ABG-VBG Analysis

Brian Locke, Anila Mehta

Table of contents

1 Data Pre-processing **1**
1.0.1 Package Set Up . . . 1
1.1 Helper functions for model diagnostics . . . 1
1.1.1 Configuration for the IPW models . . . 2
1.1.2 Outcome Variable Creation . . . 5
1.2 2) Baseline tables . . . 6
1.3 3) Three-level PCO2 categories (unweighted) . . . 16
1.4 4) Restricted cubic spline regressions (unweighted) . . . 20
1.4.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2 . . . 21
1.4.2 4.2 Unweighted, Restricted Cubic Spline - VBG . . . 23

2 Inverse Propensity Weighting **29**
2.0.1 5.1 ABG IPW weighting and diagnostics . . . 29
2.0.2 5.2 ABG IPW spline models . . . 31
2.0.3 5.3 ABG IPW spline models (2–98th percentile) . . . 33
2.0.4 5.4 VBG IPW weighting and spline models . . . 37
2.0.5 5.5 Three-level PCO2 categories (weighted; ABG, VBG) . . . 43
2.1 6) Propensity score diagnostics . . . 47

3 Multiple Imputation Analysis **50**
3.0.1 7.2 Missingness structure and drivers . . . 50
3.0.2 7.3 Monte Carlo error check after MI . . . 52
3.1 8) Pre-imputation data prep (consistent types & predictors) . . . 52

3.2 9) Imputation model specification (MICE) . . . 52
3.2.1 9.1 Predictor matrix & methods. Run MICE (moderate settings for scale) . . . 52
3.3 10) Refit propensity models within each imputation . . . 84
3.3.1 FAIL-FAST CHECKS . . . 84
3.3.2 10.1 ABG propensity (has_abg) . . . 84
3.3.3 10.2 Balance diagnostics across imputations . . . 84
3.3.4 10.3 VBG propensity (has_vbg) . . . 86
3.3.5 10.4 VBG balance . . . 87
3.4 11) Weighted outcome models within each imputation + pooling . . . 88
3.4.1 11.1 Helper: fit + extract log-OR and SE from svyglm . . . 89
3.4.2 11.3 VBG: MI pooled spline models (treated cohort only) . . . 89
3.5 12) Explainability on one representative imputation . . . 89
3.6 13) MI three-level PCO2 helpers and checks . . . 89
3.7 14) MI + IPW three-level PCO2 (ABG & VBG) . . . 90
3.7.1 14.1 ABG: MI + IPW, three-level PCO2 outcomes . . . 90
3.7.2 14.2 VBG: MI + IPW, three-level PCO2 outcomes . . . 90
3.7.3 14.3 Table 3: MI-pooled IPW associations (3-level CO) . . . 91
3.7.4 14.4 Summary: adjusted CO2-category associations across analysis tracks . . . 91
3.8 Manuscript outputs summary . . . 92
3.8.1 14.3 Visualization: pooled three-level ORs . . . 97
3.8.2 15.3 Visualization . . . 99
3.9 Diagnostics . . . 102
3.9.1 MI convergence and mixing . . . 102
3.9.2 MI stability across m . . . 102
3.9.3 MI maxit sensitivity (sampled) . . . 102
3.9.4 Balance diagnostics . . . 102
3.9.5 Outcome diagnostics . . . 104
3.9.6 Diagnostics summary and audit . . . 104
3.9.7 Performance / runtime log . . . 104
3.9.8 Performance / runtime log . . . 106
3.10 16) Save, export, and session info . . . 107

1 Data Pre-processing

This code pulls in the master database (a STATA file) and does some initial cleaning - this will only need to be run once, and then the data can be accessed in the usual way.

1.0.1 Package Set Up

`Cohort flow diagram will be generated externally and inserted here in the final manuscript packet.`

1.1 Helper functions for model diagnostics

Converts the data from a STATA format to rdata if the rdata file does not exist. If it does already exist, it just loads that.

```
# data_dir_name resolved in setup-packages from params/env
stata_file <- file.path(data_dir_name, "full_db.dta")

stata_data <- read_dta(stata_file)
var_labels <- var_label(stata_data)
value_labels <- lapply(stata_data, function(x) if (is.labelled(x)) val_labels(x))
saveRDS(
  list(var_labels = var_labels, value_labels = value_labels),
  results_path("stata_labels.rds")
)
```

1.1.1 Configuration for the IPW models

```
drop_vars_ultra_missing <- c("bnp", "spo2")
cat_vars <- c("sex", "race_ethnicity", "location", "encounter_type")
numeric_vars <- c(
  "age_at_encounter", "curr_bmi", "temp_new", "sbp", "dbp", "hr",
  "sodium", "serum_cr", "serum_hco3", "serum_cl", "serum_lac", "serum_k",
  "wbc", "plt", "serum_phos", "serum_ca"
)
```

```r
co2_vars <- c("paco2", "vbg_co2", "vbg_o2sat")

covars_gbm <- c(
  "age_at_encounter","sex","race_ethnicity","curr_bmi",
  "copd","asthma","osa","chf","acute_nmd","phtn","ckd","dm",
  "location","encounter_type","temp_new","sbp","dbp","hr",
  "sodium","serum_cr","serum_hco3","serum_cl","serum_lac","serum_k",
  "wbc","plt","serum_phos","serum_ca"
)
covars_gbm <- setdiff(covars_gbm, drop_vars_ultra_missing)
covars_ps <- covars_gbm

# Core adjustment set for conditional prognostic models
adj_core <- c("age_at_encounter", "sex", "race_ethnicity", "location", "encounter_type")

gbm_params <- list(
  n.trees           = 800,
  interaction.depth = 3,
  shrinkage         = 0.01,
  bag.fraction      = 0.8,
  n.minobsinnode    = 10,
  cv.folds          = 0,
  stop.method       = "smd.max",
  n.cores           = 1L
)
stopifnot(gbm_params$stop.method == "smd.max")
SPLINE_BASIS <- "ns"
SPLINE_DF <- 4L
stopifnot(SPLINE_BASIS %in% c("ns", "rcs"))
get_gbm_cores <- function() {
  n_rows <- nrow(subset_data)
  if (is.finite(n_rows) && n_rows > 200000L) return(1L)
  gbm_params$n.cores
}
ps_trunc_quantile <- 0.01
stopifnot(ps_trunc_quantile > 0, ps_trunc_quantile < 0.5)
```

```r
formula_abg      <- reformulate(covars_gbm, response = "has_abg")
formula_vbg      <- reformulate(covars_gbm, response = "has_vbg")

# Model diagrams: propensity models (GBM PS)
register_model_diagram("PS model: ABG test (GBM)", formula_abg, width = 10, height = 7)
register_model_diagram("PS model: VBG test (GBM)", formula_vbg, width = 10, height = 7)

# TODO: consider stop.method = "es.mean" (ATS version) if smd.max remains unstable on full N.
```

```r
run_meta <- tibble::tibble(
  run_id     = diag_run_id,
  run_mode   = RUN_MODE,
  pilot_frac = PILOT_FRAC,
  mi_batch_threshold = MI_BATCH_THRESHOLD,
  full_n     = nrow(stata_data),
  subset_n   = nrow(subset_data)
)
render_table_pdf(run_meta,
                 "Run metadata (pilot vs full)",
                 "run_metadata",
                 preview_rows = 5,
                 digits = 0)
```

Table 1: Run metadata (pilot vs full)

run_id	run_mode	pilot_frac	mi_batch_threshold	full_n	subset_n
20260204_072236	pilot	0	5000	833476	25852

```r
# Write run config JSON for portability
json_escape <- function(x) gsub("\"", "\\\"", x)
run_cfg <- list(
  run_id = diag_run_id,
  run_mode = RUN_MODE,
  pilot_frac = PILOT_FRAC,
```

```
  mi_batch_threshold = MI_BATCH_THRESHOLD,
  data_dir = data_dir_name,
  results_dir = results_dir,
  full_n = nrow(stata_data),
  subset_n = nrow(subset_data)
)
json_lines <- c(
  "{",
  paste0("  \"run_id\": \"", json_escape(run_cfg$run_id), "\","),
  paste0("  \"run_mode\": \"", json_escape(run_cfg$run_mode), "\","),
  paste0("  \"pilot_frac\": ", run_cfg$pilot_frac, ","),
  paste0("  \"mi_batch_threshold\": ", run_cfg$mi_batch_threshold, ","),
  paste0("  \"data_dir\": \"", json_escape(run_cfg$data_dir), "\","),
  paste0("  \"results_dir\": \"", json_escape(run_cfg$results_dir), "\","),
  paste0("  \"full_n\": ", run_cfg$full_n, ","),
  paste0("  \"subset_n\": ", run_cfg$subset_n),
  "}"
)
writeLines(json_lines, results_path("run_config.json"))
```

Codebook exported to Results/codebookr.docx.

1.1.2 Outcome Variable Creation

```
subset_data <- subset_data %>%
  mutate(
    ## 1. Did the patient die?
    died = if_else(!is.na(death_date), 1L, 0L),

    ## 2. Absolute death date (if death_date is an offset)
    death_abs = if_else(!is.na(death_date),
                        encounter_date + death_date,
                        as.Date(NA)),
```

```r
    ## 3. Year month (YM) for encounter and death
    enc_ym   = floor_date(encounter_date, unit = "month"),
    death_ym = floor_date(death_abs      , unit = "month"),

    ## 4. Reference censoring date: 1 Jun 2024
    ref_ym = ymd("2024-06-01"),

    ## 5. Months from encounter to death or censoring
    months_death_or_cens = case_when(
      !is.na(death_ym) ~ interval(enc_ym, death_ym) %/% months(1),
      TRUE             ~ interval(enc_ym, ref_ym)   %/% months(1)
    ),

    ## 6. Remove impossible values
    months_death_or_cens = if_else(
      months_death_or_cens < 0 | months_death_or_cens > 16,
      NA_integer_, months_death_or_cens
    ),

    ## 7. Death within one or two months
    died_1mo = if_else(died == 1 & months_death_or_cens <  1, 1L, 0L),
    died_2mo = if_else(died == 1 & months_death_or_cens <= 1, 1L, 0L),

    ## 8. Month of death (missing if censored)
    death_time = if_else(died == 1, months_death_or_cens, NA_integer_),

    ## 9. Death within 60 days (new variable)
    death_60d = if_else(died == 1 & death_abs <= (encounter_date + days(60)), 1L, 0L)
  ) %>%
  select(-enc_ym, -death_ym)

subset_data <- subset_data %>%
  mutate(
    death_60d = if_else(died == 1 & death_abs <= (encounter_date + days(60)), 1L, 0L)
  )
```

1.2 2) Baseline tables

```r
# Robust derivation of analysis variables + helper for Table 1 production
# ------------------------------------------------------------------------------

# helper: label binary 0/1 → "No"/"Yes"
bin_lab <- function(x) {
  y <- to01(x)
  if (all(is.na(y))) {
    return(factor(y, levels = c(0, 1), labels = c("No", "Yes")))
  }
  factor(y, levels = c(0, 1), labels = c("No", "Yes"))
}

# helper: preserve labeled factors if already present; otherwise map numeric codes
label_from_codes <- function(x, codes, labels) {
  if (is.factor(x)) {
    lev <- levels(x)
    if (all(lev %in% labels)) {
      return(factor(x, levels = labels))
    }
    lev_num <- suppressWarnings(as.numeric(lev))
    if (all(!is.na(lev_num)) && all(lev_num %in% codes)) {
      return(factor(as.numeric(as.character(x)), levels = codes, labels = labels))
    }
    return(x)
  }
  x_num <- suppressWarnings(as.numeric(as.character(x)))
  if (all(is.na(x_num))) return(factor(x, levels = labels))
  if (all(x_num %in% codes, na.rm = TRUE)) {
    return(factor(x_num, levels = codes, labels = labels))
  }
  factor(x)
}
```

```r
subset_data <- subset_data %>%
  mutate(
    ## ensure 0/1 numerics (avoids factor-level coercion)
    across(c(has_abg, has_vbg),
           ~ to01(.)),

    ## derive ABG / VBG status groups (binary test status only)
    abg_group = case_when(
      has_abg == 0 ~ "No ABG",
      has_abg == 1 ~ "ABG",
      TRUE         ~ "Missing"
    ),
    vbg_group = case_when(
      has_vbg == 0 ~ "No VBG",
      has_vbg == 1 ~ "VBG",
      TRUE         ~ "Missing"
    ),

    ## factorise groups with explicit NA/Missing level
    abg_group = factor(
      abg_group,
      levels = c("No ABG", "ABG", "Missing")
    ),
    vbg_group = factor(
      vbg_group,
      levels = c("No VBG", "VBG", "Missing")
    ),

    ## labelled covariates (robust to factor or numeric codes)
    sex_label = label_from_codes(sex, c(0, 1), c("Female", "Male")),
    race_ethnicity_label = label_from_codes(
      race_ethnicity,
      0:6,
      c("White", "Black or African American", "Hispanic",
        "Asian", "American Indian", "Pacific Islander", "Unknown")
    ),
```

```
    location_label = label_from_codes(
      location,
      0:3,
      c("South", "Northeast", "Midwest", "West")
    ),
    encounter_type_label = label_from_codes(
      encounter_type,
      c(2, 3),
      c("Emergency", "Inpatient")
    ),
    osa_label      = bin_lab(osa),
    asthma_label   = bin_lab(asthma),
    copd_label     = bin_lab(copd),
    chf_label      = bin_lab(chf),
    nmd_label      = bin_lab(nmd),
    phtn_label     = bin_lab(phtn),
    ckd_label      = bin_lab(ckd),
    diabetes_label = bin_lab(dm)
  )

# variables to summarise
vars <- c(
  "age_at_encounter", "curr_bmi", "sex_label", "race_ethnicity_label", "location_label",
  "osa_label", "asthma_label", "copd_label", "chf_label", "nmd_label",
  "phtn_label", "ckd_label", "diabetes_label", "encounter_type_label", "vbg_co2", "paco2"
)
vars_baseline <- setdiff(vars, c("vbg_co2", "paco2"))
vars_abg <- c(vars_baseline, "paco2")
vars_vbg <- c(vars_baseline, "vbg_co2")

# Table 1 constructor
make_table1 <- function(data, group_var, caption = "") {
  group_sym <- rlang::sym(group_var)

  df <- data %>%
    filter(!is.na(!!group_sym),                    # drop explicit NA
```

```
          !!group_sym != "Missing") %>%          # drop "Missing" cohort
    mutate(!!group_sym := droplevels(!!group_sym)) %>%  # only drop group levels
    select(all_of(c(group_var, vars_baseline)))

  empty_fac <- names(which(vapply(df, function(z) is.factor(z) && length(levels(z)) == 0L, logical(1))))
  if (length(empty_fac) > 0) {
    warning("0-level factor columns detected: ", paste(empty_fac, collapse = ", "),
            ". Converting to character to prevent gtsummary failure.", call. = FALSE)
    df[empty_fac] <- lapply(df[empty_fac], as.character)
  }

  df %>%
    gtsummary::tbl_summary(
      by   = !!group_sym,
      type = list(sex_label ~ "categorical"),
      statistic = list(
        gtsummary::all_continuous()  ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
        gtsummary::all_categorical() ~ "{n} ({p}%)"
      ),
      digits   = list(gtsummary::all_continuous() ~ 1),
      missing  = "no"                              # no gtsummary missing column/row
    ) %>%
    gtsummary::modify_header(label = "**Variable**") %>%
    gtsummary::modify_caption(strip_manual_table_number(caption))
}

if (sum(!is.na(subset_data$sex_label)) == 0L || length(levels(subset_data$sex_label)) == 0L) {
  warning("sex_label is all NA or has zero levels; check sex normalization/mapping.", call. = FALSE)
  stopifnot("sex" %in% names(subset_data))
}

# build tables
table1A <- make_table1(subset_data, "abg_group", caption = "Table 1A: ABG cohorts")
table1B <- make_table1(subset_data, "vbg_group", caption = "Table 1B: VBG cohorts")

tbl1a_pdf <- to_pdf_table(table1A, font_size = 7, landscape = FALSE, label_col_width = "2.0in",
```

```
                          longtable = TRUE)
tbl1b_pdf <- to_pdf_table(table1B, font_size = 7, landscape = FALSE, label_col_width = "2.0in",
                          longtable = TRUE)
tbl1a_pdf
```

Table 2: ABG cohorts

Variable	**No ABG** N = 16,490	**ABG** N = 9,362
age_at_encounter	58.2 ± 18.1; 0.0/16,490.0 missing (0.0%)	61.1 ± 16.9; 0.0/9,362.0 missing (0.0%)
curr_bmi	32.5 ± 8.7; 9,223.0/16,490.0 missing (55.9%)	29.0 ± 7.1; 5,500.0/9,362.0 missing (58.7%)
sex_label		
Female	8,598 (52%)	4,218 (45%)
Male	7,892 (48%)	5,144 (55%)
race_ethnicity_label		
White	10,061 (61%)	6,090 (65%)
Black or African American	3,149 (19%)	1,387 (15%)
Hispanic	1,168 (7.1%)	500 (5.3%)
Asian	252 (1.5%)	179 (1.9%)
American Indian	88 (0.5%)	101 (1.1%)
Pacific Islander	25 (0.2%)	14 (0.1%)
Unknown	1,747 (11%)	1,091 (12%)
location_label		
South	6,962 (42%)	5,171 (55%)
Northeast	4,707 (29%)	1,831 (20%)
Midwest	1,143 (6.9%)	783 (8.4%)
West	3,678 (22%)	1,577 (17%)
osa_label	3,047 (18%)	1,455 (16%)
asthma_label	2,439 (15%)	1,086 (12%)
copd_label	3,037 (18%)	2,052 (22%)
chf_label	2,964 (18%)	2,104 (22%)
nmd_label	599 (3.6%)	428 (4.6%)
phtn_label	1,231 (7.5%)	897 (9.6%)
ckd_label	2,825 (17%)	1,837 (20%)
diabetes_label	4,838 (29%)	2,776 (30%)
encounter_type_label		
Emergency	7,055 (43%)	1,451 (15%)
Inpatient	9,435 (57%)	7,911 (85%)

```
tbl1b_pdf
```

Table 3: VBG cohorts

Variable	**No VBG** N = 18,392	**VBG** N = 7,460
age_at_encounter	59.4 ± 17.7; 0.0/18,392.0 missing (0.0%)	58.9 ± 17.7; 0.0/7,460.0 missing (0.0%)
curr_bmi	31.9 ± 8.5; 9,691.0/18,392.0 missing (52.7%)	29.0 ± 7.5; 5,032.0/7,460.0 missing (67.5%)
sex_label		
Female	9,331 (51%)	3,485 (47%)
Male	9,061 (49%)	3,975 (53%)
race_ethnicity_label		
White	12,190 (66%)	3,961 (53%)
Black or African American	3,109 (17%)	1,427 (19%)
Hispanic	1,149 (6.2%)	519 (7.0%)
Asian	279 (1.5%)	152 (2.0%)
American Indian	91 (0.5%)	98 (1.3%)
Pacific Islander	33 (0.2%)	6 (<0.1%)
Unknown	1,541 (8.4%)	1,297 (17%)
location_label		
South	9,882 (54%)	2,251 (30%)
Northeast	3,358 (18%)	3,180 (43%)
Midwest	1,241 (6.7%)	685 (9.2%)
West	3,911 (21%)	1,344 (18%)
osa_label	3,273 (18%)	1,229 (16%)
asthma_label	2,520 (14%)	1,005 (13%)
copd_label	3,596 (20%)	1,493 (20%)
chf_label	3,495 (19%)	1,573 (21%)
nmd_label	757 (4.1%)	270 (3.6%)
phtn_label	1,405 (7.6%)	723 (9.7%)
ckd_label	3,119 (17%)	1,543 (21%)
diabetes_label	5,137 (28%)	2,477 (33%)
encounter_type_label		
Emergency	6,183 (34%)	2,323 (31%)
Inpatient	12,209 (66%)	5,137 (69%)

```
float_barrier()
```

```
# Status factors (column labels are taken from factor levels)
subset_data <- subset_data %>%
  mutate(
    abg_status = factor(has_abg, levels = c(0, 1),
                        labels = c("Did not get ABG", "Did get ABG")),
    vbg_status = factor(has_vbg, levels = c(0, 1),
                        labels = c("Did not get VBG", "Did get VBG"))
  )
```

```r
# ABG table with "Everyone" column first
tbl1_abg <- subset_data %>%
  select(all_of(vars_baseline), abg_status) %>%
  gtsummary::tbl_summary(
    by = abg_status,
    type = list(sex_label ~ "categorical"),
    statistic = list(
      gtsummary::all_continuous()  ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
      gtsummary::all_categorical() ~ "{n} ({p}%)"
    ),
    digits   = list(gtsummary::all_continuous() ~ 1),
    missing  = "no"
  ) %>%
  gtsummary::add_overall(last = FALSE, col_label = "Everyone") %>%
  gtsummary::modify_header(label = "**Variable**")

# VBG table (no "Everyone" here)
tbl1_vbg <- subset_data %>%
  select(all_of(vars_baseline), vbg_status) %>%
  gtsummary::tbl_summary(
    by = vbg_status,
    type = list(sex_label ~ "categorical"),
    statistic = list(
      gtsummary::all_continuous()  ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
      gtsummary::all_categorical() ~ "{n} ({p}%)"
    ),
    digits   = list(gtsummary::all_continuous() ~ 1),
    missing  = "no"
  ) %>%
  gtsummary::modify_header(label = "**Variable**")

library(gtsummary)

tbl1 <- tbl_merge(
  tbls = list(tbl1_abg, tbl1_vbg)
) %>%
```

```
  modify_caption(strip_manual_table_number("**Table 1. Baseline summary: Everyone, ABG status, and VBG status**"))

tbl1_pdf <- to_pdf_table(tbl1, font_size = 7, landscape = FALSE, label_col_width = "2.0in",
                         longtable = TRUE)
tbl1_pdf
```

Table 4: Baseline summary: Everyone, ABG status, and VBG status**

Variable	Everyone	**Did not get ABG** N = 16,490	**Did get ABG** N = 9,362	**Did not get
age_at_encounter	59.2 ± 17.7; 0.0/25,852.0 missing (0.0%)	58.2 ± 18.1; 0.0/16,490.0 missing (0.0%)	61.1 ± 16.9; 0.0/9,362.0 missing (0.0%)	59.4 ± 17.7; 0.0/1
curr_bmi	31.3 ± 8.4; 14,723.0/25,852.0 missing (57.0%)	32.5 ± 8.7; 9,223.0/16,490.0 missing (55.9%)	29.0 ± 7.1; 5,500.0/9,362.0 missing (58.7%)	31.9 ± 8.5; 9,691.0/
sex_label				
Female	12,816 (50%)	8,598 (52%)	4,218 (45%)	9,33
Male	13,036 (50%)	7,892 (48%)	5,144 (55%)	9,06
race_ethnicity_label				
White	16,151 (62%)	10,061 (61%)	6,090 (65%)	12,19
Black or African American	4,536 (18%)	3,149 (19%)	1,387 (15%)	3,10
Hispanic	1,668 (6.5%)	1,168 (7.1%)	500 (5.3%)	1,14
Asian	431 (1.7%)	252 (1.5%)	179 (1.9%)	279
American Indian	189 (0.7%)	88 (0.5%)	101 (1.1%)	91
Pacific Islander	39 (0.2%)	25 (0.2%)	14 (0.1%)	33
Unknown	2,838 (11%)	1,747 (11%)	1,091 (12%)	1,54
location_label				
South	12,133 (47%)	6,962 (42%)	5,171 (55%)	9,88
Northeast	6,538 (25%)	4,707 (29%)	1,831 (20%)	3,35
Midwest	1,926 (7.5%)	1,143 (6.9%)	783 (8.4%)	1,24
West	5,255 (20%)	3,678 (22%)	1,577 (17%)	3,91
osa_label	4,502 (17%)	3,047 (18%)	1,455 (16%)	3,27
asthma_label	3,525 (14%)	2,439 (15%)	1,086 (12%)	2,52
copd_label	5,089 (20%)	3,037 (18%)	2,052 (22%)	3,59
chf_label	5,068 (20%)	2,964 (18%)	2,104 (22%)	3,49
nmd_label	1,027 (4.0%)	599 (3.6%)	428 (4.6%)	757
phtn_label	2,128 (8.2%)	1,231 (7.5%)	897 (9.6%)	1,40
ckd_label	4,662 (18%)	2,825 (17%)	1,837 (20%)	3,11
diabetes_label	7,614 (29%)	4,838 (29%)	2,776 (30%)	5,13
encounter_type_label				
Emergency	8,506 (33%)	7,055 (43%)	1,451 (15%)	6,18
Inpatient	17,346 (67%)	9,435 (57%)	7,911 (85%)	12,20

```
float_barrier()
```

```r
# ABG cohort (has_abg == 1)
tbl2_abg <- subset_data %>%
  filter(has_abg == 1) %>%
  select(all_of(vars_abg), pco2_cat_abg) %>%
  gtsummary::tbl_summary(
    by = pco2_cat_abg,
    type = list(sex_label ~ "categorical"),
    statistic = list(
      gtsummary::all_continuous()  ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
      gtsummary::all_categorical() ~ "{n} ({p}%)"
    ),
    digits   = list(gtsummary::all_continuous() ~ 1),
    missing  = "no"
  ) %>%
  gtsummary::modify_header(
    label  = "**Variable**",
    stat_1 = "**Normal**",
    stat_2 = "**Low**",
    stat_3 = "**High**"
  )

# VBG cohort (has_vbg == 1)
tbl2_vbg <- subset_data %>%
  filter(has_vbg == 1) %>%
  select(all_of(vars_vbg), pco2_cat_vbg) %>%
  gtsummary::tbl_summary(
    by = pco2_cat_vbg,
    type = list(sex_label ~ "categorical"),
    statistic = list(
      gtsummary::all_continuous()  ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
      gtsummary::all_categorical() ~ "{n} ({p}%)"
    ),
    digits   = list(gtsummary::all_continuous() ~ 1),
    missing  = "no"
  ) %>%
  gtsummary::modify_header(
```

```
    label  = "**Variable**",
    stat_1 = "**Normal**",
    stat_2 = "**Low**",
    stat_3 = "**High**"
  )

tbl2 <- gtsummary::tbl_merge(
  tbls = list(tbl2_abg, tbl2_vbg),
  tab_spanner = c("**ABG (PaCO2)**", "**VBG (PvCO2)**")
) %>%
  gtsummary::modify_caption(strip_manual_table_number("**Table 2. Baseline summary by CO2 category within ABG and VBG cohorts**"

tbl2_pdf <- to_pdf_table(tbl2, font_size = 7, landscape = FALSE, label_col_width = "2.0in",
                         longtable = TRUE)
tbl2_pdf
```

