

# ABG-VBG Analysis

Brian Locke, Anila Mehta

## Table of contents

<b>1</b>	<b>TODO:</b>	<b>3</b>
<b>2</b>	<b>Data Pre-processing</b>	<b>3</b>
2.1	1) Seed escrow (reproducibility anchors) . . . . .	4
2.2	2) Baseline tables . . . . .	9
2.2.1	2.1 Table 1A and 1B: . . . . .	9
2.2.2	2.2 Table 1 (Overall ABG/VBG status) . . . . .	15
2.2.3	2.3 Table 2 (Hypercapnia within cohorts) . . . . .	16
2.2.4	2.4 Generating Word Docs for New Table 1 and 2 . . . . .	19
<b>3</b>	<b>Unweighted Binary Logistic Regressions</b>	<b>19</b>
3.0.1	3.1 ABG: Binary hypercapnia models . . . . .	21
3.0.2	3.2 VBG: Binary hypercapnia models . . . . .	26
3.0.3	3.3 Display model coefficients for binary hypercapnia on VBG logistic regression . . . . .	29
3.1	3) Three-level PCO <sub>2</sub> categories (unweighted) . . . . .	30
3.2	4) Restricted cubic spline regressions (unweighted) . . . . .	35
3.2.1	4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO <sub>2</sub> . . . . .	35
3.2.2	4.2 Unweighted, Restricted Cubic Spline - VBG . . . . .	38
<b>4</b>	<b>Inverse Propensity Weighting</b>	<b>41</b>
4.0.1	5.1 ABG IPW weighting and diagnostics . . . . .	41
4.0.2	5.2 ABG IPW spline models . . . . .	46
4.0.3	5.3 ABG IPW spline models (2–98th percentile) . . . . .	50
4.0.4	5.4 VBG IPW weighting and spline models . . . . .	54

4.1	5) Weighted effect estimates . . . . .	60
4.1.1	5.5 Three-level PCO <sub>2</sub> categories (weighted; ABG, VBG) . . . . .	64
4.1.2	5.6 Three-level PCO <sub>2</sub> categories (weighted; ABG vs VBG only) . . . . .	68
4.2	6) Propensity score diagnostics . . . . .	72
<b>5</b>	<b>Multiple Imputation Analysis</b>	<b>77</b>
5.1	7) Packages and reproducibility . . . . .	77
5.1.1	7.1 Missingness audit (what, where, how much) . . . . .	79
5.2	8) Pre-imputation data prep (consistent types & predictors) . . . . .	84
5.3	9) Imputation model specification (MICE) . . . . .	87
5.3.1	9.1 Predictor matrix & methods. Run MICE (moderate settings for scale) . . . . .	87
5.3.2	9.2 Convergence & plausibility checks . . . . .	91
5.4	10) Refit propensity models within each imputation . . . . .	101
5.4.1	10.1 ABG propensity (has_abg) . . . . .	101
5.4.2	10.2 Balance diagnostics across imputations . . . . .	103
5.4.3	10.3 VBG propensity (has_vbg) . . . . .	108
5.4.4	10.4 VBG balance . . . . .	109
5.5	11) Weighted outcome models within each imputation + pooling . . . . .	113
5.5.1	11.1 Helper: fit + extract log-OR and SE from svyglm . . . . .	113
5.5.2	11.2 ABG: outcomes = IMV, NIV, Death(60d), Hypercapnic RF . . . . .	114
5.5.3	11.3 Repeat for VBG . . . . .	116
5.6	12) Explainability on one representative imputation . . . . .	118
5.7	13) Imputed, weighted, three-level PCO <sub>2</sub> (ABG & VBG) . . . . .	131
5.8	14) MI + IPW three-level PCO <sub>2</sub> (ABG & VBG) . . . . .	134
5.8.1	14.1 ABG: MI + IPW, three-level PCO <sub>2</sub> outcomes . . . . .	134
5.8.2	14.2 VBG: MI + IPW, three-level PCO <sub>2</sub> outcomes . . . . .	135
5.8.3	14.3 Visualization: pooled three-level ORs . . . . .	137
5.9	15) Imputed, weighted spline PCO <sub>2</sub> (ABG & VBG) . . . . .	143
5.9.1	15.1 ABG, imputed, weighted, spline outcome . . . . .	143
5.9.2	15.2 VBG, imputed, weighted, spline outcome . . . . .	143
5.9.3	15.3 Visualization . . . . .	144
5.10	16) Save, export, and session info . . . . .	151

reminder: if rendering fails, consider deleting cached items: e.g. rm -rf ABG-VBG-analysis-2025-12-7\_cache/

## 1 TODO:

- High missingness in the 3-level CO categories.** pco2\_cat\_abg missing in 1,630/2,491; pco2\_cat\_vbg in 1,778/2,491; pco2\_cat\_calc in 2,200/2,491. This drives the complete-case drop noted above and will also affect any categorical CO models. Evidence: skim table. - evaluate if this is still a problem? **High apparent missingness in 3-level CO categories (complete-case drop).** The three-category sections trigger “Removed rows...” and show large sample loss. This is expected if many encounters lack PaCO / VBG CO (and you haven’t switched those analyses over to MI or inverse-probability-of-observation). Flag it in the text or impute/weight appropriately.
- ABG vs VBG IPW tuning asymmetry - these should be the same.** Currently, The VBG block shows different tuning prose (“changed trees and bag fraction”), implying divergence from the ABG settings; just document so readers don’t infer drift as a bias. Evidence: the VBG section text notes altered tuning.
- Shapley plots emit warnings.** loess near-singularities and pseudo-inverse messages (often due to very flat x-ranges or duplicated x). “aesthetics dropped: colour” (stat layer not using mapped colour), “label cannot be a <element\_blank> object.”
- Table 2. Baseline Characteristics by Hypercapnia Within ABG and VBG Cohorts** add a small panel with crude outcome rates (%) for each hypercapnia group to give readers intuitive anchoring.
- create ready Table 3. Primary Inverse-Probability–Weighted Associations Between CO Category and Outcomes New compact table summarizing, for ABG and VBG separately:
  - Low vs normal, High vs normal CO Odds ratios and 95% CIs for each of the four outcomes. Use MI-pooled IPSW models as primary analysis; mention in footnote. This table is the quantitative core supporting “VBG is prognostically equivalent.”
- Create Cohort Flow Diagram [New analysis/graphic needed]
  - Show numbers at each step:
  - Starting suspected-hypercapnia cohort.
  - Exclusions (age <18, missing key data, etc.).
  - Final analytic cohort.
  - Split into ABG only, VBG only, both, and neither.
  - This addresses STROBE’s “participants” and flow requirements.
- create a Missing Data / Imputation Summary (Supplement) One table summarizing missingness for key covariates and outcomes; briefly note MI settings and number of imputations. This can be simple counts/percentages drawn from the pre-imputation dataset.

## 2 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.

## 2.1 1) Seed escrow (reproducibility anchors)

Table 1: Seed escrow for MI, GBM, and SHAP runs

component	seed
Multiple imputation (mice)	20251206
ABG propensity GBM (non-MI)	42
VBG propensity GBM (non-MI)	42
SHAP (fastshap/shapviz)	123
MI GBM seeds (ABG per imputation)	20251206 + imputation index
MI GBM seeds (VBG per imputation)	30251206 + imputation index

*Chunk seed-escrow runtime: 0.00 s*

*Chunk gt-pdf-helper runtime: 0.00 s*

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 <- '/Users/blocke/Box Sync/Residency Personal Files/Scholarly Work/Locke Research Projects/abg-vbg-project/data'
data_dir_name <- '/Users/reblocke/Research/abg-vbg-project/data'

rdata_file <- file.path(data_dir_name, "full_trinetcx.rdata")
stata_file <- file.path(data_dir_name, "full_db.dta")

if (!dir.exists(data_dir_name)) {
  dir.create(data_dir_name)
  message("Directory 'data' created.")
} else {
  message("Directory 'data' already exists.")
}
```

Directory 'data' already exists.

```

if (file.exists(rdata_file)) {
  load(rdata_file)
  message("Loaded existing dataset from 'full_trinetx.rdata'.")
} else {
  message("RData file not found. Reading Stata dataset...")
  stata_data <- read_dta(stata_file)

  message("Extracting variable labels...")
  var_label(stata_data)

  message("Extracting value labels...")
  sapply(stata_data, function(x) if (is.labelled(x)) val_labels(x))

  save(stata_data, file = rdata_file)
  message("Dataset saved as 'full_trinetx.rdata'.")

  load(rdata_file)
  message("Loaded newly saved dataset from 'full_trinetx.rdata'.")
}

```

Loaded existing dataset from 'full\_trinetx.rdata'.

*Chunk load-trinetx-data runtime: 7.56 s*

```

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", "spo2",
  "sodium", "serum_cr", "serum_hco3", "serum_cl", "serum_lac", "serum_k",
  "wbc", "plt", "bnp", "serum_phos", "serum_ca"
)

gbm_params <- list(
  n.trees          = 1500,
  interaction.depth = 3,

```

```

shrinkage      = 0.01,
bag.fraction   = 0.8,
cv.folds       = 5,
stop.method    = "es.mean",
n.cores        = parallel::detectCores()
)

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

```

*Chunk propensity-config runtime: 0.01 s*

Creating subset\_data

```

set.seed(123)
rows_to_keep <- round(nrow(stata_data) * 1) #1 for real run
subset_data <- stata_data[sample(nrow(stata_data), rows_to_keep), ]

subset_data <- subset_data %>%
  filter(encounter_type != 1)

table(subset_data$encounter_type)

```

2	3
171727	343559

```
dim(subset_data)
```

```
[1] 515286    546
```

*Chunk sample-subset-data runtime: 6.18 s*

Generating Codebook for the Full Dataset

```
message("Generating codebook for the dataset...")
```

Generating codebook for the dataset...

```
study_codebook <- codebookr::codebook(  
  stata_data,  
  title = "Full TrinetX",  
  subtitle = "Dataset Documentation",  
  description = "This dataset contains patient-level records from the TrinetX database.  
    It has been processed and converted from the original Stata file."  
)  
codebook_file <- file.path(data_dir_name, "codebookr.docx")  
print(study_codebook, codebook_file)  
message("Codebook saved as 'codebookr.docx' in the data directory.")
```

Codebook saved as 'codebookr.docx' in the data directory.

*Chunk codebook-export-full runtime: 97.13 s*

New Variable - Death at 60 days

```
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)),  
  
    ## 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)
  )

```

*Chunk derive-death-60d runtime: 1.65 s*

```
table(subset_data$death_60d, useNA = "ifany")
```

```
0      1  
461485 53801
```

```
prop.table(table(subset_data$death_60d, useNA = "ifany"))
```

```
0      1  
0.89559 0.10441
```

```
summary(subset_data$death_60d)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000	0.0000	0.0000	0.1044	0.0000	1.0000

*Chunk death-60d-summary runtime: 0.04 s*

## 2.2 2) Baseline tables

### 2.2.1 2.1 Table 1A and 1B:

```
# Robust derivation of analysis variables + helper for Table 1 production  
# -----  
  
# helper: label binary 0/1 → "No"/"Yes"  
bin_lab <- function(x) factor(x, levels = c(0, 1), labels = c("No", "Yes"))  
  
subset_data <- subset_data %>%  
  mutate(
```

```

## ensure 0/1 numerics (avoids factor-level coercion)
across(c(has_abg, has_vbg, hypercap_on_abg, hypercap_on_vbg),
       ~ as.numeric(as.character(.))),

## derive ABG / VBG hypercapnia groups
abg_group
= case_when(
  has_abg == 0                      ~ "No ABG",
  has_abg == 1 & hypercap_on_abg == 0 ~ "ABG_NoHypercapnia",
  has_abg == 1 & hypercap_on_abg == 1 ~ "ABG_Hypercapnia",
  TRUE                               ~ "Missing"
),
vbg_group = case_when(
  has_vbg == 0                      ~ "No VBG",
  has_vbg == 1 & hypercap_on_vbg == 0 ~ "V рГС_NoHypercapnia",
  has_vbg == 1 & hypercap_on_vbg == 1 ~ "V рГС_Hypercapnia",
  TRUE                               ~ "Missing"
),

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

## labelled covariates
sex_label      = factor(sex, levels = c(0, 1), labels = c("Female", "Male")),
race_ethnicity_label      = factor(
  race_ethnicity,
  levels = c(0, 1, 2, 3, 4, 5, 6),
  labels = c("White", "Black or African American", "Hispanic",
            "Asian", "American Indian", "Pacific Islander", "Unknown"))

```

```

), location_label      = factor(
  location,
  levels = c(0, 1, 2, 3),
  labels = c("South", "Northeast", "Midwest", "West")
), encounter_type_label = factor(
  encounter_type,
  levels = c(2, 3),
  labels = 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"
)

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

  data %>%
    filter(!is.na (!!group_sym),                                # drop explicit NA
           !!group_sym != "Missing") %>%                      # drop "Missing" cohort
    droplevels() %>%                                         # trim empty factor levels
    select(all_of(c(group_var, vars))) %>%
    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(caption)
}

# 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")

table1A

table1B

```

*Chunk derive-table1-cohorts runtime: 7.07 s*

Generating Word Doc for Table 1A & 1B

```

ft_table1A <- as_flex_table(table1A)
ft_table1B <- as_flex_table(table1B)

doc <- read_docx() %>%
  body_add_par("Table 1A. Baseline Characteristics by ABG Group", style = "heading 1") %>%
  body_add_flextable(ft_table1A) %>%
  body_add_par("Table 1B. Baseline Characteristics by VBG Group", style = "heading 1") %>%
  body_add_flextable(ft_table1B)

print(doc, target = "Table1_ABG_VBG.docx")

