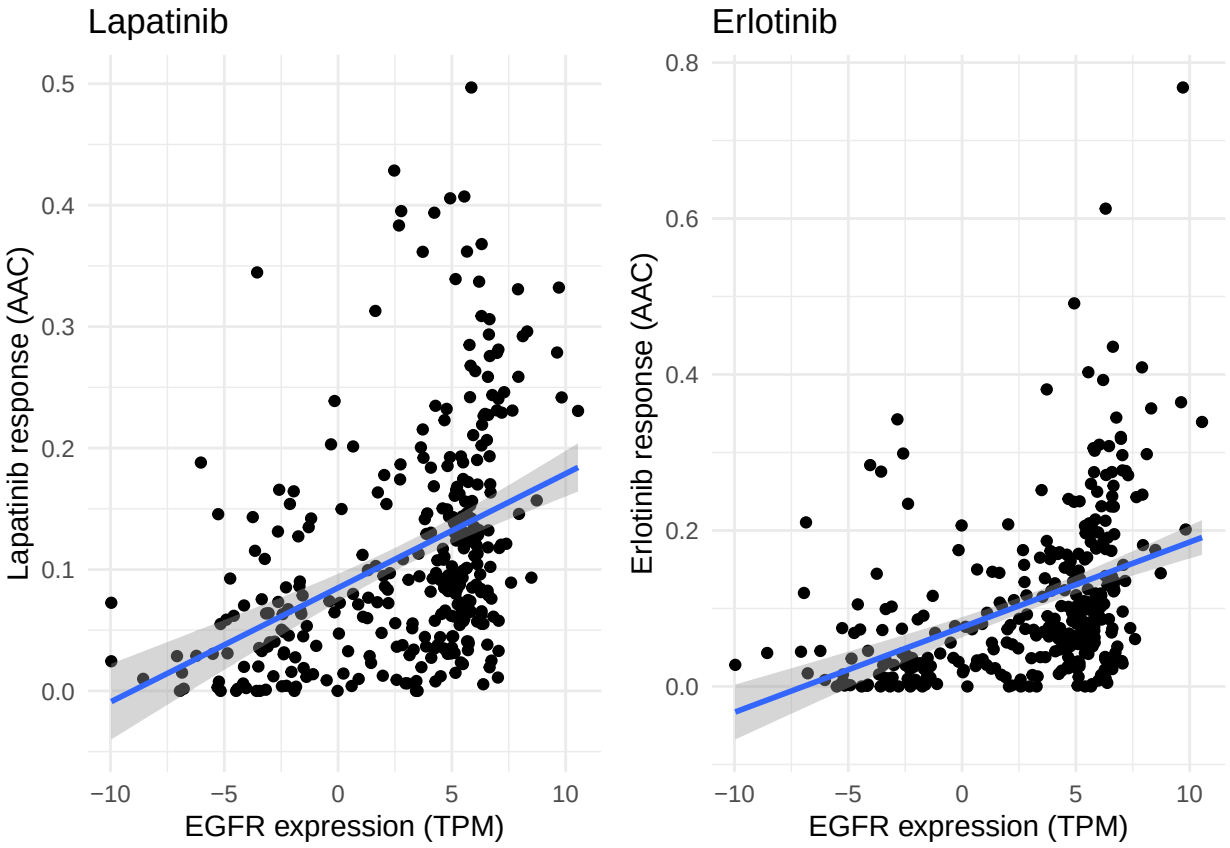
```
# get mutation profile
gcsi_mut <- summarizeMolecularProfiles(
  gcsi,
  mDataType = "mutation",
  summary.stat = "and"
) |> suppressMessages()
# get drug response profile
gcsi_sen <- summarizeSensitivityProfiles(
    gcsi,
    sensitivity.measure = "aac_recomputed",
    summary.stat = "median",
    verbose = TRUE
)
```