Table 5: Baseline summary by CO2 category within ABG and VBG cohorts**

Variable	**Normal**	**Low**	**High**	**Normal**
age_at_encounter	60.9 ± 16.9; 0.0/4,229.0 missing (0.0%)	60.4 ± 17.4; 0.0/2,488.0 missing (0.0%)	62.0 ± 16.3; 0.0/2,645.0 missing (0.0%)	58.0 ± 18.0; 0.0/3,427.0 missing (0
curr_bmi	29.0 ± 6.9; 2,440.0/4,229.0 missing (57.7%)	28.3 ± 6.8; 1,493.0/2,488.0 missing (60.0%)	29.8 ± 7.6; 1,567.0/2,645.0 missing (59.2%)	29.1 ± 7.3; 2,398.0/3,427.0 missing (
sex_label				
Female	1,878 (44%)	1,123 (45%)	1,217 (46%)	1,627 (47%)
Male	2,351 (56%)	1,365 (55%)	1,428 (54%)	1,800 (53%)
race_ethnicity_label				
White	2,742 (65%)	1,510 (61%)	1,838 (69%)	1,825 (53%)
Black or African American	609 (14%)	400 (16%)	378 (14%)	648 (19%)
Hispanic	229 (5.4%)	144 (5.8%)	127 (4.8%)	243 (7.1%)
Asian	82 (1.9%)	59 (2.4%)	38 (1.4%)	71 (2.1%)
American Indian	58 (1.4%)	32 (1.3%)	11 (0.4%)	34 (1.0%)
Pacific Islander	8 (0.2%)	5 (0.2%)	1 (<0.1%)	3 (<0.1%)
Unknown	501 (12%)	338 (14%)	252 (9.5%)	603 (18%)
location_label				
South	2,334 (55%)	1,399 (56%)	1,438 (54%)	1,106 (32%)
Northeast	795 (19%)	405 (16%)	631 (24%)	1,510 (44%)
Midwest	373 (8.8%)	184 (7.4%)	226 (8.5%)	312 (9.1%)
West	727 (17%)	500 (20%)	350 (13%)	499 (15%)
osa_label	602 (14%)	291 (12%)	562 (21%)	528 (15%)
asthma_label	474 (11%)	245 (9.8%)	367 (14%)	433 (13%)
copd_label	779 (18%)	374 (15%)	899 (34%)	548 (16%)
chf_label	834 (20%)	503 (20%)	767 (29%)	639 (19%)
nmd_label	210 (5.0%)	105 (4.2%)	113 (4.3%)	114 (3.3%)
phtn_label	346 (8.2%)	207 (8.3%)	344 (13%)	286 (8.3%)
ckd_label	799 (19%)	497 (20%)	541 (20%)	650 (19%)

Table 5: Baseline summary by CO2 category within ABG and VBG cohorts** *(continu*

Variable	**Normal**	**Low**	**High**	**Normal**
diabetes_label	1,211 (29%)	750 (30%)	815 (31%)	1,070 (31%)
encounter_type_label				
Emergency	627 (15%)	345 (14%)	479 (18%)	1,173 (34%)
Inpatient	3,602 (85%)	2,143 (86%)	2,166 (82%)	2,254 (66%)
paco2	39.7 ± 3.0; 0.0/4,229.0 missing (0.0%)	29.4 ± 4.4; 0.0/2,488.0 missing (0.0%)	59.6 ± 20.6; 0.0/2,645.0 missing (0.0%)	
vbg_co2				44.6 ± 3.0; 0.0/3,427.0 missing (0

```
float_barrier()
```

Table 6: Table 2a. Crude outcomes by CO2 category

Cohort	Outcome	Normal	Low	High
ABG	IMV	979/4229 (23.1%)	691/2488 (27.8%)	721/2645 (27.3%)
ABG	NIV	288/4229 (6.8%)	187/2488 (7.5%)	421/2645 (15.9%)
ABG	Death (60d)	604/4229 (14.3%)	537/2488 (21.6%)	504/2645 (19.1%)
ABG	Hypercapnic RF	207/4229 (4.9%)	119/2488 (4.8%)	666/2645 (25.2%)
VBG	IMV	432/3427 (12.6%)	357/2058 (17.3%)	373/1975 (18.9%)
VBG	NIV	175/3427 (5.1%)	122/2058 (5.9%)	257/1975 (13.0%)
VBG	Death (60d)	378/3427 (11.0%)	356/2058 (17.3%)	310/1975 (15.7%)
VBG	Hypercapnic RF	146/3427 (4.3%)	70/2058 (3.4%)	476/1975 (24.1%)

1.3 3) Three-level PCO2 categories (unweighted)

Three groups using low/normal/high CO2 categories

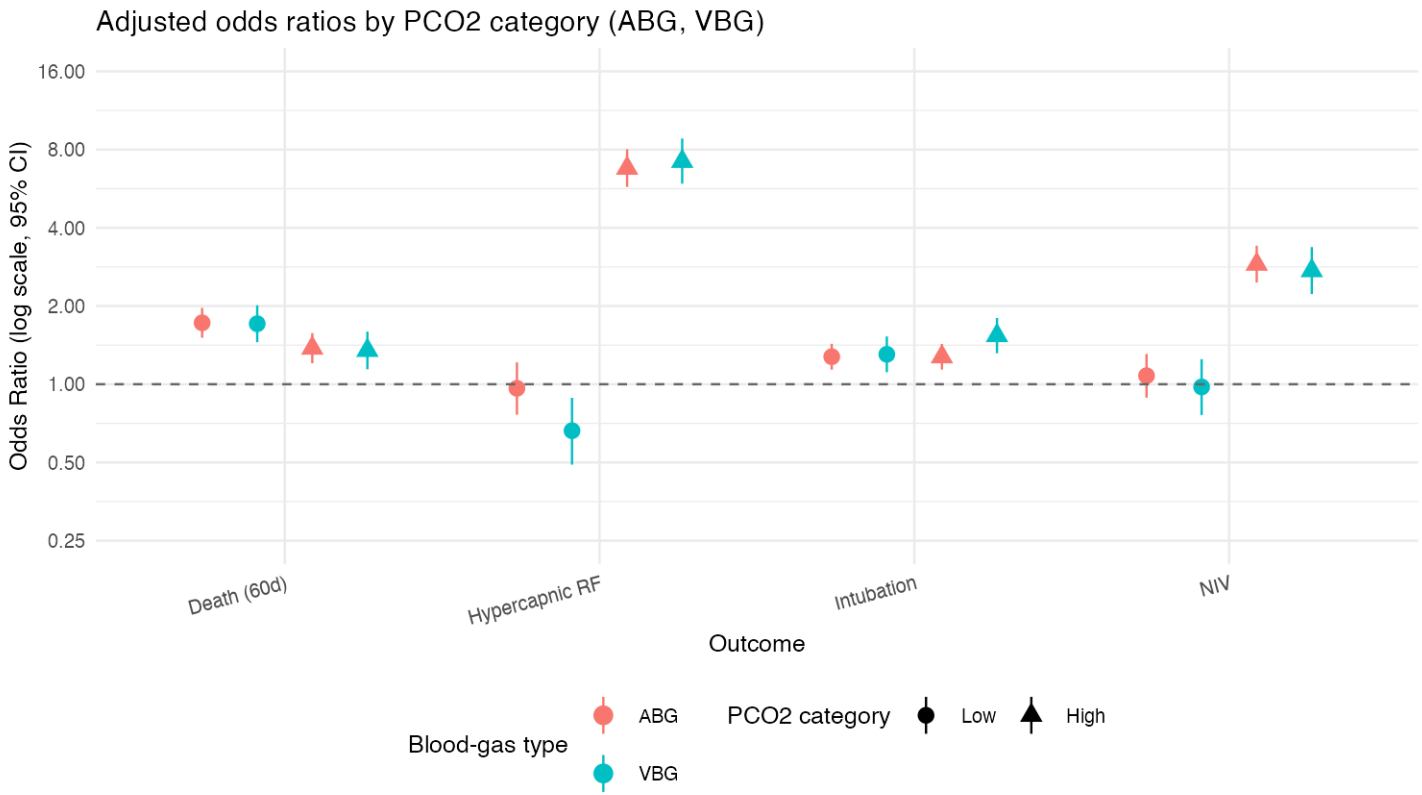
```
stopifnot(all(c("pco2_cat_abg", "pco2_cat_vbg") %in% names(subset_data)))

library(broom)
library(tidyr)
library(dplyr)

run_logit <- function(data, outcome, exposure, group_name, adj_vars = NULL, model_type = "Crude") {
  f <- if (length(adj_vars)) {
    reformulate(c(exposure, adj_vars), response = outcome)
```

```r
  } else {
    as.formula(paste(outcome, "~", exposure))
  }
  fit_res <- fit_with_diagnostics(
    function() glm(f, data = data, family = binomial, control = glm.control(maxit = 50)),
    context = make_context(
      stage = "outcome",
      component = "cat3",
      analysis_variant = "unweighted",
      model_type = "cat3",
      group = group_name,
      outcome = outcome,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  if (is.null(fit_res$fit)) {
    stop("run_logit: model fit failed for outcome=", outcome,
         " exposure=", exposure, " group=", group_name)
  }
  tidy(fit_res$fit, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term != "(Intercept)", startsWith(term, exposure)) %>%
    mutate(
      outcome = outcome,
      group   = group_name,
      model   = model_type
    )
}

outcomes_unw <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

unw_three_level_forms <- list(
  "ABG 3-level: IMV ~ CO2 category + X"      = reformulate(c("pco2_cat_abg", adj_core), response = "imv_proc"),
  "ABG 3-level: NIV ~ CO2 category + X"      = reformulate(c("pco2_cat_abg", adj_core), response = "niv_proc"),
  "ABG 3-level: Death60d ~ CO2 category + X" = reformulate(c("pco2_cat_abg", adj_core), response = "death_60d"),
```

```
  "ABG 3-level: HCRF ~ CO2 category + X"     = reformulate(c("pco2_cat_abg", adj_core), response = "hypercap_resp_failure"),
  "VBG 3-level: IMV ~ CO2 category + X"      = reformulate(c("pco2_cat_vbg", adj_core), response = "imv_proc"),
  "VBG 3-level: NIV ~ CO2 category + X"      = reformulate(c("pco2_cat_vbg", adj_core), response = "niv_proc"),
  "VBG 3-level: Death60d ~ CO2 category + X" = reformulate(c("pco2_cat_vbg", adj_core), response = "death_60d"),
  "VBG 3-level: HCRF ~ CO2 category + X"     = reformulate(c("pco2_cat_vbg", adj_core), response = "hypercap_resp_failure")
)
register_model_diagrams(unw_three_level_forms)

unw_results_crude <- bind_rows(
  lapply(outcomes_unw, function(o) run_logit(subset_data, o, "pco2_cat_abg", "ABG")),
  lapply(outcomes_unw, function(o) run_logit(subset_data, o, "pco2_cat_vbg", "VBG"))
)
unw_results_adj <- bind_rows(
  lapply(outcomes_unw, function(o) run_logit(subset_data, o, "pco2_cat_abg", "ABG", adj_core, "Adjusted")),
  lapply(outcomes_unw, function(o) run_logit(subset_data, o, "pco2_cat_vbg", "VBG", adj_core, "Adjusted"))
)

unw_threelevel_results <- unw_results_adj %>%
  mutate(method = "Unweighted adjusted")

unw_combined_or_df <- unw_results_adj %>%
  mutate(
    outcome = recode(outcome,
                     imv_proc = "Intubation",
                     niv_proc = "NIV",
                     death_60d = "Death (60d)",
                     hypercap_resp_failure = "Hypercapnic RF")
  )
unw_combined_or_df <- map_or_exposure(unw_combined_or_df, "or-plot-three-level-unweighted") |>
  select(outcome, group, exposure, estimate, conf.low, conf.high)
```

```
library(scales)

unw_combined_or_df$group <- factor(
  unw_combined_or_df$group,
  levels = c("ABG", "VBG")
```

```
)

unw_plot_df <- build_or_plot_df(unw_combined_or_df, "or-plot-three-level-unweighted",
                            expected_exposure_levels = CO2_CAT_CONTRAST_LEVELS)
stopifnot(is.data.frame(unw_plot_df))
unw_axis_spec <- compute_or_axis_spec(unw_plot_df, lo_col = "conf.low", hi_col = "conf.high",
                                      default_limits = OR_XLIM)
unw_p_or <- plot_or_safe(
  unw_plot_df,
  plot_name = "or-plot-three-level-unweighted",
  axis_spec = unw_axis_spec,
  title  = "Adjusted odds ratios by PCO2 category (ABG, VBG)",
  caption = paste(
    "Adjusted for age, sex, race/ethnicity, location, and encounter type.",
    "Reference = patients in the normal PCO2 range.",
    "Low: <35 mmHg (ABG) or <40 mmHg (VBG); High: >45 mmHg (ABG) or >50 mmHg (VBG).",
    "Because the underlying cohorts differ (ABG, VBG), denominators are not identical across groups.",
    sep = "\n"
  )
)
print_plot_once(unw_p_or, "or-plot-three-level-unweighted", width = 7.5, height = 4.8)
```



Adjusted for age, sex, race/ethnicity, location, and encounter type.
Reference = patients in the normal PCO2 range.
Low: <35 mmHg (ABG) or <40 mmHg (VBG); High: >45 mmHg (ABG) or >50 mmHg (VBG).
Because the underlying cohorts differ (ABG, VBG), denominators are not identical across groups.

1.4 4) Restricted cubic spline regressions (unweighted)

Spline curves are shown as odds ratios relative to CO2_ref (midpoint of the normal range), holding covariates at the reference profile.

```
# ABG spline dataset
subset_data_abg <- subset_data %>%
  filter(has_abg == 1, !is.na(paco2)) %>%
  select(paco2, imv_proc, niv_proc, death_60d, hypercap_resp_failure, all_of(adj_core)) %>%
  filter(complete.cases(.))
```

1.4.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```
make_spline_fml <- function(outcome, co2_var, adj_vars) {
  spline_term <- if (SPLINE_BASIS == "rcs") {
    sprintf("rms::rcs(%s, %d)", co2_var, SPLINE_DF)
  } else {
    sprintf("splines::ns(%s, %d)", co2_var, SPLINE_DF)
  }
  stats::as.formula(
    paste0(outcome, " ~ ", spline_term,
           if (length(adj_vars)) paste0(" + ", paste(adj_vars, collapse = " + ")) else "")
  )
}

#| code-block-title: "Unweighted ABG spline models (adjusted)"
abg_spline_forms <- list(
  "ABG spline (adjusted): IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "paco2", adj_core),
  "ABG spline (adjusted): NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "paco2", adj_core),
  "ABG spline (adjusted): Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "paco2", adj_core),
  "ABG spline (adjusted): HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "paco2", adj_core)
)
register_model_diagrams(abg_spline_forms)

co2_seq_abg <- stats::quantile(subset_data_abg$paco2, probs = c(0.02, 0.98), na.rm = TRUE)
grid_abg_info_unw <- make_co2_grid_ref(
  "paco2",
  seq(co2_seq_abg[1], co2_seq_abg[2], length.out = SPLINE_GRID_N),
  x_ref_abg,
  ABG_CO2_REF
)
grid_abg_unw <- grid_abg_info_unw$grid
ref_idx_abg_unw <- grid_abg_info_unw$ref_idx
co2_ref_abg_unw <- grid_abg_info_unw$co2_ref

fit_spline_glm <- function(outcome, co2_var, data, group_label) {
```

```r
  fit_res <- fit_with_diagnostics(
    function() glm(make_spline_fml(outcome, co2_var, adj_core),
                   data = data, family = binomial,
                   control = glm.control(maxit = 50)),
    context = make_context(
      stage = "outcome",
      component = "spline",
      analysis_variant = "unweighted",
      model_type = "spline",
      group = group_label,
      outcome = outcome,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  fit_res$fit
}

fit_imv <- fit_spline_glm("imv_proc", "paco2", subset_data_abg, "ABG")
fit_niv <- fit_spline_glm("niv_proc", "paco2", subset_data_abg, "ABG")
fit_death <- fit_spline_glm("death_60d", "paco2", subset_data_abg, "ABG")
fit_hcrf <- fit_spline_glm("hypercap_resp_failure", "paco2", subset_data_abg, "ABG")
if (any(vapply(list(fit_imv, fit_niv, fit_death, fit_hcrf), is.null, logical(1)))) {
  stop("Unweighted ABG spline fits failed; see model_fit_diagnostics.csv.")
}

pred_imv <- predict_or_curve_from_fit(fit_imv, grid_abg_unw, ref_idx_abg_unw, "paco2")
pred_niv <- predict_or_curve_from_fit(fit_niv, grid_abg_unw, ref_idx_abg_unw, "paco2")
pred_death <- predict_or_curve_from_fit(fit_death, grid_abg_unw, ref_idx_abg_unw, "paco2")
pred_hcrf <- predict_or_curve_from_fit(fit_hcrf, grid_abg_unw, ref_idx_abg_unw, "paco2")
## Plotting deferred until VBG curves are computed so axes can be shared.
```

1.4.2 4.2 Unweighted, Restricted Cubic Spline - VBG

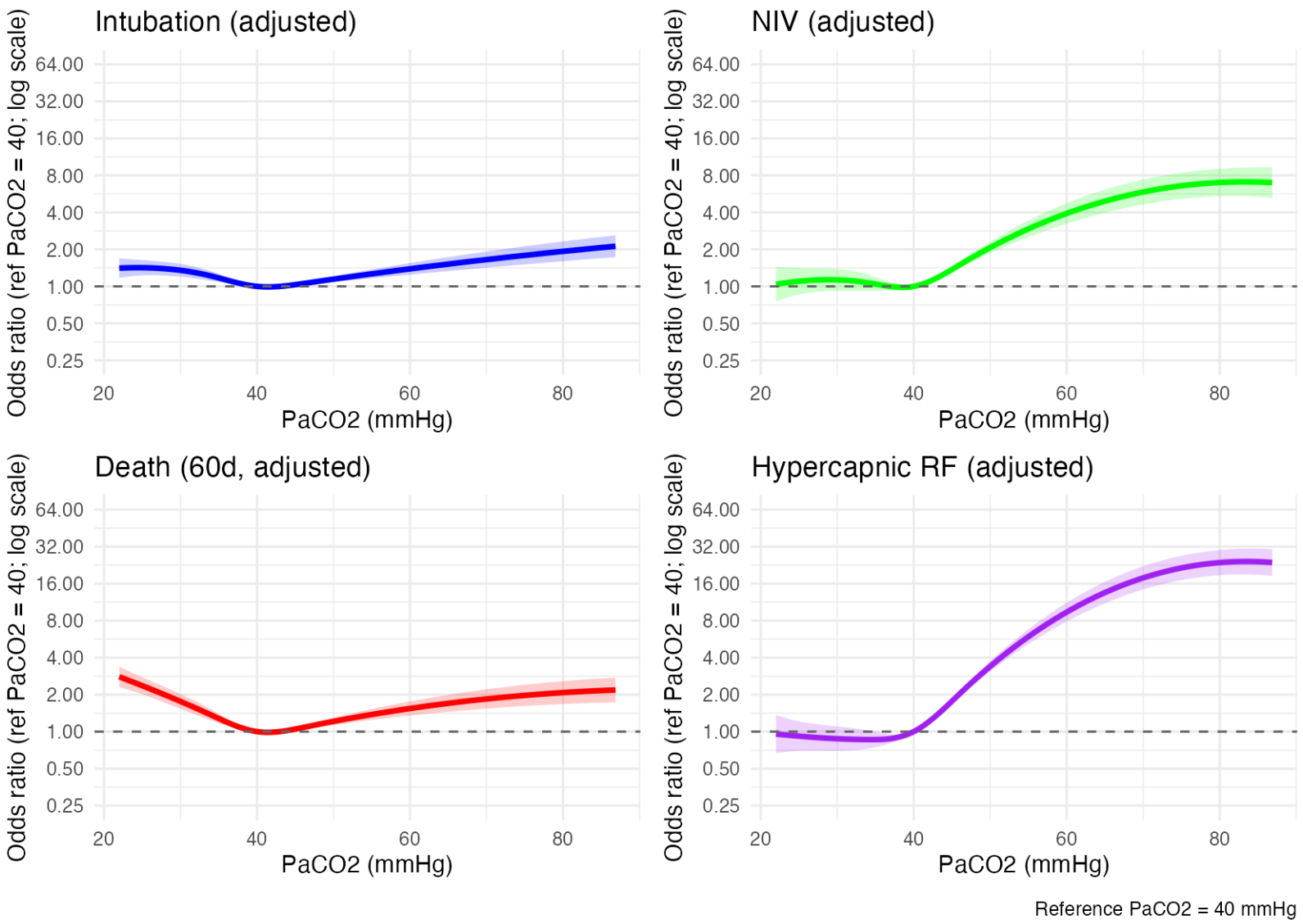
```
# --- VBG dataset ---
subset_data_vbg <- subset_data %>%
  dplyr::filter(has_vbg == 1, !is.na(vbg_co2)) %>%
  dplyr::select(vbg_co2, imv_proc, niv_proc, death_60d, hypercap_resp_failure, all_of(adj_core)) %>%
  dplyr::filter(complete.cases(.))
```