```

*Chunk export-table1a-table1b-word runtime: 0.60 s*

<b>Variable</b>	<b>No ABG N = 328,044<sup>1</sup></b>	<b>ABG_NoHypercapnia N = 129,429<sup>1</sup></b>	<b>ABG_Hypercapnia N :</b>
Age (years)	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	62.1 ± 16.4; 0.0/57,813.0 mi
Current BMI kg/m2	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.8 ± 7.9; 33,496.0/57,813.0 m
sex_label			
Female	169,023 (52%)	57,767 (45%)	27,116 (47%)
Male	159,021 (48%)	71,662 (55%)	30,697 (53%)
race_ethnicity_label			
White	200,033 (61%)	81,357 (63%)	39,784 (69%)
Black or African American	62,418 (19%)	19,197 (15%)	8,082 (14%)
Hispanic	23,548 (7.2%)	7,464 (5.8%)	2,757 (4.8%)
Asian	4,880 (1.5%)	2,739 (2.1%)	789 (1.4%)
American Indian	1,971 (0.6%)	1,768 (1.4%)	316 (0.5%)
Pacific Islander	460 (0.1%)	162 (0.1%)	56 (<0.1%)
Unknown	34,734 (11%)	16,742 (13%)	6,029 (10%)
location_label			
South	138,843 (42%)	70,729 (55%)	32,694 (57%)
Northeast	93,209 (28%)	23,262 (18%)	12,975 (22%)
Midwest	22,924 (7.0%)	10,703 (8.3%)	4,844 (8.4%)
West	73,068 (22%)	24,735 (19%)	7,300 (13%)
osa_label	60,653 (18%)	17,709 (14%)	11,965 (21%)
asthma_label	48,456 (15%)	13,049 (10%)	8,268 (14%)
copd_label	60,214 (18%)	21,195 (16%)	18,846 (33%)
chf_label	59,770 (18%)	25,469 (20%)	16,219 (28%)
nmd_label	11,891 (3.6%)	5,861 (4.5%)	2,487 (4.3%)
phtn_label	23,854 (7.3%)	10,513 (8.1%)	7,347 (13%)
ckd_label	54,528 (17%)	24,849 (19%)	11,769 (20%)
diabetes_label	93,007 (28%)	37,426 (29%)	18,521 (32%)
encounter_type_label			
Emergency	142,713 (44%)	19,196 (15%)	9,818 (17%)
Inpatient	185,331 (56%)	110,233 (85%)	47,995 (83%)
VBG PCO2	45.5 ± 10.5; 233,430.0/328,044.0 missing (71.2%)	42.0 ± 11.2; 91,782.0/129,429.0 missing (70.9%)	57.4 ± 18.4; 40,411.0/57,813.0 mi
Arterial PCO2	NA ± NA; 328,044.0/328,044.0 missing (100.0%)	35.5 ± 6.1; 0.0/129,429.0 missing (0.0%)	58.5 ± 20.4; 0.0/57,813.0 mi

<sup>1</sup>Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

<b>Variable</b>	<b>No VBG N = 365,623<sup>1</sup></b>	<b>VBG_NoHypercapnia N = 105,646<sup>1</sup></b>	<b>VBG_Hypercapnia N :</b>
Age (years)	59.4 ± 17.8; 0.0/365,623.0 missing (0.0%)	58.1 ± 17.8; 0.0/105,646.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 mi
Current BMI kg/m2	31.8 ± 8.5; 192,892.0/365,623.0 missing (52.8%)	28.7 ± 7.2; 69,615.0/105,646.0 missing (65.9%)	29.3 ± 7.9; 31,038.0/44,017.0 m
sex_label			
Female	184,619 (50%)	48,931 (46%)	20,356 (46%)
Male	181,004 (50%)	56,715 (54%)	23,661 (54%)
race_ethnicity_label			
White	241,114 (66%)	55,100 (52%)	24,960 (57%)
Black or African American	61,814 (17%)	19,199 (18%)	8,684 (20%)
Hispanic	22,951 (6.3%)	8,354 (7.9%)	2,464 (5.6%)
Asian	5,439 (1.5%)	2,293 (2.2%)	676 (1.5%)
American Indian	2,128 (0.6%)	1,683 (1.6%)	244 (0.6%)
Pacific Islander	543 (0.1%)	110 (0.1%)	25 (<0.1%)
Unknown	31,634 (8.7%)	18,907 (18%)	6,964 (16%)
location_label			
South	196,774 (54%)	30,426 (29%)	15,066 (34%)
Northeast	65,537 (18%)	44,405 (42%)	19,504 (44%)
Midwest	24,891 (6.8%)	9,178 (8.7%)	4,402 (10%)
West	78,421 (21%)	21,637 (20%)	5,045 (11%)
osa_label	65,748 (18%)	15,634 (15%)	8,945 (20%)
asthma_label	49,810 (14%)	13,419 (13%)	6,544 (15%)
copd_label	70,950 (19%)	16,459 (16%)	12,846 (29%)
chf_label	68,964 (19%)	20,573 (19%)	11,921 (27%)
nmd_label	14,796 (4.0%)	3,754 (3.6%)	1,689 (3.8%)
phtn_label	27,731 (7.6%)	8,534 (8.1%)	5,449 (12%)
ckd_label	61,091 (17%)	21,290 (20%)	8,765 (20%)
diabetes_label	101,173 (28%)	33,665 (32%)	14,116 (32%)
encounter_type_label			
Emergency	124,405 (34%)	34,711 (33%)	12,611 (29%)
Inpatient	241,218 (66%)	70,935 (67%)	31,406 (71%)
VBG PCO2	NA ± NA; 365,623.0/365,623.0 missing (100.0%)	40.1 ± 6.6; 0.0/105,646.0 missing (0.0%)	60.2 ± 12.6; 0.0/44,017.0 mi
Arterial PCO2	42.4 ± 15.5; 233,430.0/365,623.0 missing (63.8%)	38.6 ± 15.4; 68,334.0/105,646.0 missing (64.7%)	52.7 ± 19.6; 26,280.0/44,017.0 m

<sup>1</sup>Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

## 2.2.2 2.2 Table 1 (Overall ABG/VBG status)

```
# 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"))
  )

# ABG table with "Everyone" column first
tbl1_abg <- subset_data %>%
  select(all_of(vars), 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), 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}%)"
    )
  )
```

```

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

tbl1

```

*Chunk table1-everyone-abg-vbg runtime: 3.10 s*

### 2.2.3 2.3 Table 2 (Hypercapnia within cohorts)

```

# Hypercapnia factors within measured cohorts
subset_data <- subset_data %>%
  mutate(
    hyper_abg = factor(hypercap_on_abg, levels = c(1, 0),
                        labels = c("Got ABG & Hypercapnia", "Got ABG & No hypercapnia")),
    hyper_vbg = factor(hypercap_on_vbg, levels = c(1, 0),
                        labels = c("Got VBG & Hypercapnia", "Got VBG & No hypercapnia"))
  )

# ABG cohort (has_abg == 1)
tbl2_abg <- subset_data %>%
  filter(has_abg == 1) %>%
  select(all_of(vars), hyper_abg) %>%
  gtsummary::tbl_summary(

```

Table 1

Variable	Everyone <sup>1</sup>	Did not get ABG N = 328,044 <sup>1</sup>	Did get ABG N =
Age (years)	59.2 ± 17.7; 0.0/515,286.0 missing (0.0%)	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	61.2 ± 16.9; 0.0/187,242.0
Current BMI kg/m2	31.1 ± 8.4; 293,545.0/515,286.0 missing (57.0%)	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	29.0 ± 7.2; 109,322.0/187,242.0
sex_label			
Female	253,906 (49%)	169,023 (52%)	84,883 (45%)
Male	261,380 (51%)	159,021 (48%)	102,359 (55%)
race_ethnicity_label			
White	321,174 (62%)	200,033 (61%)	121,141 (65%)
Black or African American	89,697 (17%)	62,418 (19%)	27,279 (15%)
Hispanic	33,769 (6.6%)	23,548 (7.2%)	10,221 (5.5%)
Asian	8,408 (1.6%)	4,880 (1.5%)	3,528 (1.9%)
American Indian	4,055 (0.8%)	1,971 (0.6%)	2,084 (1.1%)
Pacific Islander	678 (0.1%)	460 (0.1%)	218 (0.1%)
Unknown	57,505 (11%)	34,734 (11%)	22,771 (12%)
location_label			
South	242,266 (47%)	138,843 (42%)	103,423 (55%)
Northeast	129,446 (25%)	93,209 (28%)	36,237 (19%)
Midwest	38,471 (7.5%)	22,924 (7.0%)	15,547 (8.3%)
West	105,103 (20%)	73,068 (22%)	32,035 (17%)
osa_label	90,327 (18%)	60,653 (18%)	29,674 (16%)
asthma_label	69,773 (14%)	48,456 (15%)	21,317 (11%)
copd_label	100,255 (19%)	60,214 (18%)	40,041 (21%)
chf_label	101,458 (20%)	59,770 (18%)	41,688 (22%)
nmd_label	20,239 (3.9%)	11,891 (3.6%)	8,348 (4.5%)
phtn_label	41,714 (8.1%)	23,854 (7.3%)	17,860 (9.5%)
ckd_label	91,146 (18%)	54,528 (17%)	36,618 (20%)
diabetes_label	148,954 (29%)	93,007 (28%)	55,947 (30%)
encounter_type_label			
Emergency	171,727 (33%)	142,713 (44%)	29,014 (15%)
Inpatient	343,559 (67%)	185,331 (56%)	158,228 (85%)
VBG PCO2	46.0 ± 12.7; 365,623.0/515,286.0 missing (71.0%)	45.5 ± 10.5; 233,430.0/328,044.0 missing (71.2%)	46.9 ± 15.6; 132,193.0/187,242.0
Arterial PCO2	42.6 ± 16.3; 328,044.0/515,286.0 missing (63.7%)	NA ± NA; 328,044.0/328,044.0 missing (100.0%)	42.6 ± 16.3; 0.0/187,242.0

<sup>1</sup>Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

```

by = hyper_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 = "***Got ABG & Hypercapnia***",
  stat_2 = "***Got ABG & No hypercapnia***"
)

# VBG cohort (has_vbg == 1)
tbl2_vbg <- subset_data %>%
  filter(has_vbg == 1) %>%
  select(all_of(vars), hyper_vbg) %>%
  gtsummary::tbl_summary(
    by = hyper_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 = "***Got VBG & Hypercapnia***",
  stat_2 = "***Got VBG & No hypercapnia***"
)

# Merge side-by-side (no spanners; 4 requested columns)

```

```

table2 <- gtsummary::tbl_merge(
  tbls = list(tbl2_abg, tbl2_vbg),
  tab_spanner = c(NULL, NULL)
) %>%
  gtsummary::modify_caption("**Table 2. Baseline summary by hypercapnia within ABG and VBG cohorts**")

table2

```

*Chunk table2-hypercapnia-cohorts runtime: 3.52 s*

## 2.2.4 2.4 Generating Word Docs for New Table 1 and 2

```

library(gtsummary)
library(flextable)
library(officer)

# gtsummary objects (example: table1, table2)
ft1 <- as_flex_table(tbl1)
ft2 <- as_flex_table(table2)

doc <- read_docx() %>%
  body_add_par("Table 1", style = "heading 1") %>%
  body_add_flextable(ft1) %>%
  body_add_par("Table 2", style = "heading 1") %>%
  body_add_flextable(ft2)

print(doc, target = "Tables.docx")

```

*Chunk export-table1-table2-word runtime: 0.77 s*

## 3 Unweighted Binary Logistic Regressions

Unweighted, Hypercapnia (binary yes/no) Simple (1 predictor) Regressions:

**Table 1**

<b>Variable</b>	<b>Got ABG &amp; Hypercapnia<sup>1</sup></b>	<b>Got ABG &amp; No hypercapnia<sup>1</sup></b>	<b>Got VBG &amp; Hypercapnia<sup>1</sup></b>
Age (years)	62.1 ± 16.4; 0.0/57,813.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 missing (0.0%)
Current BMI kg/m2	29.8 ± 7.9; 33,496.0/57,813.0 missing (57.9%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.3 ± 7.9; 31,038.0/44,017.0 missing (57.9%)
sex_label			
Female	27,116 (47%)	57,767 (45%)	20,356 (46%)
Male	30,697 (53%)	71,662 (55%)	23,661 (54%)
race_ethnicity_label			
White	39,784 (69%)	81,357 (63%)	24,960 (57%)
Black or African American	8,082 (14%)	19,197 (15%)	8,684 (20%)
Hispanic	2,757 (4.8%)	7,464 (5.8%)	2,464 (5.6%)
Asian	789 (1.4%)	2,739 (2.1%)	676 (1.5%)
American Indian	316 (0.5%)	1,768 (1.4%)	244 (0.6%)
Pacific Islander	56 (<0.1%)	162 (0.1%)	25 (<0.1%)
Unknown	6,029 (10%)	16,742 (13%)	6,964 (16%)
location_label			
South	32,694 (57%)	70,729 (55%)	15,066 (34%)
Northeast	12,975 (22%)	23,262 (18%)	19,504 (44%)
Midwest	4,844 (8.4%)	10,703 (8.3%)	4,402 (10%)
West	7,300 (13%)	24,735 (19%)	5,045 (11%)
osa_label	11,965 (21%)	17,709 (14%)	8,945 (20%)
asthma_label	8,268 (14%)	13,049 (10%)	6,544 (15%)
copd_label	18,846 (33%)	21,195 (16%)	12,846 (29%)
chf_label	16,219 (28%)	25,469 (20%)	11,921 (27%)
nmd_label	2,487 (4.3%)	5,861 (4.5%)	1,689 (3.8%)
phtn_label	7,347 (13%)	10,513 (8.1%)	5,449 (12%)
ckd_label	11,769 (20%)	24,849 (19%)	8,765 (20%)
diabetes_label	18,521 (32%)	37,426 (29%)	14,116 (32%)
encounter_type_label			
Emergency	9,818 (17%)	19,196 (15%)	12,611 (29%)
Inpatient	47,995 (83%)	110,233 (85%)	31,406 (71%)
VBG PCO2	57.4 ± 18.4; 40,411.0/57,813.0 missing (69.9%)	42.0 ± 11.2; 91,782.0/129,429.0 missing (70.9%)	60.2 ± 12.6; 0.0/44,017.0 missing (0.0%)
Arterial PCO2	58.5 ± 20.4; 0.0/57,813.0 missing (0.0%)	35.5 ± 6.1; 0.0/129,429.0 missing (0.0%)	52.7 ± 19.6; 26,280.0/44,017.0 missing (0.0%)

<sup>1</sup>Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

Unweighted, ABG Group: hypercapnia treated as a binary (yes/no) predictor

### 3.0.1 3.1 ABG: Binary hypercapnia models

```
logit_intubated_abg <- glm(imv_proc ~ hypercap_on_abg, data = subset_data, family = binomial)
summary(logit_intubated_abg)
```

Call:

```
glm(formula = imv_proc ~ hypercap_on_abg, family = binomial,
     data = subset_data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.275471	0.005086	-447.4	<2e-16 ***
hypercap_on_abg	1.259993	0.010700	117.8	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 362580 on 515285 degrees of freedom  
Residual deviance: 350435 on 515284 degrees of freedom  
AIC: 350439

Number of Fisher Scoring iterations: 5

```
tidy(logit_intubated_abg,
      exponentiate = TRUE, # turns log-odds → OR
      conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1027485	0.0050863	-447.3684	0	0.1017279	0.1037765
hypercap_on_abg	3.5253956	0.0106997	117.7601	0	3.4521736	3.6000445

```
logit_niv_abg <- glm(niv_proc ~ hypercap_on_abg, data = subset_data, family = binomial)
summary(logit_niv_abg)
```

Call:  
`glm(formula = niv_proc ~ hypercap_on_abg, family = binomial,  
 data = subset_data)`

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.860418	0.006533	-437.84	<2e-16 ***
hypercap_on_abg	1.201665	0.013093	91.78	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 250639 on 515285 degrees of freedom  
Residual deviance: 243461 on 515284 degrees of freedom  
AIC: 243465