Step 3: Create plots of associations between EGFR expression and EGFR-targeted therapy response

```r
# specify genes and drugs of interest
drugs <- c("Erlotinib", "Lapatinib")
genes <- c("EGFR", "BRCA1")

# subset drug sensitivity
sen <- t(gcsi_sen[drugs,]) |> as.data.frame()

# subset mutations
mut <- t(assay(gcsi_mut)) |> as.data.frame()
mut <- mut[,genes]

# subset gene expression
rna <- assay(gcsi_rna) |> as.data.frame()
keep <- gcsi_rna@elementMetadata$gene_id[gcsi_rna@elementMetadata$gene_name %in% genes]
rna <- t(rna[keep,]) |> as.data.frame()
colnames(rna) <- gcsi_rna@elementMetadata$gene_name[match(colnames(rna), gcsi_rna@elementMetadata$gene_

# keep common cell lines
common <- intersect(rownames(sen), intersect(rownames(rna), rownames(mut)))
sen <- sen[match(common, rownames(sen)),]
rna <- rna[match(common, rownames(rna)),]
mut <- mut[match(common, rownames(mut)),]

# compile dataframe for plotting
toPlot <- data.frame(
  sample = common,
  EGFR = rna[["EGFR"]],
  Lapatinib = sen[["Lapatinib"]],
  Erlotinib = sen[["Erlotinib"]]
)

# plot associations between genes of interest and TKi
plot_egfr <- function(drug) {
  return(
    ggplot(toPlot, aes(x = EGFR, y = .data[[drug]])) +
      geom_point() + geom_smooth(method = "lm") +
      theme_minimal() +
      labs(x = "EGFR expression (TPM)", y = paste(drug, "response (AAC)"), title = drug)
  )
}

p1 <- plot_egfr("Lapatinib")
p2 <- plot_egfr("Erlotinib")

ggarrange(p1, p2)
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 416 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 416 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 415 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 415 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Step 4: assess if BRCA mutation status influences the association

```
# get samples with BRCA1 mutation
mutated <- mut[!is.na(mut$BRCA1),] |> rownames()
toPlot$brca1 <- ifelse(toPlot$sample %in% mutated, "Mutated", "Unmutated")

# plot association faceted by BRCA1 mutation status
plot_egfr <- function(drug) {
  return(
    ggplot(toPlot, aes(x = EGFR, y = .data[[drug]])) +
      geom_point() + geom_smooth(method = "lm") +
      facet_wrap(~brca1, ncol = 1) +
      theme_minimal() +
      labs(x = "EGFR expression (TPM)", y = paste(drug, "response (AAC)"), title = drug)
  )
}
```

```
p1 <- plot_egfr("Lapatinib")
p2 <- plot_egfr("Erlotinib")
ggarrange(p1, p2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

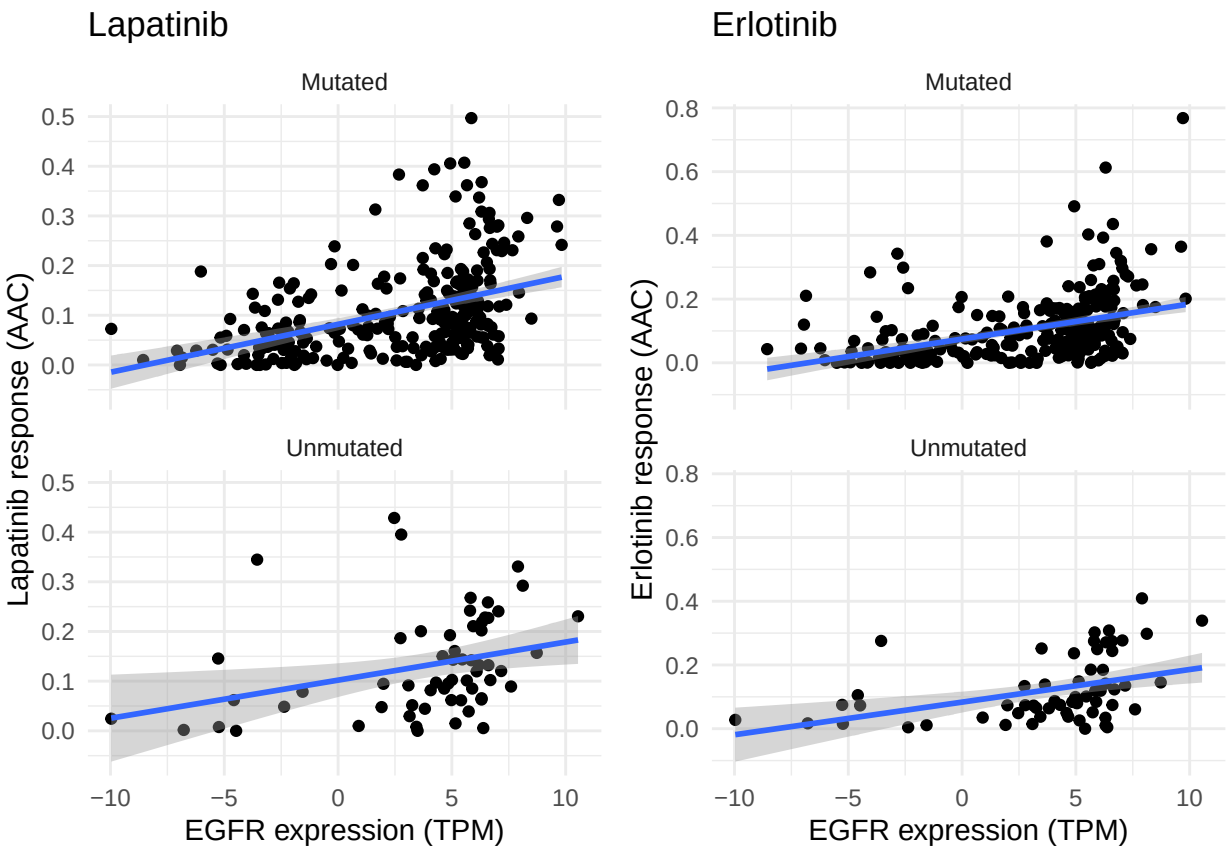
```
## Warning: Removed 416 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 416 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 415 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 415 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Case 3:

Curated clinical genomic datasets provide an opportunity for exploratory analysis to assist in experiment and analysis design. While SummarizedExperiment objects are ideal for storing molecular profiles and associated metadata, navitgating these objets can be time-consuming and challenging. An agentic framework can be leveraged to facilitate data extraction and perform quick exploratory analyses to identify preliminary associations between variables of interest.

Prompt to Agent:

High tumour cellularity can be associated with more aggressive cancers. Gene signatures that are surrogate markers of tumour cellularity may be predictive of response to immunotherapies. Using the PDAC Dataset in `./data/rawdata/`, can you provide evidence that a gene signature can be stratify high and low tumour cellularity?

Example Agent Output:

Step 1: load in dataset

