# ============================================================

# 01) Baseline (no outcomes): summary by group

# Continuous: mean +/- SD (Welch t) if normal in both; else median [IQR] (Wilcoxon)

# Categorical: n (%) + Chi-square (Fisher if any expected cell < 5)

# Derived rows: BMI < 18.5; Albumin <= 30 g/L; Hemoglobin <= 100 g/L

# ============================================================

set.seed(20240908)

options(stringsAsFactors = FALSE)

# --- Packages (install manually if missing) ---

need <- c(

 "tidyverse","janitor","writexl","broom","sandwich","lmtest",

 "glmnet","geepack","emmeans","logistf"

)

miss <- need[!sapply(need, requireNamespace, quietly = TRUE)]

if (length(miss)) stop("Missing packages: ", paste(miss, collapse = ", "))

invisible(lapply(need, function(p) library(p, character.only = TRUE)))

# --- Paths / dirs ---

dir.create("outputs/01\_baseline", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/02\_pih", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/03\_models", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/04\_lasso", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/05\_variab", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/06\_gee", recursive = TRUE, showWarnings = FALSE)

dir.create("outputs/07\_subgroup", recursive = TRUE, showWarnings = FALSE)

path\_baseline <- "data/baseline.csv"

path\_gee <- "data/GEE.csv"

# ============================================================

# 01) Baseline table (no outcomes included)

# - Continuous: mean±SD + Welch's t if normal in both groups; else median[IQR] + Wilcoxon

# - Categorical: n(%) + Chi-square (switch to Fisher if expected cell <5)

# - Derived rows: BMI<18.5, Albumin ≤30 g/L, Hemoglobin ≤100 g/L

# ============================================================

if (!file.exists(path\_baseline)) stop("baseline.csv not found in ./data")

df <- readr::read\_csv(path\_baseline, show\_col\_types = FALSE) |> janitor::clean\_names()

# Group must have two levels

if (!"group" %in% names(df)) stop("'group' column is required in baseline.csv")

if (is.numeric(df$group) || is.integer(df$group)) {

 df <- df |> mutate(group = factor(group, levels = c(1,2), labels = c("Group 1","Group 2")))

} else {

 if (dplyr::n\_distinct(df$group) != 2) stop("'group' must have exactly two levels")

 df <- df |> mutate(group = factor(group))

}

# Optional factor conversions when present

maybe\_factor <- c("sex","asa","hypertension","copd","diabetes","cad",

 "arb\_acei","ccb","diuretic","betablocker","chemo",

 "surgery\_approch","surgery\_approach","field","am\_pm")

for (v in intersect(maybe\_factor, names(df))) df[[v]] <- factor(df[[v]])

# Derived baseline categories (for table display only)

if ("bmi" %in% names(df)) {

 df <- df |> mutate(bmi\_lt\_18\_5 = factor(if\_else(bmi < 18.5, "BMI < 18.5", "BMI ≥ 18.5"),

 levels = c("BMI < 18.5","BMI ≥ 18.5")))

}

if ("alb" %in% names(df)) {

 df <- df |> mutate(alb\_le\_30 = factor(if\_else(alb <= 30, "Albumin ≤ 30", "Albumin > 30"),

 levels = c("Albumin ≤ 30","Albumin > 30")))

}

if ("hb" %in% names(df)) {

 df <- df |> mutate(hb\_le\_100 = factor(if\_else(hb <= 100, "Hemoglobin ≤ 100", "Hemoglobin > 100"),

 levels = c("Hemoglobin ≤ 100","Hemoglobin > 100")))

}

# Treat low-cardinality numerics (≤5 unique) as categorical, excluding "group"

num\_cols <- names(df)[sapply(df, is.numeric)]

num\_small <- num\_cols[sapply(num\_cols, function(x) dplyr::n\_distinct(df[[x]][!is.na(df[[x]])]) <= 5)]

for (v in setdiff(num\_small, "group")) df[[v]] <- factor(df[[v]])

# Lists

is\_cont <- sapply(df, is.numeric)

cont\_vars <- setdiff(names(df)[is\_cont], "group")

is\_cat <- sapply(df, is.factor) | sapply(df, is.character)

cat\_vars <- setdiff(names(df)[is\_cat], "group")

# Helpers

fmt\_mean\_sd <- function(x) sprintf("%.2f \u00B1 %.2f", mean(x, na.rm=TRUE), sd(x, na.rm=TRUE))

fmt\_median\_iqr <- function(x) {

 q <- stats::quantile(x, c(.25,.5,.75), na.rm=TRUE)

 sprintf("%.2f [%.2f\u2013%.2f]", q[2], q[1], q[3]) # median [Q1–Q3]

}

make\_cont\_row <- function(data, var, by) {

 v <- dplyr::pull(data, {{var}}); g <- dplyr::pull(data, {{by}})

 if (!is.factor(g)) g <- factor(g)

 lv <- levels(g)

 x1 <- v[g == lv[1]]; x1 <- x1[is.finite(x1)]

 x2 <- v[g == lv[2]]; x2 <- x2[is.finite(x2)]

 sh1 <- if (length(x1) >= 3) stats::shapiro.test(x1)$p.value else 0

 sh2 <- if (length(x2) >= 3) stats::shapiro.test(x2)$p.value else 0

 normal\_both <- (sh1 > 0.05) & (sh2 > 0.05)

 if (normal\_both) {

 tibble::tibble(

 Variable = as\_label(enquo(var)), Display = "mean \u00B1 SD",

 `Group 1` = fmt\_mean\_sd(x1), `Group 2` = fmt\_mean\_sd(x2), Overall = fmt\_mean\_sd(v),

 `P value` = sprintf("%.3f", stats::t.test(x1, x2, var.equal = FALSE)$p.value)

 )

 } else {

 tibble::tibble(

 Variable = as\_label(enquo(var)), Display = "median [IQR]",

 `Group 1` = fmt\_median\_iqr(x1), `Group 2` = fmt\_median\_iqr(x2), Overall = fmt\_median\_iqr(v),

 `P value` = sprintf("%.3f", stats::wilcox.test(x1, x2)$p.value)

 )