Number of Fisher Scoring iterations: 5

```
tidy(logit_niv_abg,
  exponentiate = TRUE, # turns log-odds → OR
  conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0572448	0.0065330	-437.84069	0	0.0565151	0.0579811

term	estimate	std.error	statistic	p.value	conf.low	conf.high
hypercap_on_abg	3.3256498	0.0130929	91.77958	0	3.2412752	3.4119744

```
logit_death_abg <- glm(death_60d ~ hypercap_on_abg, data = subset_data, family = binomial)
summary(logit_death_abg)
```

Call:

```
glm(formula = death_60d ~ hypercap_on_abg, family = binomial,
  data = subset_data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.258743	0.005052	-447.11	<2e-16 ***
hypercap_on_abg	0.756240	0.011903	63.53	<2e-16 ***
---				
Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1	' '	1	

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 344897 on 515285 degrees of freedom
Residual deviance: 341286 on 515284 degrees of freedom
AIC: 341290
```

Number of Fisher Scoring iterations: 5

```
tidy(logit_death_abg,
  exponentiate = TRUE, # turns log-odds → OR
  conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1044817	0.0050519	-447.10674	0	0.1034509	0.105520
hypercap_on_abg	2.1302521	0.0119030	63.53375	0	2.0810535	2.180455

```
logit_icd_abg <- glm(hypercap_resp_failure ~ hypercap_on_abg, data = subset_data, family = binomial)
summary(logit_icd_abg)
```

Call:  
`glm(formula = hypercap_resp_failure ~ hypercap_on_abg, family = binomial,  
 data = subset_data)`

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )		
(Intercept)	-3.355697	0.008192	-409.6	<2e-16 ***		
hypercap_on_abg	2.175594	0.012779	170.2	<2e-16 ***		
---						
Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'	0.1 ' '	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 223278 on 515285 degrees of freedom  
Residual deviance: 197920 on 515284 degrees of freedom  
AIC: 197924

Number of Fisher Scoring iterations: 6

```
tidy(logit_icd_abg,
  exponentiate = TRUE, # turns log-odds → OR
  conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.034885	0.0081920	-409.6314	0	0.034328	0.0354483
hypercap_on_abg	8.807416	0.0127795	170.2414	0	8.589528	9.0307810

Chunk abg-binary-logit-models runtime: 20.11 s

Display the regression coefficients for the binary (hypercapnia yes/no) predictor logistic regressions

	Intubated	NIV	Death	ICD Hyper
hypercap_on_abg	3.53 (3.45, 3.60)	3.33 (3.24, 3.41)	2.13 (2.08, 2.18)	8.81 (8.59, 9.03)

```
modelsummary(
  list("Intubated" = logit_intubated_abg,
       "NIV"      = logit_niv_abg,
       "Death"     = logit_death_abg,
       "ICD Hyper" = logit_icd_abg),
  exponentiate = TRUE,
  conf_level   = 0.95,
  estimate     = "{estimate}",
  statistic    = "({conf.low}, {conf.high})",
  coef_omit    = "(Intercept)",
  gof_omit     = ".*",
  fmt          = 2,
  output       = "gt")
) |>
  gt_pdf(title = "Odds Ratios for ABG Hypercapnia (>45 mmHg)'s association with...")
```

Profiled confidence intervals may take longer time to compute.  
 Use `ci\_method="wald"`` for faster computation of CIs.  
 Profiled confidence intervals may take longer time to compute.  
 Use `ci\_method="wald"`` for faster computation of CIs.  
 Profiled confidence intervals may take longer time to compute.  
 Use `ci\_method="wald"`` for faster computation of CIs.  
 Profiled confidence intervals may take longer time to compute.  
 Use `ci\_method="wald"`` for faster computation of CIs.

*Chunk abg-binary-or-table runtime: 24.56 s*

Unweighted VBG Group

### 3.0.2 3.2 VBG: Binary hypercapnia models

```
logit_intubated_vbg <- glm(imv_proc ~ hypercap_on_vbg, data = subset_data, family = binomial)
summary(logit_intubated_vbg)
```

Call:

```
glm(formula = imv_proc ~ hypercap_on_vbg, family = binomial,
     data = subset_data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.134026	0.004735	-450.69	<2e-16 ***
hypercap_on_vbg	0.648004	0.013168	49.21	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 362580 on 515285 degrees of freedom  
Residual deviance: 360405 on 515284 degrees of freedom  
AIC: 360409

Number of Fisher Scoring iterations: 4

```
tidy(logit_intubated_vbg,
      exponentiate = TRUE, # turns log-odds → OR
      conf.int = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1183598	0.0047351	-450.68580	0	0.1172651	0.1194621
hypercap_on_vbg	1.9117219	0.0131682	49.20969	0	1.8629159	1.9616038

```
logit_niv_vbg <- glm(niv_proc ~ hypercap_on_vbg, data = subset_data, family = binomial)
summary(logit_niv_vbg)
```

Call:  
`glm(formula = niv_proc ~ hypercap_on_vbg, family = binomial,  
 data = subset_data)`

Coefficients:  

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.735699	0.006091	-449.13	<2e-16 ***
hypercap_on_vbg	0.750555	0.015845	47.37	<2e-16 ***

  
 ---  
 Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 250639 on 515285 degrees of freedom  
 Residual deviance: 248688 on 515284 degrees of freedom  
 AIC: 248692

Number of Fisher Scoring iterations: 5

```
tidy(logit_niv_vbg,
  exponentiate = TRUE, # turns log-odds → OR
  conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0648486	0.0060911	-449.12691	0	0.0640777	0.0656261
hypercap_on_vbg	2.1181752	0.0158446	47.36965	0	2.0532293	2.1848014

```
logit_death_vbg <- glm(death_60d ~ hypercap_on_vbg, data = subset_data, family = binomial)
summary(logit_death_vbg)
```

```
Call:  
glm(formula = death_60d ~ hypercap_on_vbg, family = binomial,  
    data = subset_data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.197765	0.004856	-452.56	<2e-16 ***
hypercap_on_vbg	0.479895	0.014131	33.96	<2e-16 ***
---				

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 344897 on 515285 degrees of freedom  
Residual deviance: 343841 on 515284 degrees of freedom  
AIC: 343845

Number of Fisher Scoring iterations: 4

```
tidy(logit_death_vbg,  
      exponentiate = TRUE, # turns log-odds → OR  
      conf.int     = TRUE) # adds 95 % CI
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1110511	0.0048563	-452.56297	0	0.1099977	0.1121119
hypercap_on_vbg	1.6159045	0.0141315	33.95924	0	1.5716549	1.6611746

```
logit_icd_vbg <- glm(hypercap_resp_failure ~ hypercap_on_vbg, data = subset_data, family = binomial)  
summary(logit_icd_vbg)
```

Call:

```
glm(formula = hypercap_resp_failure ~ hypercap_on_vbg, family = binomial,
```

```

data = subset_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.136462   0.007293 -430.1 <2e-16 ***
hypercap_on_vbg 1.831609   0.013731   133.4 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 223278  on 515285  degrees of freedom
Residual deviance: 208772  on 515284  degrees of freedom
AIC: 208776

Number of Fisher Scoring iterations: 6

```

```

tidy(logit_icd_vbg,
      exponentiate = TRUE, # turns log-odds → OR
      conf.int     = TRUE) # adds 95 % CI

```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0434362	0.0072930	-430.0654	0	0.0428184	0.0440602
hypercap_on_vbg	6.2439262	0.0137314	133.3879	0	6.0779584	6.4140808

Chunk vbg-binary-logit-models runtime: 18.10 s

### 3.0.3 3.3 Display model coefficients for binary hypercapnia on VBG logistic regression

```

modelsummary(
  list("Intubated" = logit_intubated_vbg,
       "NIV"        = logit_niv_vbg,
       "Death"       = logit_death_vbg,

```

	Intubated	NIV	Death	ICD Hyper
hypercap_on_vbg	1.91 (1.86, 1.96)	2.12 (2.05, 2.18)	1.62 (1.57, 1.66)	6.24 (6.08, 6.41)

```

  "ICD Hyper" = logit_icd_vbg),
exponentiate = TRUE,
conf_level   = 0.95,
estimate     = "{estimate}",
statistic    = "{(conf.low}, {conf.high})",
coef_omit    = "(Intercept)",
gof_omit     = ".*",
# drop all goodness-of-fit rows
fmt          = 2,
# 2 decimal places everywhere
output       = "gt"
) |>
  gt_pdf(title = "Odds ratios for VBG hypercapnia (>45 mmHg) on outcomes")

```

Profiled confidence intervals may take longer time to compute.

Use `ci\_method="wald"` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci\_method="wald"` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci\_method="wald"` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci\_method="wald"` for faster computation of CIs.

*Chunk vbg-binary-or-table runtime: 23.13 s*

### 3.1 3) Three-level PCO2 categories (unweighted)

Now doing 3 groups instead of binary (above, normal and below)

```

subset_data <- subset_data %>%
  mutate(
  pco2_cat_abg = case_when(

```

```

!is.na(paco2) & paco2 < 35 ~ "Hypocapnia",
!is.na(paco2) & paco2 > 45 ~ "Hypercapnia",
!is.na(paco2) ~ "Eucapnia"
),
pco2_cat_vbg = case_when(
  !is.na(vbg_co2) & vbg_co2 < 40 ~ "Hypocapnia",
  !is.na(vbg_co2) & vbg_co2 > 50 ~ "Hypercapnia",
  !is.na(vbg_co2) ~ "Eucapnia"
)
) %>%
mutate(
  across(c(pco2_cat_abg, pco2_cat_vbg),
         ~factor(.x, levels = c("Eucapnia", "Hypocapnia", "Hypercapnia")))
)

library(broom)
library(dplyr)

run_logit <- function(data, outcome, exposure, group_name) {
  f <- as.formula(paste(outcome, "~", exposure))
  glm(f, data = data, family = binomial) %>%
    tidy(exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term != "(Intercept)") %>%
    mutate(
      outcome = outcome,
      group = group_name
    )
}

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

results <- bind_rows(
  lapply(outcomes, function(o) run_logit(subset_data, o, "pco2_cat_abg", "ABG")),
  lapply(outcomes, function(o) run_logit(subset_data, o, "pco2_cat_vbg", "VBG"))
)

```

```

combined_or_df <- results %>%
  mutate(
    exposure = recode(term,
      "pco2_cat_abgHypocapnia" = "Hypocapnia",
      "pco2_cat_abgHypercapnia" = "Hypercapnia",
      "pco2_cat_vbgHypocapnia" = "Hypocapnia",
      "pco2_cat_vbgHypercapnia" = "Hypercapnia"),
    outcome = recode(outcome,
      imv_proc = "Intubation",
      niv_proc = "NIV",
      death_60d = "Death (60d)",
      hypercap_resp_failure = "Hypercapnic RF")
  ) %>%
  select(outcome, group, exposure, estimate, conf.low, conf.high)

```

*Chunk or-data-three-level-unweighted runtime: 19.60 s*

```

library(scales)

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

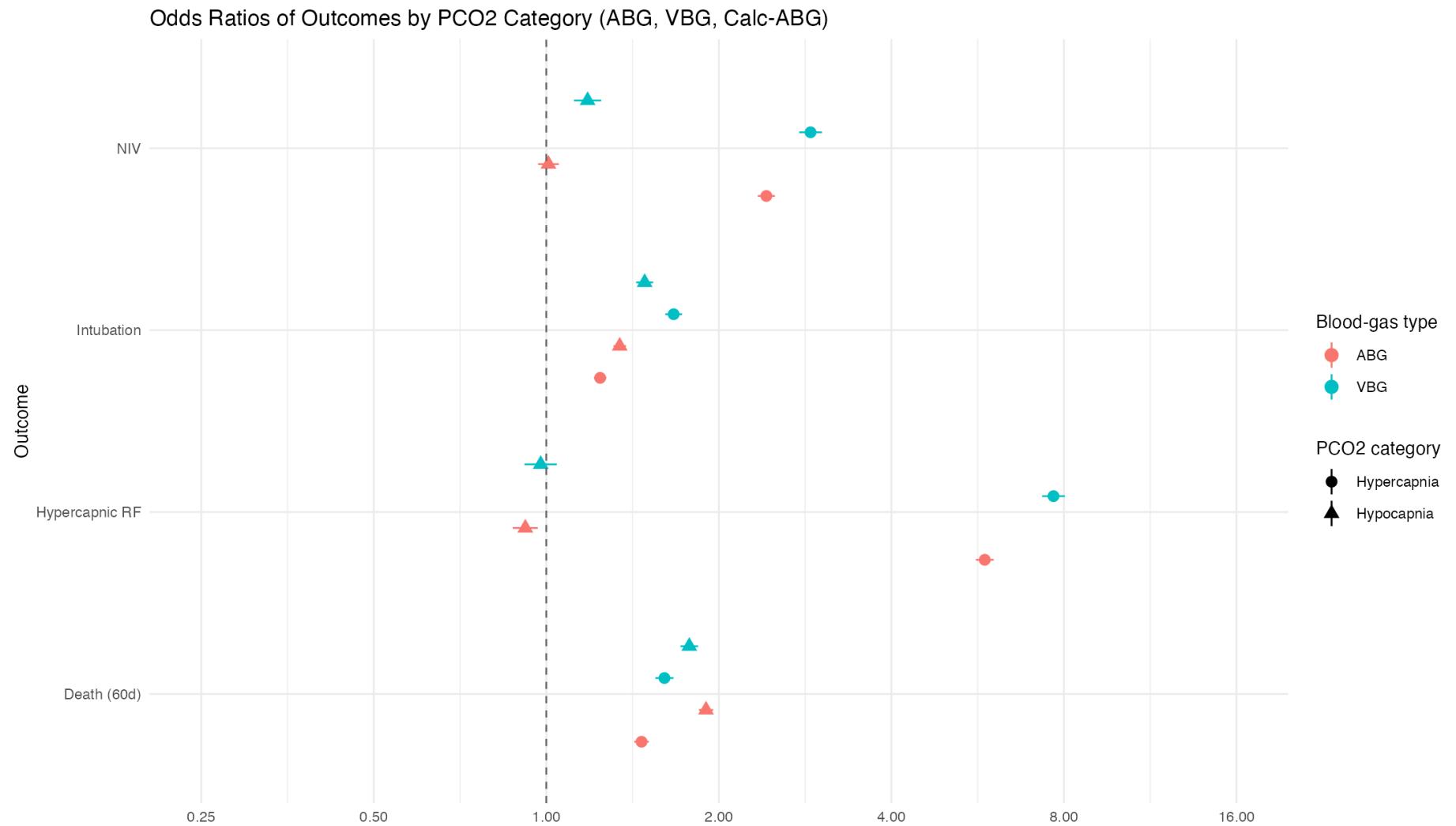
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +
  geom_pointrange(

```

```

position = position_dodge(width = 0.7),
size      = 0.6
) +
geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
scale_y_log10(
  breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
  limits = c(0.25, 16),
  labels = number_format(accuracy = 0.01)
) +
coord_flip() +
labs(
  title  = "Odds Ratios of Outcomes by PCO2 Category (ABG, VBG, Calc-ABG)",
  x       = "Outcome",
  y       = "Odds Ratio (log scale, 95% CI)",
  color   = "Blood-gas type",
  shape   = "PCO2 category",
  caption = paste(
    "Odds ratios are computed within each blood-gas cohort.",
    "Reference = patients in the normal PCO2 range.",
    "Hypocapnia: <35 mmHg (ABG/Calc) or <40 mmHg (VBG); Hypercapnia: >45 mmHg (ABG/Calc) or >50 mmHg (VBG).",
    "Because the underlying cohorts differ (ABG, VBG), denominators are not identical across groups.",
    sep = "\n"
  )
) +
theme_minimal(base_size = 10) +
theme(plot.caption = element_text(hjust = 0))

```



*Chunk or-plot-three-level-unweighted runtime: 0.33 s*

### 3.2 4) Restricted cubic spline regressions (unweighted)

```
# ABG spline dataset
subset_data_abg <- subset_data %>%
  select(paco2, imv_proc, niv_proc, death_60d, hypercap_resp_failure) %>%
  filter(!is.na(paco2))

dd_abg <- datadist(subset_data_abg)
options(datadist = "dd_abg")
```

Chunk rcs-abg-data-prep runtime: 0.03 s

#### 3.2.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```
fit_imv <- lrm(imv_proc ~ rcs(paco2, 4), data = subset_data_abg)
pred_imv <- as.data.frame(Predict(fit_imv, paco2, fun = plogis))

plot_imv <- ggplot(pred_imv, aes(x = paco2, y = yhat)) +
  geom_line(color = "blue", size = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "blue", alpha = 0.2) +
  labs(title = "Probability of Intubation by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()
```

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
i Please use `linewidth` instead.

