Agentic AI for Adaptive Pharmacogenomic Biomarker Discovery in Cancer

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.

Example Agent Output:

Step 1: Load ICB data

```
# 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.

```
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
```

```
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.

```
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"))
```

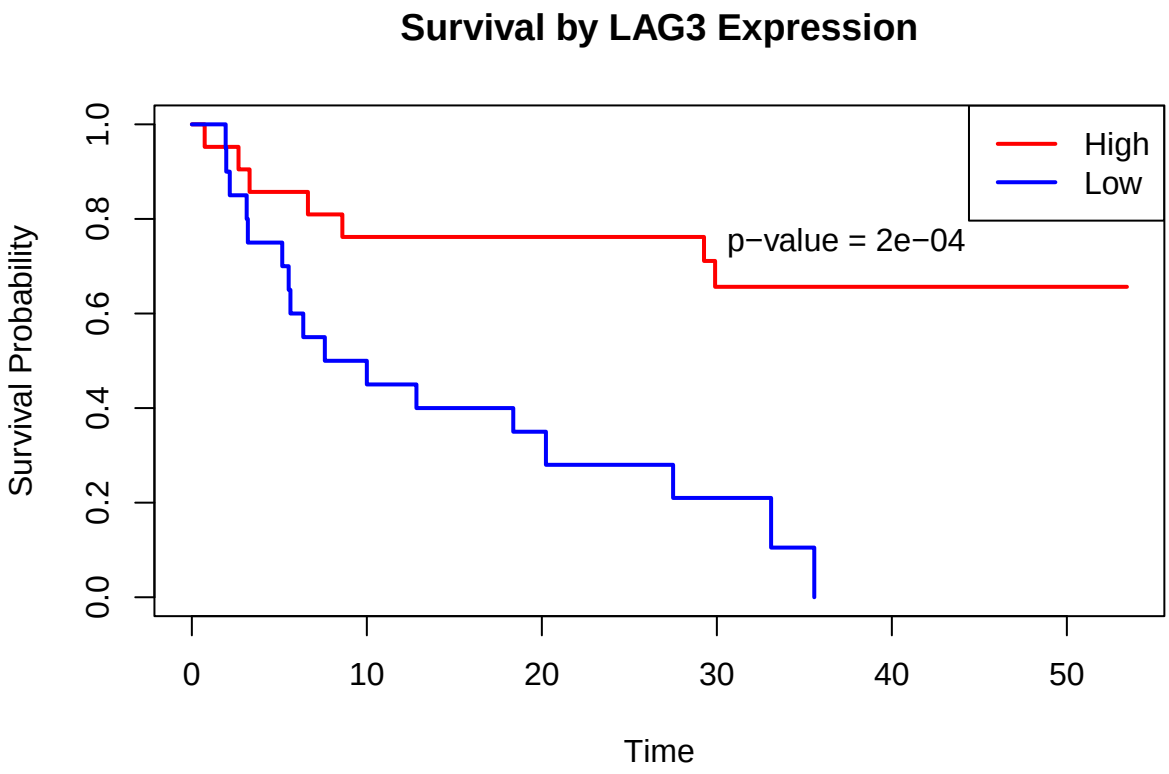
Notes / next steps

- 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
)
```