 }

}

make\_cat\_block <- function(data, var, by) {

 tab <- data |> dplyr::count({{by}}, {{var}}) |> dplyr::group\_by({{by}}) |>

 dplyr::mutate(pct = n/sum(n)) |> dplyr::ungroup()

 overall <- data |> dplyr::count({{var}}) |>

 dplyr::mutate({{by}} := factor("Overall",

 levels = c(levels(data[[as\_label(enquo(by))]]), "Overall"))) |>

 dplyr::group\_by({{by}}) |> dplyr::mutate(pct = n/sum(n)) |> dplyr::ungroup()

 mat <- data |> dplyr::count({{by}}, {{var}}) |>

 tidyr::pivot\_wider(names\_from = {{var}}, values\_from = n, values\_fill = 0) |>

 dplyr::select(-{{by}}) |> as.matrix()

 pval <- tryCatch({

 cs <- suppressWarnings(stats::chisq.test(mat, correct = FALSE))

 if (any(cs$expected < 5)) stats::fisher.test(mat)$p.value else cs$p.value

 }, error = function(e) stats::fisher.test(mat)$p.value)

 if (nrow(mat) < 2 || ncol(mat) < 2) pval <- NA\_real\_

 tab |> dplyr::rename(Level = {{var}}) |>

 dplyr::mutate(`n(%)` = sprintf("%d (%.1f%%)", n, pct\*100)) |>

 dplyr::select({{by}}, Level, `n(%)`) |>

 tidyr::pivot\_wider(names\_from = {{by}}, values\_from = `n(%)`) |>

 dplyr::left\_join(

 overall |> dplyr::rename(Level = {{var}}) |>

 dplyr::transmute(Level, Overall = sprintf("%d (%.1f%%)", n, pct\*100)),

 by = "Level"

 ) |>

 dplyr::mutate(Variable = as\_label(enquo(var))) |>

 dplyr::relocate(Variable, Level) |>

 dplyr::mutate(`P value` = ifelse(dplyr::row\_number() == 1,

 ifelse(is.na(pval), "", sprintf("%.3f", pval)), ""))

}

tab\_cont <- purrr::map\_df(cont\_vars, ~ make\_cont\_row(df, all\_of(.x), group))

tab\_cat <- purrr::map\_df(cat\_vars, ~ make\_cat\_block(df, all\_of(.x), group))

writexl::write\_xlsx(list("Table1\_Continuous" = tab\_cont, "Table1\_Categorical" = tab\_cat),

 path = "outputs/01\_baseline/Table1\_baseline.xlsx")

print(tab\_cont, n = nrow(tab\_cont))

print(tab\_cat, n = nrow(tab\_cat))

message("[01] Baseline table saved to outputs/01\_baseline/Table1\_baseline.xlsx")

# ============================================================

# 02) PIH incidence (SBP/MAP) with hemodynamic QC

# Definitions:

# PIH (SBP): min(SBP\_0/5/10/15) < 90 OR >= 30% drop from baseline

# PIH (MAP): min(MAP\_0/5/10/15) < 65 OR >= 20% drop from baseline

# ============================================================

if (!file.exists(path\_gee)) stop("GEE.csv not found in ./data")

dat <- readr::read\_csv(path\_gee, show\_col\_types = FALSE) |> janitor::clean\_names()

# Exposure coding (Fentanyl = reference)

if (!is.factor(dat$group)) {

 dat <- dat |>

 dplyr::mutate(

 group = dplyr::case\_when(

 group %in% c("R","r","Remifentanil","remifentanil",1,"1") ~ "Remifentanil",

 TRUE ~ "Fentanyl"

 ),

 group = factor(group, levels = c("Fentanyl","Remifentanil"))

 )

}

# ---------- Hemodynamic QC (drop implausible values -> NA) ----------

qc\_params <- list(

 sbp\_lo = 40, sbp\_hi = 300,

 map\_lo = 30, map\_hi = 200,

 hr\_lo = 20, hr\_hi = 220

)

qc\_clip\_vec <- function(x, lo, hi) {

 x <- as.numeric(x)

 x[!is.finite(x)] <- NA\_real\_

 x[x <= 0] <- NA\_real\_ # 0/negative -> invalid

 x[x < lo | x > hi] <- NA\_real\_ # outside physiologic bounds -> NA

 x

}

dat\_raw <- dat # keep an untouched copy for comparison

dat\_qc <- dat

# Candidate columns in wide/long structures

hemo\_cols <- intersect(

 names(dat\_qc),

 c("sbp\_base","map\_base","hr\_base",

 "sbp\_0","sbp\_5","sbp\_10","sbp\_15",

 "map\_0","map\_5","map\_10","map\_15",

 "sbp","map","hr")

)

for (nm in hemo\_cols) {

 if (grepl("^sbp(\_|$)", nm)) dat\_qc[[nm]] <- qc\_clip\_vec(dat\_qc[[nm]], qc\_params$sbp\_lo, qc\_params$sbp\_hi)

 if (grepl("^map(\_|$)", nm)) dat\_qc[[nm]] <- qc\_clip\_vec(dat\_qc[[nm]], qc\_params$map\_lo, qc\_params$map\_hi)

 if (grepl("^hr(\_|$)", nm)) dat\_qc[[nm]] <- qc\_clip\_vec(dat\_qc[[nm]], qc\_params$hr\_lo, qc\_params$hr\_hi)

}

# If both SBP and MAP are present at a time point and MAP > SBP, set MAP to NA

times <- c("base","0","5","10","15")

for (t in times) {

 sb <- paste0("sbp\_", t); mp <- paste0("map\_", t)

 if (t == "base") { sb <- "sbp\_base"; mp <- "map\_base" }

 if (all(c(sb, mp) %in% names(dat\_qc))) {

 idx <- which(!is.na(dat\_qc[[mp]]) & !is.na(dat\_qc[[sb]]) & dat\_qc[[mp]] > dat\_qc[[sb]])

 if (length(idx)) dat\_qc[[mp]][idx] <- NA\_real\_

 }

}

# QC summary: how many values were dropped to NA per column

dir.create("outputs/02\_pih", recursive = TRUE, showWarnings = FALSE)

qc\_count <- function(raw, qc, vars) {

 tibble::tibble(

 variable = vars,

 n\_pre = sapply(vars, function(v) sum(!is.na(raw[[v]]))),

 n\_dropped = sapply(vars, function(v) sum(!is.na(raw[[v]]) & is.na(qc[[v]]))),

 pct\_dropped = round(100 \* n\_dropped / pmax(n\_pre, 1), 1)

 )

}

if (length(hemo\_cols)) {

 qc\_summary <- qc\_count(dat\_raw, dat\_qc, hemo\_cols)

 readr::write\_csv(qc\_summary, "outputs/02\_pih/qc\_summary.csv")