```
fit_niv <- lrm(niv_proc ~ rcs(paco2, 4), data = subset_data_abg)
pred_niv <- as.data.frame(Predict(fit_niv, paco2, fun = plogis))

plot_niv <- ggplot(pred_niv, aes(x = paco2, y = yhat)) +
  geom_line(color = "green", size = 1.2) +
```

```

geom_ribbon(aes(ymin = lower, ymax = upper), fill = "green", alpha = 0.2) +
  labs(title = "Probability of NIV by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

fit_death <- lrm(death_60d ~ rcs(paco2, 4), data = subset_data_abg)
pred_death <- as.data.frame(Predict(fit_death, paco2, fun = plogis))

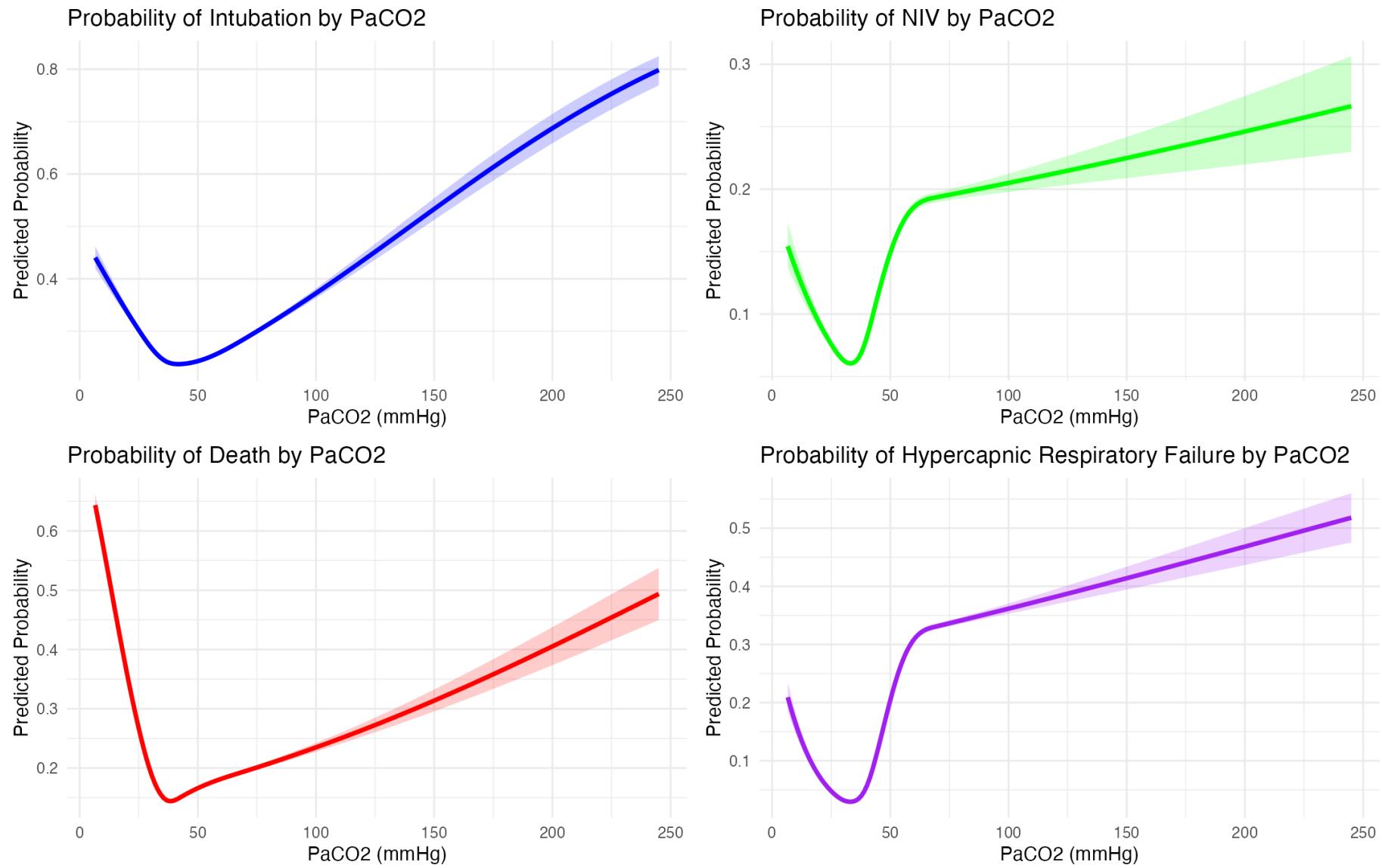
plot_death <- ggplot(pred_death, aes(x = paco2, y = yhat)) +
  geom_line(color = "red", size = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.2) +
  labs(title = "Probability of Death by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

fit_hcrcf <- lrm(hypercap_resp_failure ~ rcs(paco2, 4), data = subset_data_abg)
pred_hcrcf <- as.data.frame(Predict(fit_hcrcf, paco2, fun = plogis))

plot_hcrcf <- ggplot(pred_hcrcf, aes(x = paco2, y = yhat)) +
  geom_line(color = "purple", size = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.2) +
  labs(title = "Probability of Hypercapnic Respiratory Failure by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

(plot_imv | plot_niv) / (plot_death | plot_hcrcf)

```



Chunk rcs-abg-unweighted-models runtime: 2.01 s

### 3.2.2 4.2 Unweighted, Restricted Cubic Spline - VBG

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

dd_vbg <- datadist(subset_data_vbg)    # create datadist for VBG
# activate when doing VBG models:
options(datadist = "dd_vbg")
```

Chunk rcs-vbg-data-prep runtime: 0.03 s

```
subset_data_vbg <- subset_data %>%
  select(vbg_co2, imv_proc, niv_proc, death_60d, hypercap_resp_failure) %>%
  filter(!is.na(vbg_co2) & complete.cases(.))

dd <- datadist(subset_data_vbg)
options(datadist = "dd")

fit_imv_vbg <- lrm(imv_proc ~ rcs(vbg_co2, 4), data = subset_data_vbg)
fit_niv_vbg <- lrm(niv_proc ~ rcs(vbg_co2, 4), data = subset_data_vbg)
fit_death_vbg <- lrm(death_60d ~ rcs(vbg_co2, 4), data = subset_data_vbg)
fit_hcrcf_vbg <- lrm(hypercap_resp_failure ~ rcs(vbg_co2, 4), data = subset_data_vbg)

pred_imv_vbg <- as.data.frame(Predict(fit_imv_vbg, vbg_co2, fun = plogis))
pred_niv_vbg <- as.data.frame(Predict(fit_niv_vbg, vbg_co2, fun = plogis))
pred_death_vbg <- as.data.frame(Predict(fit_death_vbg, vbg_co2, fun = plogis))
pred_hcrcf_vbg <- as.data.frame(Predict(fit_hcrcf_vbg, vbg_co2, fun = plogis))

plot_imv_vbg <- ggplot(pred_imv_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "blue", alpha = 0.2) +
  labs(title = "IMV", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()
```

```

plot_niv_vbg <- ggplot(pred_niv_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "green") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "green", alpha = 0.2) +
  labs(title = "NIV", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

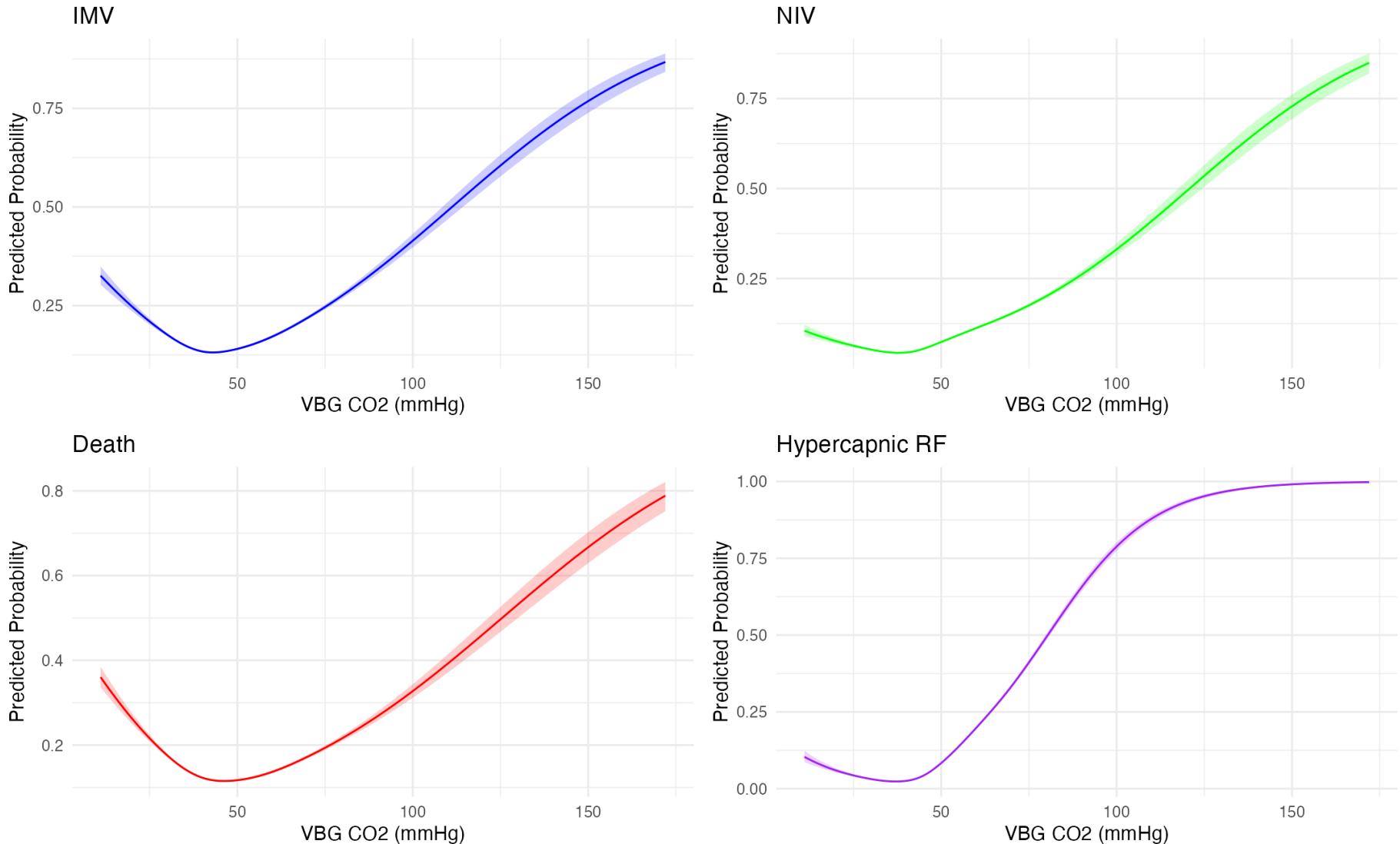
plot_death_vbg <- ggplot(pred_death_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "red") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.2) +
  labs(title = "Death", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

plot_hcrf_vbg <- ggplot(pred_hcrf_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "purple") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.2) +
  labs(title = "Hypercapnic RF", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

((plot_imv_vbg | plot_niv_vbg) /
 (plot_death_vbg | plot_hcrf_vbg)) +
 plot_annotation(title = "Predicted Probability by VBG CO2 (RCS Models)")

```

### Predicted Probability by VBG CO<sub>2</sub> (RCS Models)



*Chunk rcs-vbg-unweighted-models runtime: 1.27 s*

## 4 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: 39947224<https://pmc.ncbi.nlm.nih.gov/articles/PMC11825193/>

Current propensity score uses **age\_at\_encounter + sex + race\_ethnicity** (remember - have to specify to use this as a factor variable) + **curr\_bmi + copd + asthma + osa + chf + acute\_nmd + phtn + location** (as a factor variable)

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

1. Smaller GBM & stopping rule → faster fit, avoids over-fitting, lighter tails (which lead to extreme weights that are problematic).
2. bal.tab() documents balance; aim is to adjust spec until standard mean difference (SMD) < 0.1.
3. Weight stabilization (divide by mean) mitigates a few huge weights. I also winsorized, which is a way to avoid very extreme weights (ie you set <1st percentile to the 1st percentile value, and >99th percentile to 99th percentile).
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.