```
vbg_spline_forms <- list(
  "VBG spline (adjusted): IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "vbg_co2", adj_core),
  "VBG spline (adjusted): NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "vbg_co2", adj_core),
  "VBG spline (adjusted): Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "vbg_co2", adj_core),
  "VBG spline (adjusted): HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "vbg_co2", adj_core)
)
register_model_diagrams(vbg_spline_forms)

co2_seq_vbg <- stats::quantile(subset_data_vbg$vbg_co2, probs = c(0.02, 0.98), na.rm = TRUE)
grid_vbg_info_unw <- make_co2_grid_ref(
  "vbg_co2",
  seq(co2_seq_vbg[1], co2_seq_vbg[2], length.out = SPLINE_GRID_N),
  x_ref_vbg,
  VBG_CO2_REF
)
grid_vbg_unw <- grid_vbg_info_unw$grid
ref_idx_vbg_unw <- grid_vbg_info_unw$ref_idx
co2_ref_vbg_unw <- grid_vbg_info_unw$co2_ref

fit_imv_vbg <- fit_spline_glm("imv_proc", "vbg_co2", subset_data_vbg, "VBG")
fit_niv_vbg <- fit_spline_glm("niv_proc", "vbg_co2", subset_data_vbg, "VBG")
fit_death_vbg <- fit_spline_glm("death_60d", "vbg_co2", subset_data_vbg, "VBG")
fit_hcrf_vbg <- fit_spline_glm("hypercap_resp_failure", "vbg_co2", subset_data_vbg, "VBG")
if (any(vapply(list(fit_imv_vbg, fit_niv_vbg, fit_death_vbg, fit_hcrf_vbg), is.null, logical(1)))) {
  stop("Unweighted VBG spline fits failed; see model_fit_diagnostics.csv.")
}
```

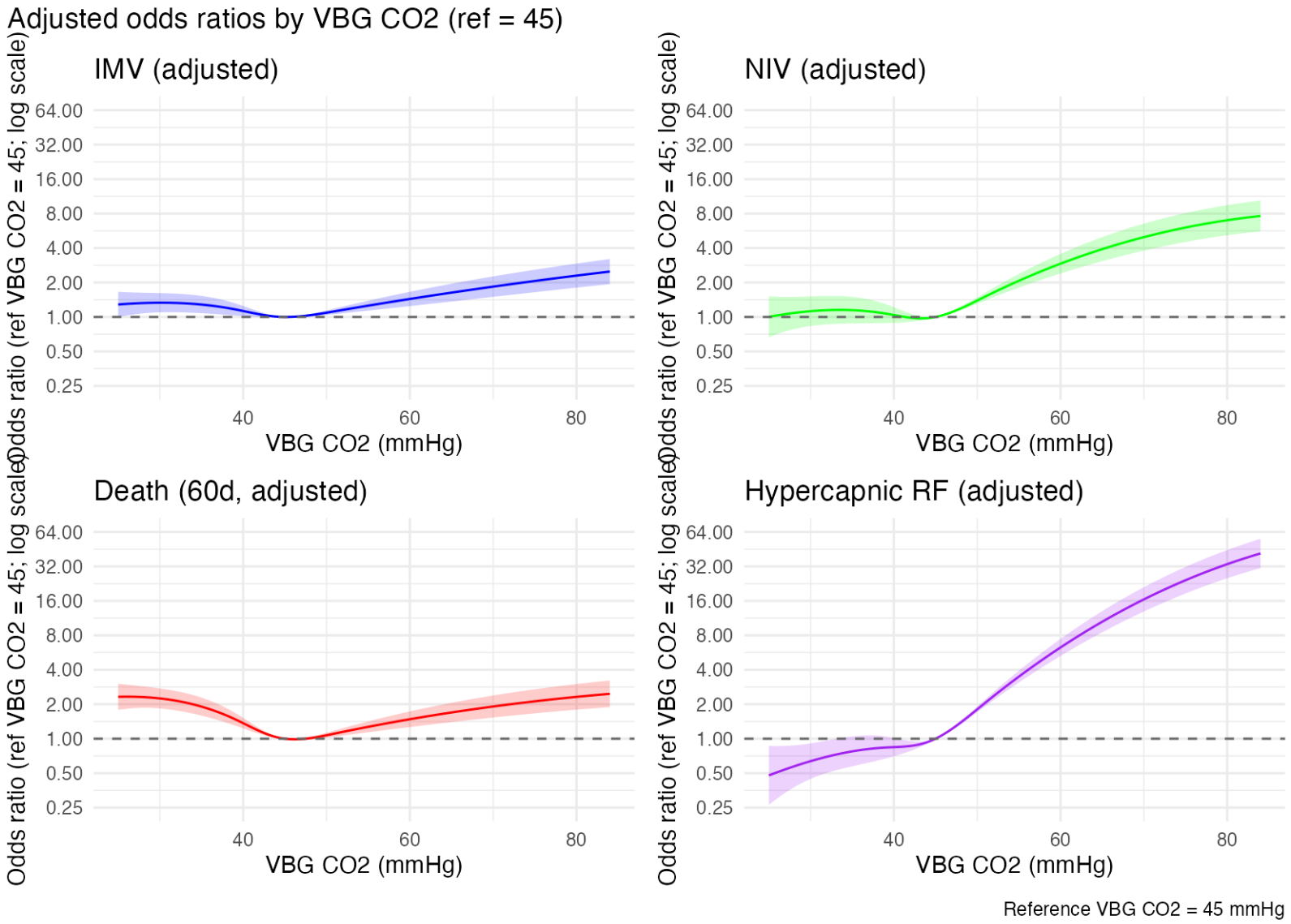
```r
pred_imv_vbg <- predict_or_curve_from_fit(fit_imv_vbg, grid_vbg_unw, ref_idx_vbg_unw, "vbg_co2")
pred_niv_vbg <- predict_or_curve_from_fit(fit_niv_vbg, grid_vbg_unw, ref_idx_vbg_unw, "vbg_co2")
pred_death_vbg <- predict_or_curve_from_fit(fit_death_vbg, grid_vbg_unw, ref_idx_vbg_unw, "vbg_co2")
pred_hcrf_vbg <- predict_or_curve_from_fit(fit_hcrf_vbg, grid_vbg_unw, ref_idx_vbg_unw, "vbg_co2")
axis_unw_common <- compute_or_axis_spec(
  list(pred_imv, pred_niv, pred_death, pred_hcrf,
       pred_imv_vbg, pred_niv_vbg, pred_death_vbg, pred_hcrf_vbg),
  lo_col = "LCL", hi_col = "UCL"
)

plot_imv <- ggplot(pred_imv, aes(x = paco2, y = OR)) +
  geom_line(color = "blue", linewidth = 1.2) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "blue", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "Intubation (adjusted)", x = "PaCO2 (mmHg)",
       y = paste0("Odds ratio (ref PaCO2 = ", co2_ref_abg_unw, "; log scale)")) +
  theme_minimal()

plot_niv <- ggplot(pred_niv, aes(x = paco2, y = OR)) +
  geom_line(color = "green", linewidth = 1.2) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "green", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "NIV (adjusted)", x = "PaCO2 (mmHg)",
       y = paste0("Odds ratio (ref PaCO2 = ", co2_ref_abg_unw, "; log scale)")) +
  theme_minimal()

plot_death <- ggplot(pred_death, aes(x = paco2, y = OR)) +
  geom_line(color = "red", linewidth = 1.2) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "red", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "Death (60d, adjusted)", x = "PaCO2 (mmHg)",
       y = paste0("Odds ratio (ref PaCO2 = ", co2_ref_abg_unw, "; log scale)")) +
  theme_minimal()
```

```r
plot_hcrf <- ggplot(pred_hcrf, aes(x = paco2, y = OR)) +
  geom_line(color = "purple", linewidth = 1.2) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "purple", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "Hypercapnic RF (adjusted)", x = "PaCO2 (mmHg)",
       y = paste0("Odds ratio (ref PaCO2 = ", co2_ref_abg_unw, "; log scale)")) +
  theme_minimal()

plot_imv_vbg <- ggplot(pred_imv_vbg, aes(x = vbg_co2, y = OR)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "blue", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "IMV (adjusted)", x = "VBG CO2 (mmHg)",
       y = paste0("Odds ratio (ref VBG CO2 = ", co2_ref_vbg_unw, "; log scale)")) +
  theme_minimal()

plot_niv_vbg <- ggplot(pred_niv_vbg, aes(x = vbg_co2, y = OR)) +
  geom_line(color = "green") +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "green", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "NIV (adjusted)", x = "VBG CO2 (mmHg)",
       y = paste0("Odds ratio (ref VBG CO2 = ", co2_ref_vbg_unw, "; log scale)")) +
  theme_minimal()

plot_death_vbg <- ggplot(pred_death_vbg, aes(x = vbg_co2, y = OR)) +
  geom_line(color = "red") +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "red", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "Death (60d, adjusted)", x = "VBG CO2 (mmHg)",
       y = paste0("Odds ratio (ref VBG CO2 = ", co2_ref_vbg_unw, "; log scale)")) +
  theme_minimal()
```

```r
plot_hcrf_vbg <- ggplot(pred_hcrf_vbg, aes(x = vbg_co2, y = OR)) +
  geom_line(color = "purple") +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "purple", alpha = 0.2) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_unw_common) +
  labs(title = "Hypercapnic RF (adjusted)", x = "VBG CO2 (mmHg)",
       y = paste0("Odds ratio (ref VBG CO2 = ", co2_ref_vbg_unw, "; log scale)")) +
  theme_minimal()

unw_abg_panel <- (plot_imv | plot_niv) / (plot_death | plot_hcrf) +
  plot_annotation(caption = paste0("Reference PaCO2 = ", co2_ref_abg_unw, " mmHg"))

unw_vbg_panel <- ((plot_imv_vbg | plot_niv_vbg) /
  (plot_death_vbg | plot_hcrf_vbg)) +
  plot_annotation(
    title = paste0("Adjusted odds ratios by VBG CO2 (ref = ", co2_ref_vbg_unw, ")"),
    caption = paste0("Reference VBG CO2 = ", co2_ref_vbg_unw, " mmHg")
  )

print_plot_once(unw_abg_panel, "spline-unweighted-abg", width = 8.5, height = 6)
```




```
print_plot_once(unw_vbg_panel, "spline-unweighted-vbg", width = 8.5, height = 6)
```




```
float_barrier()
```

2 Inverse Propensity Weighting

IPW done using Gradient Boosting Methods (GBM) - a type of decision-tree based machine learning. "***Random forests and GBM are designed to automatically include relevant interactions for variables included in the model.*** As such, using a GBM to estimate the PS model, can reduce model misspecification, since ***the analyst is not required to identify relevant interactions or nonlinearities.***" from this citation: PMID: 39947224https://pmc.ncbi.nlm.nih.gov/articles/PMC11825193/

Current propensity score uses `covars_gbm` (demographics, comorbidities, encounter type, vitals, labs) as defined above; in this block only `encounter_type` is explicitly factored before weighting.

Note: for all these, I suggested new GBM adjustments that accomplish the following:

1. Smaller GBM & balance-based stopping (stop.method = "smd.max") → faster fit, avoids over-fitting, lighter tails (which lead to extreme weights that are problematic).
2. Target balance compares weighted treated cohort to the full sample; aim for $|\text{SMD}| < 0.1$.
3. Weight stabilization (divide by mean) mitigates a few huge weights. We use one-sided truncation at very small propensities (caps large weights only).
4. Uses robust variance estimation (e.g. allows the variances to change by PaCO2) for IP-weighted GLM; works with splines via rcs(). This is a bit nuanced but I think good to change even though it adds complexity
5. Deterministic seed ensures result replication.

2.0.1 5.1 ABG IPW weighting and diagnostics

```
# Already normalized globally; just drop unused levels
subset_data$encounter_type <- droplevels(subset_data$encounter_type)
```

GBM tuning is shared across ABG and VBG via `gbm_params` to keep symmetry; update there if needed.

```
#   1. fit GBM propensity model, ABG
set.seed(42)
gbm_df_abg <- subset_data[, c("has_abg", "has_vbg", covars_gbm), drop = FALSE]
gbm_df_abg <- normalize_types(gbm_df_abg, levels_ref)
gbm_df_abg <- droplevels_all(gbm_df_abg)
```

```r
gbm_preflight(gbm_df_abg, covars_gbm, "unimp_abg")
append_mem_snapshot("gbm_unimp", "unimp_abg", "pre")

weight_model <- do.call(
  weightit,
  c(
    list(
      formula_abg,
      data        = gbm_df_abg,
      method      = "gbm",
      estimand    = "ATE",
      missing    = "ind",
      include.obj = FALSE
    ),
    gbm_params
  )
)
append_mem_snapshot("gbm_unimp", "unimp_abg", "post")

#   2. One-sided IPSW (ABG observed only) + truncation of small propensities
ipow_abg <- compute_ipow_weights(
  weight_model,
  treat = gbm_df_abg$has_abg,
  ps_floor_quantile = ps_trunc_quantile,
  stabilize = TRUE
)
w_abg <- ipow_abg$weights
ps_floor_abg <- ipow_abg$ps_floor
subset_data$trunc_abg <- ipow_abg$truncated
subset_data$ps_abg <- ipow_abg$ps
subset_data$w_abg  <- w_abg
assert_finite_weights(w_abg[subset_data$has_abg == 1], "w_abg")
rm(weight_model, gbm_df_abg)
invisible(gc())

# Balance diagnostics and treated-only outcome models are handled later.
```

Inverse Propensity-Weighted Logistic Regressions with CO2 predictor represented as a restricted cubic spline.

These are covariate-adjusted outcome models (outcome ~ spline(CO2) + X), fit separately for ABG and VBG cohorts using survey::svyglm with robust (design-based) SEs. Spline curves are shown as odds ratios relative to CO2_ref (midpoint of the normal range).

2.0.2 5.2 ABG IPW spline models

```
# set.seed(42)  # reproducible GBM fit
#
# #   1. inverse-probability weights for receiving an ABG
#
# # done in the last block, so not needed
#

# Model diagrams: IPW ABG spline models
ipw_abg_rcs_forms <- list(
  "ABG IPW spline (adjusted): IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "paco2", adj_core),
  "ABG IPW spline (adjusted): NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "paco2", adj_core),
  "ABG IPW spline (adjusted): Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "paco2", adj_core),
  "ABG IPW spline (adjusted): HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "paco2", adj_core)
)
register_model_diagrams(ipw_abg_rcs_forms)

#   2. analysis sample: rows with a measured PaCO2
subset_data_abg <- subset_data %>%
  filter(!is.na(paco2)) %>%                    # implies has_abg == 1
  select(paco2, imv_proc, niv_proc, death_60d,
         hypercap_resp_failure, w_abg, all_of(adj_core)) %>%
  filter(complete.cases(.))


#   3. weighted logistic spline models with robust SEs
fitfun <- function(formula, outcome) {
  fit_res <- fit_with_diagnostics(
    function() svyglm(
```

```
      formula,
      design = svydesign(ids = ~1, weights = ~w_abg, data = subset_data_abg),
      family = quasibinomial(),
      control = glm.control(maxit = 50)
    ),
    context = make_context(
      stage = "outcome",
      component = "spline",
      analysis_variant = "ipw",
      model_type = "spline",
      group = "ABG",
      outcome = outcome,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  fit_res$fit
}

fit_imv_abg   <- fitfun(make_spline_fml("imv_proc", "paco2", adj_core), "imv_proc")
fit_niv_abg   <- fitfun(make_spline_fml("niv_proc", "paco2", adj_core), "niv_proc")
fit_death_abg <- fitfun(make_spline_fml("death_60d", "paco2", adj_core), "death_60d")
fit_hcrf_abg  <- fitfun(make_spline_fml("hypercap_resp_failure", "paco2", adj_core),
                        "hypercap_resp_failure")
if (any(vapply(list(fit_imv_abg, fit_niv_abg, fit_death_abg, fit_hcrf_abg), is.null, logical(1)))) {
  stop("IPW ABG spline fits failed; see model_fit_diagnostics.csv.")
}

#   4. prediction helper
mkpred <- function(fit, data_ref, co2_var, ref_df, co2_ref) {
  co2_seq <- stats::quantile(data_ref[[co2_var]], probs = c(0.02, 0.98), na.rm = TRUE)
  grid_info <- make_co2_grid_ref(
    co2_var,
    seq(co2_seq[1], co2_seq[2], length.out = SPLINE_GRID_N),
    ref_df,
```

```
    co2_ref
  )
  predict_or_curve_from_fit(fit, grid_info$grid, grid_info$ref_idx, co2_var)
}

pred_imv_abg   <- mkpred(fit_imv_abg,   subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_niv_abg   <- mkpred(fit_niv_abg,   subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_death_abg <- mkpred(fit_death_abg, subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_hcrf_abg  <- mkpred(fit_hcrf_abg,  subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
axis_abg_ipw_trim <- compute_or_axis_spec(
  list(pred_imv_abg, pred_niv_abg, pred_death_abg, pred_hcrf_abg),
  lo_col = "LCL", hi_col = "UCL"
)
#   5. plotting
# Plotting deferred until VBG curves are computed so axes can be shared.
```

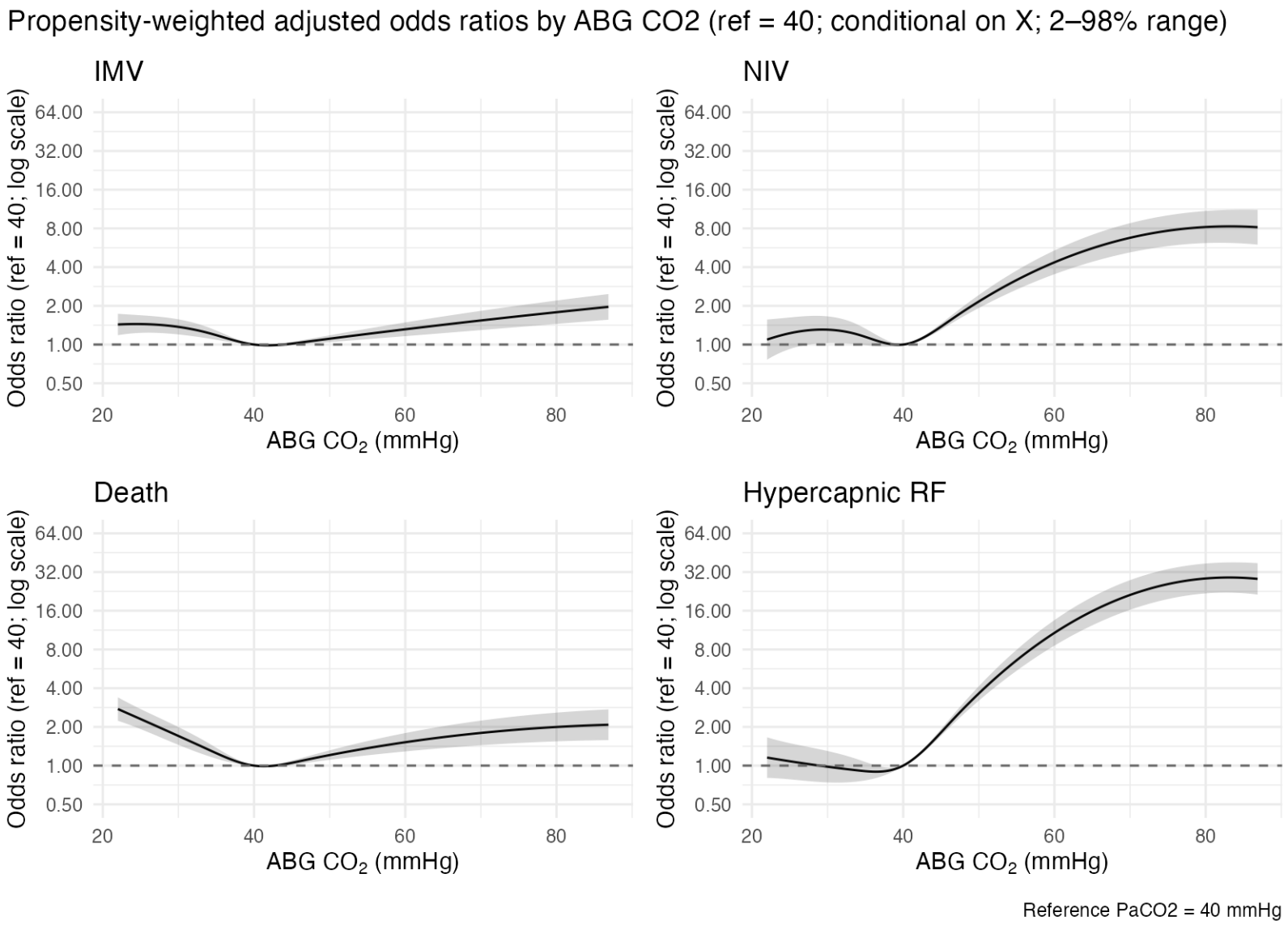
Restricting plots bewtween 0.02 and 0.98

2.0.3 5.3 ABG IPW spline models (2–98th percentile)

```
subset_data_abg <- subset_data %>%
  filter(!is.na(paco2)) %>%                    # implies has_abg == 1
  select(paco2, imv_proc, niv_proc, death_60d,
         hypercap_resp_failure, w_abg, all_of(adj_core)) %>%
  filter(complete.cases(.))

fitfun <- function(formula, outcome) {
  fit_res <- fit_with_diagnostics(
    function() svyglm(
      formula,
      design = svydesign(ids = ~1, weights = ~w_abg, data = subset_data_abg),
      family = quasibinomial(),
      control = glm.control(maxit = 50)
    ),
```

```
    context = make_context(
      stage = "outcome",
      component = "spline",
      analysis_variant = "ipw",
      model_type = "spline",
      group = "ABG",
      outcome = outcome,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  fit_res$fit
}

fit_imv_abg   <- fitfun(make_spline_fml("imv_proc", "paco2", adj_core), "imv_proc")
fit_niv_abg   <- fitfun(make_spline_fml("niv_proc", "paco2", adj_core), "niv_proc")
fit_death_abg <- fitfun(make_spline_fml("death_60d", "paco2", adj_core), "death_60d")
fit_hcrf_abg  <- fitfun(make_spline_fml("hypercap_resp_failure", "paco2", adj_core),
                        "hypercap_resp_failure")
if (any(vapply(list(fit_imv_abg, fit_niv_abg, fit_death_abg, fit_hcrf_abg), is.null, logical(1)))) {
  stop("IPW ABG spline fits (trimmed) failed; see model_fit_diagnostics.csv.")
}

#   4. prediction helper
mkpred <- function(fit, data_ref, co2_var, ref_df, co2_ref) {
  q <- stats::quantile(data_ref[[co2_var]], probs = c(0.02, 0.98), na.rm = TRUE)
  grid_info <- make_co2_grid_ref(
    co2_var,
    seq(q[1], q[2], length.out = SPLINE_GRID_N),
    ref_df,
    co2_ref
  )
  predict_or_curve_from_fit(fit, grid_info$grid, grid_info$ref_idx, co2_var)
}
```

```r
pred_imv_abg   <- mkpred(fit_imv_abg,   subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_niv_abg   <- mkpred(fit_niv_abg,   subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_death_abg <- mkpred(fit_death_abg, subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)
pred_hcrf_abg  <- mkpred(fit_hcrf_abg,  subset_data_abg, "paco2", x_ref_abg, ABG_CO2_REF)

#   5. plotting
xlab <- expression(paste("ABG CO"[2], " (mmHg)"))

plt <- function(dat, title)
  ggplot(dat, aes(paco2, OR)) +
    geom_line() +
    geom_ribbon(aes(ymin = LCL, ymax = UCL), alpha = 0.2) +
    geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
    or_axis_scale(axis_abg_ipw_trim) +
    labs(title = title, x = xlab,
         y = paste0("Odds ratio (ref = ", ABG_CO2_REF, "; log scale)")) +
    theme_minimal()

ipw_abg_panel <- (patchwork::wrap_plots(
  plt(pred_imv_abg,   "IMV"),
  plt(pred_niv_abg,   "NIV"),
  plt(pred_death_abg, "Death"),
  plt(pred_hcrf_abg,  "Hypercapnic RF"),
  ncol = 2
)) +
  plot_annotation(
    title = paste0("Propensity-weighted adjusted odds ratios by ABG CO2 (ref = ",
                   ABG_CO2_REF, "; conditional on X; 2-98% range)"),
    caption = paste0("Reference PaCO2 = ", ABG_CO2_REF, " mmHg")
  )

print_plot_once(ipw_abg_panel, "spline-ipw-abg-trimmed", width = 8.5, height = 6)
```



VBG uses the same GBM tuning as ABG (shared `gbm_params`).

2.0.4 5.4 VBG IPW weighting and spline models

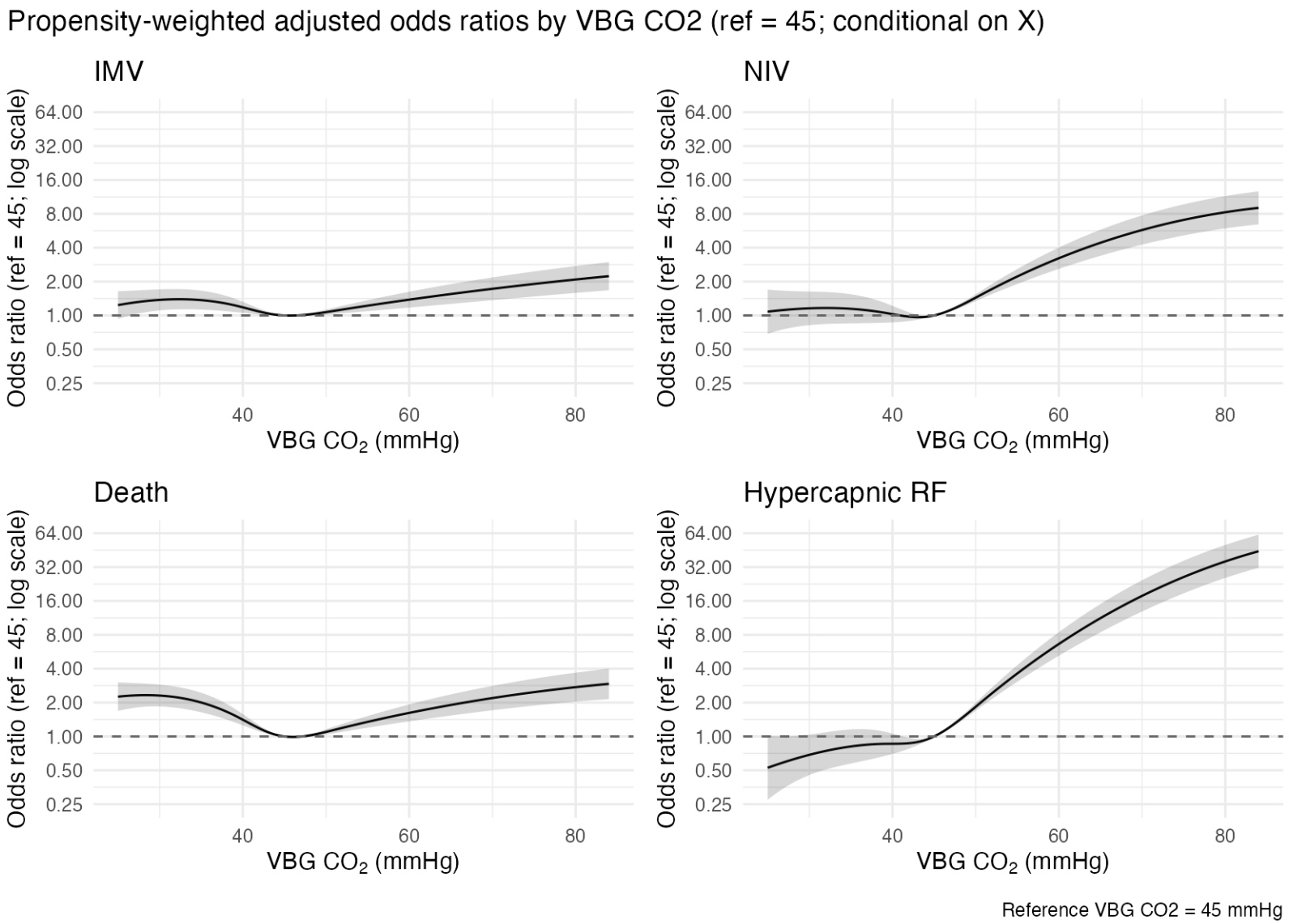
```
#  Inverse-propensity weighting & outcome modelling for **VBG** cohort
#    - mirrored 1-to-1 to the validated ABG workflow

set.seed(42)