}

# ---------- Build PIH flags from wide SBP/MAP columns: raw vs QC ----------

make\_pih\_flags <- function(D) {

 need\_sbp <- c("sbp\_base","sbp\_0","sbp\_5","sbp\_10","sbp\_15")

 need\_map <- c("map\_base","map\_0","map\_5","map\_10","map\_15")

 if (!("pih\_sbp" %in% names(D)) && all(need\_sbp %in% names(D))) {

 D <- D |>

 dplyr::mutate(

 min\_sbp = pmin(sbp\_0, sbp\_5, sbp\_10, sbp\_15, na.rm = TRUE),

 # if all four SBP timepoints are missing -> min\_sbp := NA

 min\_sbp = dplyr::if\_else(

 rowSums(is.na(dplyr::select(D, sbp\_0, sbp\_5, sbp\_10, sbp\_15))) == 4,

 NA\_real\_, min\_sbp

 ),

 sbp\_drop\_pct = (sbp\_base - min\_sbp) / sbp\_base,

 pih\_sbp = as.integer((min\_sbp < 90) | (sbp\_drop\_pct >= 0.30))

 )

 }

 if (!("pih\_map" %in% names(D)) && all(need\_map %in% names(D))) {

 D <- D |>

 dplyr::mutate(

 min\_map = pmin(map\_0, map\_5, map\_10, map\_15, na.rm = TRUE),

 # if all four MAP timepoints are missing -> min\_map := NA

 min\_map = dplyr::if\_else(

 rowSums(is.na(dplyr::select(D, map\_0, map\_5, map\_10, map\_15))) == 4,

 NA\_real\_, min\_map

 ),

 map\_drop\_pct = (map\_base - min\_map) / map\_base,

 pih\_map = as.integer((min\_map < 65) | (map\_drop\_pct >= 0.20))

 )

 }

 D

}

dat\_raw <- make\_pih\_flags(dat\_raw)

dat\_qc <- make\_pih\_flags(dat\_qc)

prop\_ci <- function(x, n) {

 pt <- stats::prop.test(x, n)

 tibble::tibble(n = n, events = x, prop = as.numeric(pt$estimate),

 ci\_low = unname(pt$conf.int[1]), ci\_high = unname(pt$conf.int[2]))

}

inc\_tbl <- function(D, which = c("SBP","MAP")) {

 out <- list()

 if ("SBP" %in% which && "pih\_sbp" %in% names(D)) {

 a <- D |>

 dplyr::group\_by(group) |>

 dplyr::summarise(n\_nonmiss = sum(!is.na(pih\_sbp)),

 events = sum(pih\_sbp == 1, na.rm = TRUE), .groups = "drop") |>

 dplyr::rowwise() |>

 dplyr::mutate(tmp = list(prop\_ci(events, n\_nonmiss))) |>

 tidyr::unnest\_wider(tmp) |>

 dplyr::mutate(metric = "PIH (SBP)") |>

 dplyr::relocate(metric)

 out$sbp <- a

 }

 if ("MAP" %in% which && "pih\_map" %in% names(D)) {

 b <- D |>

 dplyr::group\_by(group) |>

 dplyr::summarise(n\_nonmiss = sum(!is.na(pih\_map)),

 events = sum(pih\_map == 1, na.rm = TRUE), .groups = "drop") |>

 dplyr::rowwise() |>

 dplyr::mutate(tmp = list(prop\_ci(events, n\_nonmiss))) |>

 tidyr::unnest\_wider(tmp) |>

 dplyr::mutate(metric = "PIH (MAP)") |>

 dplyr::relocate(metric)

 out$map <- b

 }

 dplyr::bind\_rows(out)

}

inc\_raw <- inc\_tbl(dat\_raw, c("SBP","MAP"))

inc\_qc <- inc\_tbl(dat\_qc, c("SBP","MAP"))

readr::write\_csv(inc\_raw, "outputs/02\_pih/pih\_incidence\_by\_group\_raw.csv")

readr::write\_csv(inc\_qc, "outputs/02\_pih/pih\_incidence\_by\_group\_qc.csv")

inc\_cmp <- inc\_qc |>

 dplyr::rename(n\_nonmiss\_qc = n\_nonmiss, events\_qc = events,

 prop\_qc = prop, ci\_low\_qc = ci\_low, ci\_high\_qc = ci\_high) |>

 dplyr::left\_join(

 inc\_raw |>

 dplyr::rename(n\_nonmiss\_raw = n\_nonmiss, events\_raw = events,

 prop\_raw = prop, ci\_low\_raw = ci\_low, ci\_high\_raw = ci\_high),

 by = c("metric","group")

 )

readr::write\_csv(inc\_cmp, "outputs/02\_pih/pih\_incidence\_compare.csv")

message("[02] Incidence (raw & QC) saved to outputs/02\_pih/")

# From here on, analyses use QC data only.

dat <- dat\_qc

# ============================================================

# 03) Multivariable logistic models (DAG covariates; QC data)

# ============================================================

dag\_covars <- c("age","bmi","hbp","cad","am\_pm")

tidy\_coeftest <- function(ct) {

 out <- as.data.frame(ct) |> tibble::rownames\_to\_column("term")

 names(out) <- c("term","estimate","std.error","statistic","p.value")

 tibble::as\_tibble(out)

}

# SBP model

df\_sbp <- dat |> dplyr::select(dplyr::all\_of(c("pih\_sbp","group", dag\_covars))) |> tidyr::drop\_na()

fit\_sbp <- stats::glm(pih\_sbp ~ group + age + bmi + hbp + cad + am\_pm, data = df\_sbp, family = stats::binomial())

tab\_sbp <- broom::tidy(fit\_sbp, conf.int = TRUE, conf.level = 0.95, exponentiate = TRUE) |>

 dplyr::mutate(dplyr::across(c(estimate, conf.low, conf.high), ~ round(.x, 3)),

 p.value = signif(p.value, 3))

readr::write\_csv(tab\_sbp, "outputs/03\_models/logit\_sbp\_or\_ci.csv")

rob\_sbp <- lmtest::coeftest(fit\_sbp, vcov = sandwich::vcovHC(fit\_sbp, type = "HC0"))

tidy\_coeftest(rob\_sbp) |> readr::write\_csv("outputs/03\_models/logit\_sbp\_robust\_se.csv")

# MAP model

df\_map <- dat |> dplyr::select(dplyr::all\_of(c("pih\_map","group", dag\_covars))) |> tidyr::drop\_na()

fit\_map <- stats::glm(pih\_map ~ group + age + bmi + hbp + cad + am\_pm, data = df\_map, family = stats::binomial())

tab\_map <- broom::tidy(fit\_map, conf.int = TRUE, conf.level = 0.95, exponentiate = TRUE) |>

 dplyr::mutate(dplyr::across(c(estimate, conf.low, conf.high), ~ round(.x, 3)),

 p.value = signif(p.value, 3))

readr::write\_csv(tab\_map, "outputs/03\_models/logit\_map\_or\_ci.csv")

rob\_map <- lmtest::coeftest(fit\_map, vcov = sandwich::vcovHC(fit\_map, type = "HC0"))

tidy\_coeftest(rob\_map) |> readr::write\_csv("outputs/03\_models/logit\_map\_robust\_se.csv")

message("[03] Logistic models (QC) saved.")