### 4.0.1 5.1 ABG IPW weighting and diagnostics

```
subset_data$encounter_type <- factor(subset_data$encounter_type,
                                         levels = c(2, 3),
                                         labels = c("Emergency", "Inpatient"))
```

Chunk encode-encounter-type runtime: 0.09 s

\*\*Removed lactate from weights, decreased n.trees, increased bagging

```

# 1. fit GBM propensity model, ABG
set.seed(42)

weight_model <- do.call(
  weightit,
  c(
    list(
      formula_abg,
      data      = subset_data,
      method    = "gbm",
      estimand  = "ATE",
      missing   = "ind",
      include.obj = TRUE      # ← REQUIRED for importance/SHAP
    ),
    gbm_params
  )
)
w_abg <- weight_model # Canonical alias so later code can use `w_abg`

# 2. Winsorise / stabilise weights (two-sided)
w <- weight_model$weights          # original GBM weights
w <- w / mean(w)                  # stabilise
cut <- quantile(w, c(0.01, 0.99), na.rm = TRUE)
w  <- pmin(pmax(w, cut[1]), cut[2]) # two-tail Winsorisation
w <- w / mean(w)                  # re-stabilise so E[w]=1

# overwrite inside the object and attach to data
weight_model$weights <- w
subset_data$w_abg    <- w

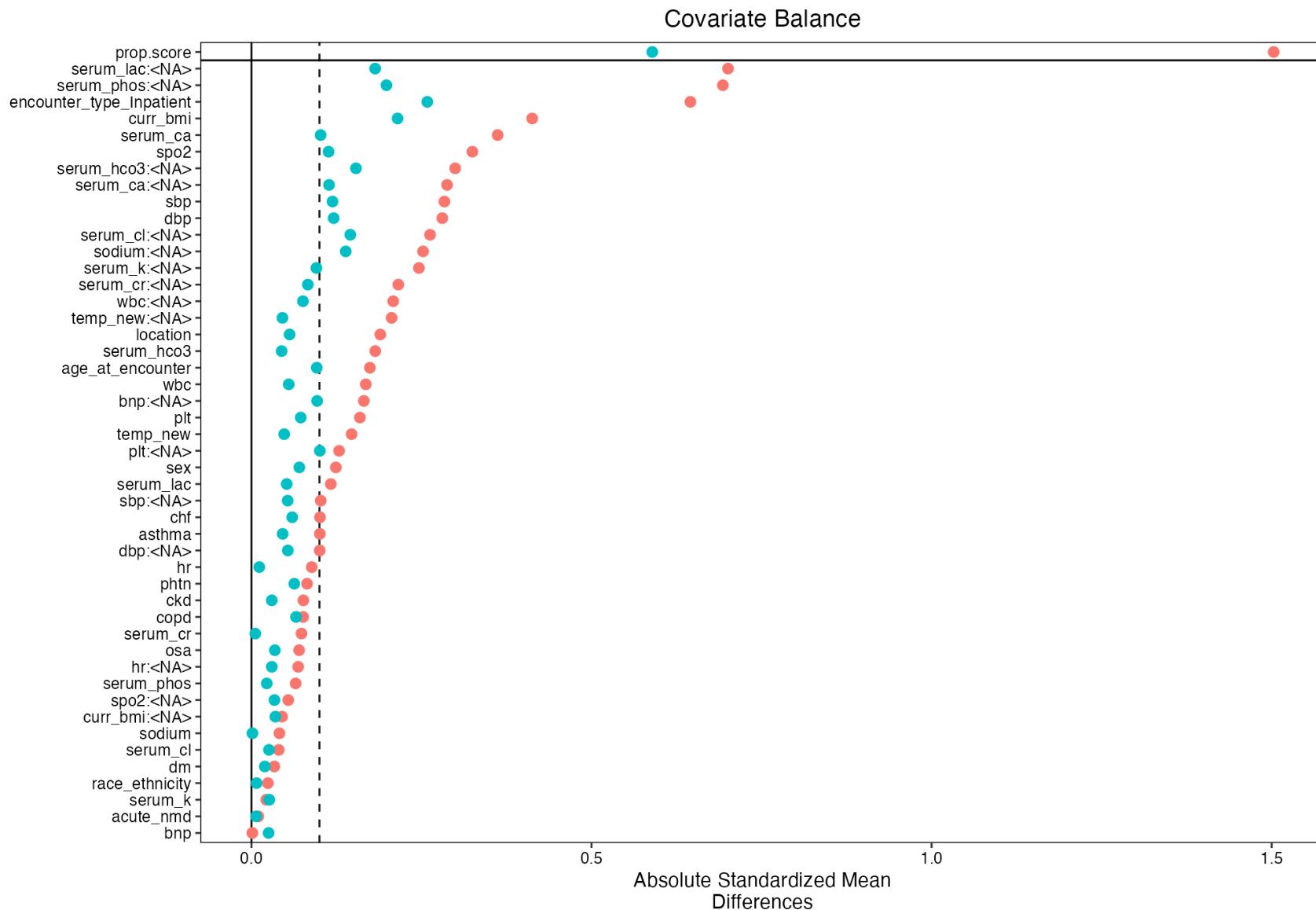
# 3. balance diagnostics (only raw vs. IPW)
bal <- bal.tab(
  weight_model,
  un = TRUE,
  m.threshold = 0.1,
  binary = "std",

```

```
  s.d.denom = "pooled"  
)
```

Warning: Missing values exist in the covariates. Displayed values omit these observations.

```
love.plot(  
  bal,  
  stats      = "m",           # standardized mean differences only  
  abs        = TRUE,  
  var.order   = "unadjusted",  
  sample.names = c("Raw", "IPW")  
)
```



```
# 4. survey design with the same weights
design <- svydesign(ids = ~1, weights = ~w_abg, data = subset_data)

# 5. outcome models (examples)
```

```

fit_niv  <- svyglm(niv_proc ~ has_abg, design = design, family = quasibinomial())
fit_imv  <- svyglm(imv_proc ~ has_abg, design = design, family = quasibinomial())
fit_death <- svyglm(death_60d ~ has_abg, design = design, family = quasibinomial())
fit_icd   <- svyglm(hypercap_resp_failure ~ has_abg, design = design, family = quasibinomial())

# quick effect estimates
lapply(list(IMV = fit_imv, NIV = fit_niv, Death = fit_death, ICD = fit_icd), function(m) {
  c(OR = exp(coef(m)[2]),
    LCL = exp(confint(m)[2,1]),
    UCL = exp(confint(m)[2,2]))
})

```

\$IMV  
 OR.has\_abg        LCL        UCL  
 6.664119    6.497030    6.835505

\$NIV  
 OR.has\_abg        LCL        UCL  
 1.978913    1.930275    2.028777

\$Death  
 OR.has\_abg        LCL        UCL  
 2.034938    1.993179    2.077571

\$ICD  
 OR.has\_abg        LCL        UCL  
 3.367251    3.270841    3.466503

*Chunk ipw-abg-weighting runtime: 774.24 s*

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

#### 4.0.2 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
#
#
# 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) %>%
  filter(complete.cases(.))
  
# 3. weighted logistic spline models with robust SEs
dd <- datadist(subset_data_abg); options(datadist = "dd")

fitfun <- function(formula)
  svyglm(
    formula,
    design = svydesign(ids = ~1, weights = ~w_abg, data = subset_data_abg),
    family = quasibinomial()
  )

fit_imv_abg   <- fitfun(imv_proc           ~ rcs(paco2, 4))
fit_niv_abg   <- fitfun(niv_proc           ~ rcs(paco2, 4))
fit_death_abg <- fitfun(death_60d          ~ rcs(paco2, 4))
fit_hcrf_abg  <- fitfun(hypercap_resp_failure ~ rcs(paco2, 4))

# 4. prediction helper
mkpred <- function(fit, data_ref) {
  # 1. Grid of PaCO2 values
```

```

newd <- data.frame(
  paco2 = seq(min(data_ref$paco2, na.rm = TRUE),
              max(data_ref$paco2, na.rm = TRUE),
              length.out = 200)
)

# 2. Design (model) matrix for the new data
mm <- model.matrix(delete.response(terms(fit)), # drop outcome
                    data = newd)

# 3. Linear predictor and its standard error
eta <- mm %*% coef(fit)                      # 'x
vcov <- vcov(fit)                            # robust VCov from svyglm
se   <- sqrt(rowSums((mm %*% vcov) * mm))    # √diag(X Σ X)

# 4. Transform to probability scale
transform(
  newd,
  yhat  = plogis(eta),
  lower = plogis(eta - 1.96 * se),
  upper = plogis(eta + 1.96 * se)
)
}

pred_imv_abg  <- mkpred(fit_imv_abg, subset_data_abg)
pred_niv_abg  <- mkpred(fit_niv_abg, subset_data_abg)
pred_death_abg <- mkpred(fit_death_abg, subset_data_abg)
pred_hcrcf_abg <- mkpred(fit_hcrcf_abg, subset_data_abg)

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

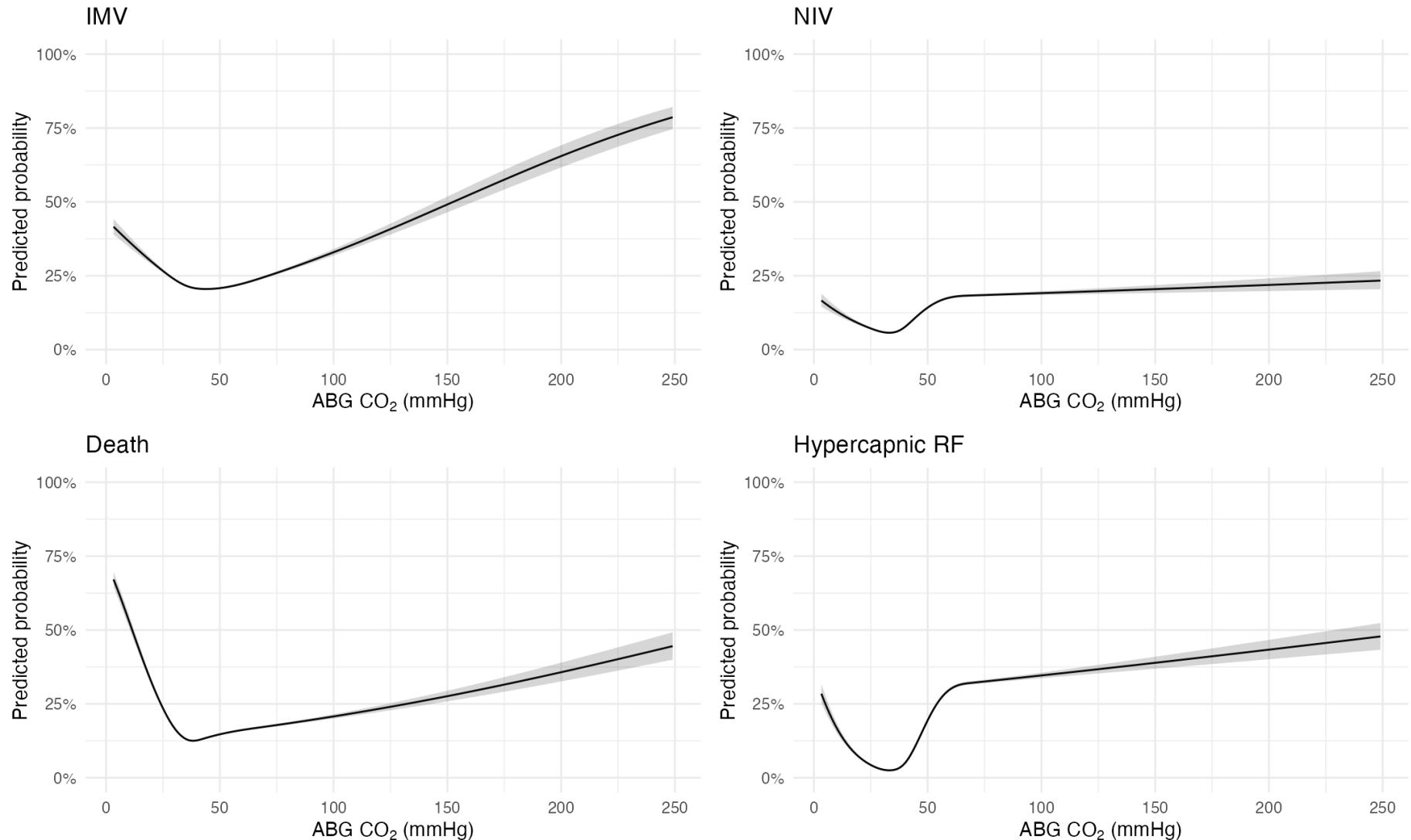
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +

```

```
scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
  labs(title = title, x = xlab, y = "Predicted probability") +
  theme_minimal()

(patchwork::wrap_plots(
  plt(pred_imv_abg,    "IMV"),
  plt(pred_niv_abg,    "NIV"),
  plt(pred_death_abg,  "Death"),
  plt(pred_hcrcf_abg,  "Hypercapnic RF"),
  ncol = 2
)
) +
  plot_annotation(
    title = expression(
      paste("Propensity-weighted predicted probability by ABG CO"[2],
            " (restricted cubic spline)")
    )
)
```

### Propensity-weighted predicted probability by ABG CO<sub>2</sub> (restricted cubic spline)



Chunk ipw-abg-rcts-models runtime: 4.15 s

Restricting plots bewtween 0.02 and 0.98

#### 4.0.3 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) %>%
  filter(complete.cases(.))

# 3. weighted logistic spline models with robust SEs
dd <- datadist(subset_data_abg); options(datadist = "dd")

fitfun <- function(formula)
  svyglm(
    formula,
    design = svydesign(ids = ~1, weights = ~w_abg, data = subset_data_abg),
    family = quasibinomial()
  )

fit_imv_abg   <- fitfun(imv_proc           ~ rcs(paco2, 4))
fit_niv_abg   <- fitfun(niv_proc           ~ rcs(paco2, 4))
fit_death_abg <- fitfun(death_60d          ~ rcs(paco2, 4))
fit_hcrf_abg  <- fitfun(hypercap_resp_failure ~ rcs(paco2, 4))

# 4. prediction helper
mkpred <- function(fit, data_ref) {
  # 1. Grid of PaCO2 values restricted to 2nd–98th percentile
  q <- quantile(data_ref$paco2, probs = c(0.02, 0.98), na.rm = TRUE)
  newd <- data.frame(
    paco2 = seq(q[1], q[2], length.out = 200)
  )

  # 2. Design (model) matrix for the new data
  mm <- model.matrix(delete.response(terms(fit)), data = newd)
```

```

# 3. Linear predictor and its standard error
eta  <- mm %*% coef(fit)
vcov <- vcov(fit)
se   <- sqrt(rowSums((mm %*% vcov) * mm))

# 4. Transform to probability scale
transform(
  newd,
  yhat  = plogis(eta),
  lower = plogis(eta - 1.96 * se),
  upper = plogis(eta + 1.96 * se)
)
}

pred_imv_abg  <- mkpred(fit_imv_abg,    subset_data_abg)
pred_niv_abg  <- mkpred(fit_niv_abg,    subset_data_abg)
pred_death_abg <- mkpred(fit_death_abg,  subset_data_abg)
pred_hcrcf_abg <- mkpred(fit_hcrcf_abg, subset_data_abg)

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

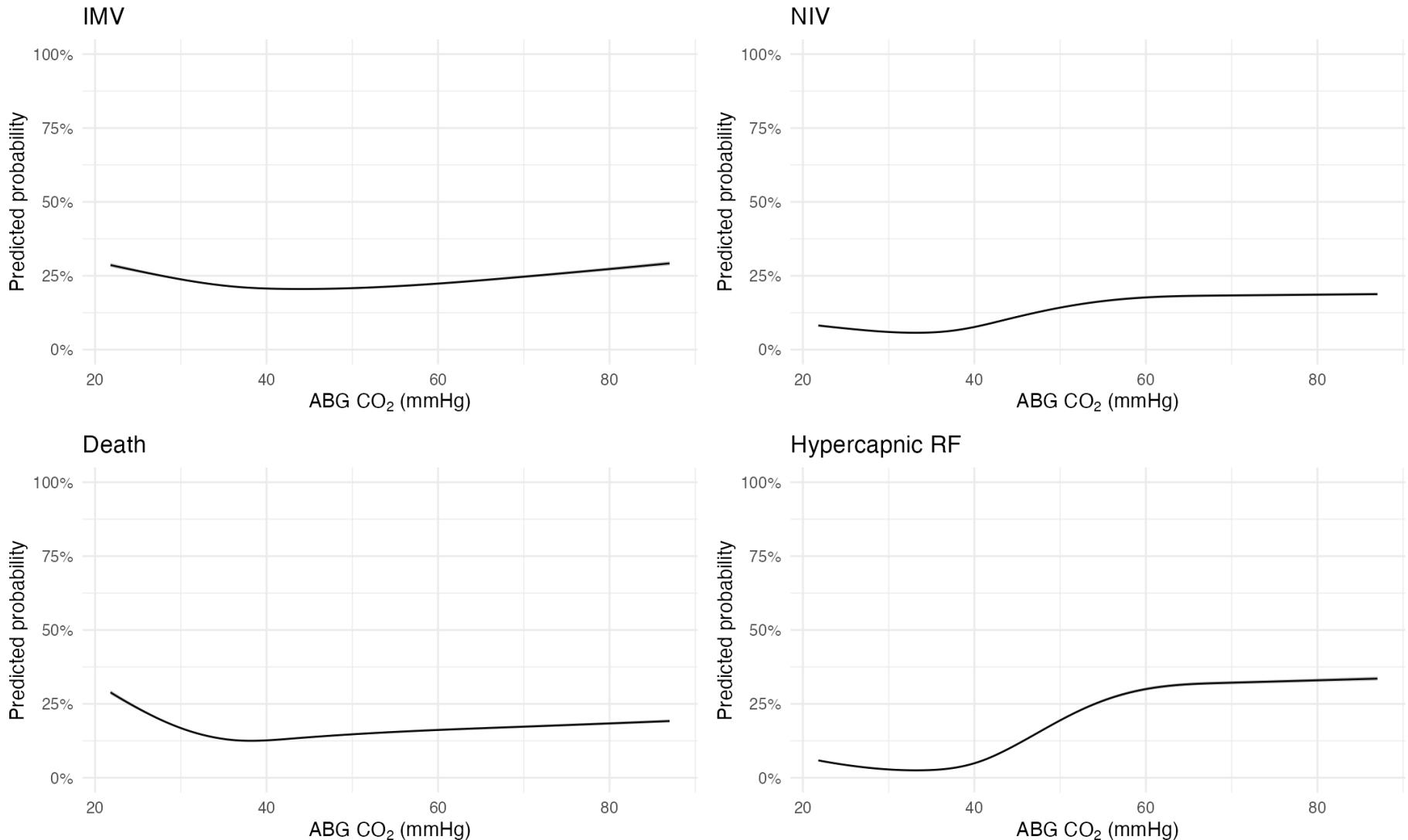
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

(patchwork:::wrap_plots(
  plt(pred_imv_abg,    "IMV"),
  plt(pred_niv_abg,    "NIV"),
  plt(pred_death_abg,  "Death"),
  plt(pred_hcrcf_abg,  "Hypercapnic RF"),
  ncol = 2
)