# 1. IPW for VBG ---------------------------------------------------------------
set.seed(42)
gbm_df_vbg <- subset_data[, c("has_abg", "has_vbg", covars_gbm), drop = FALSE]
gbm_df_vbg <- normalize_types(gbm_df_vbg, levels_ref)
gbm_df_vbg <- droplevels_all(gbm_df_vbg)
gbm_preflight(gbm_df_vbg, covars_gbm, "unimp_vbg")
append_mem_snapshot("gbm_unimp", "unimp_vbg", "pre")
w_vbg <- do.call(
  weightit,
  c(
    list(
      formula_vbg,
      data        = gbm_df_vbg,
      method      = "gbm",
      estimand    = "ATE",
      missing     = "ind",
      include.obj = FALSE
    ),
    gbm_params
  )
)
append_mem_snapshot("gbm_unimp", "unimp_vbg", "post")

# One-sided IPSW (VBG observed only) + truncation of small propensities
ipow_vbg <- compute_ipow_weights(
  w_vbg,
  treat = gbm_df_vbg$has_vbg,
  ps_floor_quantile = ps_trunc_quantile,
```

```
  stabilize = TRUE
)
w_vbg_ipow <- ipow_vbg$weights
ps_floor_vbg <- ipow_vbg$ps_floor
subset_data$trunc_vbg <- ipow_vbg$truncated
subset_data$ps_vbg <- ipow_vbg$ps
subset_data$w_vbg  <- w_vbg_ipow
assert_finite_weights(w_vbg_ipow[subset_data$has_vbg == 1], "w_vbg")
rm(w_vbg, gbm_df_vbg)
invisible(gc())

# Balance diagnostics are handled later.

# 2. Analysis set (VBG only) ----------------------------------------------------
subset_data_vbg <- subset_data %>%
  filter(!is.na(vbg_co2)) %>%
  select(vbg_co2, imv_proc, niv_proc, death_60d,
         hypercap_resp_failure, w_vbg, all_of(adj_core)) %>%
  filter(complete.cases(.))

fitfun <- function(formula, outcome) {
  fit_res <- fit_with_diagnostics(
    function() svyglm(
      formula,
      design = svydesign(ids = ~1, weights = ~w_vbg, data = subset_data_vbg),
      family = quasibinomial(),
      control = glm.control(maxit = 50)
    ),
    context = make_context(
      stage = "outcome",
      component = "spline",
      analysis_variant = "ipw",
      model_type = "spline",
      group = "VBG",
      outcome = outcome,
      imputation = NA_integer_,
```

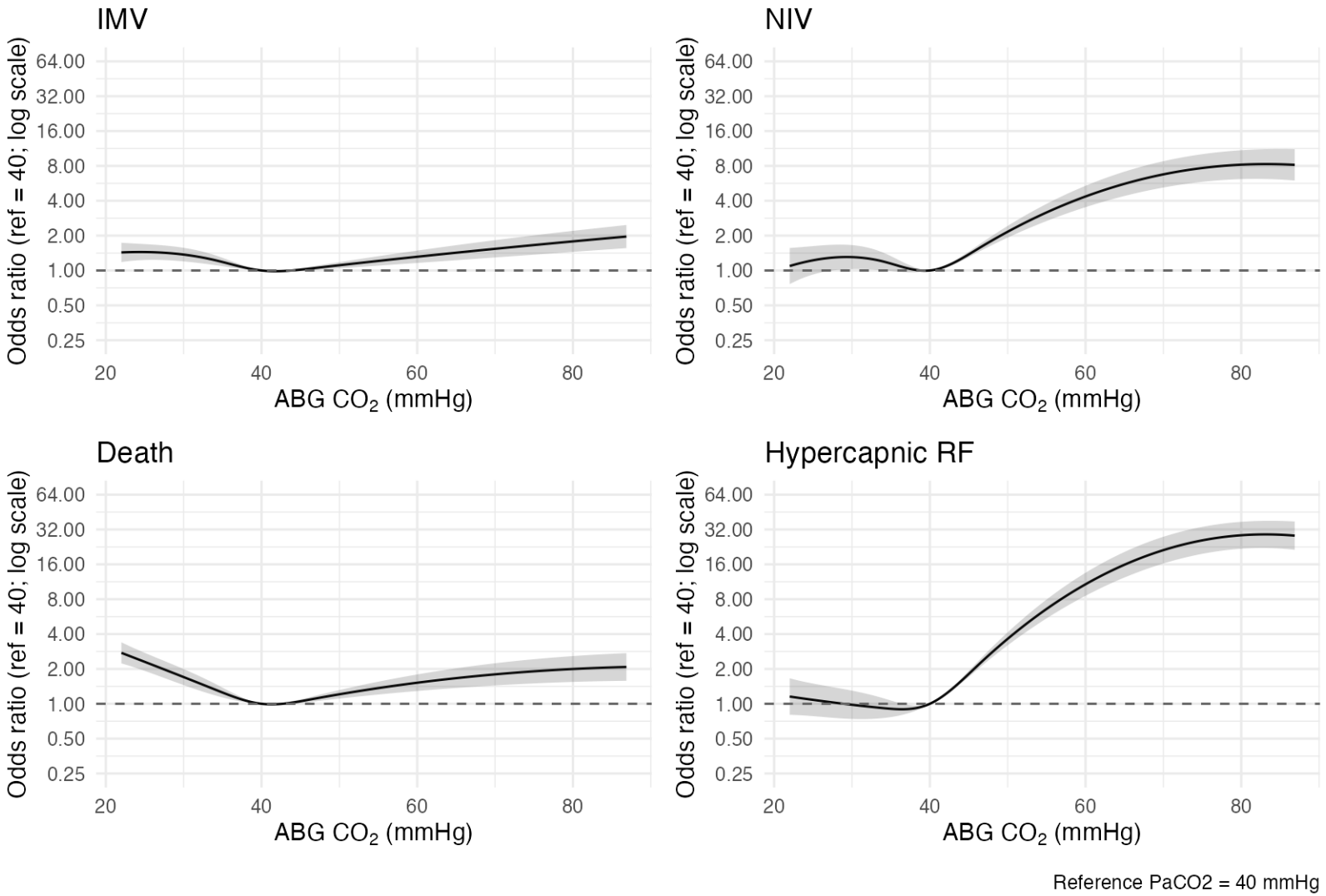
```
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  fit_res$fit
}

# Model diagrams: IPW VBG spline models
ipw_vbg_rcs_forms <- list(
  "VBG IPW spline (adjusted): IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "vbg_co2", adj_core),
  "VBG IPW spline (adjusted): NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "vbg_co2", adj_core),
  "VBG IPW spline (adjusted): Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "vbg_co2", adj_core),
  "VBG IPW spline (adjusted): HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "vbg_co2", adj_core)
)
register_model_diagrams(ipw_vbg_rcs_forms)

fit_imv_vbg   <- fitfun(make_spline_fml("imv_proc", "vbg_co2", adj_core), "imv_proc")
fit_niv_vbg   <- fitfun(make_spline_fml("niv_proc", "vbg_co2", adj_core), "niv_proc")
fit_death_vbg <- fitfun(make_spline_fml("death_60d", "vbg_co2", adj_core), "death_60d")
fit_hcrf_vbg  <- fitfun(make_spline_fml("hypercap_resp_failure", "vbg_co2", adj_core),
                        "hypercap_resp_failure")
if (any(vapply(list(fit_imv_vbg, fit_niv_vbg, fit_death_vbg, fit_hcrf_vbg), is.null, logical(1)))) {
  stop("IPW VBG spline fits failed; see model_fit_diagnostics.csv.")
}

# 4. Prediction helper ------------------------------------------------------
mkpred <- function(fit, data_ref, co2_var, ref_df, co2_ref) {
  co2_seq <- stats::quantile(data_ref[[co2_var]], probs = c(0.02, 0.98), na.rm = TRUE)
  grid_info <- make_co2_grid_ref(
    co2_var,
    seq(co2_seq[1], co2_seq[2], length.out = SPLINE_GRID_N),
    ref_df,
    co2_ref
  )
  predict_or_curve_from_fit(fit, grid_info$grid, grid_info$ref_idx, co2_var)
}
```

```r
pred_imv_vbg   <- mkpred(fit_imv_vbg,   subset_data_vbg, "vbg_co2", x_ref_vbg, VBG_CO2_REF)
pred_niv_vbg   <- mkpred(fit_niv_vbg,   subset_data_vbg, "vbg_co2", x_ref_vbg, VBG_CO2_REF)
pred_death_vbg <- mkpred(fit_death_vbg, subset_data_vbg, "vbg_co2", x_ref_vbg, VBG_CO2_REF)
pred_hcrf_vbg  <- mkpred(fit_hcrf_vbg,  subset_data_vbg, "vbg_co2", x_ref_vbg, VBG_CO2_REF)
axis_ipw_common <- compute_or_axis_spec(
  list(pred_imv_abg, pred_niv_abg, pred_death_abg, pred_hcrf_abg,
       pred_imv_vbg, pred_niv_vbg, pred_death_vbg, pred_hcrf_vbg),
  lo_col = "LCL", hi_col = "UCL"
)

# 5. Plotting (gray scheme) ----------------------------------------------------
xlab <- expression(paste("VBG CO"[2], " (mmHg)"))

plt <- function(dat, title)
  ggplot(dat, aes(vbg_co2, OR)) +
    geom_line() +
    geom_ribbon(aes(ymin = LCL, ymax = UCL), alpha = 0.2) +
    geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
    or_axis_scale(axis_ipw_common) +
    labs(title = title, x = xlab,
         y = paste0("Odds ratio (ref = ", VBG_CO2_REF, "; log scale)")) +
    theme_minimal()

ipw_vbg_panel <- (patchwork::wrap_plots(
  plt(pred_imv_vbg,   "IMV"),
  plt(pred_niv_vbg,   "NIV"),
  plt(pred_death_vbg, "Death"),
  plt(pred_hcrf_vbg,  "Hypercapnic RF"),
  ncol = 2
)) +
  plot_annotation(
    title = paste0("Propensity-weighted adjusted odds ratios by VBG CO2 (ref = ",
                   VBG_CO2_REF, "; conditional on X)"),
    caption = paste0("Reference VBG CO2 = ", VBG_CO2_REF, " mmHg")
  )
```

```
print_plot_once(ipw_vbg_panel, "spline-ipw-vbg", width = 8.5, height = 6)
```



```r
# ABG plots with the same axis (shared with VBG)
xlab_abg <- expression(paste("ABG CO"[2], " (mmHg)"))

plt_abg <- function(dat, title)
  ggplot(dat, aes(paco2, OR)) +
    geom_line() +
    geom_ribbon(aes(ymin = LCL, ymax = UCL), alpha = 0.2) +
    geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
    or_axis_scale(axis_ipw_common) +
    labs(title = title, x = xlab_abg,
         y = paste0("Odds ratio (ref = ", ABG_CO2_REF, "; log scale)")) +
    theme_minimal()

ipw_abg_shared_panel <- (patchwork::wrap_plots(
  plt_abg(pred_imv_abg,   "IMV"),
  plt_abg(pred_niv_abg,   "NIV"),
  plt_abg(pred_death_abg, "Death"),
  plt_abg(pred_hcrf_abg,  "Hypercapnic RF"),
  ncol = 2
)) +
  plot_annotation(
    title = paste0("Propensity-weighted adjusted odds ratios by ABG CO2 (ref = ",
                   ABG_CO2_REF, "; conditional on X)"),
    caption = paste0("Reference PaCO2 = ", ABG_CO2_REF, " mmHg")
  )

print_plot_once(ipw_abg_shared_panel, "spline-ipw-abg-shared", width = 8.5, height = 6)
```



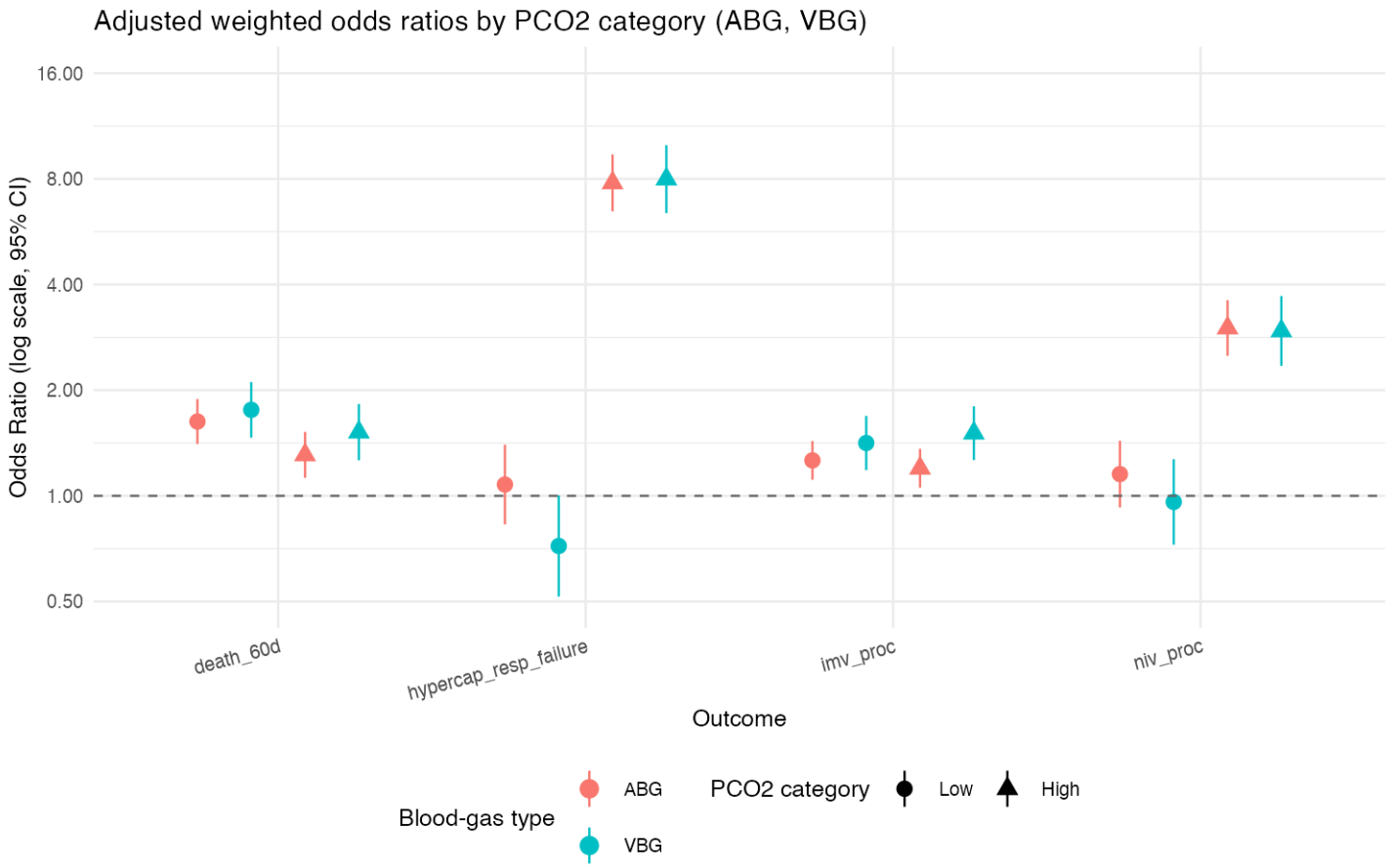
2.0.5 5.5 Three-level PCO2 categories (weighted; ABG, VBG)

Three groups with weights and covariate adjustment

```r
library(dplyr)
library(survey)
library(broom)
library(ggplot2)
library(scales)

#   1. Ensure PCO2 categories are present
stopifnot(all(c("pco2_cat_abg", "pco2_cat_vbg") %in% names(subset_data)))

#   2. Function: weighted logistic regression & OR extraction
run_weighted_or <- function(data, outcome, cat_var, weight_var, group_name,
                            treat_var, adj_vars) {
  stopifnot(!is.null(treat_var))
  stopifnot(!is.null(adj_vars))
  dat <- data %>%
    filter(
      .data[[treat_var]] == 1,
      !is.na(.data[[cat_var]]),
      !is.na(.data[[outcome]]),
      !is.na(.data[[weight_var]]),
      .data[[weight_var]] > 0
    ) %>%
    mutate(
      !!cat_var := factor(.data[[cat_var]],
                          levels = CO2_CAT_LEVELS)
    ) %>%
    droplevels()

  design <- svydesign(
    ids = ~1,
    weights = as.formula(paste0("~", weight_var)),
    data = dat
  )

  rhs_terms <- c(cat_var, adj_vars)
  fml <- stats::reformulate(rhs_terms, response = outcome)
```

```
  fit_res <- fit_with_diagnostics(
    function() svyglm(fml, design = design, family = quasibinomial(),
                      control = glm.control(maxit = 50)),
    context = make_context(
      stage = "outcome",
      component = "cat3",
      analysis_variant = "ipw",
      model_type = "cat3",
      group = group_name,
      outcome = outcome,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_outcome_diag(fit_res$diag)
  if (is.null(fit_res$fit)) {
    stop("run_weighted_or: model fit failed for outcome=", outcome,
         " cat_var=", cat_var, " group=", group_name)
  }

  tidy(fit_res$fit, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term != "(Intercept)", startsWith(term, cat_var)) %>%
    mutate(
      group    = group_name,
      outcome  = outcome
    )
}

#   3. Run across outcomes & cohorts
outcomes_ipw <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

ipw_three_level_forms <- list(
  "ABG IPW 3-level: IMV ~ CO2 category + X"      = reformulate(c("pco2_cat_abg", adj_core), response = "imv_proc"),
  "ABG IPW 3-level: NIV ~ CO2 category + X"      = reformulate(c("pco2_cat_abg", adj_core), response = "niv_proc"),
  "ABG IPW 3-level: Death60d ~ CO2 category + X" = reformulate(c("pco2_cat_abg", adj_core), response = "death_60d"),
  "ABG IPW 3-level: HCRF ~ CO2 category + X"     = reformulate(c("pco2_cat_abg", adj_core), response = "hypercap_resp_failure"),
```

```
  "VBG IPW 3-level: IMV ~ CO2 category + X"      = reformulate(c("pco2_cat_vbg", adj_core), response = "imv_proc"),
  "VBG IPW 3-level: NIV ~ CO2 category + X"      = reformulate(c("pco2_cat_vbg", adj_core), response = "niv_proc"),
  "VBG IPW 3-level: Death60d ~ CO2 category + X" = reformulate(c("pco2_cat_vbg", adj_core), response = "death_60d"),
  "VBG IPW 3-level: HCRF ~ CO2 category + X"     = reformulate(c("pco2_cat_vbg", adj_core), response = "hypercap_resp_failure")
)
register_model_diagrams(ipw_three_level_forms)

ipw_combined_or_df <- bind_rows(
  lapply(outcomes_ipw, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_abg",  "w_abg",      "ABG",
                    treat_var = "has_abg", adj_vars = adj_core)),
  lapply(outcomes_ipw, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_vbg",  "w_vbg",      "VBG",
                    treat_var = "has_vbg", adj_vars = adj_core))
)

ipw_threelevel_results <- ipw_combined_or_df %>%
  mutate(method = "IPW adjusted")

ipw_combined_or_df <- map_or_exposure(ipw_combined_or_df, "or-plot-three-level-weighted")
ipw_combined_or_df$group <- factor(ipw_combined_or_df$group, levels = c("ABG", "VBG"))

#   4. Plot weighted odds ratios
ipw_plot_df <- build_or_plot_df(ipw_combined_or_df, "or-plot-three-level-weighted",
                            expected_exposure_levels = CO2_CAT_CONTRAST_LEVELS)
ipw_axis_spec <- compute_or_axis_spec(ipw_plot_df, lo_col = "conf.low", hi_col = "conf.high",
                                      default_limits = OR_XLIM)
ipw_p_or <- plot_or_safe(
  ipw_plot_df,
  plot_name = "or-plot-three-level-weighted",
  axis_spec = ipw_axis_spec,
  title  = "Adjusted weighted odds ratios by PCO2 category (ABG, VBG)"
)
print_plot_once(ipw_p_or, "or-plot-three-level-weighted", width = 7.5, height = 4.8)
```



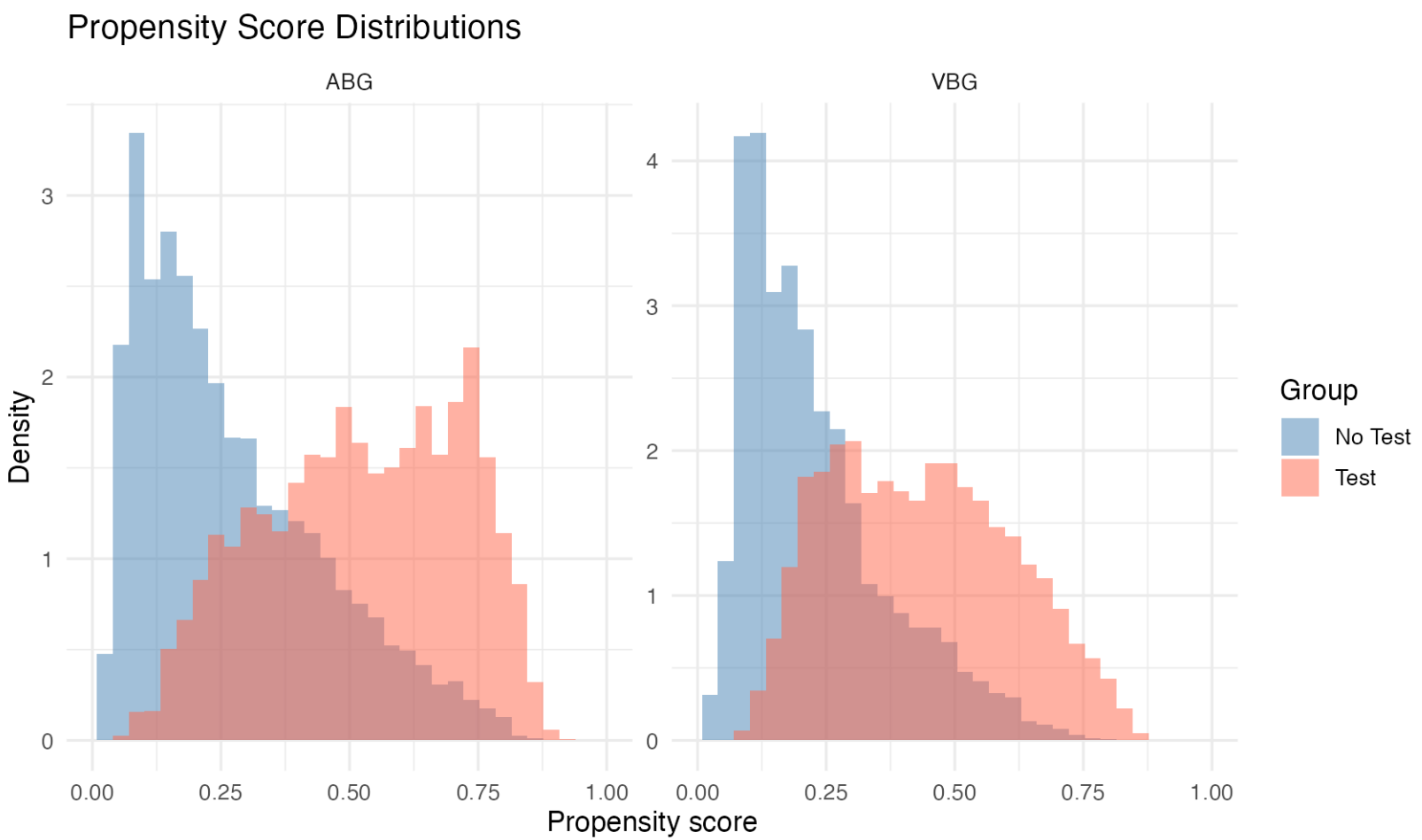
2.1 6) Propensity score diagnostics

Plotting propensity scores