# ============================================================

# 04) LASSO (primary SBP PIH, QC) + refit

# ============================================================

if (!("arb\_acei" %in% names(dat))) {

 if (all(c("arb","acei") %in% names(dat))) {

 dat <- dat |> dplyr::mutate(arb\_acei = as.integer((arb %in% c(1,"1",TRUE)) | (acei %in% c(1,"1",TRUE))))

 } else if ("arb" %in% names(dat)) {

 dat <- dat |> dplyr::mutate(arb\_acei = as.integer(arb %in% c(1,"1",TRUE)))

 } else if ("acei" %in% names(dat)) {

 dat <- dat |> dplyr::mutate(arb\_acei = as.integer(acei %in% c(1,"1",TRUE)))

 }

}

candidates <- intersect(

 c("group","age","sex","bmi","hbp","cad","am\_pm","chemo","arb\_acei","ccb","alb","hb","sbp\_base","map\_base"),

 names(dat)

)

df\_lasso <- dat |> dplyr::select(dplyr::all\_of(c("pih\_sbp", candidates))) |> tidyr::drop\_na()

if (nrow(df\_lasso) < 10) stop("Too few complete cases for LASSO.")

x <- model.matrix(pih\_sbp ~ ., data = df\_lasso)[, -1]

y <- df\_lasso$pih\_sbp

set.seed(20240908)

cvfit <- glmnet::cv.glmnet(x, y, family = "binomial", alpha = 1, nfolds = 10, standardize = TRUE)

lambda\_min <- cvfit$lambda.min; lambda\_1se <- cvfit$lambda.1se

coef\_min <- coef(cvfit, s = lambda\_min)

sel\_terms <- rownames(coef\_min)[as.numeric(coef\_min) != 0]

sel\_terms <- setdiff(sel\_terms, "(Intercept)")

orig\_cols <- intersect(sel\_terms, colnames(df\_lasso))

if (!"group" %in% orig\_cols && "group" %in% colnames(df\_lasso)) orig\_cols <- c("group", orig\_cols)

if (!length(orig\_cols)) orig\_cols <- "group"

form\_lasso <- as.formula(paste("pih\_sbp ~", paste(orig\_cols, collapse = " + ")))

fit\_lasso <- stats::glm(form\_lasso, data = df\_lasso, family = stats::binomial())

tab\_lasso <- broom::tidy(fit\_lasso, conf.int = TRUE, conf.level = 0.95, exponentiate = TRUE) |>

 dplyr::mutate(dplyr::across(c(estimate, conf.low, conf.high), ~ round(.x, 3)), p.value = signif(p.value, 3))

dir.create("outputs/04\_lasso", recursive = TRUE, showWarnings = FALSE)

readr::write\_csv(tibble::tibble(lambda\_min = lambda\_min, lambda\_1se = lambda\_1se), "outputs/04\_lasso/lasso\_lambdas.csv")

readr::write\_csv(tibble::tibble(selected\_design\_terms = sel\_terms), "outputs/04\_lasso/lasso\_selected\_terms\_design\_matrix.csv")

readr::write\_csv(tibble::tibble(selected\_original\_columns = orig\_cols), "outputs/04\_lasso/lasso\_selected\_original\_columns.csv")

readr::write\_csv(tab\_lasso, "outputs/04\_lasso/lasso\_glm\_or\_ci.csv")

png("outputs/04\_lasso/lasso\_cv\_curve.png", width = 1400, height = 900, res = 300); plot(cvfit); dev.off()

png("outputs/04\_lasso/lasso\_coef\_path.png", width = 1400, height = 900, res = 300)

plot(glmnet::glmnet(x, y, family = "binomial", alpha = 1, standardize = TRUE)); dev.off()

message("[04] LASSO results (QC) saved.")

# ============================================================

# 05) SBP variability 0-15 min (ARV & CV) + group tests

# ============================================================

need\_sbp4 <- c("sbp\_0","sbp\_5","sbp\_10","sbp\_15")

if (!all(need\_sbp4 %in% names(dat))) {

 warning("[05] Variability skipped: sbp\_0/5/10/15 not all present.")

} else {

 row\_arv <- function(v) { v <- v[is.finite(v)]; if (length(v) < 2) return(NA\_real\_); mean(abs(diff(v))) }

 sbp\_mat <- dat |> dplyr::select(dplyr::all\_of(need\_sbp4))

 sbp\_mean <- apply(sbp\_mat, 1, function(z) mean(z, na.rm = TRUE))

 sbp\_sd <- apply(sbp\_mat, 1, function(z) sd(z, na.rm = TRUE))

 sbp\_arv <- apply(sbp\_mat, 1, row\_arv)

 sbp\_cv <- 100 \* (sbp\_sd / sbp\_mean)

 dat\_var <- dat |> dplyr::mutate(sbp\_arv\_0\_15 = sbp\_arv, sbp\_cv\_0\_15 = sbp\_cv)

 fmt\_mean\_sd2 <- function(x) sprintf("%.2f +/- %.2f", mean(x, na.rm=TRUE), sd(x, na.rm=TRUE))

 fmt\_med\_iqr2 <- function(x) { q <- stats::quantile(x, c(.25,.5,.75), na.rm=TRUE); sprintf("%.2f [%.2f, %.2f]", q[2], q[1], q[3]) }

 summ\_by\_group <- dat\_var |>

 dplyr::group\_by(group) |>

 dplyr::summarise(n = dplyr::n(),

 ARV\_mean\_SD = fmt\_mean\_sd2(sbp\_arv\_0\_15),

 ARV\_median\_IQR = fmt\_med\_iqr2(sbp\_arv\_0\_15),

 CV\_mean\_SD = fmt\_mean\_sd2(sbp\_cv\_0\_15),

 CV\_median\_IQR = fmt\_med\_iqr2(sbp\_cv\_0\_15),

 .groups = "drop")

 is\_normal <- function(x) { x <- x[is.finite(x)]; if (length(x) < 3) return(FALSE); stats::shapiro.test(x)$p.value > 0.05 }

 x\_arv\_F <- dat\_var$sbp\_arv\_0\_15[dat\_var$group == "Fentanyl"]

 x\_arv\_R <- dat\_var$sbp\_arv\_0\_15[dat\_var$group == "Remifentanil"]

 x\_cv\_F <- dat\_var$sbp\_cv\_0\_15[ dat\_var$group == "Fentanyl"]

 x\_cv\_R <- dat\_var$sbp\_cv\_0\_15[ dat\_var$group == "Remifentanil"]

 arv\_test <- if (is\_normal(x\_arv\_F) & is\_normal(x\_arv\_R)) {

 list(metric = "SBP-ARV (0-15 min)", test = "Welch t-test", p = stats::t.test(x\_arv\_F, x\_arv\_R, var.equal = FALSE)$p.value)