```

```
)  
) +  
  plot_annotation(  
    title = expression(  
      paste("Propensity-weighted predicted probability by ABG CO"[2],  
            " (restricted cubic spline)"))  
  )  
)
```

### Propensity-weighted predicted probability by ABG CO<sub>2</sub> (restricted cubic spline)



Chunk ipw-abg-rcts-trimmed runtime: 3.48 s

VBG - changed trees and bag fraction

#### 4.0.4 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)
w_vbg <- do.call(
  weightit,
  c(
    list(
      formula_vbg,
      data      = subset_data,
      method    = "gbm",
      estimand  = "ATE",
      missing   = "ind",
      include.obj = TRUE
    ),
    gbm_params
  )
)

# Stabilise & winsorise weights
w <- w_vbg$weights
w <- w / mean(w)
cut <- quantile(w, c(0.01, 0.99), na.rm = TRUE)
w  <- pmin(pmax(w, cut[1]), cut[2])
w <- w / mean(w)

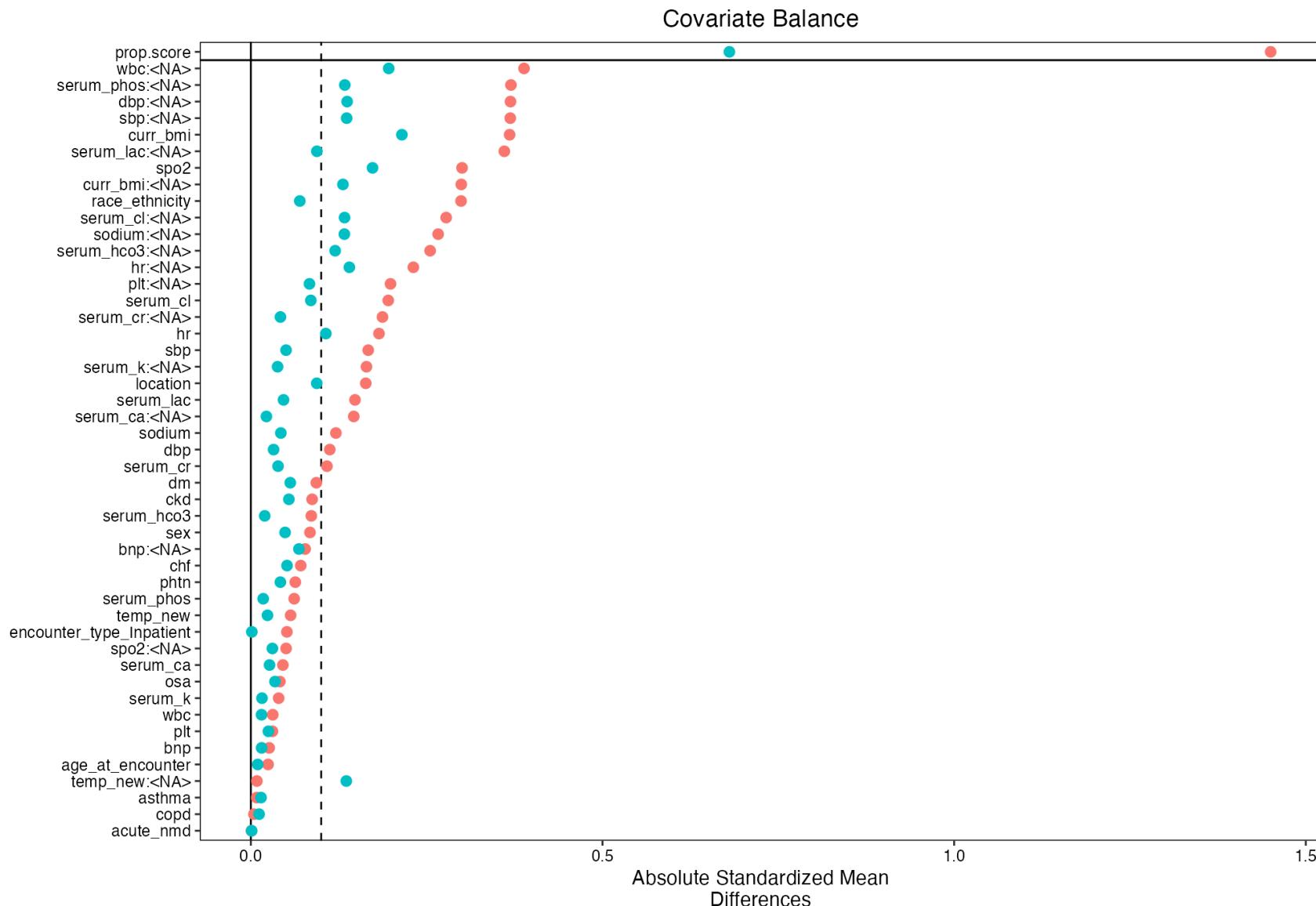
w_vbg$weights  <- w
subset_data$w_vbg <- w

v_bal <- bal.tab(
```

```
w_vbg,  
un = TRUE,  
m.threshold = 0.1,  
binary = "std",  
s.d.denom = "pooled"  
)
```

Warning: Missing values exist in the covariates. Displayed values omit these observations.

```
love.plot(  
  v_bal,  
  stats      = "m",           # standardized mean differences only  
  abs        = TRUE,  
  var.order   = "unadjusted",  
  sample.names = c("Raw", "IPW")  
)
```



```
# 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) %>%
filter(complete.cases(.))

# 3. Weighted spline models -----
dd_vbg <- datadist(subset_data_vbg)
options(datadist = "dd_vbg")

fitfun <- function(formula)
  svyglm(
    formula,
    design = svydesign(ids = ~1, weights = ~w_vbg, data = subset_data_vbg),
    family = quasibinomial()
  )

fit_imv_vbg   <- fitfun(imv_proc           ~ rcs(vbg_co2, 4))
fit_niv_vbg   <- fitfun(niv_proc          ~ rcs(vbg_co2, 4))
fit_death_vbg <- fitfun(death_60d         ~ rcs(vbg_co2, 4))
fit_hcrf_vbg  <- fitfun(hypercap_resp_failure ~ rcs(vbg_co2, 4))

# 4. Prediction helper -----
mkpred <- function(fit, data_ref) {
  newd <- data.frame(
    vbg_co2 = seq(min(data_ref$vbg_co2, na.rm = TRUE),
                  max(data_ref$vbg_co2, na.rm = TRUE),
                  length.out = 200)
  )
  mm   <- model.matrix(delete.response(terms(fit)), newd)
  eta  <- mm %*% coef(fit)
  vcov <- vcov(fit)
  se   <- sqrt(rowSums((mm %*% vcov) * mm))
  transform(
    newd,
    yhat = plogis(eta),
    lower = plogis(eta - 1.96 * se),
    upper = plogis(eta + 1.96 * se)
  )
}

```

```

}

pred_imv_vbg    <- mkpred(fit_imv_vbg,    subset_data_vbg)
pred_niv_vbg    <- mkpred(fit_niv_vbg,    subset_data_vbg)
pred_death_vbg <- mkpred(fit_death_vbg,  subset_data_vbg)
pred_hcrf_vbg   <- mkpred(fit_hcrf_vbg,  subset_data_vbg)

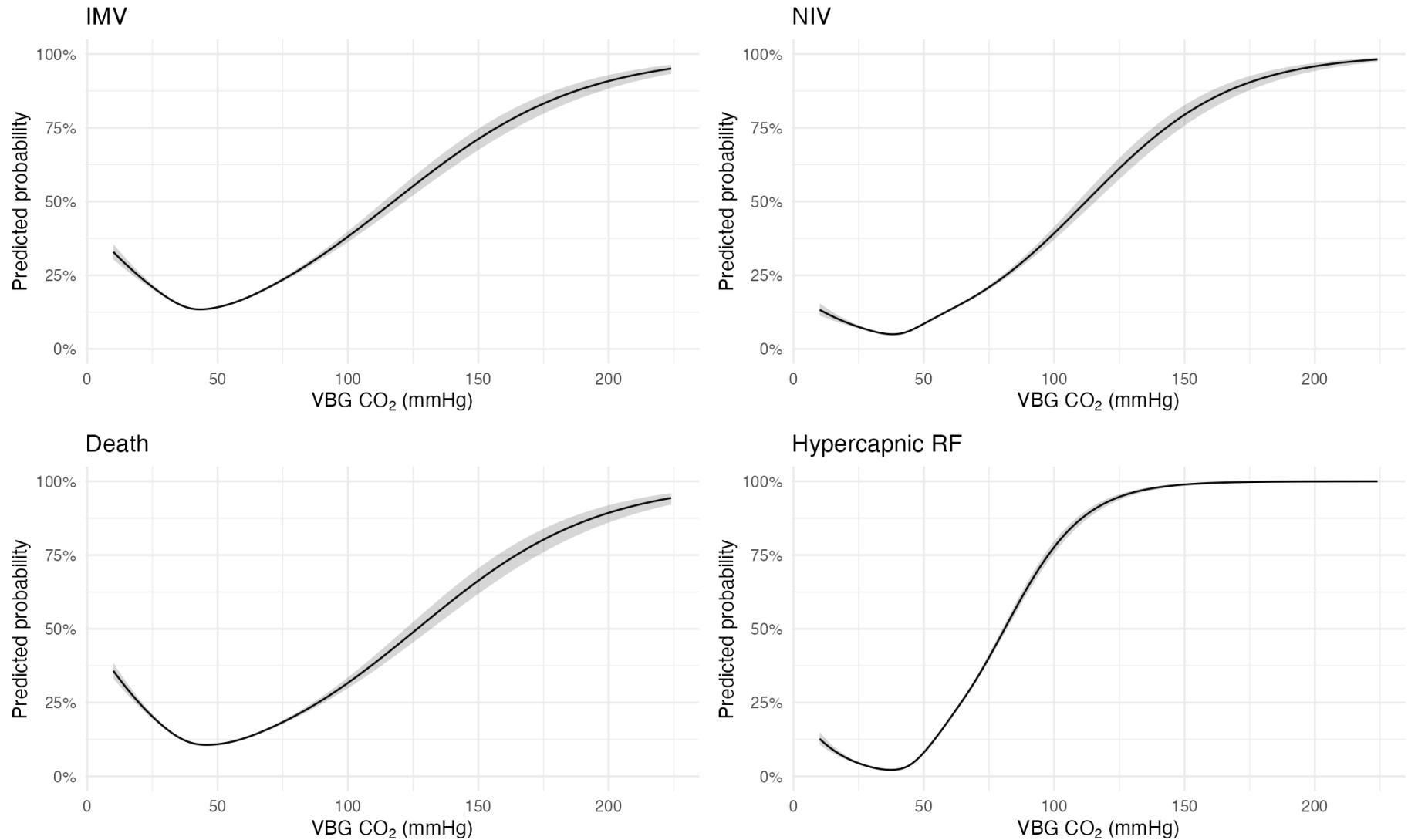
# 5. Plotting (gray scheme) -----
xlab <- expression(paste("VBG CO"[2], " (mmHg)"))

plt <- function(dat, title)
  ggplot(dat, aes(vbg_co2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

(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 = expression(
      paste("Propensity-weighted predicted probability by VBG CO"[2],
            " (restricted cubic spline)")
    )
)

```

### Propensity-weighted predicted probability by VBG CO<sub>2</sub> (restricted cubic spline)



Chunk ipw-vbg-workflow runtime: 1421.65 s

Calculated VBG to ABG / Farkas

## 4.1 5) Weighted effect estimates

New weighted binary regression figures.