```
# --- Propensity score histograms (ABG / VBG) --------------------------------
# ABG = arterial blood gas; VBG = venous blood gas

library(dplyr)
library(ggplot2)
```

```r
library(scales)

stopifnot("has_abg" %in% names(subset_data))
stopifnot("has_vbg" %in% names(subset_data))
stopifnot(all(c("ps_abg", "ps_vbg") %in% names(subset_data)))

# Build list of per-cohort PS data frames conditionally (so missing cohorts don't error)
ps_dfs_cond <- list(
  ABG = data.frame(
    ps        = subset_data$ps_abg,
    treat     = subset_data$has_abg,
    ScoreType = "ABG"
  ),
  VBG = data.frame(
    ps        = subset_data$ps_vbg,
    treat     = subset_data$has_vbg,
    ScoreType = "VBG"
  )
)

# Bind, clean, and factorize for plotting
df_ps_cond <- bind_rows(ps_dfs_cond) %>%
  filter(!is.na(ps), !is.na(treat)) %>%
  mutate(
    treat     = factor(treat, levels = c(0, 1), labels = c("No Test", "Test")),
    ScoreType = factor(ScoreType, levels = c("ABG", "VBG"))
  )

# Plot
p_ps_cond <- ggplot(df_ps_cond, aes(x = ps, fill = treat)) +
  geom_histogram(aes(y = after_stat(density)), alpha = 0.5,
                 position = "identity", bins = 30) +
  scale_fill_manual(values = c("No Test" = "steelblue", "Test" = "tomato")) +
  facet_wrap(~ScoreType, scales = "free_y") +
  coord_cartesian(xlim = c(0, 1)) +
  labs(
```

```
    title = "Propensity Score Distributions",
    x     = "Propensity score",
    y     = "Density",
    fill  = "Group"
  ) +
  theme_minimal(base_size = 12)
print_plot_once(p_ps_cond, "propensity-histograms-conditional", width = 8.5, height = 5)
```



3 Multiple Imputation Analysis

added 12/6/2025

3.0.1 7.2 Missingness structure and drivers

```
Preview (top 20 rows). Full table saved to Results/missingness_by_strata_preview.csv.
```

Pre-imputation missingness

Variable	Missing (n)	Missing (%)
vbg_o2sat	22,366	86.5%
bnp	21,181	81.9%
vbg_co2	18,392	71.1%
spo2	18,319	70.9%
paco2	16,490	63.8%
serum_lac	15,571	60.2%
curr_bmi	14,723	57.0%
serum_phos	13,762	53.2%
temp_new	12,460	48.2%
hr	9,418	36.4%
dbp	7,738	29.9%
sbp	7,666	29.7%
wbc	4,542	17.6%
serum_ca	2,627	10.2%
serum_cr	2,396	9.3%
plt	2,080	8.0%
serum_k	2,021	7.8%
serum_cl	1,498	5.8%
serum_hco3	1,460	5.6%
sodium	1,326	5.1%
age_at_encounter	0	0.0%
sex	0	0.0%
race_ethnicity	0	0.0%
copd	0	0.0%
asthma	0	0.0%
osa	0	0.0%
chf	0	0.0%
acute_nmd	0	0.0%
phtn	0	0.0%
ckd	0	0.0%
dm	0	0.0%
location	0	0.0%
encounter_type	0	0.0%
has_abg	0	0.0%
has_vbg	0	0.0%
imv_proc	0	0.0%

Table 7: Missingness by key strata (pre-imputation; top 10 variables; full table saved to Results/missingness-by-strata.csv).

level	variable	pct_missing	stratum	pct_missing_overall
0	vbg_o2sat	86.4	has_abg	86.5
1	vbg_o2sat	86.7	has_abg	86.5
0	vbg_o2sat	99.2	has_vbg	86.5
1	vbg_o2sat	55.3	has_vbg	86.5
0	vbg_o2sat	87.1	imv_proc	86.5
1	vbg_o2sat	82.1	imv_proc	86.5
0	bnp	84.4	has_abg	81.9
1	bnp	77.7	has_abg	81.9
0	bnp	82.8	has_vbg	81.9
1	bnp	79.8	has_vbg	81.9
0	bnp	82.4	imv_proc	81.9
1	bnp	78.5	imv_proc	81.9
0	vbg_co2	71.5	has_abg	71.1
1	vbg_co2	70.5	has_abg	71.1
0	vbg_co2	100.0	has_vbg	71.1
1	vbg_co2	0.0	has_vbg	71.1
0	vbg_co2	72.6	imv_proc	71.1
1	vbg_co2	59.8	imv_proc	71.1
0	spo2	69.6	has_abg	70.9
1	spo2	73.1	has_abg	70.9

3.0.2 7.3 Monte Carlo error check after MI

3.1 8) Pre-imputation data prep (consistent types & predictors)

Why: MI models need coherent types; using exactly the same covariates as the propensity score models avoids model drift.

3.2 9) Imputation model specification (MICE)

3.2.1 9.1 Predictor matrix & methods. Run MICE (moderate settings for scale)

```
# --- variables for GBM propensity (kept identical to main analysis) ---
# ---- MICE setup: include PaCO2/VBG CO2 as predictors but do not impute ----
library(mice)
library(dplyr)
```

Table 8: Predictors of missingness (logit OR; top 20 by p-value; full table saved to Results/missingness-drivers.csv).

variable	term	OR	LCL	UCL	p.value
temp_new	location1	18.41	15.92	21.29	0
temp_new	location3	21.44	18.95	24.26	0
serum_phos	encounter_typeInpatient	0.12	0.11	0.14	0
hr	location1	11.68	10.14	13.46	0
sbp	encounter_typeInpatient	0.04	0.04	0.05	0
dbp	encounter_typeInpatient	0.05	0.04	0.06	0
serum_lac	has_abg	0.24	0.22	0.26	0
vbg_o2sat	has_vbg	0.00	0.00	0.00	0
dbp	location3	1607.67	847.74	3048.81	0
serum_lac	location3	3.24	2.89	3.62	0
serum_phos	has_abg	0.34	0.31	0.38	0
sbp	location3	3373.24	1382.62	8229.86	0
hr	location2	4.34	3.68	5.13	0
wbc	location3	3.13	2.72	3.61	0
temp_new	location2	3.76	3.17	4.45	0
serum_cl	age_at_encounter	0.96	0.96	0.97	0
sodium	age_at_encounter	0.96	0.96	0.97	0
serum_ca	age_at_encounter	0.97	0.97	0.97	0
serum_hco3	age_at_encounter	0.97	0.96	0.97	0
serum_phos	has_vbg	0.38	0.33	0.43	0

```r
# --- add analysis targets and CO2 measures explicitly -------------------------
mi_vars <- setdiff(unique(c(
  covars_gbm,
  "has_abg","has_vbg",                                  # treatments (NOT imputed)
  "imv_proc","niv_proc","death_60d","hypercap_resp_failure",  # outcomes (NOT imputed)
  co2_vars
)), drop_vars_ultra_missing)

mi_df <- subset_data[, mi_vars, drop = FALSE]
mi_df <- normalize_types(mi_df, levels_ref)

mi_df_size <- utils::object.size(mi_df)
message("MI data size (bytes): ", format(mi_df_size, units = "auto"))

# Rough memory preflight based on total missing cells (imputed values only).
miss_counts <- vapply(mi_df, function(x) sum(is.na(x)), numeric(1))
miss_total <- sum(miss_counts)
```

```r
if (is.finite(miss_total) && miss_total > 0) {
  est_bytes <- miss_total * M_IMP * 8
  message("MI imputation storage estimate: ", format(structure(est_bytes, class = "object_size"), units = "auto"))
  if (est_bytes > MI_MAX_BYTES) {
    m_max <- floor(MI_MAX_BYTES / (miss_total * 8))
    m_target <- max(50L, m_max)
    if (m_target < 50L) {
      stop("Estimated MI storage exceeds memory (m=", M_IMP,
           ", estimated=", format(structure(est_bytes, class = "object_size"), units = "auto"),
           "). Reduce missingness or MI scope, or increase available memory.")
    }
    if (m_target < M_IMP) {
      message("Reducing M_IMP from ", M_IMP, " to ", m_target,
              " to stay within MI_MAX_BYTES.")
      M_IMP <- m_target
    }
  }
}

# Make binary comorbids factors so "logreg" is used (and stays binary)
bin_covars <- c("copd","asthma","osa","chf","acute_nmd","phtn","ckd","dm")
missing_bin <- setdiff(bin_covars, names(mi_df))
stopifnot(length(missing_bin) == 0)
mi_df[bin_covars] <- lapply(mi_df[bin_covars], function(z) {
  if (is.factor(z)) return(droplevels(z))
  zz <- suppressWarnings(as.integer(z))
  factor(zz, levels = c(0L,1L), labels = c("0","1"))
})

# For MICE: convert any remaining characters → factors
mi_df <- dplyr::mutate(mi_df, across(where(is.character), ~ factor(.x)))

# Guardrail: high-cardinality factors can blow up MICE model matrices.
# Exclude them from predictorMatrix and (if missing) make "Missing" explicit.
high_card <- names(which(vapply(mi_df, function(x) is.factor(x) && nlevels(x) > MAX_LEVELS_PRED, logical(1))))
if (length(high_card)) {
```

```
  message("MICE: high-cardinality factors detected (nlevels > ", MAX_LEVELS_PRED, "): ",
          paste(high_card, collapse = ", "))
  for (v in high_card) {
    if (any(is.na(mi_df[[v]]))) {
      lv <- levels(mi_df[[v]])
      tmp <- as.character(mi_df[[v]])
      tmp[is.na(tmp)] <- "Missing"
      mi_df[[v]] <- factor(tmp, levels = unique(c(lv, "Missing")))
      if (!is.null(levels_ref) && !is.null(levels_ref[[v]]) &&
          !"Missing" %in% levels_ref[[v]]) {
        levels_ref[[v]] <- c(levels_ref[[v]], "Missing")
      }
    }
  }
}

# --- methods & predictor matrix aligned to *mi_df* -----------------------------
meth <- mice::make.method(mi_df)

is_fac      <- vapply(mi_df, is.factor,  logical(1))
is_num      <- vapply(mi_df, is.numeric, logical(1))
is_bin_fac  <- vapply(mi_df, function(x) is.factor(x)  && nlevels(x) == 2, logical(1))
is_multicat <- vapply(mi_df, function(x) is.factor(x)  && nlevels(x) >  2, logical(1))

# robust defaults
meth[is_num]      <- "pmm"      # numerics: predictive mean matching
meth[is_multicat] <- "polyreg"  # unordered multicategory
meth[is_bin_fac]  <- "logreg"   # binary factors: logistic regression

# never impute treatments, outcomes, or CO2 exposures
no_imp <- c("has_abg","has_vbg","imv_proc","niv_proc","death_60d","hypercap_resp_failure",
            "paco2","vbg_co2")
if (length(high_card)) no_imp <- unique(c(no_imp, high_card))
meth[intersect(names(meth), no_imp)] <- ""

# predictor matrix; force has_abg/has_vbg as predictors, but do not impute no_imp
```

```r
pred <- mice::quickpred(mi_df, mincor = MINCOR_QUICKPRED, minpuc = MINPUC_QUICKPRED)
for (nm in intersect(c("has_abg", "has_vbg"), colnames(pred))) {
  pred[, nm] <- 1
}
pred[intersect(rownames(pred), no_imp), ] <- 0

if (length(high_card)) {
  pred[, intersect(high_card, colnames(pred))] <- 0
  pred[intersect(high_card, rownames(pred)), ] <- 0
}

# Ensure dropped covariates do not appear as predictors
drop_covars <- intersect(drop_vars_ultra_missing, colnames(pred))
if (length(drop_covars)) {
  pred[, drop_covars] <- 0
}

# Non-imputed variables with missingness should NOT be predictors
no_imp_with_missing <- intersect(no_imp, names(mi_df))
no_imp_with_missing <- no_imp_with_missing[
  vapply(mi_df[no_imp_with_missing], function(x) any(is.na(x)), logical(1))
]
no_imp_with_missing <- setdiff(no_imp_with_missing, c("has_abg", "has_vbg"))

pred_cols_check <- intersect(c("paco2", "vbg_co2"), colnames(pred))
if (length(pred_cols_check)) {
  message(
    "MICE: predictor column sums (pre-exclude) for ",
    paste(pred_cols_check, collapse = ", "),
    ": ",
    paste(colSums(pred[, pred_cols_check, drop = FALSE]), collapse = ", ")
  )
}
if (length(no_imp_with_missing)) {
  pred[, intersect(no_imp_with_missing, colnames(pred))] <- 0
  message("MICE: excluded non-imputed missing predictors: ",
```

```
          paste(no_imp_with_missing, collapse = ", "))
}
if (length(pred_cols_check)) {
  message(
    "MICE: predictor column sums (post-exclude) for ",
    paste(pred_cols_check, collapse = ", "),
    ": ",
    paste(colSums(pred[, pred_cols_check, drop = FALSE]), collapse = ", ")
  )
}

# Ensure key covariates with missingness have predictors (avoid zero-row pred)
core_preds <- intersect(
  c("age_at_encounter", "sex", "race_ethnicity", "location", "encounter_type",
    "has_abg", "has_vbg", "imv_proc", "niv_proc", "death_60d",
    "hypercap_resp_failure", "paco2", "vbg_co2"),
  colnames(pred)
)
core_preds <- setdiff(core_preds, no_imp_with_missing)
vars_need_pred <- intersect(covars_gbm, rownames(pred))
vars_need_pred <- setdiff(vars_need_pred, no_imp)
vars_need_pred <- vars_need_pred[vapply(mi_df[vars_need_pred], function(x) any(is.na(x)), logical(1))]
zero_pred <- vars_need_pred[rowSums(pred[vars_need_pred, , drop = FALSE] != 0, na.rm = TRUE) == 0]
if (length(zero_pred)) {
  message("MICE predictor rows empty for: ", paste(zero_pred, collapse = ", "),
          ". Adding core predictors.")
  pred[zero_pred, core_preds] <- 1
}
pred[intersect(rownames(pred), no_imp), ] <- 0
pred[c(zero_pred), c(zero_pred)] <- 0

# --- Preflight and cap predictor rows to avoid huge model matrices -------------

n_mm_cols <- function(pred_row, df) {
  preds <- names(which(pred_row != 0))
  cols <- 1L
```

```
  for (p in preds) {
    x <- df[[p]]
    if (is.factor(x)) {
      lv <- nlevels(x)
      cols <- cols + max(1L, lv - 1L)
    } else {
      cols <- cols + 1L
    }
  }
  cols
}

pred_width_preflight <- function(pred, df, meth) {
  vars <- names(meth)[meth != ""]
  vars <- vars[vapply(df[vars], function(x) any(is.na(x)), logical(1))]
  rows <- lapply(vars, function(v) {
    pred_row <- pred[v, ]
    n_pred <- sum(pred_row != 0, na.rm = TRUE)
    mm_cols <- n_mm_cols(pred_row, df)
    x <- df[[v]]
    nlevels_v <- if (is.factor(x)) nlevels(x) else NA_integer_
    miss_n <- sum(is.na(x))
    data.frame(
      variable = v,
      method = meth[[v]],
      n_pred = n_pred,
      mm_cols = mm_cols,
      nlevels_v = nlevels_v,
      miss_n = miss_n,
      stringsAsFactors = FALSE
    )
  })
  if (!length(rows)) {
    return(data.frame(variable = character(), method = character(), n_pred = integer(),
                      mm_cols = integer(), nlevels_v = integer(), miss_n = integer()))
  }
```

```
  dplyr::bind_rows(rows)
}

preflight_pre <- pred_width_preflight(pred, mi_df, meth) |>
  dplyr::mutate(stage = "pre")
if (nrow(preflight_pre)) {
  top_mm <- preflight_pre |> dplyr::arrange(dplyr::desc(mm_cols)) |> dplyr::slice_head(n = 10)
  top_np <- preflight_pre |> dplyr::arrange(dplyr::desc(n_pred)) |> dplyr::slice_head(n = 10)
  print(top_mm)
  print(top_np)
}

core_preds <- intersect(
  c("age_at_encounter", "sex", "race_ethnicity", "location", "encounter_type",
    "has_abg", "has_vbg", "imv_proc", "niv_proc", "death_60d",
    "hypercap_resp_failure", "paco2", "vbg_co2"),
  colnames(pred)
)
core_preds <- setdiff(core_preds, no_imp_with_missing)

set.seed(MI_SEED + 100)
idx <- sample.int(nrow(mi_df), min(COR_SAMPLE_N, nrow(mi_df)))

vars_cap <- names(meth)[meth != ""]
vars_cap <- vars_cap[vapply(mi_df[vars_cap], function(x) any(is.na(x)), logical(1))]

pred_nlevels <- function(x) {
  if (is.factor(x)) nlevels(x) else 1L
}

for (v in vars_cap) {
  cand <- names(which(pred[v, ] != 0))
  if (!length(cand)) next
  keep <- intersect(core_preds, cand)
  rem  <- setdiff(cand, keep)
```

```
  if (length(rem)) {
    y_raw <- mi_df[[v]][idx]
    y <- if (is.factor(y_raw)) as.integer(y_raw) else suppressWarnings(as.numeric(y_raw))
    scores <- vapply(rem, function(r) {
      x_raw <- mi_df[[r]][idx]
      x <- if (is.factor(x_raw)) as.integer(x_raw) else suppressWarnings(as.numeric(x_raw))
      suppressWarnings(abs(stats::cor(y, x, use = "pairwise.complete.obs")))
    }, numeric(1))
    ord <- order(is.na(scores), -scores)
    rem_keep <- rem[ord]
    rem_keep <- rem_keep[seq_len(min(length(rem_keep), max(0L, MAX_PRED_PER_VAR - length(keep))))]
  } else {
    rem_keep <- character()
  }

  keep_all <- unique(c(keep, rem_keep))
  pred[v, ] <- 0
  pred[v, keep_all] <- 1
  pred[v, v] <- 0

  mm_cols <- n_mm_cols(pred[v, ], mi_df)
  if (mm_cols > MAX_MM_COLS) {
    drop_pool <- setdiff(keep_all, keep)
    if (!length(drop_pool)) drop_pool <- keep
    drop_order <- drop_pool[order(vapply(drop_pool, function(p) pred_nlevels(mi_df[[p]]), integer(1)),
                                  decreasing = TRUE)]
    for (p in drop_order) {
      if (p %in% keep && length(keep) == 1) break
      pred[v, p] <- 0
      mm_cols <- n_mm_cols(pred[v, ], mi_df)
      if (mm_cols <= MAX_MM_COLS) break
    }
  }
}

preflight_post <- pred_width_preflight(pred, mi_df, meth) |>
```

```
  dplyr::mutate(stage = "post")
preflight_all <- dplyr::bind_rows(preflight_pre, preflight_post)
write_csv_safely(preflight_all, results_path("mice_pred_width_preflight.csv"), row_names = FALSE)
if (nrow(preflight_post)) {
  top_mm_post <- preflight_post |> dplyr::arrange(dplyr::desc(mm_cols)) |> dplyr::slice_head(n = 10)
  print(top_mm_post)
}

# MI integrity: treatments/outcomes excluded; binaries use logreg
stopifnot(all(meth[no_imp] == ""))
stopifnot(all(rowSums(pred[intersect(rownames(pred), no_imp), , drop = FALSE]) == 0))
bin_fac <- names(which(vapply(mi_df, function(x) is.factor(x) && nlevels(x) == 2, logical(1))))
bin_fac <- setdiff(bin_fac, no_imp)
stopifnot(all(meth[bin_fac] == "logreg"))

# integrity checks
stopifnot(
  ncol(pred) == ncol(mi_df),
  nrow(pred) == ncol(mi_df),
  length(meth) == ncol(mi_df),
  identical(names(meth), colnames(mi_df))
)

# --- run MICE ----------------------------------------------------------------
mi_df_run <- mi_df
M_IMP_RUN <- M_IMP
MAXIT_RUN <- MAXIT_MI
FORCE_MI_BATCHED <- nrow(mi_df_run) > MI_BATCH_THRESHOLD

if (!requireNamespace("digest", quietly = TRUE)) {
  stop("Package 'digest' is required for MI checkpoint signatures.")
}

make_mi_signature <- function(df) {
  classes <- vapply(df, function(x) class(x)[1], character(1))
  na_counts <- vapply(df, function(x) sum(is.na(x)), integer(1))
```

```
  nlevels <- vapply(df, function(x) if (is.factor(x)) nlevels(x) else NA_integer_, integer(1))
  lvl_hash <- vapply(df, function(x) {
    if (!is.factor(x)) return(NA_character_)
    digest::digest(levels(x), algo = "xxhash64")
  }, character(1))
  col_sig <- data.frame(
    name = names(df),
    class = classes,
    na = na_counts,
    nlevels = nlevels,
    lvl_hash = lvl_hash,
    stringsAsFactors = FALSE
  )
  hash <- digest::digest(
    list(dim = dim(df), col_sig = col_sig),
    algo = "xxhash64"
  )
  list(
    hash = hash,
    nrow = nrow(df),
    ncol = ncol(df),
    run_mode = RUN_MODE,
    pilot_frac = PILOT_FRAC,
    mi_pilot_mode = RUN_MODE,
    sample_seed = SAMPLE_SEED,
    mi_seed = MI_SEED,
    col_sig = col_sig
  )
}

mi_smoke_log_path <- results_path("mice_smoketest.log")
write_smoke_log <- function(lines) {
  write_diag_lines(lines, mi_smoke_log_path)
}

mem_max <- tryCatch(mem.maxVSize(), error = function(e) NA_real_)
```

```r
mem_env <- Sys.getenv("R_MAX_VSIZE", unset = NA_character_)
gc_pre <- utils::capture.output(gc())
write_smoke_log(c(
  paste0("MI smoke test log: ", Sys.time()),
  paste0("mem.maxVSize: ", ifelse(is.na(mem_max), "NA", format(mem_max, scientific = FALSE))),
  paste0("R_MAX_VSIZE env: ", ifelse(nchar(mem_env), mem_env, "NA")),
  paste0("mi_df size: ", format(utils::object.size(mi_df), units = "auto")),
  paste0("mi_df_run size: ", format(utils::object.size(mi_df_run), units = "auto")),
  "gc() pre:",
  gc_pre
))
if (RUN_MODE == "full" && is.finite(mem_max) && mem_max < 2.2e10) {
  message("Full run on ~16GB memory: consider running MI/weights on a >32GB machine for speed.")
}

subset_data_saved <- FALSE
subset_data_path <- results_path("subset_data_pre_mi.rds")
if (MI_MEMORY_HYGIENE) {
  saveRDS(subset_data, subset_data_path)
  rm(subset_data)
  subset_data_saved <- TRUE
  invisible(gc())
}

run_mice_call <- function(m_val, maxit, seed_val, print_flag = FALSE) {
  mice::mice(
    data            = mi_df_run,
    m               = m_val,
    maxit           = maxit,
    predictorMatrix = pred,
    method          = meth,
    printFlag       = print_flag,
    seed            = seed_val
  )
}
```

```r
if (MI_SMOKE_TEST) {
  smoke_con <- file(mi_smoke_log_path, open = "at")
  sink(smoke_con, type = "output")

  cap_smoke <- capture_warnings(
    tryCatch(
      run_mice_call(m_val = 1L, maxit = 1L, seed_val = MI_SEED, print_flag = TRUE),
      error = function(e) e
    ),
    context = make_context(
      stage = "MI",
      component = "mice_smoketest",
      analysis_variant = "mi",
      model_type = "mice",
      group = NA_character_,
      outcome = NA_character_,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_warnings(cap_smoke$warnings)
  sink(type = "output")
  close(smoke_con)

  if (inherits(cap_smoke$value, "error")) {
    smoke_msg <- conditionMessage(cap_smoke$value)
    message("MICE smoke test failed: ", smoke_msg)
    write_smoke_log(c(
      "Smoke test failed.",
      paste0("Error: ", smoke_msg)
    ))
    if (grepl("vector memory limit", smoke_msg, fixed = TRUE)) {
      MI_DEBUG_PRINTFLAG <- TRUE
    }
    if (subset_data_saved) {
      subset_data <- readRDS(subset_data_path)
```

```
    }
    if (MI_DEBUG_PRINTFLAG) {
      debug_path <- results_path("mice_debug_print.txt")
      dbg_con <- file(debug_path, open = "wt")
      sink(dbg_con, type = "output")
      message("Debug MI: running m=1, maxit=1 with printFlag=TRUE.")
      tryCatch(
        run_mice_call(m_val = 1L, maxit = 1L, seed_val = MI_SEED, print_flag = TRUE),
        error = function(e) e
      )
      sink(type = "output")
      close(dbg_con)
    }
    stop("MICE smoke test failed; see ", mi_smoke_log_path, " and Results/mice_debug_print.txt.")
  } else {
    message("MICE smoke test succeeded (m=1, maxit=1).")
  }
}

get_vcells_stats <- function() {
  g <- gc()
  cn <- colnames(g)
  mb_cols <- cn[grepl("\\(Mb\\)", cn)]
  used_mb_col <- if (length(mb_cols)) mb_cols[1] else NA_character_
  limit_col <- cn[grepl("limit", cn, ignore.case = TRUE)]
  limit_col <- if (length(limit_col)) limit_col[1] else NA_character_
  trig_col <- cn[grepl("trigger", cn, ignore.case = TRUE)]
  trig_col <- if (length(trig_col)) trig_col[1] else NA_character_
  max_col <- cn[grepl("max", cn, ignore.case = TRUE)]
  max_col <- if (length(max_col)) max_col[1] else NA_character_

  used_mb <- if (!is.na(used_mb_col)) as.numeric(g["Vcells", used_mb_col]) else NA_real_
  limit_mb <- if (!is.na(limit_col)) as.numeric(g["Vcells", limit_col]) else NA_real_
  trig_mb <- if (!is.na(trig_col)) as.numeric(g["Vcells", trig_col]) else NA_real_
  max_mb <- if (!is.na(max_col)) as.numeric(g["Vcells", max_col]) else NA_real_
```