 } else {

 list(metric = "SBP-ARV (0-15 min)", test = "Wilcoxon rank-sum", p = stats::wilcox.test(x\_arv\_F, x\_arv\_R)$p.value)

 }

 cv\_test <- if (is\_normal(x\_cv\_F) & is\_normal(x\_cv\_R)) {

 list(metric = "SBP-CV (0-15 min)", test = "Welch t-test", p = stats::t.test(x\_cv\_F, x\_cv\_R, var.equal = FALSE)$p.value)

 } else {

 list(metric = "SBP-CV (0-15 min)", test = "Wilcoxon rank-sum", p = stats::wilcox.test(x\_cv\_F, x\_cv\_R)$p.value)

 }

 pvals\_tbl <- tibble::tibble(metric = c(arv\_test$metric, cv\_test$metric),

 test = c(arv\_test$test, cv\_test$test),

 p\_value = signif(c(arv\_test$p, cv\_test$p), 3))

 dir.create("outputs/05\_variab", recursive = TRUE, showWarnings = FALSE)

 readr::write\_csv(dat\_var |> dplyr::select(group, sbp\_arv\_0\_15, sbp\_cv\_0\_15), "outputs/05\_variab/sbp\_variability\_rowwise.csv")

 readr::write\_csv(summ\_by\_group, "outputs/05\_variab/sbp\_variability\_group\_summary.csv")

 readr::write\_csv(pvals\_tbl, "outputs/05\_variab/sbp\_variability\_pvalues.csv")

 message("[05] Variability results (QC) saved.")

}

# ============================================================

# 06) GEE: SBP & HR trajectories at 0/5/10/15 (QC data)

# ============================================================

req <- c("patient\_id","time","sbp","hr","age","bmi","hbp","am\_pm","cad","group")

if (!all(req %in% names(dat))) {

 message("[06] GEE skipped (required long-format columns not all present).")

} else {

 dgee <- dat |>

 dplyr::mutate(time = factor(time, levels = c(0,5,10,15), ordered = TRUE),

 hbp = factor(hbp), am\_pm = factor(am\_pm), cad = factor(cad)) |>

 tidyr::drop\_na(patient\_id, time, sbp, hr, group, hbp, am\_pm, bmi, age, cad)

 dir.create("outputs/06\_gee", recursive = TRUE, showWarnings = FALSE)

 gee\_sbp <- geepack::geeglm(sbp ~ group \* time + hbp + am\_pm + bmi + age + cad,

 id = patient\_id, data = dgee, family = gaussian(), corstr = "ar1")

 emm\_sbp <- emmeans::emmeans(gee\_sbp, ~ group | time)

 cmp\_sbp <- emmeans::contrast(emm\_sbp, method = "revpairwise")

 res\_tbl\_sbp <- broom::tidy(cmp\_sbp) |>

 dplyr::transmute(time, contrast,

 estimate = round(estimate, 2),

 SE = round(std.error, 2),

 `95% CI low` = round(estimate - 1.96\*std.error, 2),

 `95% CI high` = round(estimate + 1.96\*std.error, 2),

 p.value = signif(p.value, 3))

 readr::write\_csv(res\_tbl\_sbp, "outputs/06\_gee/gee\_sbp\_diffs.csv")

 gee\_hr <- geepack::geeglm(hr ~ group \* time + hbp + am\_pm + bmi + age + cad,

 id = patient\_id, data = dgee, family = gaussian(), corstr = "ar1")

 emm\_hr <- emmeans::emmeans(gee\_hr, ~ group | time)

 cmp\_hr <- emmeans::contrast(emm\_hr, method = "revpairwise")

 res\_tbl\_hr <- broom::tidy(cmp\_hr) |>

 dplyr::transmute(time, contrast,

 estimate = round(estimate, 2),

 SE = round(std.error, 2),

 `95% CI low` = round(estimate - 1.96\*std.error, 2),

 `95% CI high` = round(estimate + 1.96\*std.error, 2),

 p.value = signif(p.value, 3))

 readr::write\_csv(res\_tbl\_hr, "outputs/06\_gee/gee\_hr\_diffs.csv")

 # Optional plotting omitted (keep your plotting code if needed)

 message("[06] GEE results (QC) saved.")

}

# ============================================================

# 07) Subgroup analysis (SBP only; QC data)

# ============================================================

stopifnot("pih\_sbp" %in% names(dat))

to\_yesno <- function(x) {

 y <- ifelse(is.na(x), NA\_character\_, ifelse(x %in% c(1,"1",TRUE,"TRUE","Yes","yes","Y","y"), "Yes", "No"))

 factor(y, levels = c("Yes","No"))

}

if ("age" %in% names(dat)) dat <- dat |> dplyr::mutate(age\_gt70 = factor(dplyr::if\_else(age > 70, ">70", "<=70"),

 levels = c(">70","<=70")))

if ("hbp" %in% names(dat)) dat$hbp\_fac <- to\_yesno(dat$hbp)

if ("arb\_acei" %in% names(dat)) dat$arb\_acei\_use <- to\_yesno(dat$arb\_acei)

if ("ccb" %in% names(dat)) dat$ccb\_use <- to\_yesno(dat$ccb)

if ("chemo" %in% names(dat)) dat$chemo\_use <- to\_yesno(dat$chemo)

if ("alb" %in% names(dat)) dat$alb\_grp <- factor(dplyr::if\_else(dat$alb <= 35, "<=35 g/L", ">35 g/L"),

 levels = c("<=35 g/L",">35 g/L"))

if ("hb" %in% names(dat)) dat$hb\_grp <- factor(dplyr::if\_else(dat$hb <= 100,"<=100 g/L",">100 g/L"),

 levels = c("<=100 g/L",">100 g/L"))

subgroups <- c("age\_gt70","hbp\_fac","arb\_acei\_use","ccb\_use","chemo\_use","alb\_grp","hb\_grp")

subgroups <- subgroups[subgroups %in% names(dat)]

is\_sparse <- function(df, subgroup, outcome = "pih\_sbp") {

 levs <- levels(df[[subgroup]])

 for (lv in levs) {

 sub <- df[df[[subgroup]] == lv, , drop = FALSE]

 if (nrow(sub) < 15) return(TRUE)

 tab <- table(sub$group, sub[[outcome]])

 if (any(tab == 0)) return(TRUE)