```
pdac <- readRDS(here("./data/rawdata/PDAC_Haider.rds"))
pdac
```

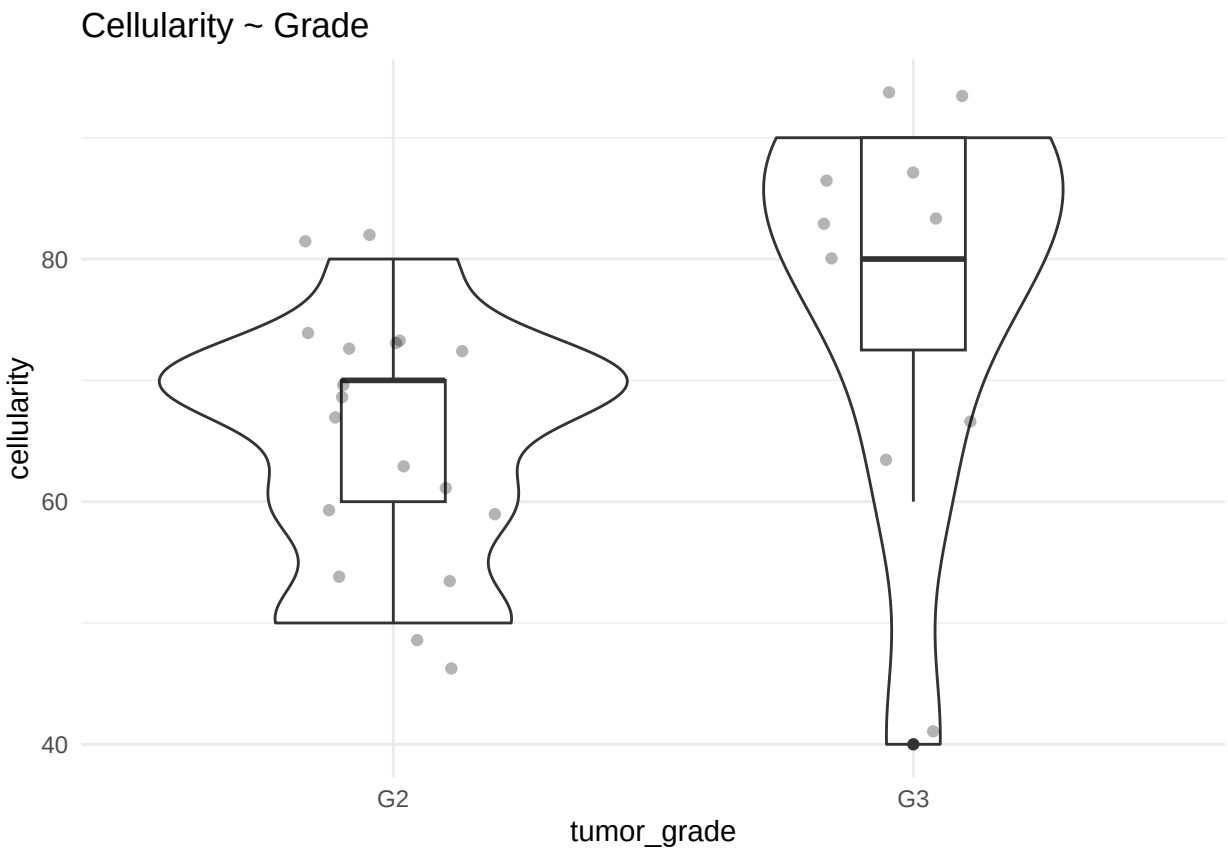
```
## class: RangedSummarizedExperiment
## dim: 16171 28
## metadata(3): '' annotation protocolData
## assays(1): ''
## rownames(16171): ENSG00000237613 ENSG00000273547 ... ENSG00000240450
##   ENSG00000172288
## rowData names(6): gene_id gene_name ... symbol entrezid
## colnames(28): GSM1363848 GSM1363849 ... GSM1363878 GSM1363879
## colData names(8): sample_id cellularity ... sample_type
##   unique_patient_id
```

. . .

Step 2: show the associations between tumour cellularity and tumour grade

Here, we can observe that cells of higher tumour grade tend to also present with greater cellularity.