```
  if (!is.finite(limit_mb) || limit_mb <= 0) {
    mem_lim <- tryCatch(mem.maxVSize(), error = function(e) NA_real_)
    if (is.finite(mem_lim) && mem_lim > 0) {
      limit_mb <- if (mem_lim < 1e8) mem_lim else mem_lim / 1024^2
    }
  }

  data.frame(
    gc_vcells_used_mb = used_mb,
    gc_vcells_limit_mb = limit_mb,
    gc_vcells_frac = ifelse(is.finite(limit_mb) && limit_mb > 0, used_mb / limit_mb, NA_real_),
    gc_vcells_trigger_mb = trig_mb,
    gc_vcells_max_used_mb = max_mb,
    stringsAsFactors = FALSE
  )
}

set.seed(MI_SEED)
run_mice_single <- function(m_val) {
  runtime_logger(
    "mice_imputation",
    run_mice_call(m_val = m_val, maxit = MAXIT_RUN, seed_val = MI_SEED, print_flag = FALSE),
    notes = paste0("m=", m_val, "; maxit=", MAXIT_RUN)
  )
}

mc_progress <- list()
sentinel_specs <- list(
  list(name = "abg_imv", outcome = "imv_proc", treat = "has_abg", co2_var = "paco2",
       low = ABG_CO2_LOW, high = ABG_CO2_HIGH),
  list(name = "vbg_imv", outcome = "imv_proc", treat = "has_vbg", co2_var = "vbg_co2",
       low = VBG_CO2_LOW, high = VBG_CO2_HIGH)
)

mcerr_ratio_for_spec <- function(imp_obj, spec) {
  fits <- lapply(seq_len(imp_obj$m), function(i) {
```

```
    d <- mice::complete(imp_obj, action = i)
    d <- d[, c(spec$outcome, spec$treat, spec$co2_var), drop = FALSE]
    d <- d[d[[spec$treat]] == 1 & is.finite(d[[spec$co2_var]]), , drop = FALSE]
    if (nrow(d) == 0L) return(NULL)
    d$co2_cat <- make_co2_cat3(d[[spec$co2_var]], spec$low, spec$high)
    d$co2_cat <- stats::relevel(base::droplevels(d$co2_cat), ref = "Normal")
    if (dplyr::n_distinct(d[[spec$outcome]]) < 2L) return(NULL)
    fit <- tryCatch(
      stats::glm(stats::reformulate("co2_cat", response = spec$outcome),
                 data = d, family = binomial(), x = FALSE, y = FALSE, model = FALSE),
      error = function(e) NULL
    )
    if (is.null(fit)) return(NULL)
    list(coef = stats::coef(fit), vcov = stats::vcov(fit))
  })
  fits <- fits[!vapply(fits, is.null, logical(1))]
  if (length(fits) < 2L) return(NA_real_)
  results <- lapply(fits, `[[`, "coef")
  variances <- lapply(fits, `[[`, "vcov")
  pooled <- mitools::MIcombine(results = results, variances = variances)
  est <- as.numeric(stats::coef(pooled))
  names(est) <- names(stats::coef(pooled))
  se <- sqrt(diag(pooled$variance))
  names(se) <- names(stats::coef(pooled))
  coef_mat <- do.call(cbind, lapply(results, function(v) v[names(est)]))
  B <- apply(coef_mat, 1, stats::var, na.rm = TRUE)
  mcerr <- sqrt(B / length(results))
  ratio <- mcerr / se
  idx <- grepl("^co2_cat", names(ratio))
  if (!any(idx)) return(max(ratio, na.rm = TRUE))
  max(ratio[idx], na.rm = TRUE)
}

run_mice_batched <- function(m_total, m_batch_start, maxit, base_seed) {
  imp_acc <- NULL
  m_done <- 0L
```

```
batch_attempt_idx <- 0L
m_batch <- m_batch_start
batch_log <- list()
logged_events_acc <- list()

while (m_done < m_total) {
  if (MI_GC_EVERY_BATCH) invisible(gc())
  mem_stats <- get_vcells_stats()
  if (MI_PREEMPTIVE_BATCH_REDUCE &&
      is.finite(mem_stats$gc_vcells_frac) &&
      mem_stats$gc_vcells_frac > MI_VCELLS_FRAC_THRESHOLD &&
      m_batch > MI_BATCH_MIN) {
    m_batch <- max(MI_BATCH_MIN, floor(m_batch / 2))
    message("Preemptively reducing MI batch size to ", m_batch,
            " (Vcells pressure: ", round(mem_stats$gc_vcells_frac, 2), ").")
  }

  m_b <- min(m_batch, m_total - m_done)
  batch_attempt_idx <- batch_attempt_idx + 1L
  seed_b <- base_seed + batch_attempt_idx * MI_BATCH_SEED_STRIDE
  message("MICE batch ", batch_attempt_idx, " (m=", m_b, ", seed=", seed_b, ")")

  t0 <- Sys.time()
  mem_pre <- get_vcells_stats()
  cap <- capture_warnings(
    tryCatch(
      runtime_logger(
        paste0("mice_batch_", batch_attempt_idx),
        mice::mice(
          data            = mi_df_run,
          m               = m_b,
          maxit           = maxit,
          predictorMatrix = pred,
          method          = meth,
          printFlag       = FALSE,
          seed            = seed_b
```

```
        ),
        notes = paste0("batch=", batch_attempt_idx, "; m=", m_b, "; maxit=", maxit)
      ),
      error = function(e) e
    ),
    context = make_context(
      stage = "MI",
      component = "mice_batch",
      analysis_variant = "mi",
      model_type = "mice",
      group = NA_character_,
      outcome = NA_character_,
      imputation = NA_integer_,
      batch = batch_attempt_idx
    )
  )
  append_warnings(cap$warnings)

  imp_b <- cap$value
  if (inherits(imp_b, "error")) {
    err_msg <- conditionMessage(imp_b)
    message("MICE batch ", batch_attempt_idx, " failed: ", err_msg)
    mem_stats <- get_vcells_stats()
    batch_log[[batch_attempt_idx]] <- data.frame(
      batch = batch_attempt_idx,
      m_batch = m_b,
      seed = seed_b,
      ok = FALSE,
      error_message = err_msg,
      seconds = as.numeric(difftime(Sys.time(), t0, units = "secs")),
      gc_vcells_used_mb_pre = mem_pre$gc_vcells_used_mb,
      gc_vcells_limit_mb_pre = mem_pre$gc_vcells_limit_mb,
      gc_vcells_frac_pre = mem_pre$gc_vcells_frac,
      gc_vcells_used_mb_post = mem_stats$gc_vcells_used_mb,
      gc_vcells_limit_mb_post = mem_stats$gc_vcells_limit_mb,
      gc_vcells_frac_post = mem_stats$gc_vcells_frac,
```

```
      stringsAsFactors = FALSE
    )
    if (grepl("vector memory limit", err_msg, fixed = TRUE) && m_batch > MI_BATCH_MIN) {
      m_batch <- max(MI_BATCH_MIN, floor(m_batch / 2))
      message("Reducing MI batch size to ", m_batch, " and retrying.")
      invisible(gc())
      next
    }
    write_csv_safely(dplyr::bind_rows(batch_log), results_path("mice_batches_log.csv"), row_names = FALSE)
    stop("MICE batch ", batch_attempt_idx, " failed; see log: ", results_path("mice_batches_log.csv"))
  }

  if (is.null(imp_acc)) {
    imp_acc <- imp_b
  } else {
    imp_acc <- mice::ibind(imp_acc, imp_b)
  }

  le_b <- imp_b$loggedEvents
  le_b <- if (is.null(le_b)) data.frame() else as.data.frame(le_b)
  if (NROW(le_b) > 0) {
    le_b <- le_b |>
      dplyr::mutate(
        batch = batch_attempt_idx,
        seed = seed_b,
        m_global_start = m_done + 1L
      )
    logged_events_acc[[length(logged_events_acc) + 1L]] <- le_b
  }

  m_done <- imp_acc$m
  mem_stats <- get_vcells_stats()
  batch_log[[batch_attempt_idx]] <- data.frame(
    batch = batch_attempt_idx,
    m_batch = m_b,
    seed = seed_b,
```

```
      ok = TRUE,
      error_message = NA_character_,
      seconds = as.numeric(difftime(Sys.time(), t0, units = "secs")),
      gc_vcells_used_mb_pre = mem_pre$gc_vcells_used_mb,
      gc_vcells_limit_mb_pre = mem_pre$gc_vcells_limit_mb,
      gc_vcells_frac_pre = mem_pre$gc_vcells_frac,
      gc_vcells_used_mb_post = mem_stats$gc_vcells_used_mb,
      gc_vcells_limit_mb_post = mem_stats$gc_vcells_limit_mb,
      gc_vcells_frac_post = mem_stats$gc_vcells_frac,
      stringsAsFactors = FALSE
    )
    rm(imp_b)
    if (MI_GC_EVERY_BATCH) invisible(gc())

    if (ALLOW_M_IMP_EARLY_STOP &&
        m_done >= M_IMP_MIN &&
        (m_done %% M_IMP_STEP == 0 || m_done == m_total)) {
      ratios <- vapply(sentinel_specs, function(s) mcerr_ratio_for_spec(imp_acc, s), numeric(1))
      mc_progress[[length(mc_progress) + 1L]] <<- data.frame(
        m = m_done,
        abg_ratio = ratios[["abg_imv"]],
        vbg_ratio = ratios[["vbg_imv"]],
        max_ratio = max(ratios, na.rm = TRUE),
        stringsAsFactors = FALSE
      )
      write_csv_safely(dplyr::bind_rows(mc_progress),
                       results_path("mi_mcerr_progress.csv"),
                       row_names = FALSE)
      if (all(is.finite(ratios)) && max(ratios, na.rm = TRUE) <= MCERR_RATIO_TARGET) {
        message("MC error criterion met at m=", m_done,
                " (max MCerr/SE=", round(max(ratios, na.rm = TRUE), 3), "). Stopping early.")
        break
      }
    }
  }
```

```
  write_csv_safely(dplyr::bind_rows(batch_log), results_path("mice_batches_log.csv"), row_names = FALSE)
  log_events_raw_batched <- dplyr::bind_rows(logged_events_acc)
  attr(imp_acc, "logged_events_batched") <- log_events_raw_batched
  imp_acc
}


imp <- NULL
use_batched <- isTRUE(FORCE_MI_BATCHED)
if (!use_batched) {
  cap_mice <- capture_warnings(
    tryCatch(run_mice_single(M_IMP_RUN), error = function(e) e),
    context = make_context(
      stage = "MI",
      component = "mice",
      analysis_variant = "mi",
      model_type = "mice",
      group = NA_character_,
      outcome = NA_character_,
      imputation = NA_integer_,
      batch = NA_integer_
    )
  )
  append_warnings(cap_mice$warnings)
  imp <- cap_mice$value
  if (inherits(imp, "error") && grepl("vector memory limit", conditionMessage(imp), fixed = TRUE)) {
    message("MICE memory limit hit; switching to batched mode.")
    use_batched <- TRUE
  }
}
if (use_batched) {
  message("MICE: running in batches (start=", MI_BATCH_START, ").")
  imp <- run_mice_batched(M_IMP_RUN, MI_BATCH_START, MAXIT_RUN, MI_SEED)
} else {
  write_csv_safely(data.frame(), results_path("mice_batches_log.csv"), row_names = FALSE)
}
```

```r
if (inherits(imp, "error")) stop(imp)
stopifnot(inherits(imp, "mids"))
if (ALLOW_M_IMP_EARLY_STOP && imp$m < M_IMP_RUN) {
  message("Early stop: stopping at m=", imp$m, " (target ", M_IMP_RUN, ").")
  M_IMP_RUN <- imp$m
} else {
  stopifnot(imp$m == M_IMP_RUN)
}
if (M_IMP != M_IMP_RUN) M_IMP <- M_IMP_RUN
if (MAXIT_MI != MAXIT_RUN) MAXIT_MI <- MAXIT_RUN
write_csv_safely(dplyr::bind_rows(mc_progress), results_path("mi_mcerr_progress.csv"), row_names = FALSE)
saveRDS(imp, file = mi_mids_file)

# Save MICE spec for reproducibility
saveRDS(
  list(method = imp$method, predictorMatrix = imp$predictorMatrix),
  results_path("mice_spec.rds")
)
if (use_batched) {
  message("Multiple imputation was run in batches and combined via mice::ibind().")
}
if (subset_data_saved) {
  subset_data <- readRDS(subset_data_path)
}

# Logged events: raw + summary (by dep/out/meth)
log_events_raw <- as.data.frame(imp$loggedEvents)
log_events_batched <- as.data.frame(attr(imp, "logged_events_batched"))
log_events_raw <- dplyr::bind_rows(log_events_raw, log_events_batched)

if (nrow(log_events_raw)) {
  write_csv_safely(log_events_raw, results_path("mice_logged_events_raw.csv"), row_names = FALSE,
                   required_cols = c("dep", "out", "meth"))
  log_events_summary <- log_events_raw |>
    dplyr::count(dep, out, meth, name = "n") |>
    dplyr::mutate(variable = dep) |>
```

```
    dplyr::arrange(dplyr::desc(n)) |>
    dplyr::mutate(pct = n / sum(n))
  write_csv_safely(log_events_summary, results_path("mice_logged_events_summary.csv"), row_names = FALSE,
                   required_cols = c("variable", "n", "pct"))
} else {
  log_events_raw_empty <- data.frame(
    dep = character(), out = character(), meth = character(),
    stringsAsFactors = FALSE
  )
  log_events_summary_empty <- data.frame(
    variable = character(), n = integer(), pct = numeric(),
    stringsAsFactors = FALSE
  )
  write_csv_safely(log_events_raw_empty, results_path("mice_logged_events_raw.csv"), row_names = FALSE,
                   required_cols = c("dep", "out", "meth"))
  write_csv_safely(log_events_summary_empty, results_path("mice_logged_events_summary.csv"), row_names = FALSE,
                   required_cols = c("variable", "n", "pct"))
  log_events_summary <- log_events_summary_empty
}
if (nrow(mi_info_log)) {
  warns_events <- mi_info_log |>
    dplyr::filter(stage == "MI", component %in% c("mice", "mice_batch"))
  if (nrow(warns_events) && nrow(log_events_raw) == 0L) {
    warning("Mismatch: main MI run reported logged events but loggedEvents table is empty; ",
            "check batch capture and loggedEvents exports.", call. = FALSE)
  }
}

# Chain diagnostics (lightweight; no complete("long"))
chain_diag <- data.frame()
chain_diag_stats <- list(
  n_imputed_vars = 0L,
  n_with_chainMean = 0L,
  n_with_drift_tail = 0L,
  drift_tail_na_frac = NA_real_,
  tail_window_na_mean = NA_real_
```

```
)
stopifnot(!is.null(imp$method))
impute_vars <- names(imp$method)[imp$method != ""]
impute_vars <- intersect(impute_vars, names(imp$data))
if (length(impute_vars)) {
  impute_vars <- impute_vars[vapply(imp$data[impute_vars], function(x) any(is.na(x)), logical(1))]
}
stopifnot(!is.null(imp$chainMean))
{
  cm <- imp$chainMean
  dims <- dim(cm)
  dn <- dimnames(cm)
  iter_candidates <- unique(c(imp$iteration, MAXIT_MI, MAXIT_MI + 1L))
  iter_candidates <- iter_candidates[is.finite(iter_candidates) & iter_candidates > 0]
  imp_m <- imp$m

  extract_chain_mean <- function(cm_obj, dims_obj, dn_obj, imp_m_val, iter_cand) {
    mean_chain <- NULL
    var_names <- NULL
    iter_dim <- NA_integer_
    var_dim <- NA_integer_
    m_dim <- NA_integer_
    if (!is.null(dims_obj) && length(dims_obj) == 2) {
      iter_dim <- which(dims_obj %in% iter_cand)[1]
      if (length(iter_dim) == 0) iter_dim <- 1L
      var_dim <- setdiff(seq_along(dims_obj), iter_dim)[1]
      mean_chain <- cm_obj
      if (iter_dim == 2L) mean_chain <- t(mean_chain)
    } else if (!is.null(dims_obj) && length(dims_obj) == 3) {
      m_dim <- which(dims_obj == imp_m_val)
      if (length(m_dim)) {
        m_dim <- m_dim[1]
      } else {
        m_dim <- 3L
      }
      iter_dim <- setdiff(which(dims_obj %in% iter_cand), m_dim)[1]
```

```
    if (length(iter_dim) == 0) iter_dim <- setdiff(1:3, c(m_dim, NA_integer_))[1]
    var_dim <- setdiff(1:3, c(iter_dim, m_dim))[1]
    if (all(is.finite(c(iter_dim, var_dim, m_dim)))) {
      cm_std <- aperm(cm_obj, c(iter_dim, var_dim, m_dim))
      mean_chain <- apply(cm_std, c(1, 2), mean, na.rm = TRUE)
      if (!is.null(dimnames(cm_std))) {
        if (!is.null(dimnames(cm_std)[[2]])) {
          var_names <- dimnames(cm_std)[[2]]
        }
        if (!is.null(dimnames(cm_std)[[1]])) {
          rownames(mean_chain) <- dimnames(cm_std)[[1]]
        }
      }
    }
  }
  if (is.null(mean_chain)) {
    return(list(mean_chain = NULL, var_names = NULL, iter_dim = iter_dim, var_dim = var_dim, m_dim = m_dim))
  }
  if (is.null(var_names) && !is.null(dn_obj) && length(dn_obj) >= var_dim) {
    var_names <- dn_obj[[var_dim]]
  }
  if (is.null(var_names)) {
    var_names <- colnames(mean_chain)
  }
  list(mean_chain = mean_chain, var_names = var_names, iter_dim = iter_dim, var_dim = var_dim, m_dim = m_dim)
}

res_chain <- extract_chain_mean(cm, dims, dn, imp_m, iter_candidates)
mean_chain <- res_chain$mean_chain
var_names <- res_chain$var_names
if (is.null(mean_chain)) {
  stop("Chain diagnostics: unable to construct mean_chain from imp$chainMean.")
}

{
  iter_idx <- seq_len(nrow(mean_chain))
```

```
  numeric_names <- !is.null(var_names) && all(grepl("^\\d+$", var_names))
  if (is.null(var_names) || numeric_names) {
    if (length(impute_vars) && length(impute_vars) == ncol(mean_chain)) {
      var_names <- impute_vars
      numeric_names <- FALSE
    }
  }
  if (is.null(var_names)) {
    stop("Chain diagnostics: variable names missing and could not be matched to imp$method.")
  }
  colnames(mean_chain) <- var_names

  if (numeric_names) {
    warn_df <- data.frame(
      time = as.character(Sys.time()),
      stage = "MI",
      component = "chain_diagnostics",
      analysis_variant = NA_character_,
      model_type = NA_character_,
      group = NA_character_,
      outcome = NA_character_,
      imputation = NA_integer_,
      batch = NA_integer_,
      message = "Chain diagnostics variable name mapping failed; using numeric/fallback names.",
      stringsAsFactors = FALSE
    )
    append_warnings(warn_df)
  }

  vars_imputed <- impute_vars
  chain_diag_stats$n_imputed_vars <- length(vars_imputed)
  keep_vars <- intersect(vars_imputed, var_names)
  missing_in_chain <- setdiff(vars_imputed, var_names)
  if (length(missing_in_chain)) {
    missing_df <- data.frame(
      variable = missing_in_chain,
```

```
      stringsAsFactors = FALSE
    )
    write_csv_safely(missing_df, results_path("mice_chain_diagnostics_missing_vars.csv"), row_names = FALSE)
  }
  if (length(vars_imputed) && length(keep_vars) == 0L) {
    warn_df <- data.frame(
      time = as.character(Sys.time()),
      stage = "MI",
      component = "chain_diagnostics",
      analysis_variant = NA_character_,
      model_type = NA_character_,
      group = NA_character_,
      outcome = NA_character_,
      imputation = NA_integer_,
      batch = NA_integer_,
      message = "Chain diagnostics: no imputed variables matched chainMean names; skipping drift metrics.",
      stringsAsFactors = FALSE
    )
    append_warnings(warn_df)
    mean_chain <- mean_chain[, 0, drop = FALSE]
    var_names <- character()
  } else if (length(keep_vars)) {
    mean_chain <- mean_chain[, keep_vars, drop = FALSE]
    var_names <- keep_vars
  }

  safe_sd <- function(x) if (sum(is.finite(x)) < 2) NA_real_ else stats::sd(x, na.rm = TRUE)
  safe_maxdiff <- function(x) {
    x <- x[is.finite(x)]
    if (length(x) < 2) return(NA_real_)
    max(abs(diff(x)))
  }
  safe_slope <- function(x, iter) {
    ok <- is.finite(x)
    if (sum(ok) < 2) return(NA_real_)
    coef(stats::lm(x[ok] ~ iter[ok]))[2]
```

```
}
tail_finite <- function(x, k) tail(x[is.finite(x)], k)
safe_maxdiff_tail <- function(x, k) {
  xf <- tail_finite(x, k)
  if (length(xf) < 2) return(NA_real_)
  max(abs(diff(xf)))
}

n_vars <- ncol(mean_chain)
n_finite <- integer(n_vars)
drift_all <- numeric(n_vars)
drift_tail <- numeric(n_vars)
tail_n_finite <- integer(n_vars)
tail_window_na_frac <- numeric(n_vars)
slope <- numeric(n_vars)
sd_chain <- numeric(n_vars)
overall_reason <- character(n_vars)
tail_reason <- character(n_vars)
flag_reason <- character(n_vars)
flag <- logical(n_vars)
diagnostic_available <- logical(n_vars)
sd_obs <- rep(NA_real_, n_vars)

chain_src_df <- mi_df_run
stopifnot(is.data.frame(chain_src_df))
sd_obs <- vapply(var_names, function(v) {
  stopifnot(v %in% names(chain_src_df))
  x <- chain_src_df[[v]]
  if (inherits(x, "haven_labelled")) x <- suppressWarnings(as.numeric(x))
  if (!is.numeric(x)) return(NA_real_)
  safe_sd(x)
}, numeric(1))

for (j in seq_len(n_vars)) {
  x <- mean_chain[, j]
  n_finite[j] <- sum(is.finite(x))
```

```
    drift_all[j] <- safe_maxdiff(x)
    slope[j] <- safe_slope(x, iter_idx)
    sd_chain[j] <- safe_sd(x)
    tail_vals <- tail_finite(x, TAIL_N_FINITE)
    tail_n_finite[j] <- length(tail_vals)
    drift_tail[j] <- safe_maxdiff_tail(x, TAIL_N_FINITE)
    tail_window <- tail(x, min(TAIL_WINDOW_ITERS, length(x)))
    tail_window_na_frac[j] <- mean(!is.finite(tail_window))
    overall_reason[j] <- if (n_finite[j] >= 2) "ok" else "insufficient_finite_overall"
    tail_reason[j] <- if (tail_n_finite[j] >= 2) "ok" else "insufficient_finite_tail"
  }

  drift_all_scaled <- drift_all / sd_obs
  drift_tail_scaled <- drift_tail / sd_obs
  slope_scaled <- slope / sd_obs

  for (j in seq_len(n_vars)) {
    if (!is.finite(sd_obs[j]) || sd_obs[j] <= 0) {
      flag[j] <- NA
      flag_reason[j] <- "missing_scale"
      diagnostic_available[j] <- FALSE
      next
    }
    if (tail_n_finite[j] < 2) {
      flag[j] <- NA
      flag_reason[j] <- "tail_insufficient"
      diagnostic_available[j] <- FALSE
      next
    }
    if (!is.finite(drift_tail_scaled[j]) || !is.finite(slope_scaled[j])) {
      flag[j] <- NA
      flag_reason[j] <- "insufficient_data"
      diagnostic_available[j] <- FALSE
      next
    }
    flag_tail <- drift_tail_scaled[j] > 0.01
```