 }

 FALSE

}

rhs\_from <- function(vars) if (length(vars)) paste(vars, collapse = " + ") else "1"

or\_by\_level\_mle <- function(fit, subgroup, levels\_vec) {

 cf <- coef(fit); V <- sandwich::vcovHC(fit, type = "HC0")

 grp2 <- grep("^group", names(cf), value = TRUE)[1]

 purrr::map\_dfr(seq\_along(levels\_vec), function(i){

 lev <- as.character(levels\_vec[i])

 if (i == 1) {

 L <- rep(0, length(cf)); names(L) <- names(cf); L[grp2] <- 1

 } else {

 int\_candidates <- c(paste0(grp2, ":", subgroup, lev), paste0(subgroup, lev, ":", grp2))

 int\_name <- int\_candidates[int\_candidates %in% names(cf)][1]

 L <- rep(0, length(cf)); names(L) <- names(cf); L[grp2] <- 1

 if (!is.na(int\_name)) L[int\_name] <- 1

 }

 logOR <- sum(L\*cf); SE <- sqrt(as.numeric(t(L) %\*% V %\*% L))

 z <- logOR/SE; p <- 2\*pnorm(-abs(z))

 tibble::tibble(level = lev, OR = exp(logOR),

 CI\_low = exp(logOR - 1.96\*SE), CI\_high = exp(logOR + 1.96\*SE), p = p)

 })

}

counts\_display <- function(df, subgroup) {

 tot <- df |> dplyr::count(group, name = "N\_group")

 df |> dplyr::count(group, !!rlang::sym(subgroup), name = "n\_level") |>

 dplyr::left\_join(tot, by = "group") |>

 dplyr::mutate(pct = 100 \* n\_level / N\_group,

 display = sprintf("%d (%.1f%%)", n\_level, pct)) |>

 dplyr::select(group, level = !!rlang::sym(subgroup), display) |>

 tidyr::pivot\_wider(names\_from = group, values\_from = display, values\_fill = "")

}

fit\_firth\_level <- function(df, outcome, covars) {

 fml <- as.formula(paste(outcome, "~", paste(c("group", covars), collapse = " + ")))

 fit <- tryCatch(logistf::logistf(fml, data = df), error = function(e) NULL)

 if (is.null(fit)) return(tibble::tibble(OR = NA\_real\_, CI\_low = NA\_real\_, CI\_high = NA\_real\_, p = NA\_real\_))

 param <- grep("^group", names(fit$coefficients), value = TRUE)[1]

 or <- unname(exp(fit$coefficients[param]))

 ci <- tryCatch(exp(confint(fit, parm = param)), error = function(e) c(NA\_real\_, NA\_real\_))

 p <- unname(fit$prob[param])

 tibble::tibble(OR = or, CI\_low = ci[1], CI\_high = ci[2], p = p)

}

run\_one\_subgroup <- function(data, subgroup, dag\_covars, outcome = "pih\_sbp") {

 keep <- unique(c(outcome, "group", subgroup, dag\_covars))

 df <- data |> dplyr::select(dplyr::all\_of(keep)) |> tidyr::drop\_na()

 if (nrow(df) < 10 || dplyr::n\_distinct(df$group) < 2 || dplyr::n\_distinct(df[[subgroup]]) < 2) return(tibble::tibble())

 if (!is.factor(df[[subgroup]])) df[[subgroup]] <- factor(df[[subgroup]])

 levs <- levels(df[[subgroup]]); dag\_use <- setdiff(dag\_covars, subgroup)

 if (!is\_sparse(df, subgroup, outcome)) {

 f\_base <- as.formula(paste(outcome, "~", rhs\_from(c("group", subgroup, dag\_use))))

 f\_int <- as.formula(paste(outcome, "~", rhs\_from(c(paste0("group\*", subgroup), dag\_use))))

 fit\_base <- stats::glm(f\_base, data = df, family = stats::binomial())

 fit\_int <- stats::glm(f\_int, data = df, family = stats::binomial())

 p\_inter <- tryCatch(tail(anova(fit\_base, fit\_int, test = "LRT")$`Pr(>Chi)`, 1), error = function(e) NA\_real\_)

 or\_tbl <- or\_by\_level\_mle(fit\_int, subgroup, levs) |>

 dplyr::mutate(OR = round(OR, 3), CI\_low = round(CI\_low, 3), CI\_high = round(CI\_high, 3),

 `P value` = signif(p, 3)) |>

 dplyr::select(level, OR, CI\_low, CI\_high, `P value`)

 cnt <- counts\_display(df, subgroup)

 out <- cnt |> dplyr::right\_join(or\_tbl, by = "level") |>

 dplyr::mutate(Subgroup = subgroup,

 `P for interaction` = ifelse(is.na(p\_inter), NA, signif(p\_inter, 3)),

 Method = "Interaction (MLE)") |>

 dplyr::relocate(Subgroup, level, OR, CI\_low, CI\_high, `P value`, `P for interaction`, Method)

 return(out)

 } else {

 rows <- purrr::map\_dfr(levs, function(lv){

 df\_lv <- df |> dplyr::filter(.data[[subgroup]] == lv)

 est <- fit\_firth\_level(df\_lv, outcome, dag\_use)

 cnt <- counts\_display(df\_lv, subgroup) |> dplyr::mutate(level = lv)

 dplyr::left\_join(cnt, est, by = "level") |>

 dplyr::mutate(Subgroup = subgroup, `P for interaction` = NA, Method = "Firth (per level)")

 })

 rows |>

 dplyr::mutate(OR = round(OR, 3), CI\_low = round(CI\_low, 3), CI\_high = round(CI\_high, 3),

 `P value` = signif(p, 3)) |>

 dplyr::relocate(Subgroup, level, OR, CI\_low, CI\_high, `P value`, `P for interaction`, Method)

 }

}

dir.create("outputs/07\_subgroup", recursive = TRUE, showWarnings = FALSE)

res\_sbp <- purrr::map\_dfr(subgroups, ~ run\_one\_subgroup(dat, .x, dag\_covars, outcome = "pih\_sbp"))

readr::write\_csv(res\_sbp, "outputs/07\_subgroup/subgroup\_primary\_sbp.csv")

message("[07] Subgroup (SBP, QC) saved.")