```
# get clinical metadata of samples
sample_meta <- pdac@colData
sample_meta$cellularity <- as.numeric(sample_meta$cellularity)
sample_meta$age <- as.numeric(sample_meta$age)

# plot distribution of tumour cellularity by tumour grade
ggplot(sample_meta, aes(x = tumor_grade, y = cellularity)) +
  geom_violin() +
  geom_boxplot(width = 0.2) +
  geom_jitter(width = 0.2, alpha = 0.3) +
  theme_minimal() + labs(title = "Cellularity ~ Grade")
```



. . .

Step 3: identify a list of genes that demonstrate association with cellularity

Here are some genes that were identified to be correlated with cellularity. We will use these to attempt and cluster samples based on cellularity.

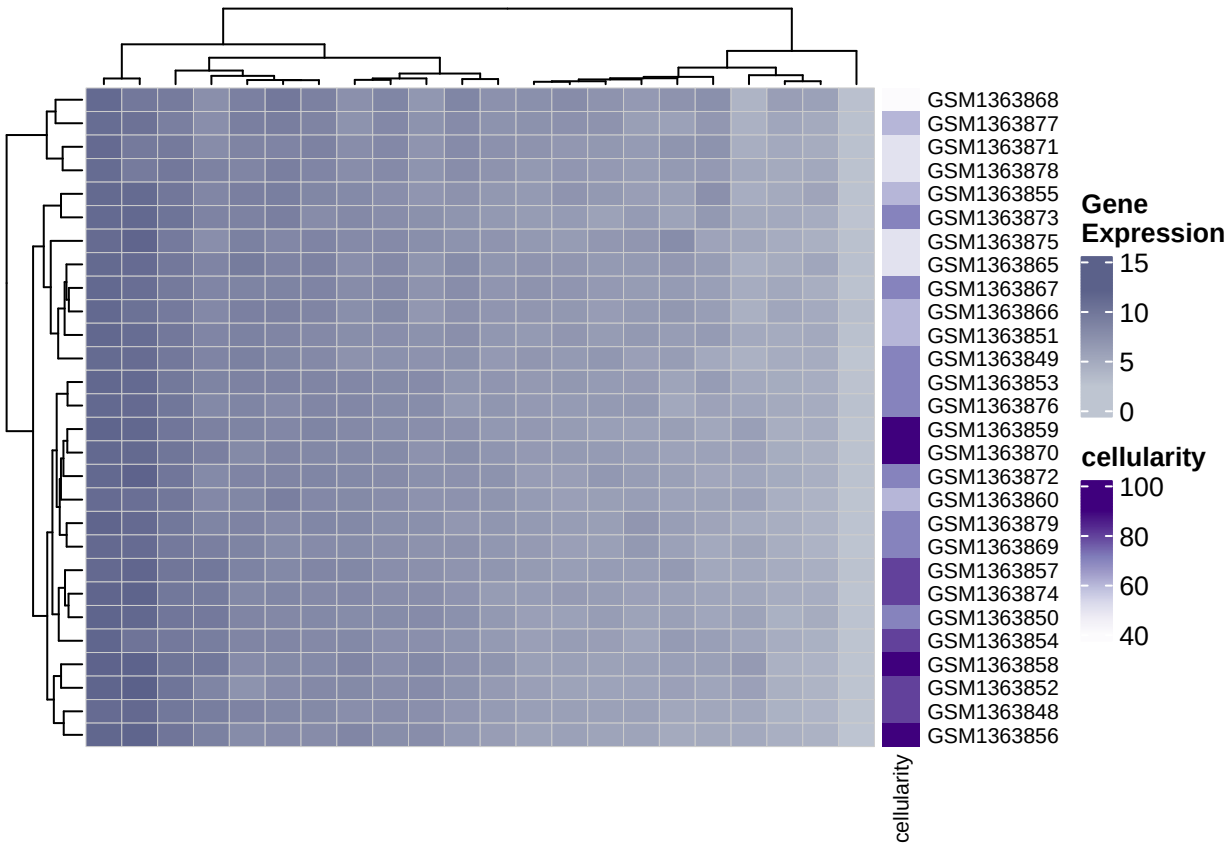
```
# get gene expression
rna <- assay(pdac) |> t() |> as.data.frame()