```
    flag_slope <- abs(slope_scaled[j]) > 0.001
    flag[j] <- flag_tail | flag_slope
    flag_reason[j] <- if (flag_tail & flag_slope) {
      "both"
    } else if (flag_tail) {
      "tail_drift"
    } else if (flag_slope) {
      "slope"
    } else {
      "none"
    }
    diagnostic_available[j] <- TRUE
  }

  chain_diag <- data.frame(
    variable = var_names,
    method = imp$method[var_names],
    n_finite = n_finite,
    drift_all = drift_all,
    drift_tail = drift_tail,
    drift_all_scaled = drift_all_scaled,
    drift_tail_scaled = drift_tail_scaled,
    tail_n_finite = tail_n_finite,
    tail_window_na_frac = tail_window_na_frac,
    slope = slope,
    slope_scaled = slope_scaled,
    sd_chain = sd_chain,
    sd_obs = sd_obs,
    overall_reason = overall_reason,
    tail_reason = tail_reason,
    flag_reason = flag_reason,
    diagnostic_available = diagnostic_available,
    flag = flag,
    stringsAsFactors = FALSE
  )
```

```
    chain_diag_stats$n_with_chainMean <- nrow(chain_diag)
    chain_diag_stats$n_with_drift_tail <- sum(is.finite(chain_diag$drift_tail))
    if (chain_diag_stats$n_with_chainMean > 0) {
      chain_diag_stats$drift_tail_na_frac <- 1 - (chain_diag_stats$n_with_drift_tail /
                                                    chain_diag_stats$n_with_chainMean)
      chain_diag_stats$tail_window_na_mean <- mean(chain_diag$tail_window_na_frac, na.rm = TRUE)
    }
    write_csv_safely(chain_diag, results_path("mice_chain_diagnostics.csv"), row_names = FALSE)
    if (any(isTRUE(chain_diag$flag), na.rm = TRUE)) {
      frac_flag <- mean(chain_diag$flag, na.rm = TRUE)
      if (is.finite(frac_flag) && frac_flag < 0.05) {
        message("Chain diagnostics: low drift flags (", round(frac_flag * 100, 1),
                "%). MAXIT_MI may be reduced safely (consider 10 or 5).")
      }
    }
  }
}

# quick sanity: these must exist and be numeric in completed data
imp_n <- imp$m
get_imp_stats <- list(count = 0L, seconds = 0)
get_imp <- function(i, imp_obj = imp) {
  t0 <- Sys.time()
  d <- normalize_types(mice::complete(imp_obj, action = i), levels_ref)
  get_imp_stats$count <<- get_imp_stats$count + 1L
  get_imp_stats$seconds <<- get_imp_stats$seconds +
    as.numeric(difftime(Sys.time(), t0, units = "secs"))
  d
}
d1 <- get_imp(1)
stopifnot(all(c("paco2","vbg_co2") %in% names(d1)))
stopifnot(is.numeric(d1$paco2), is.numeric(d1$vbg_co2))

# post-MICE sanity: no remaining NA in covars_gbm
covars_check <- intersect(covars_gbm, names(d1))
na_counts <- vapply(d1[, covars_check, drop = FALSE], function(x) sum(is.na(x)), numeric(1))
```

Monte Carlo error vs SE (diagnostic only)

Term	Estimate	SE	MC error	MC error / SE	2.5%	97.5%
(Intercept)	-3.821	0.189	0.014	0.074	-4.19136254	-3.4497745159
has_abg	2.166	0.053	0.001	0.016	2.06223732	2.2704301035
age_at_encounter	-0.003	0.001	0.000	0.014	-0.00596037	-0.0009504728
curr_bmi	-0.009	0.005	0.000	0.086	-0.01787779	0.0001613787
sexMale	0.236	0.043	0.001	0.014	0.15097989	0.3212478672
encounter_typeInpatient	1.044	0.068	0.000	0.005	0.91072250	1.1766322121

```
na_counts <- na_counts[na_counts > 0]
if (length(na_counts)) {
  message("Post-MICE NA counts (covars_gbm): ",
          paste(names(na_counts), na_counts, collapse = ", "))
  ev_sum <- summarize_logged_events(imp)
  if (nrow(ev_sum)) {
    ev_sub <- ev_sum[ev_sum$variable %in% names(na_counts), , drop = FALSE]
    if (nrow(ev_sub)) {
      print(utils::head(ev_sub, 10))
    } else {
      message("No loggedEvents entries for covars_gbm with NA.")
    }
  } else {
    message("No loggedEvents recorded.")
  }
  stop("Post-MICE check failed: remaining NA in covars_gbm. See loggedEvents summary above.")
}
```

MC error diagnostic model: Diagnostic model: imv_proc ~ has_abg + age_at_encounter + curr_bmi + sex + encounter_type (unweighted).

3.3 10) Refit propensity models within each imputation

MI propensity scores use logistic regression with restricted cubic splines (`rms::rcs`, 4 knots by default) for continuous covariates; the same covariate set used in non-MI models is reused here (`covars_ps`). IPSW truncation rules are unchanged.

Note: the MI computations below run in a single pass per imputation (weights, balance, cat3, spline). Subsequent MI sections reuse those outputs and will stop if they are missing.

3.3.1 FAIL-FAST CHECKS

```
            used   (Mb) gc trigger   (Mb) limit (Mb)   max used   (Mb)
Ncells   6628748  354.1   12180610  650.6         NA   12180610  650.6
Vcells 610437722 4657.3 1053929965 8040.9      16384 1053929965 8040.9
```

3.3.2 10.1 ABG propensity (has_abg)

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("W_abg_list"))
message("ABG MI weights were computed in the single-pass MI loop above.")
```

Table 9: ABG weight diagnostics (MI; median across imputations)

n	min	p99.99.	max	ess
9362	0.384	3.819	3.821	6847.997

3.3.3 10.2 Balance diagnostics across imputations

```
stopifnot(exists("target_balance_table"))

# Use a fixed x-axis max so ABG/VBG balance plots are directly comparable.
```

```r
stopifnot(exists("bal_imp_abg"))

bal_abg_pooled <- bal_imp_abg$bal_long |>
  mutate(label = ifelse(is.na(level), variable, paste0(variable, ":", level))) |>
  group_by(label) |>
  summarise(
    pre_med = median(abs(smd_pre), na.rm = TRUE),
    post_med = median(abs(smd_post), na.rm = TRUE),
    pre_mean = mean(abs(smd_pre), na.rm = TRUE),
    post_mean = mean(abs(smd_post), na.rm = TRUE),
    post_max = max(abs(smd_post), na.rm = TRUE),
    .groups = "drop"
  )

bal_abg_plot <- bal_abg_pooled |>
  mutate(label = factor(label, levels = label[order(post_med, decreasing = TRUE)])) |>
  pivot_longer(c(pre_med, post_med), names_to = "type", values_to = "smd") |>
  mutate(type = recode(type, pre_med = "Pre", post_med = "Post"))

p_abg <- ggplot(bal_abg_plot, aes(x = smd, y = label, shape = type)) +
  geom_vline(xintercept = 0.1, linetype = 2, linewidth = 0.3) +
  geom_point(size = 1.2) +
  labs(title = "MI target balance (ABG): pooled |SMD|", x = "|Target SMD|", y = NULL, shape = "Stage") +
  scale_x_continuous(limits = c(0, BAL_XLIM_MAX),
                     expand = expansion(mult = c(0, 0.02))) +
  theme_minimal(base_size = 10)
save_diag_plot(p_abg, results_path("figs", "diag-mi-balance-pooled-abg.png"), width = 7, height = 6)

knitr::kable(bal_imp_abg$worst_rows_overall, caption = "ABG: worst target SMD rows across imputations (top 10)")
```

Table 10: ABG: worst target SMD rows across imputations (top 10)

variable	level	type	smd_pre	smd_post	group	imp	abs_post
encounter_type	Inpatient	factor	0.3704053	0.1001128	ABG	45	0.1001128
encounter_type	Emergency	factor	-0.3704053	-0.1001128	ABG	45	0.1001128

variable	level	type	smd_pre	smd_post	group	imp	abs_post
encounter_type	Inpatient	factor	0.3704053	0.0979839	ABG	40	0.0979839
encounter_type	Emergency	factor	-0.3704053	-0.0979839	ABG	40	0.0979839
encounter_type	Inpatient	factor	0.3704053	0.0959636	ABG	38	0.0959636
encounter_type	Emergency	factor	-0.3704053	-0.0959636	ABG	38	0.0959636
encounter_type	Emergency	factor	-0.3704053	-0.0952833	ABG	51	0.0952833
encounter_type	Inpatient	factor	0.3704053	0.0952833	ABG	51	0.0952833
encounter_type	Emergency	factor	-0.3704053	-0.0952657	ABG	8	0.0952657
encounter_type	Inpatient	factor	0.3704053	0.0952657	ABG	8	0.0952657

```
abg_imp_summary <- bal_imp_abg$bal_imp_summary |>
  summarise(
    med = median(max_abs_post, na.rm = TRUE),
    p90 = quantile(max_abs_post, 0.9, na.rm = TRUE),
    max = max(max_abs_post, na.rm = TRUE)
  )
knitr::kable(abg_imp_summary, caption = "ABG: max |Target SMD| summary across imputations")
```

Table 11: ABG: max |Target SMD| summary across imputations

med	p90	max
0.0911384	0.0940709	0.1001128

3.3.4 10.3 VBG propensity (has_vbg)

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("W_vbg_list"))
message("VBG MI weights were computed in the single-pass MI loop above.")
```

Table 12: VBG weight diagnostics (MI; median across imputations)

n	min	p99.99.	max	ess
7460	0.308	3.892	3.895	5076.89

3.3.5 10.4 VBG balance

```
stopifnot(exists("bal_imp_vbg"))

bal_vbg_pooled <- bal_imp_vbg$bal_long |>
  mutate(label = ifelse(is.na(level), variable, paste0(variable, ":", level))) |>
  group_by(label) |>
  summarise(
    pre_med = median(abs(smd_pre), na.rm = TRUE),
    post_med = median(abs(smd_post), na.rm = TRUE),
    pre_mean = mean(abs(smd_pre), na.rm = TRUE),
    post_mean = mean(abs(smd_post), na.rm = TRUE),
    post_max = max(abs(smd_post), na.rm = TRUE),
    .groups = "drop"
  )

bal_vbg_plot <- bal_vbg_pooled |>
  mutate(label = factor(label, levels = label[order(post_med, decreasing = TRUE)])) |>
  pivot_longer(c(pre_med, post_med), names_to = "type", values_to = "smd") |>
  mutate(type = recode(type, pre_med = "Pre", post_med = "Post"))

p_vbg <- ggplot(bal_vbg_plot, aes(x = smd, y = label, shape = type)) +
  geom_vline(xintercept = 0.1, linetype = 2, linewidth = 0.3) +
  geom_point(size = 1.2) +
  labs(title = "MI target balance (VBG): pooled |SMD|", x = "|Target SMD|", y = NULL, shape = "Stage") +
  scale_x_continuous(limits = c(0, BAL_XLIM_MAX),
                     expand = expansion(mult = c(0, 0.02))) +
  theme_minimal(base_size = 10)
save_diag_plot(p_vbg, results_path("figs", "diag-mi-balance-pooled-vbg.png"), width = 7, height = 6)
```

```
knitr::kable(bal_imp_vbg$worst_rows_overall, caption = "VBG: worst target SMD rows across imputations (top 10)")
```

Table 13: VBG: worst target SMD rows across imputations (top 10)

variable	level	type	smd_pre	smd_post	group	imp	abs_post
encounter_type	Emergency	factor	-0.0375275	-0.0505432	VBG	26	0.0505432
encounter_type	Inpatient	factor	0.0375275	0.0505432	VBG	26	0.0505432
encounter_type	Emergency	factor	-0.0375275	-0.0493456	VBG	68	0.0493456
encounter_type	Inpatient	factor	0.0375275	0.0493456	VBG	68	0.0493456
encounter_type	Emergency	factor	-0.0375275	-0.0489478	VBG	36	0.0489478
encounter_type	Inpatient	factor	0.0375275	0.0489478	VBG	36	0.0489478
encounter_type	Inpatient	factor	0.0375275	0.0484923	VBG	9	0.0484923
encounter_type	Emergency	factor	-0.0375275	-0.0484923	VBG	9	0.0484923
encounter_type	Inpatient	factor	0.0375275	0.0479686	VBG	53	0.0479686
encounter_type	Emergency	factor	-0.0375275	-0.0479686	VBG	53	0.0479686

```
vbg_imp_summary <- bal_imp_vbg$bal_imp_summary |>
  summarise(
    med = median(max_abs_post, na.rm = TRUE),
    p90 = quantile(max_abs_post, 0.9, na.rm = TRUE),
    max = max(max_abs_post, na.rm = TRUE)
  )
knitr::kable(vbg_imp_summary, caption = "VBG: max |Target SMD| summary across imputations")
```

Table 14: VBG: max |Target SMD| summary across imputations

med	p90	max
0.044039	0.0474142	0.0505432

3.4 11) Weighted outcome models within each imputation + pooling

Within each imputation, fit covariate-adjusted CO2 spline outcome models **only in the measured cohort** (has_abg==1 for PaCO2; has_vbg==1 for VBG CO2), using IPSW weights to address nonrandom testing. Curves are pooled pointwise across imputations (Rubin's

rules on the log-OR scale) and displayed as odds ratios relative to CO2_ref at a reference covariate profile.

3.4.1 11.1 Helper: fit + extract log-OR and SE from svyglm

```
stopifnot(exists("fit_spline_imp"), exists("pool_spline_curve"),
          exists("pool_spline_coefs"), exists("pool_terms"))
```

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("abg_spline"), exists("abg_curves"), exists("abg_coefs"))
message("ABG MI spline results were computed in the single-pass MI loop above.")
```

3.4.2 11.3 VBG: MI pooled spline models (treated cohort only)

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("vbg_spline"), exists("vbg_curves"), exists("vbg_coefs"))
message("VBG MI spline results were computed in the single-pass MI loop above.")
```

3.5 12) Explainability on one representative imputation

MI propensity scores are estimated via logistic regression with splines; SHAP summaries are not computed for the MI PS model. A placeholder diagnostic file is written for audit completeness.

3.6 13) MI three-level PCO2 helpers and checks

```
stopifnot(exists("pool_terms"))
message("MI 3-level helpers defined in the single-pass MI section.")
```

3.7 14) MI + IPW three-level PCO2 (ABG & VBG)

3.7.1 14.1 ABG: MI + IPW, three-level PCO2 outcomes

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("abg_cat_results"))

mi_abg_cat_forms <- list(
  "MI IPW ABG 3-level: IMV ~ CO2 category + X"      = reformulate(c("co2_cat", adj_core), response = "imv_proc"),
  "MI IPW ABG 3-level: NIV ~ CO2 category + X"      = reformulate(c("co2_cat", adj_core), response = "niv_proc"),
  "MI IPW ABG 3-level: Death60d ~ CO2 category + X" = reformulate(c("co2_cat", adj_core), response = "death_60d"),
  "MI IPW ABG 3-level: HCRF ~ CO2 category + X"     = reformulate(c("co2_cat", adj_core), response = "hypercap_resp_failure")
)
register_model_diagrams(mi_abg_cat_forms)
message("ABG MI 3-level results were computed in the single-pass MI loop above.")
```

3.7.2 14.2 VBG: MI + IPW, three-level PCO2 outcomes

```
stopifnot(exists("mi_single_pass_done"), isTRUE(mi_single_pass_done))
stopifnot(exists("vbg_cat_results"))

mi_vbg_cat_forms <- list(
  "MI IPW VBG 3-level: IMV ~ CO2 category + X"      = reformulate(c("co2_cat", adj_core), response = "imv_proc"),
  "MI IPW VBG 3-level: NIV ~ CO2 category + X"      = reformulate(c("co2_cat", adj_core), response = "niv_proc"),
  "MI IPW VBG 3-level: Death60d ~ CO2 category + X" = reformulate(c("co2_cat", adj_core), response = "death_60d"),
  "MI IPW VBG 3-level: HCRF ~ CO2 category + X"     = reformulate(c("co2_cat", adj_core), response = "hypercap_resp_failure")
)
register_model_diagrams(mi_vbg_cat_forms)
message("VBG MI 3-level results were computed in the single-pass MI loop above.")
```

Table 3. MI-pooled IPW associations between CO category and outcomes (adjusted)

Cohort	Outcome	Low vs normal OR (95% CI)	High vs normal OR (95% CI)
ABG	IMV	1.27 (1.11, 1.45)	1.26 (1.10, 1.44)
ABG	NIV	1.14 (0.91, 1.44)	3.06 (2.52, 3.70)
ABG	Death (60d)	1.63 (1.40, 1.90)	1.30 (1.12, 1.52)
ABG	Hypercapnic RF	1.03 (0.79, 1.35)	7.66 (6.29, 9.32)
VBG	IMV	1.50 (1.23, 1.82)	1.54 (1.27, 1.87)
VBG	NIV	0.91 (0.68, 1.24)	2.83 (2.21, 3.63)
VBG	Death (60d)	1.75 (1.42, 2.15)	1.42 (1.16, 1.74)
VBG	Hypercapnic RF	0.68 (0.46, 1.00)	6.53 (5.08, 8.38)

Weighted survey GLMs adjusted for baseline covariates; weights = MI-specific GLM (RCS) IPW; m = 80 imputations (seed 20251206); reference = Normal.

3.7.3 14.3 Table 3: MI-pooled IPW associations (3-level CO)

3.7.4 14.4 Summary: adjusted CO2-category associations across analysis tracks

Table 15: Adjusted odds ratios (low/high vs normal) across analysis tracks; n/events reflect model sample size (median across imputations for MI).

method	group	outcome_label	n_model	events	Low vs normal	High vs normal
Unweighted adjusted	ABG	IMV	9362	2391	1.28 (1.14, 1.43)	1.27 (1.14, 1.43)
Unweighted adjusted	ABG	NIV	9362	896	1.08 (0.89, 1.31)	2.90 (2.46, 3.41)
Unweighted adjusted	ABG	Death (60d)	9362	1645	1.72 (1.51, 1.97)	1.38 (1.20, 1.57)
Unweighted adjusted	ABG	Hypercapnic RF	9362	992	0.96 (0.76, 1.21)	6.78 (5.75, 8.03)
Unweighted adjusted	VBG	IMV	7460	1162	1.30 (1.11, 1.53)	1.54 (1.32, 1.80)
Unweighted adjusted	VBG	NIV	7460	554	0.98 (0.76, 1.25)	2.74 (2.22, 3.37)
Unweighted adjusted	VBG	Death (60d)	7460	1044	1.71 (1.45, 2.01)	1.35 (1.14, 1.59)
Unweighted adjusted	VBG	Hypercapnic RF	7460	692	0.66 (0.49, 0.89)	7.21 (5.92, 8.82)
IPW adjusted	ABG	IMV	9362	2391	1.26 (1.11, 1.43)	1.20 (1.05, 1.36)
IPW adjusted	ABG	NIV	9362	896	1.15 (0.93, 1.44)	3.01 (2.50, 3.61)
IPW adjusted	ABG	Death (60d)	9362	1645	1.63 (1.41, 1.89)	1.31 (1.13, 1.52)

method	group	outcome_label	n_model	events	Low vs normal	High vs normal
IPW adjusted	ABG	Hypercapnic RF	9362	992	1.08 (0.83, 1.40)	7.79 (6.46, 9.39)
IPW adjusted	VBG	IMV	7460	1162	1.41 (1.18, 1.69)	1.51 (1.26, 1.80)
IPW adjusted	VBG	NIV	7460	554	0.96 (0.73, 1.27)	2.95 (2.35, 3.71)
IPW adjusted	VBG	Death (60d)	7460	1044	1.76 (1.46, 2.11)	1.52 (1.26, 1.83)
IPW adjusted	VBG	Hypercapnic RF	7460	692	0.72 (0.52, 1.00)	7.99 (6.39, 9.98)
IPW + MI adjusted	ABG	IMV	9362	2391	1.27 (1.11, 1.45)	1.26 (1.10, 1.44)
IPW + MI adjusted	ABG	NIV	9362	896	1.14 (0.91, 1.44)	3.06 (2.52, 3.70)
IPW + MI adjusted	ABG	Death (60d)	9362	1645	1.63 (1.40, 1.90)	1.30 (1.12, 1.52)
IPW + MI adjusted	ABG	Hypercapnic RF	9362	992	1.03 (0.79, 1.35)	7.66 (6.29, 9.32)
IPW + MI adjusted	VBG	IMV	7460	1162	1.50 (1.23, 1.82)	1.54 (1.27, 1.87)
IPW + MI adjusted	VBG	NIV	7460	554	0.91 (0.68, 1.24)	2.83 (2.21, 3.63)
IPW + MI adjusted	VBG	Death (60d)	7460	1044	1.75 (1.42, 2.15)	1.42 (1.16, 1.74)
IPW + MI adjusted	VBG	Hypercapnic RF	7460	692	0.68 (0.46, 1.00)	6.53 (5.08, 8.38)

3.8 Manuscript outputs summary

```r
# Cohort flow / sample sizes
flow_tbl <- tibble::tibble(
  metric = c(
    "Full cohort (raw)",
    "Analytic subset",
    "ABG tested",
    "ABG with PaCO2",
    "VBG tested",
    "VBG with VBG CO2"
  ),
  n = c(
    nrow(stata_data),
    nrow(subset_data),
    sum(subset_data$has_abg == 1, na.rm = TRUE),
    sum(subset_data$has_abg == 1 & !is.na(subset_data$paco2), na.rm = TRUE),
    sum(subset_data$has_vbg == 1, na.rm = TRUE),
    sum(subset_data$has_vbg == 1 & !is.na(subset_data$vbg_co2), na.rm = TRUE)
```

```
  )
)
render_table_pdf(flow_tbl, "Cohort flow summary", "cohort_flow_summary",
                 preview_rows = 10, digits = 0)
```

Table 16: Cohort flow summary

metric	n
Full cohort (raw)	833476
Analytic subset	25852
ABG tested	9362
ABG with PaCO2	9362
VBG tested	7460
VBG with VBG CO2	7460

```
# Event counts by cohort
outcome_vars <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")
outcome_labels <- c(
  imv_proc = "IMV",
  niv_proc = "NIV",
  death_60d = "Death (60d)",
  hypercap_resp_failure = "Hypercapnic RF"
)
event_tbl <- dplyr::bind_rows(
  lapply(outcome_vars, function(o) {
    dplyr::tibble(
      outcome = outcome_labels[[o]],
      group = "ABG",
      n = sum(subset_data$has_abg == 1 & !is.na(subset_data[[o]]), na.rm = TRUE),
      events = sum(subset_data$has_abg == 1 & subset_data[[o]] == 1, na.rm = TRUE)
    )
  }),
  lapply(outcome_vars, function(o) {
    dplyr::tibble(
      outcome = outcome_labels[[o]],
      group = "VBG",
      n = sum(subset_data$has_vbg == 1 & !is.na(subset_data[[o]]), na.rm = TRUE),
```

```
      events = sum(subset_data$has_vbg == 1 & subset_data[[o]] == 1, na.rm = TRUE)
    )
  })
)
event_tbl <- event_tbl |>
  dplyr::mutate(pct = ifelse(n > 0, 100 * events / n, NA_real_))
render_table_pdf(event_tbl,
                 "Outcome counts by cohort (ABG/VBG tested)",
                 "outcome_counts_by_cohort",
                 preview_rows = 20,
                 digits = 1)
```

Table 17: Outcome counts by cohort (ABG/VBG tested)

outcome	group	n	events	pct
IMV	ABG	9362	2391	25.5
NIV	ABG	9362	896	9.6
Death (60d)	ABG	9362	1645	17.6
Hypercapnic RF	ABG	9362	992	10.6
IMV	VBG	7460	1162	15.6
NIV	VBG	7460	554	7.4
Death (60d)	VBG	7460	1044	14.0
Hypercapnic RF	VBG	7460	692	9.3

```
# Weighting diagnostics (non-MI weights)
stopifnot(all(c("w_abg", "w_vbg", "ps_abg", "ps_vbg") %in% names(subset_data)))
wt_abg <- subset_data$w_abg[subset_data$has_abg == 1]
wt_vbg <- subset_data$w_vbg[subset_data$has_vbg == 1]
ps_abg <- subset_data$ps_abg[subset_data$has_abg == 1]
ps_vbg <- subset_data$ps_vbg[subset_data$has_vbg == 1]
trunc_abg <- subset_data$trunc_abg[subset_data$has_abg == 1]
trunc_vbg <- subset_data$trunc_vbg[subset_data$has_vbg == 1]

wt_sum <- dplyr::bind_rows(
  weight_summary(wt_abg, ps = ps_abg, ps_floor = ps_floor_abg,
                 truncated = trunc_abg) |>
    dplyr::mutate(group = "ABG"),
```

```
  weight_summary(wt_vbg, ps = ps_vbg, ps_floor = ps_floor_vbg,
                 truncated = trunc_vbg) |>
    dplyr::mutate(group = "VBG")
)
wt_sum_display <- wt_sum |>
  dplyr::select(group, n, ess, min, p99, max, trunc_rate, ps_p01) |>
  dplyr::rename(
    trunc = trunc_rate,
    p01_ps = ps_p01
  )
render_table_pdf(wt_sum_display,
                 "Weighting diagnostics summary (non-MI)",
                 "weighting_diagnostics_non_mi",
                 preview_rows = 10,
                 preview_cols = c("group", "n", "ess", "min", "p99", "max", "trunc", "p01_ps"),
                 wide = TRUE,
                 digits = 3)
```

Preview (top 10 rows). Full table saved to Results/weighting_diagnostics_non_mi.csv.