```
# IP-weighted odds-ratio plot (ABG, VBG, Calculated-ABG)
#   - exact analogue of the un-weighted figure
#
#
# weights already attached earlier:
#   • w_abg      - propensity for *ABG*    (column in subset_data)
#   • w_vbg      - propensity for *VBG*    (column in subset_data)
#   •           - same weights, used for calculated ABG CO2
#
# 1. helper to fit an IP-weighted GLM and return tidy OR -----
tidy_ipw <- function(data, outcome, exposure, weight_var,
                      group_label, outcome_label) {
  des <- svydesign(ids = ~1, weights = as.formula(paste0("~", weight_var)),
                   data = data)
  mod <- svyglm(
    as.formula(paste0(outcome, " ~ ", exposure)),
    design = des,
    family = quasibinomial()
  )
  tidy(mod, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term == exposure) %>% # keep the exposure row
    mutate(group = group_label, outcome = outcome_label)
}
#
# 2. cohort-specific data frames -----
abg_df   <- subset_data %>% filter(has_abg == 1)
vbg_df   <- subset_data %>% filter(has_vbg == 1)
#
# 3. fit models & collect estimates -----
ipw_estimates <- bind_rows(
  # ABG
```

```

tidy_ipw(abg_df, "imv_proc", "hypercap_on_abg", "w_abg", "ABG", "Intubation"),
tidy_ipw(abg_df, "niv_proc", "hypercap_on_abg", "w_abg", "ABG", "NIV"),
tidy_ipw(abg_df, "death_60d", "hypercap_on_abg", "w_abg", "ABG", "Death"),
tidy_ipw(abg_df, "hypercap_resp_failure", "hypercap_on_abg", "w_abg", "ABG", "ICD Code"),

# VBG
tidy_ipw(vbg_df, "imv_proc", "hypercap_on_vbg", "w_vbg", "VBG", "Intubation"),
tidy_ipw(vbg_df, "niv_proc", "hypercap_on_vbg", "w_vbg", "VBG", "NIV"),
tidy_ipw(vbg_df, "death_60d", "hypercap_on_vbg", "w_vbg", "VBG", "Death"),
tidy_ipw(vbg_df, "hypercap_resp_failure", "hypercap_on_vbg", "w_vbg", "VBG", "ICD Code")
)

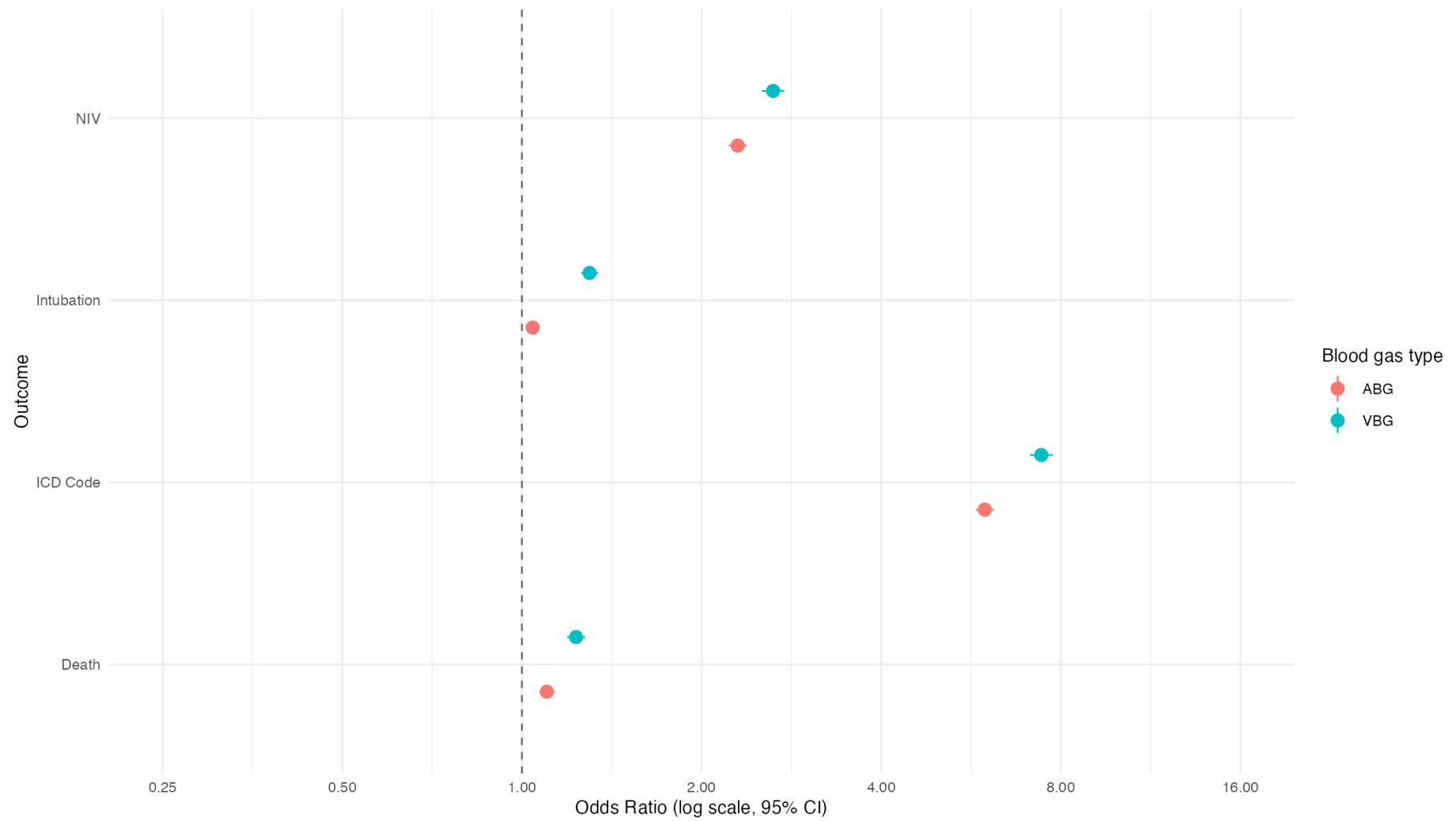
# 4. plotting -----
ipw_estimates$group <- factor(
  ipw_estimates$group,
  levels = c("ABG", "VBG")
)

ggplot(
  ipw_estimates,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group
  )
) +
  geom_pointrange(position = position_dodge(width = 0.6), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = number_format(accuracy = 0.01)
  ) +
  coord_flip() +

```

```
labs(  
  title = "IP Weighted Odds Ratio of Outcomes When Hypercapnia Present",  
  x      = "Outcome",  
  y      = "Odds Ratio (log scale, 95% CI)",  
  color  = "Blood gas type",  
  caption = paste(  
    "Inverse-probability weights adjust for covariates associated with receiving each blood-gas.",  
    "Models are fitted within their respective cohorts:",  
    "ABG (weights = w_abg), VBG (w_vbg).",  
    "Numerator = hypercapnic; denominator = normocapnic within cohort.",  
    sep = "\n"  
  )  
) +  
theme_minimal(base_size = 10) +  
theme(plot.caption = element_text(hjust = 0))
```

### IP Weighted Odds Ratio of Outcomes When Hypercapnia Present



Inverse-probability weights adjust for covariates associated with receiving each blood-gas.

Models are fitted within their respective cohorts:

ABG (weights =  $w_{abg}$ ), VBG ( $w_{vbg}$ ).

Numerator = hypercapnic; denominator = normocapnic within cohort.

*Chunk ipw-binary-or-plot runtime: 12.28 s*

#### 4.1.1 5.5 Three-level PCO<sub>2</sub> categories (weighted; ABG, VBG)

Three Groups with Weights

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

# 1. Create PCO2 categories
subset_data <- subset_data %>%
  mutate(
    pco2_cat_abg = case_when(
      !is.na(paco2) & paco2 < 35 ~ "Hypocapnia",
      !is.na(paco2) & paco2 >= 35 & paco2 <= 45 ~ "Eucapnia",
      !is.na(paco2) & paco2 > 45 ~ "Hypercapnia",
      TRUE ~ NA_character_
    ),
    pco2_cat_vbg = case_when(
      !is.na(vbg_co2) & vbg_co2 < 40 ~ "Hypocapnia",
      !is.na(vbg_co2) & vbg_co2 >= 40 & vbg_co2 <= 50 ~ "Eucapnia",
      !is.na(vbg_co2) & vbg_co2 > 50 ~ "Hypercapnia",
      TRUE ~ NA_character_
    )
  )

# 2. Function: weighted logistic regression & OR extraction
run_weighted_or <- function(data, outcome, cat_var, weight_var, group_name) {
  dat <- data %>%
    filter(
      !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 = c("Eucapnia", "Hypocapnia", "Hypercapnia"))
) %>%
  droplevels()

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

fit <- svyglm(as.formula(paste(outcome, "~", cat_var)),
              design = design, family = quasibinomial())

tidy(fit, exponentiate = TRUE, conf.int = TRUE) %>%
  filter(term != "(Intercept)") %>%
  mutate(
    group      = group_name,
    outcome    = outcome,
    exposure   = gsub(paste0(cat_var), "", term) %>%
                  gsub("`", "", .)
  )
}

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

combined_or_df <- bind_rows(
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_abg", "w_abg", "ABG")),
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_vbg", "w_vbg", "VBG")))
)

# Ensure nice ordering

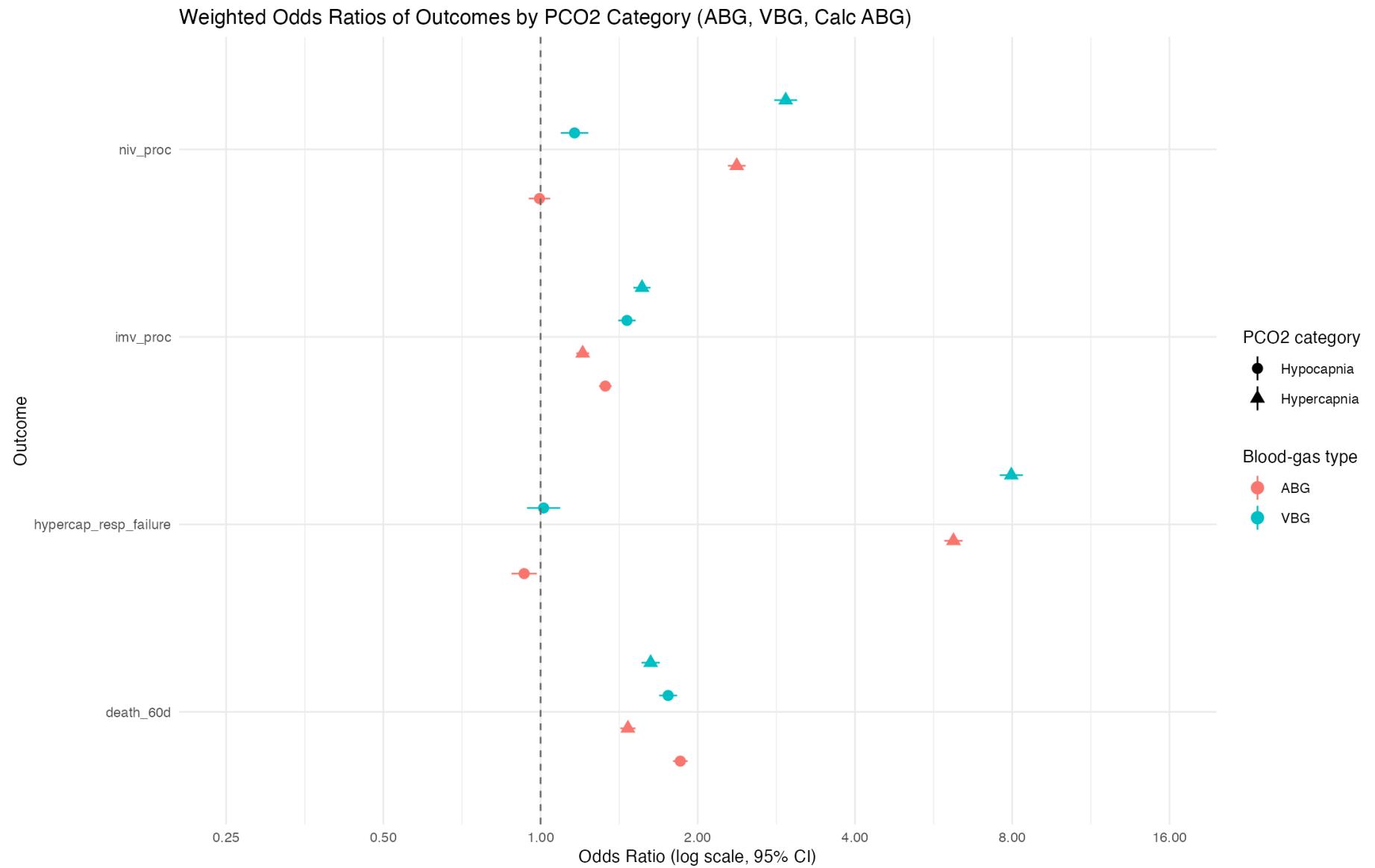
```

```

combined_or_df$group      <- factor(combined_or_df$group,
                                      levels = c("ABG", "VBG"))
combined_or_df$exposure <- factor(combined_or_df$exposure,
                                      levels = c("Eucapnia", "Hypocapnia", "Hypercapnia"))

# 4. Plot weighted odds ratios
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +
  geom_pointrange(position = position_dodge(width = 0.7), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = number_format(accuracy = 0.01)
  ) +
  coord_flip() +
  labs(
    title  = "Weighted Odds Ratios of Outcomes by PCO2 Category (ABG, VBG, Calc ABG)",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95% CI)",
    color  = "Blood-gas type",
    shape  = "PCO2 category"
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))

```



Chunk ipw-three-level-pco2-all runtime: 21.35 s

#### 4.1.2 5.6 Three-level PCO<sub>2</sub> categories (weighted; ABG vs VBG only)

Three groups with weights: Just ABG and VBG

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

# 2. Function: weighted logistic regression & OR extraction
run_weighted_or <- function(data, outcome, cat_var, weight_var, group_name) {
  dat <- data %>%
    filter(
      !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 = c("Eucapnia", "Hypocapnia", "Hypercapnia"))
    ) %>%
    droplevels()

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

  fit <- svyglm(as.formula(paste(outcome, "~", cat_var)),
                design = design, family = quasibinomial())

  tidy(fit, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term != "(Intercept)") %>%
```

```

    mutate(
      group    = group_name,
      outcome  = outcome,
      exposure = gsub(paste0(cat_var), "", term) %>%
                    gsub(``, "", .)
    )
  }