# compute pearson's correlation coefficient
results <- sapply(rna, function(gene) {
  res <- cor.test(gene, sample_meta$cellularity)
  c(estimate = res$estimate,
    p_value = res$p.value)
}) |> t() |> as.data.frame()
cellularity_genes <- results[abs(results$estimate.cor) > 0.6 & results$p_value < 0.05,]

print(head(cellularity_genes))
```

```
##                 estimate.cor      p_value
## ENSG00000116183   -0.6322636 3.066962e-04
## ENSG00000152578   -0.6746983 8.223576e-05
## ENSG00000139567   -0.6077667 6.026029e-04
## ENSG00000111057    0.6076023 6.052284e-04
## ENSG00000180263    0.6448192 2.120910e-04
## ENSG00000165795   -0.6453598 2.086724e-04
```

Step 4: visualize distribution of identified genes and tumour cellularity

Observe that, using identified genes, we are able to somewhat cluster samples by cellularity. The next step would be to validate this signature in an independent dataset, or perform this analysis on a larger dataset to obtain more confidence results.

```
# keep cellularity genes
cellularity_rna <- rna[,rownames(cellularity_genes)] |> log2()

# plot heatmap of cellularity ~ gene expression
row_ha <- rowAnnotation(
  cellularity = sample_meta$cellularity,
  annotation_name_gp = gpar(fontsize = 8),
  col = list(cellularity = colorRamp2(
    breaks = seq(min(sample_meta$cellularity),
                 max(sample_meta$cellularity),
                 length.out = 9),
    colors = brewer.pal(9, "Purples")
  ))
)
ht <- Heatmap(
    cellularity_rna, name = "Gene\nExpression",
    right_annotation = row_ha,
    show_column_names = FALSE,
    col = c("#BEC5D1", "#5B618A"),
    rect_gp = gpar(col = "grey80", lwd = 0.5),
    row_names_gp = gpar(fontsize = 8)
) |> suppressWarnings()
print(ht)
```


GSM1363868
GSM1363877
GSM1363871
GSM1363878
GSM1363855
GSM1363873
GSM1363875
GSM1363865
GSM1363867
GSM1363866
GSM1363851
GSM1363849
GSM1363853
GSM1363876
GSM1363859
GSM1363870
GSM1363872
GSM1363860
GSM1363879
GSM1363869
GSM1363857
GSM1363874
GSM1363850
GSM1363854
GSM1363858
GSM1363852
GSM1363848
GSM1363856
cellularity
Gene Expression
15
10
5
0
cellularity
100
80
60
40