Table 18: Weighting diagnostics summary (non-MI)

	group	n	ess	min	p99	max	trunc	p01_ps
1%...1	ABG	9362	7199.526	0.468	3.330	3.334	0.01	0.129
1%...2	VBG	7460	5982.854	0.410	2.784	2.784	0.01	0.128

```
# Missingness + MI spec summary
miss_vars <- c(covars_gbm, "paco2", "vbg_co2",
               "imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")
miss_vars <- intersect(miss_vars, names(subset_data_raw))
miss_tbl <- subset_data_raw |>
  dplyr::summarise(dplyr::across(dplyr::all_of(miss_vars), ~ mean(is.na(.)) * 100)) |>
  tidyr::pivot_longer(dplyr::everything(), names_to = "variable", values_to = "pct_missing") |>
  dplyr::arrange(dplyr::desc(pct_missing)) |>
  dplyr::slice_head(n = 10)
render_table_pdf(miss_tbl,
```

```
                "Top 10 variables by missingness (pre-imputation)",
                "missingness_top10",
                preview_rows = 10,
                digits = 1)
```

Table 19: Top 10 variables by missingness (pre-imputation)

variable	pct_missing
vbg_co2	71.1
paco2	63.8
serum_lac	60.2
curr_bmi	57.0
serum_phos	53.2
temp_new	48.2
hr	36.4
dbp	29.9
sbp	29.7
wbc	17.6

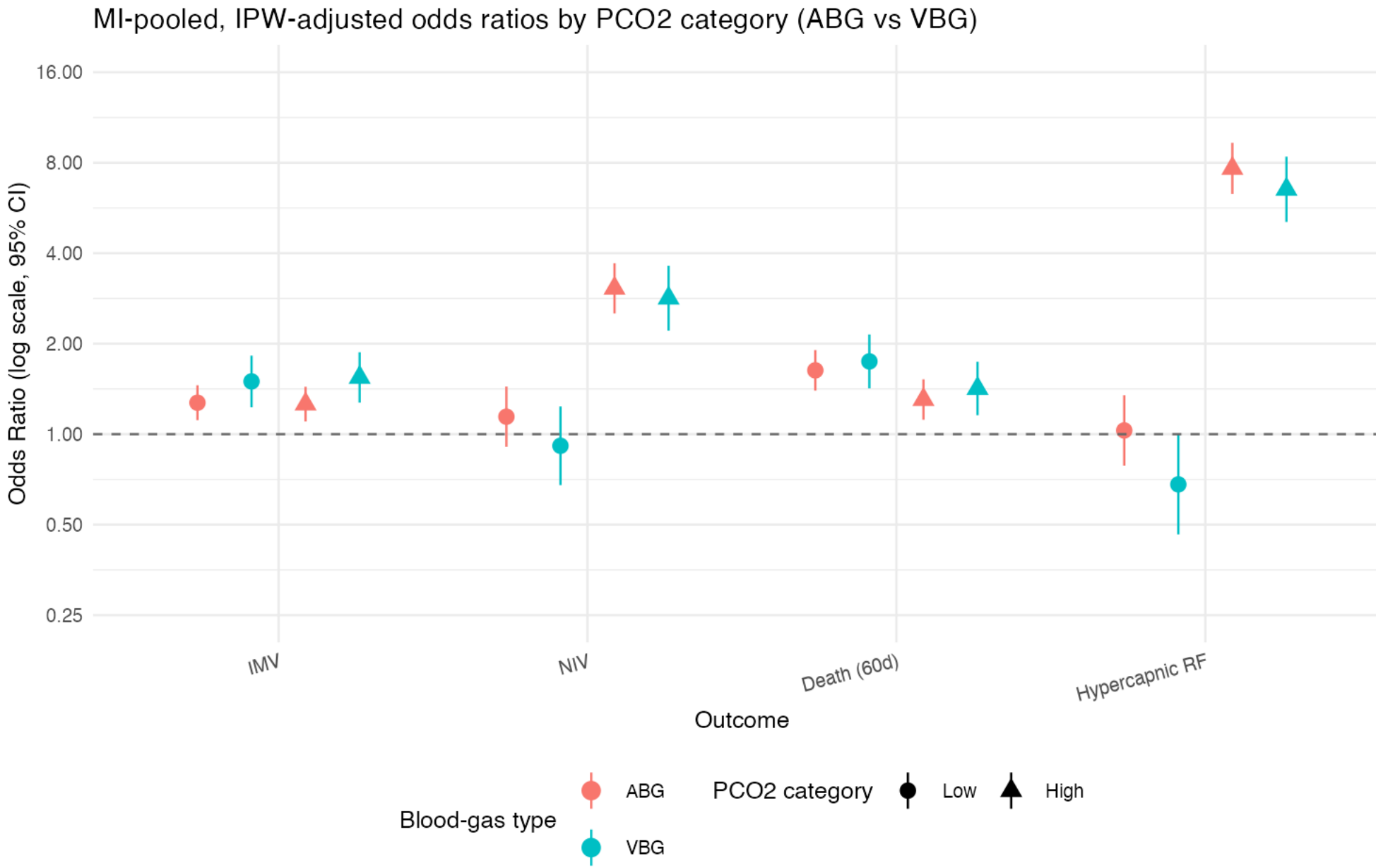
```
mi_spec_tbl <- tibble::tibble(
  m = imp$m,
  maxit = MAXIT_MI,
  methods = paste(unique(imp$method[imp$method != ""]), collapse = ", "),
  ps_model = MI_PS_METHOD,
  ps_spline_k = MI_PS_SPLINE_K,
  ps_glm_maxit = MI_GLM_MAXIT
)
render_table_pdf(mi_spec_tbl,
                 "MI specification (methods used)",
                 "mi_specification",
                 preview_rows = 10)
```

Table 20: MI specification (methods used)

m	maxit	methods	ps_model	ps_spline_k	ps_glm_maxit
80	20	pmm	glm_rcs4	4	25

3.8.1 14.3 Visualization: pooled three-level ORs

```
stopifnot(exists("abg_cat_results"), exists("vbg_cat_results"))

mi_combined_or_df <- dplyr::bind_rows(
  dplyr::mutate(abg_cat_results, group = "ABG"),
  dplyr::mutate(vbg_cat_results, group = "VBG")
) |>
  dplyr::mutate(
    outcome = factor(outcome,
                     levels = c("IMV", "NIV", "Death60d", "HCRF"),
                     labels = c("IMV", "NIV", "Death (60d)", "Hypercapnic RF")),
    group = factor(group, levels = c("ABG", "VBG"))
  )

mi_combined_or_df <- map_or_exposure(mi_combined_or_df, "or-plot-mi-weighted")
mi_plot_df <- mi_combined_or_df |>
  dplyr::mutate(estimate = OR, conf.low = LCL, conf.high = UCL) |>
  dplyr::select(-OR, -LCL, -UCL)

mi_plot_df <- build_or_plot_df(
  mi_plot_df,
  "or-plot-mi-weighted",
  expected_exposure_levels = CO2_CAT_CONTRAST_LEVELS
)
mi_axis_spec <- compute_or_axis_spec(list(mi_plot_df), lo_col = "conf.low", hi_col = "conf.high")

mi_p_or <- plot_or_safe(
  mi_plot_df,
  plot_name = "or-plot-mi-weighted",
  axis_spec = mi_axis_spec,
  title = "MI-pooled, IPW-adjusted odds ratios by PCO2 category (ABG vs VBG)"
)
print_plot_once(mi_p_or, "or-plot-mi-weighted", width = 7.5, height = 4.8)
```

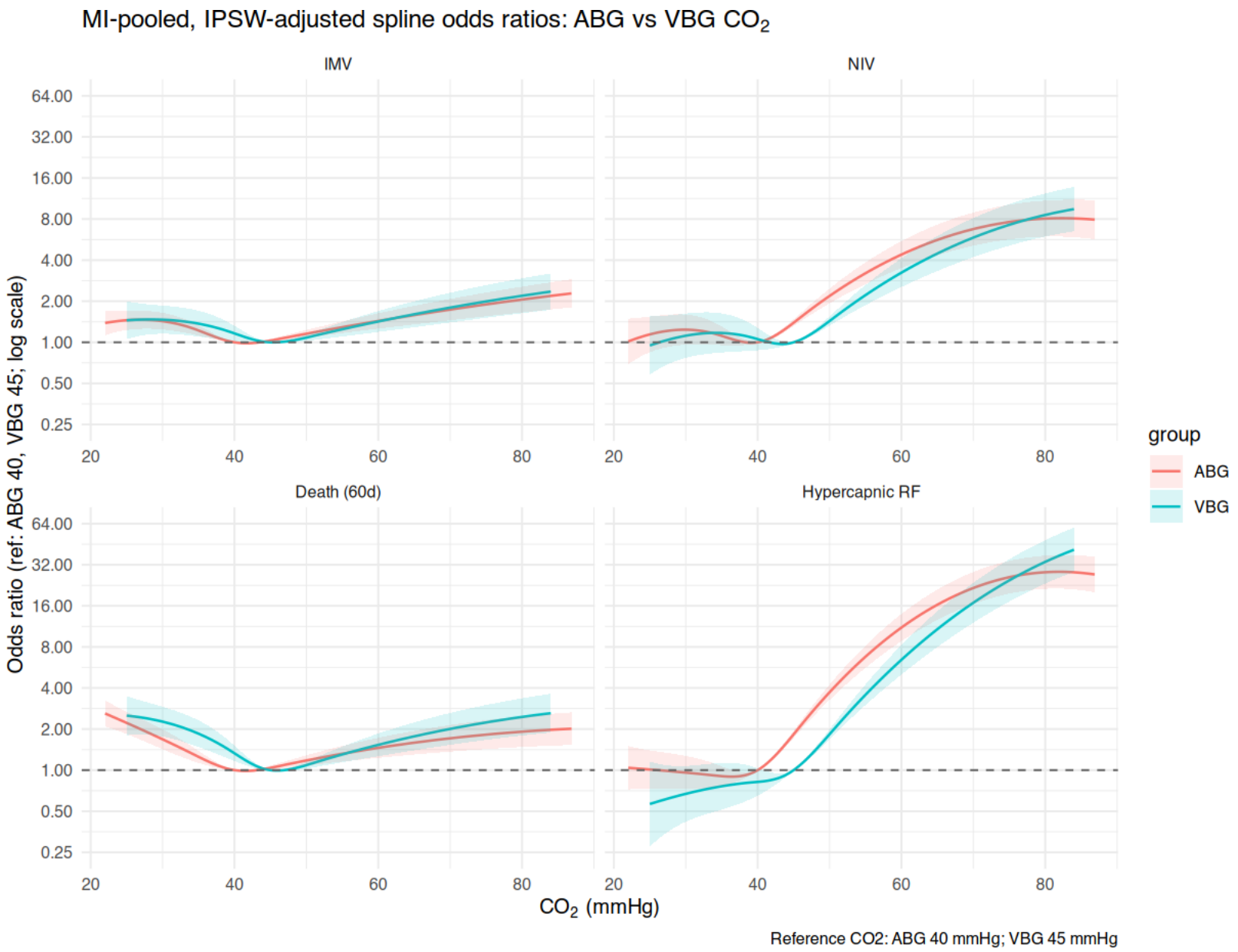

MI-pooled, IPW-adjusted odds ratios by PCO2 category (ABG vs VBG)
Odds Ratio (log scale, 95% CI)
16.00
8.00
4.00
2.00
1.00
0.50
0.25
IMV
NIV
Death (60d)
Hypercapnic RF
Outcome
Blood-gas type
ABG
VBG
PCO2 category
Low
High

```
float_barrier()
```

3.8.2 15.3 Visualization

```
library(dplyr)
library(ggplot2)
library(patchwork)
library(purrr)

mi_ipw_rcs_forms <- list(
  "MI IPW spline (adjusted) ABG: IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "paco2", adj_core),
  "MI IPW spline (adjusted) ABG: NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "paco2", adj_core),
  "MI IPW spline (adjusted) ABG: Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "paco2", adj_core),
  "MI IPW spline (adjusted) ABG: HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "paco2", adj_core),
  "MI IPW spline (adjusted) VBG: IMV ~ CO2 spline + X"      = make_spline_fml("imv_proc", "vbg_co2", adj_core),
  "MI IPW spline (adjusted) VBG: NIV ~ CO2 spline + X"      = make_spline_fml("niv_proc", "vbg_co2", adj_core),
  "MI IPW spline (adjusted) VBG: Death60d ~ CO2 spline + X" = make_spline_fml("death_60d", "vbg_co2", adj_core),
  "MI IPW spline (adjusted) VBG: HCRF ~ CO2 spline + X"     = make_spline_fml("hypercap_resp_failure", "vbg_co2", adj_core)
)
register_model_diagrams(mi_ipw_rcs_forms)

stopifnot(exists("abg_curves"), exists("vbg_curves"))

curve_abg <- abg_curves |>
  mutate(co2 = paco2) |>
  select(-paco2)
curve_vbg <- vbg_curves |>
  mutate(co2 = vbg_co2) |>
  select(-vbg_co2)
curve_all <- bind_rows(curve_abg, curve_vbg) |>
  mutate(outcome = factor(outcome,
                          levels = c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure"),
                          labels = c("IMV", "NIV", "Death (60d)", "Hypercapnic RF")))
axis_mi_ipw <- compute_or_axis_spec(list(curve_abg, curve_vbg), lo_col = "LCL", hi_col = "UCL")
```

```r
ggplot(curve_all, aes(x = co2, y = OR, color = group, fill = group)) +
  geom_line(linewidth = 0.6) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), alpha = 0.15, color = NA) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "grey40") +
  or_axis_scale(axis_mi_ipw) +
  facet_wrap(~ outcome, scales = "free_x") +
  labs(
    title = expression(
      paste("MI-pooled, IPSW-adjusted spline odds ratios: ABG vs VBG CO"[2])
    ),
    x = expression(CO[2]~"(mmHg)"),
    y = paste0("Odds ratio (ref: ABG ", ABG_CO2_REF, ", VBG ", VBG_CO2_REF, "; log scale)"),
    caption = paste0("Reference CO2: ABG ", ABG_CO2_REF, " mmHg; VBG ", VBG_CO2_REF, " mmHg")
  ) +
  theme_minimal(base_size = 10)
```


MI-pooled, IPSW-adjusted spline odds ratios: ABG vs VBG CO_2
IMV
NIV
Death (60d)
Hypercapnic RF
Odds ratio (ref: ABG 40, VBG 45; log scale)
64.00
32.00
16.00
8.00
4.00
2.00
1.00
0.50
0.25
20
40
60
80
CO_2 (mmHg)
group
ABG
VBG
Reference CO2: ABG 40 mmHg; VBG 45 mmHg

```
float_barrier()
```

3.9 Diagnostics

3.9.1 MI convergence and mixing

3.9.2 MI stability across m

3.9.3 MI maxit sensitivity (sampled)

3.9.4 Balance diagnostics

Table 21: Target balance (top 10 by max |SMD|)

group	variable	max_abs_pre	max_abs_post
ABG	curr_bmi	0.2691867	0.1938276
ABG	encounter_type	0.3704053	0.1924810
ABG	location	0.1663410	0.1140054
ABG	sbp	0.1792273	0.0872274
ABG	serum_ca	0.2235344	0.0851847
ABG	age_at_encounter	0.1055368	0.0815615
ABG	copd	0.0561648	0.0782714
ABG	chf	0.0722894	0.0742121
ABG	dbp	0.1615796	0.0728835
ABG	sex	0.0904038	0.0668771
VBG	curr_bmi	0.2734533	0.1797502
VBG	location	0.3988553	0.1506953
VBG	serum_cl	0.1455225	0.0789445
VBG	race_ethnicity	0.2049873	0.0784599
VBG	hr	0.1307602	0.0744904
VBG	dm	0.0822989	0.0716654
VBG	dbp	0.0927166	0.0671185
VBG	sbp	0.1213141	0.0556533
VBG	ckd	0.0689314	0.0537622

group	variable	max_abs_pre	max_abs_post
VBG	serum_lac	0.0969344	0.0483619

Table 22: Worst target SMD rows across imputations (top 10)

variable	level	type	smd_pre	smd_post	group	imp	abs_post
encounter_type	Inpatient	factor	0.3704053	0.1001128	ABG	45	0.1001128
encounter_type	Emergency	factor	-0.3704053	-0.1001128	ABG	45	0.1001128
encounter_type	Inpatient	factor	0.3704053	0.0979839	ABG	40	0.0979839
encounter_type	Emergency	factor	-0.3704053	-0.0979839	ABG	40	0.0979839
encounter_type	Inpatient	factor	0.3704053	0.0959636	ABG	38	0.0959636
encounter_type	Emergency	factor	-0.3704053	-0.0959636	ABG	38	0.0959636
encounter_type	Emergency	factor	-0.3704053	-0.0952833	ABG	51	0.0952833
encounter_type	Inpatient	factor	0.3704053	0.0952833	ABG	51	0.0952833
encounter_type	Emergency	factor	-0.3704053	-0.0952657	ABG	8	0.0952657
encounter_type	Inpatient	factor	0.3704053	0.0952657	ABG	8	0.0952657
encounter_type	Emergency	factor	-0.0375275	-0.0505432	VBG	26	0.0505432
encounter_type	Inpatient	factor	0.0375275	0.0505432	VBG	26	0.0505432
encounter_type	Emergency	factor	-0.0375275	-0.0493456	VBG	68	0.0493456
encounter_type	Inpatient	factor	0.0375275	0.0493456	VBG	68	0.0493456
encounter_type	Emergency	factor	-0.0375275	-0.0489478	VBG	36	0.0489478
encounter_type	Inpatient	factor	0.0375275	0.0489478	VBG	36	0.0489478
encounter_type	Inpatient	factor	0.0375275	0.0484923	VBG	9	0.0484923
encounter_type	Emergency	factor	-0.0375275	-0.0484923	VBG	9	0.0484923
encounter_type	Inpatient	factor	0.0375275	0.0479686	VBG	53	0.0479686
encounter_type	Emergency	factor	-0.0375275	-0.0479686	VBG	53	0.0479686

Table 23: Distribution of max |Target SMD| across imputations

group	med	iqr	max
ABG	0.0911384	0.0034712	0.1001128
VBG	0.0440390	0.0037890	0.0505432

Table 24: Most frequent worst-balance terms (top 10 per group)

group	term	n
ABG	age_at_encounter	80
ABG	curr_bmi	80
ABG	encounter_type:Emergency	80
ABG	encounter_type:Inpatient	80
ABG	location:0	80
ABG	location:1	80
ABG	chf:1	76
ABG	chf:0	74
ABG	ckd:0	68
ABG	ckd:1	61
VBG	age_at_encounter	80
VBG	encounter_type:Emergency	80
VBG	encounter_type:Inpatient	80
VBG	location:3	80
VBG	serum_ca	80
VBG	serum_phos	51
VBG	race_ethnicity:6	46
VBG	temp_new	45
VBG	serum_cl	43
VBG	location:0	40

3.9.5 Outcome diagnostics

3.9.6 Diagnostics summary and audit

```
Preview (top 10 rows). Full table saved to Results/diagnostics_summary_display.csv.
```

3.9.7 Performance / runtime log

Table 25: Diagnostics summary (IPSW + MI)

block	m	ess	trunc	med_max_smd	max_max_smd
ABG weights	80	7199.526	0.01	0.091	0.100
VBG weights	80	5982.854	0.01	0.044	0.051
ABG outcomes	80	7199.526	0.01	0.091	0.100
VBG outcomes	80	5982.854	0.01	0.044	0.051

Table 26: Worst fitted-probability extremes (top 10)

stage	component	analysis_variant	model_type	group	outcome	imputation	n_used	events	converged	iter	sep_flag	nonconv_flag	min_phat	max_phat	warn_n	top_warning	error_message	extreme
outcome	spline	ipw	spline	ABG	hypercap_resp_failure	NA	9362	992	TRUE	7	TRUE	FALSE	0	0.7398233	0	NA	NA	0
outcome	spline	ipw	spline	ABG	hypercap_resp_failure	NA	9362	992	TRUE	7	TRUE	FALSE	0	0.7398233	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	1	9362	992	TRUE	7	TRUE	FALSE	0	0.7837141	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	2	9362	992	TRUE	7	TRUE	FALSE	0	0.7842440	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	3	9362	992	TRUE	7	TRUE	FALSE	0	0.7898782	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	4	9362	992	TRUE	7	TRUE	FALSE	0	0.7820752	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	5	9362	992	TRUE	7	TRUE	FALSE	0	0.7788758	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	6	9362	992	TRUE	7	TRUE	FALSE	0	0.7855521	0	NA	NA	0

stage	component	analysis_variant	model_type	group	outcome	imputation	n_used	events	converged	iter	sep_flag	nonconv_flag	min_phat	max_phat	warn_n	top_warning	error_message	extreme
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	7	9362	992	TRUE	7	TRUE	FALSE	0	0.7769311	0	NA	NA	0
outcome	spline	mi_ipw	spline	ABG	hypercap_resp_failure	8	9362	992	TRUE	7	TRUE	FALSE	0	0.7807939	0	NA	NA	0

3.9.8 Performance / runtime log

Table 27: Top runtime steps (seconds)

step_name	seconds	start_time	end_time	notes	run_id	run_mode	n_subset
mi_single_pass	195.22984	2026-02-04 07:39:15.373768	2026-02-04 07:42:30.603606	m=80	20260204_072236	pilot	25852
mice_batch_36	18.20452	2026-02-04 07:36:52.950617	2026-02-04 07:37:11.155134	batch=36; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_37	18.12006	2026-02-04 07:37:12.528671	2026-02-04 07:37:30.648726	batch=37; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_35	18.06604	2026-02-04 07:36:33.511232	2026-02-04 07:36:51.577272	batch=35; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_34	18.02548	2026-02-04 07:36:14.115993	2026-02-04 07:36:32.141469	batch=34; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_38	18.01687	2026-02-04 07:37:31.977403	2026-02-04 07:37:49.994278	batch=38; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_1	17.99264	2026-02-04 07:25:50.91151	2026-02-04 07:26:08.90415	batch=1; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_40	17.78155	2026-02-04 07:38:10.337816	2026-02-04 07:38:28.119368	batch=40; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_6	17.76175	2026-02-04 07:27:25.250658	2026-02-04 07:27:43.012407	batch=6; m=2; maxit=20	20260204_072236	pilot	25852

step_name	seconds	start_time	end_time	notes	run_id	run_mode	n_subset
mice_batch_14	17.74737	2026-02-04 07:29:55.877432	2026-02-04 07:30:13.624806	batch=14; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_18	17.71192	2026-02-04 07:31:11.903159	2026-02-04 07:31:29.615077	batch=18; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_39	17.67599	2026-02-04 07:37:51.287994	2026-02-04 07:38:08.963982	batch=39; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_10	17.65232	2026-02-04 07:28:40.65581	2026-02-04 07:28:58.308133	batch=10; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_31	17.65083	2026-02-04 07:35:17.437083	2026-02-04 07:35:35.087911	batch=31; m=2; maxit=20	20260204_072236	pilot	25852
mice_batch_15	17.64427	2026-02-04 07:30:15.026542	2026-02-04 07:30:32.67081	batch=15; m=2; maxit=20	20260204_072236	pilot	25852

Table 28: Runtime summary (total + top 5 steps)

step_name	seconds
TOTAL	900.53355
mi_single_pass	195.22984
mice_batch_36	18.20452
mice_batch_37	18.12006
mice_batch_35	18.06604
mice_batch_34	18.02548

3.10 16) Save, export, and session info

```
audit_issues <- results_path("diagnostics_audit_issues.csv")
if (file.exists(audit_issues)) {
  issues_df <- read.csv(audit_issues)
  issues_df <- issues_df |>
    dplyr::arrange(factor(severity, levels = c("blocker", "high", "medium", "low"))) |>
```

```
    dplyr::slice_head(n = 10)
  knitr::kable(issues_df, caption = "Diagnostics audit: top issues (see Results/diagnostics_audit.md for full details)")
} else {
  cat("Diagnostics audit summary not available.\n")
}
```

Table 29: Diagnostics audit: top issues (see Results/diagnostics_audit.md for full details)

severity	component	evidence_file	evidence_snippet	why_it_matters	recommended_fix
high	Balance	Results/balance_target_imp_summary.csv	ABG max|SMD|=0.100	ABG target balance exceeds 0.10 threshold across imputations.	Revisit GBM tuning, covariate set, or truncation to improve ABG balance.
high	Outcome	Results/model_fit_diagnostics.csv	sep_flag TRUE for 905 / 1324 fits	High rate of separation/near-separation can bias ORs and CIs.	Inspect flagged outcomes; consider penalized fits or check data sparsity.

```
stopifnot(exists("abg_curves"), exists("vbg_curves"), exists("abg_coefs"), exists("vbg_coefs"))
saveRDS(
  list(
    abg_curves = abg_curves,
    vbg_curves = vbg_curves,
    abg_coefs  = abg_coefs,
    vbg_coefs  = vbg_coefs
  ),
  mi_pooled_file
)
```

Software: R 4.5.2 ; key packages: mice, WeightIt, cobalt, survey, rms.