# ============================================================

# Session info

# ============================================================

writeLines(capture.output(sessionInfo()), "outputs/session\_info.txt")

message("All steps completed. See ./outputs/")

# ============================================================

# 02b) Secondary outcomes (incidence only; hospital stay = IQR)

# ============================================================

dir.create("outputs/02\_secondary", recursive = TRUE, showWarnings = FALSE)

stopifnot(file.exists(path\_baseline), file.exists(path\_gee))

bl <- readr::read\_csv(path\_baseline, show\_col\_types = FALSE) |> janitor::clean\_names()

ge <- readr::read\_csv(path\_gee, show\_col\_types = FALSE) |> janitor::clean\_names()

# group standardization

std\_group <- function(d){

 if (!"group" %in% names(d)) stop("'group' column missing.")

 if (!is.factor(d$group)) {

 d <- d |>

 dplyr::mutate(group = dplyr::case\_when(

 group %in% c("R","r","remifentanil","Remifentanil",1,"1") ~ "Remifentanil",

 group %in% c("F","f","fentanyl","Fentanyl",0,"2","0") ~ "Fentanyl",

 TRUE ~ as.character(group)

 ),

 group = factor(group))

 }

 if (all(c("Fentanyl","Remifentanil") %in% levels(d$group)))

 d$group <- factor(d$group, levels = c("Fentanyl","Remifentanil"))

 d

}

bl <- std\_group(bl)

ge <- std\_group(ge)

# helpers

prop\_ci <- function(x, n) { pt <- stats::prop.test(x, n); c(pt$conf.int[1], pt$conf.int[2]) }

qc\_clip <- function(x, lo, hi){ x <- as.numeric(x); x[!is.finite(x)|x<=0] <- NA\_real\_; x[x<lo|x>hi] <- NA\_real\_; x }

find\_col <- function(d, cand) { nm <- names(d); for (c in cand) if (c %in% nm) return(c); NA\_character\_ }

# hospital stay (median[IQR] + Wilcoxon)

cont\_wilcox <- function(df, var, grp="group"){

 d <- df |> dplyr::select(dplyr::all\_of(c(var, grp))) |> tidyr::drop\_na()

 if (!is.factor(d[[grp]])) d[[grp]] <- factor(d[[grp]])

 lv <- levels(d[[grp]]); x1 <- d[[var]][d[[grp]]==lv[1]]; x2 <- d[[var]][d[[grp]]==lv[2]]

 fmt <- function(x){ q <- stats::quantile(x, c(.25,.5,.75), na.rm=TRUE); sprintf("%.2f [%.2f, %.2f]", q[2], q[1], q[3]) }

 tibble::tibble(

 Outcome = var, Test = "Wilcoxon rank-sum",

 `Group 1` = fmt(x1), `Group 2` = fmt(x2), Overall = fmt(c(x1,x2)),

 `P value` = sprintf("%.3f", stats::wilcox.test(x1, x2)$p.value)

 )

}

# binary outcomes: n(%), p-value (Chi-square/Fisher), OR + 95% CI (Fisher)

bin\_incidence <- function(df, outcome, grp="group"){

 dd <- df |> dplyr::filter(!is.na(.data[[grp]])) |>

 dplyr::mutate(y = as.integer(.data[[outcome]] %in% c(1,"1","yes","Yes","Y","y",TRUE)))

 if (length(unique(dd[[grp]])) != 2) return(tibble::tibble())

 agg <- dd |>

 dplyr::group\_by(.data[[grp]]) |>

 dplyr::summarise(n = dplyr::n(), events = sum(y, na.rm=TRUE), prop = events/n, .groups="drop") |>

 dplyr::mutate(

 ci\_low = mapply(function(e,n) prop\_ci(e,n)[1], events, n),

 ci\_high = mapply(function(e,n) prop\_ci(e,n)[2], events, n),

 `n(%)` = sprintf("%d (%.1f%%)", events, 100\*prop)

 )

 tab <- table(dd[[grp]], dd$y)

 pval <- tryCatch({

 cs <- suppressWarnings(stats::chisq.test(tab, correct=FALSE))

 if (any(cs$expected < 5)) stats::fisher.test(tab)$p.value else cs$p.value

 }, error = function(e) stats::fisher.test(tab)$p.value)

 or\_ci <- tryCatch({

 ft <- stats::fisher.test(tab); c(or=unname(ft$estimate), lo=ft$conf.int[1], hi=ft$conf.int[2])

 }, error = function(e) c(or=NA\_real\_, lo=NA\_real\_, hi=NA\_real\_))

 agg |>

 dplyr::mutate(Outcome = outcome) |>

 dplyr::relocate(Outcome) |>

 dplyr::bind\_rows(

 tibble::tibble(

 Outcome = paste0(outcome, " (overall)"),

 `n(%)` = NA\_character\_, !!colnames(agg)[1] := NA, n = sum(agg$n),

 events = sum(agg$events), prop = NA\_real\_, ci\_low = NA\_real\_, ci\_high = NA\_real\_,

 `P value` = sprintf("%.3f", pval), OR = round(or\_ci["or"],3),

 `95% CI (OR)` = ifelse(is.na(or\_ci["lo"]), "", sprintf("%.3f-%.3f", or\_ci["lo"], or\_ci["hi"]))

 )

 )

}

# hospital stay / POC / vasopressor columns

col\_hs <- find\_col(bl, c("hospital\_stay","length\_of\_stay","los"))

col\_poc <- find\_col(bl, c("poc","severe\_cardiovascular\_complication","major\_cv\_complication","stroke\_or\_mi"))

col\_vaso <- find\_col(bl, c("vasopressor","vasopressor\_use","vasopressors","vaso\_use","pressor\_use"))

if (!is.na(col\_hs)) bl[[col\_hs]] <- suppressWarnings(as.numeric(bl[[col\_hs]]))

if (!is.na(col\_vaso)) bl[[col\_vaso]] <- as.integer(bl[[col\_vaso]] %in% c(1,"1","yes","Yes","Y","y",TRUE))

# SBP/HR extremes 0-15 min; baseline SBP

sbp\_lo <- 40; sbp\_hi <- 300; hr\_lo <- 20; hr\_hi <- 220

wide\_sbp <- all(c("sbp\_0","sbp\_5","sbp\_10","sbp\_15") %in% names(ge))

wide\_hr <- all(c("hr\_0","hr\_5","hr\_10","hr\_15") %in% names(ge))

if (wide\_sbp) {

 ge$sbp\_0 <- qc\_clip(ge$sbp\_0, sbp\_lo, sbp\_hi)

 ge$sbp\_5 <- qc\_clip(ge$sbp\_5, sbp\_lo, sbp\_hi)

 ge$sbp\_10 <- qc\_clip(ge$sbp\_10, sbp\_lo, sbp\_hi)

 ge$sbp\_15 <- qc\_clip(ge$sbp\_15, sbp\_lo, sbp\_hi)

 ge <- ge |>

 dplyr::rowwise() |>

 dplyr::mutate(sbp\_max\_0\_15 = {v <- c\_across(c(sbp\_0,sbp\_5,sbp\_10,sbp\_15)); if (all(is.na(v))) NA\_real\_ else max(v, na.rm=TRUE)}) |>

 dplyr::ungroup()

}

if (wide\_hr) {

 ge$hr\_0 <- qc\_clip(ge$hr\_0, hr\_lo, hr\_hi)

 ge$hr\_5 <- qc\_clip(ge$hr\_5, hr\_lo, hr\_hi)

 ge$hr\_10 <- qc\_clip(ge$hr\_10, hr\_lo, hr\_hi)

 ge$hr\_15 <- qc\_clip(ge$hr\_15, hr\_lo, hr\_hi)

 ge <- ge |>

 dplyr::rowwise() |>

 dplyr::mutate(hr\_min\_0\_15 = {v <- c\_across(c(hr\_0,hr\_5,hr\_10,hr\_15)); if (all(is.na(v))) NA\_real\_ else min(v, na.rm=TRUE)}) |>

 dplyr::ungroup()

}

if (!wide\_sbp || !wide\_hr) {

 need <- c("patient\_id","time","sbp","hr")

 if (!all(need %in% names(ge))) stop("Require patient\_id/time/sbp/hr for long format.")

 ge\_long <- ge |> dplyr::select(patient\_id, time, sbp, hr) |> dplyr::filter(time %in% c(0,5,10,15))

 ge\_long$sbp <- qc\_clip(ge\_long$sbp, sbp\_lo, sbp\_hi)

 ge\_long$hr <- qc\_clip(ge\_long$hr, hr\_lo, hr\_hi)

 sbp\_max <- ge\_long |> dplyr::group\_by(patient\_id) |> dplyr::summarise(sbp\_max\_0\_15 = if (all(is.na(sbp))) NA\_real\_ else max(sbp, na.rm=TRUE), .groups="drop")

 hr\_min <- ge\_long |> dplyr::group\_by(patient\_id) |> dplyr::summarise(hr\_min\_0\_15 = if (all(is.na(hr))) NA\_real\_ else min(hr, na.rm=TRUE), .groups="drop")

 ge <- ge |> dplyr::left\_join(sbp\_max, by="patient\_id") |> dplyr::left\_join(hr\_min, by="patient\_id")

}

if (!"sbp\_base" %in% names(ge)) {

 if ("sbp\_0" %in% names(ge)) ge$sbp\_base <- ge$sbp\_0 else {

 sbp0 <- if (exists("ge\_long")) ge\_long |> dplyr::filter(time==0) |> dplyr::select(patient\_id, sbp0=sbp) else NULL

 if (!is.null(sbp0)) ge <- ge |> dplyr::left\_join(sbp0, by="patient\_id") |> dplyr::mutate(sbp\_base = sbp0)

 }

}

if ("sbp\_base" %in% names(ge)) ge$sbp\_base <- qc\_clip(ge$sbp\_base, sbp\_lo, sbp\_hi)

ge <- ge |>

 dplyr::mutate(

 hyper\_sbp = dplyr::case\_when(

 is.finite(sbp\_max\_0\_15) & is.finite(sbp\_base) ~ as.integer(sbp\_max\_0\_15 > 180 | (sbp\_max\_0\_15 - sbp\_base)/sbp\_base >= 0.20),

 is.finite(sbp\_max\_0\_15) & !is.finite(sbp\_base) ~ as.integer(sbp\_max\_0\_15 > 180),

 TRUE ~ NA\_integer\_

 ),

 brady\_hr = dplyr::if\_else(is.finite(hr\_min\_0\_15), as.integer(hr\_min\_0\_15 < 50), NA\_integer\_)

 )

# compute tables

sec\_dir <- "outputs/02\_secondary"

hs\_tbl <- if (!is.na(col\_hs)) cont\_wilcox(bl, col\_hs, "group") else tibble::tibble()

poc\_tbl <- if (!is.na(col\_poc)) bin\_incidence(bl, col\_poc, "group") else tibble::tibble()

hyper\_tbl<- if ("hyper\_sbp" %in% names(ge)) bin\_incidence(ge, "hyper\_sbp", "group") else tibble::tibble()

brady\_tbl<- if ("brady\_hr" %in% names(ge)) bin\_incidence(ge, "brady\_hr", "group") else tibble::tibble()

if (is.na(col\_vaso)) {

 col\_vaso\_ge <- find\_col(ge, c("vasopressor","vasopressor\_use","vasopressors","vaso\_use","pressor\_use"))

 if (!is.na(col\_vaso\_ge)) ge[[col\_vaso\_ge]] <- as.integer(ge[[col\_vaso\_ge]] %in% c(1,"1","yes","Yes","Y","y",TRUE))

 vaso\_tbl <- if (!is.na(col\_vaso\_ge)) bin\_incidence(ge, col\_vaso\_ge, "group") else tibble::tibble()

} else {

 vaso\_tbl <- bin\_incidence(bl, col\_vaso, "group")

}

if (nrow(hs\_tbl)) readr::write\_csv(hs\_tbl, file.path(sec\_dir, "hospital\_stay\_by\_group.csv"))

if (nrow(poc\_tbl)) readr::write\_csv(poc\_tbl, file.path(sec\_dir, "poc\_by\_group.csv"))

if (nrow(hyper\_tbl)) readr::write\_csv(hyper\_tbl, file.path(sec\_dir, "hypertension\_by\_group.csv"))

if (nrow(brady\_tbl)) readr::write\_csv(brady\_tbl, file.path(sec\_dir, "bradycardia\_by\_group.csv"))

if (exists("vaso\_tbl") && nrow(vaso\_tbl)) readr::write\_csv(vaso\_tbl, file.path(sec\_dir, "vasopressor\_use\_by\_group.csv"))

sheets <- list()

if (nrow(hs\_tbl)) sheets[["Hospital\_stay"]] <- hs\_tbl

if (nrow(poc\_tbl)) sheets[["POC"]] <- poc\_tbl

if (nrow(hyper\_tbl)) sheets[["Hypertension\_SBP"]] <- hyper\_tbl

if (nrow(brady\_tbl)) sheets[["Bradycardia\_HR"]] <- brady\_tbl

if (exists("vaso\_tbl") && nrow(vaso\_tbl)) sheets[["Vasopressor\_use"]] <- vaso\_tbl

if (length(sheets)) writexl::write\_xlsx(sheets, path = file.path(sec\_dir, "secondary\_outcomes.xlsx"))