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

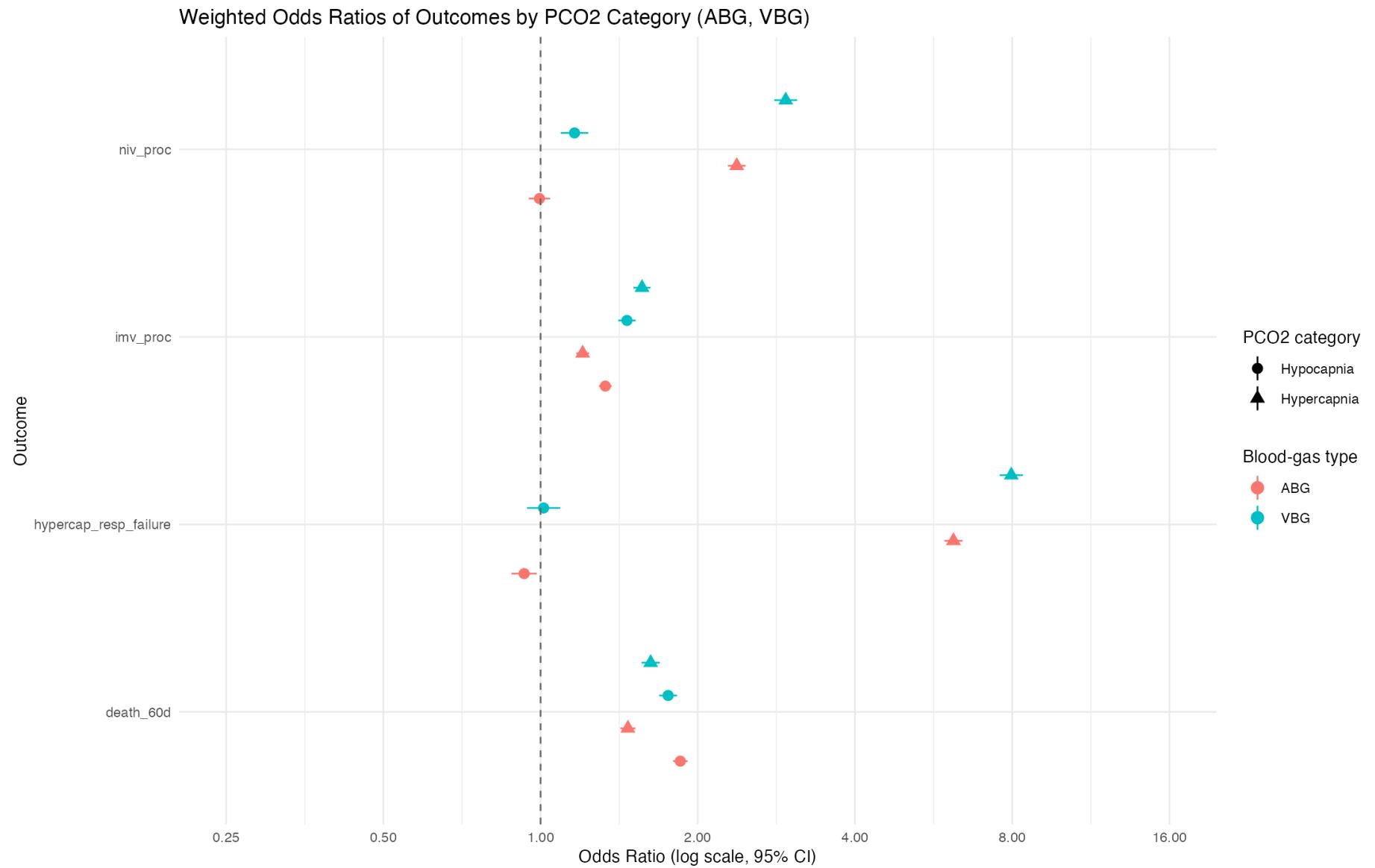
combined_or_df <- bind_rows(
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_abg", "w_abg", "ABG")),
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_vbg", "w_vbg", "VBG"))
)

# Ensure nice ordering
combined_or_df$group     <- factor(combined_or_df$group,
                                      levels = c("ABG", "VBG"))
combined_or_df$exposure <- factor(combined_or_df$exposure,
                                      levels = c("Eucapnia", "Hypocapnia", "Hypercapnia"))

# 4. Plot weighted odds ratios
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +

```

```
geom_pointrange(position = position_dodge(width = 0.7), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = number_format(accuracy = 0.01)
  ) +
  coord_flip() +
  labs(
    title = "Weighted Odds Ratios of Outcomes by PCO2 Category (ABG, VBG)",
    x = "Outcome",
    y = "Odds Ratio (log scale, 95% CI)",
    color = "Blood-gas type",
    shape = "PCO2 category"
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```



Chunk ipw-three-level-pco2-abg-vbg runtime: 21.72 s

## 4.2 6) Propensity score diagnostics

Plotting propensity scores

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

library(dplyr)
library(ggplot2)
library(scales)

# Resolve WeightIt objects regardless of naming used upstream
w_abg_obj <- if (exists("w_abg")) w_abg else if (exists("weight_model")) weight_model else NULL
w_vbg_obj <- if (exists("w_vbg")) w_vbg else NULL

if (is.null(w_abg_obj)) stop("ABG WeightIt object not found. Define `w_abg` or `weight_model` before this block.")
if (!"has_abg" %in% names(subset_data)) stop("`subset_data` must contain `has_abg` for ABG PS plotting.")

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

if (!is.null(w_vbg_obj) && "has_vbg" %in% names(subset_data)) {
  ps_dfs$VBG <- data.frame(
    ps      = w_vbg_obj$ps,
    treat   = subset_data$has_vbg,
    ScoreType = "VBG"
  )
} else if (is.null(w_vbg_obj)) {
  message("Note: VBG WeightIt object `w_vbg` not found; skipping VBG panel.")
}
```

```

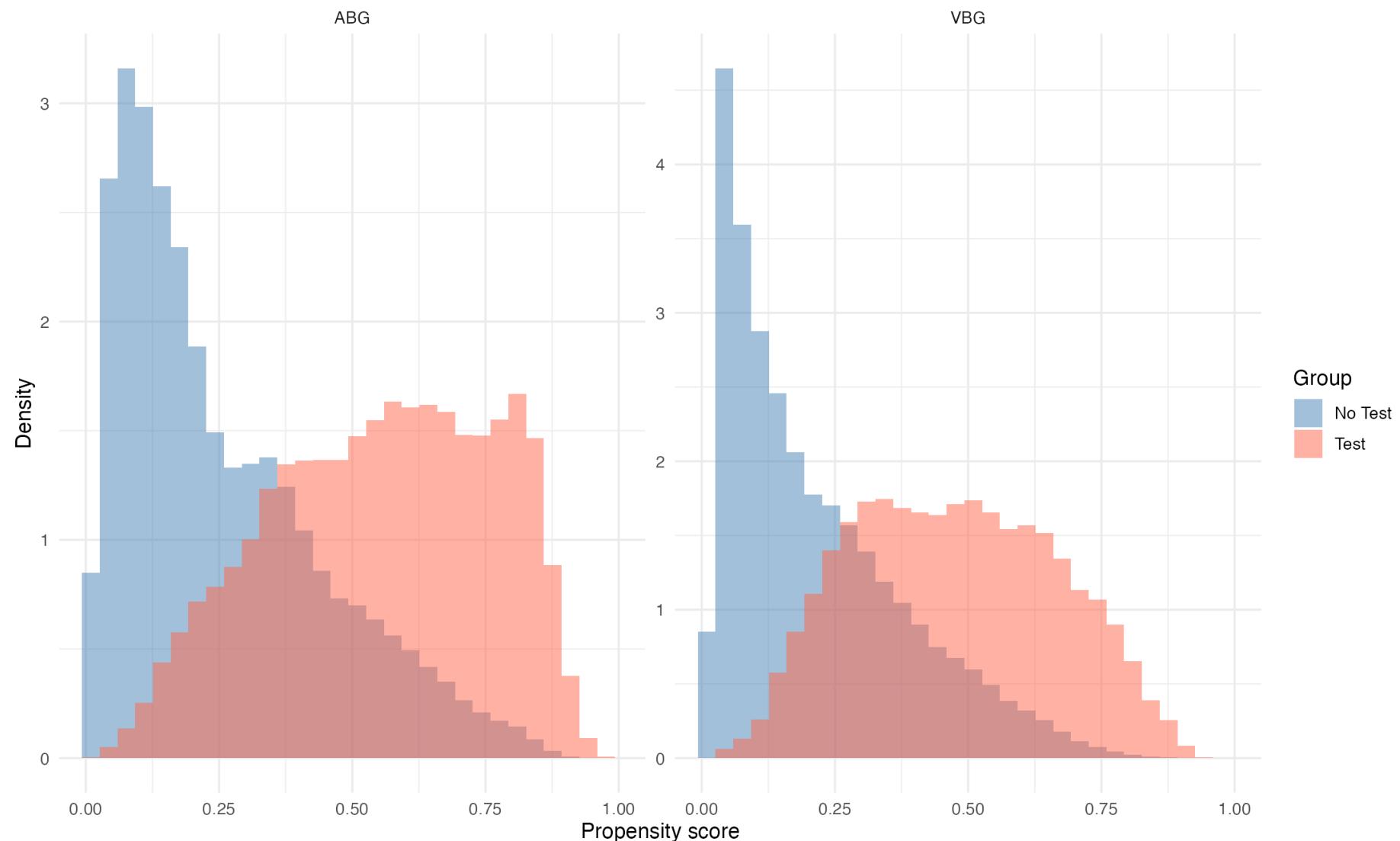
# Bind, clean, and factorize for plotting
df_ps <- bind_rows(psdfs) %>%
  filter(!is.na(ps), !is.na(treat)) %>%
  mutate(
    treat      = factor(treat, levels = c(0, 1), labels = c("No Test", "Test")),
    ScoreType = factor(ScoreType, levels = c("ABG", "VBG"))
  )

# Plot
ggplot(df_ps, aes(x = ps, fill = treat)) +
  geom_histogram(aes(y = ..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)

```

Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.  
 i Please use `after\_stat(density)` instead.

## Propensity Score Distributions



Chunk propensity-histograms-conditional runtime: 0.67 s

```
df_ps <- bind_rows(  
  data.frame(
```

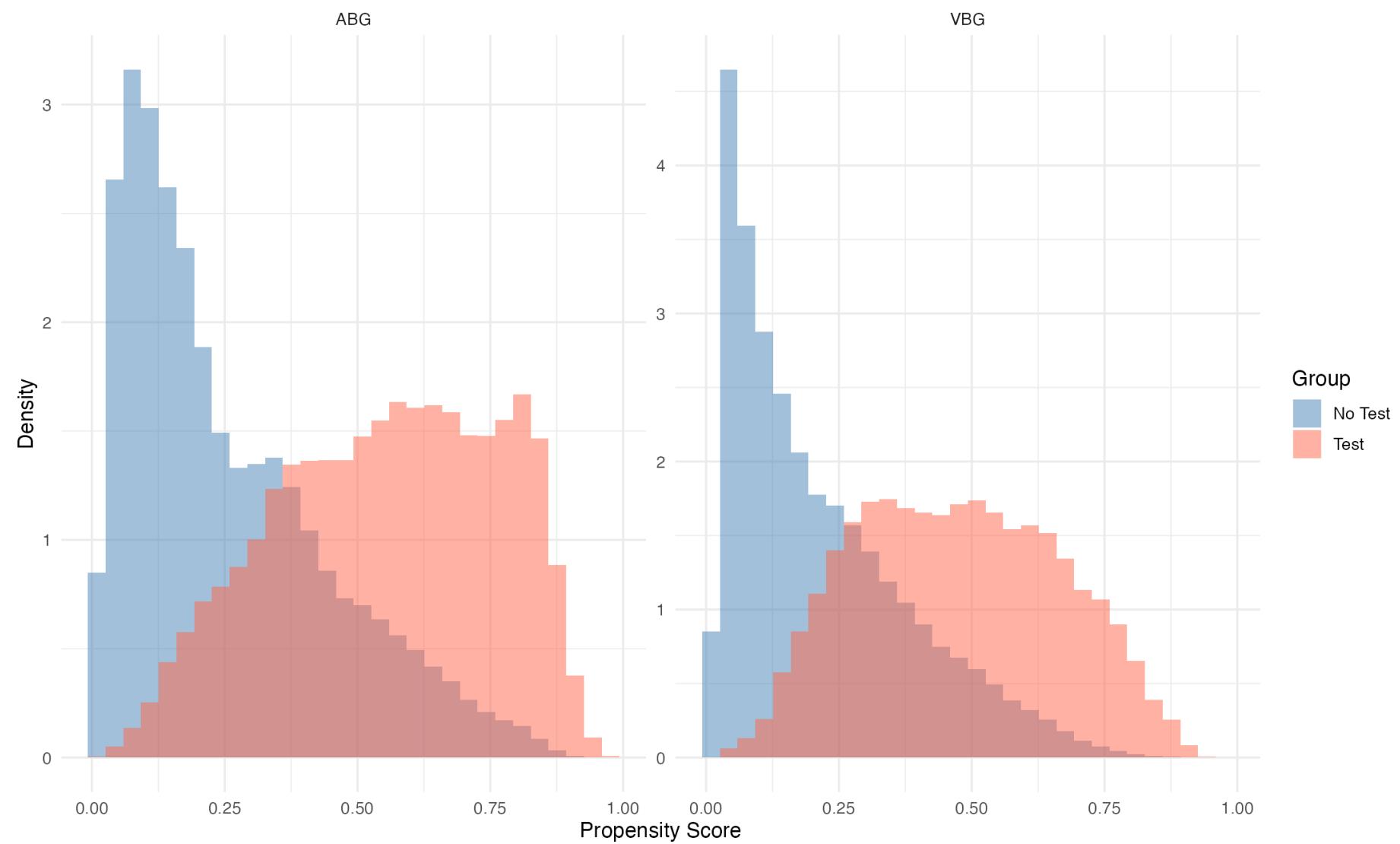
```

ps      = w_abg$ps,
treat   = subset_data$has_abg,
ScoreType = "ABG"
),
data.frame(
  ps      = w_vbg$ps,
  treat   = subset_data$has_vbg,
  ScoreType = "VBG"
)
) %>%
  mutate(
    treat   = factor(treat, levels = c(0,1), labels = c("No Test", "Test")),
    ScoreType = factor(ScoreType, levels = c("ABG","V р"))
)

ggplot(df_ps, aes(x = ps, fill = treat)) +
  geom_histogram(aes(y = ..density..), alpha = 0.5,
                 position = "identity", bins = 30) +
  scale_fill_manual(values = c("No Test" = "steelblue", "Test" = "tomato")) +
  facet_wrap(~ScoreType, scales = "free_y") +
  labs(
    title = "Propensity Score Distributions",
    x = "Propensity Score",
    y = "Density",
    fill = "Group"
  ) +
  theme_minimal(base_size = 12)

```

## Propensity Score Distributions



*Chunk propensity-histograms-all runtime: 0.52 s*

## 5 Multiple Imputation Analysis

added 12/6/2025

### 5.1 7) Packages and reproducibility

```
# Core MI + diagnostics
library(mice)      # chained equations (MICE)
```

Attaching package: 'mice'

The following object is masked from 'package:stats':

filter

The following objects are masked from 'package:base':

cbind, rbind

```
library(miceadds)    # pooling helpers & utilities
```

\* miceadds 3.18-36 (2025-09-12 09:54:54)

```
library(naniar)      # missingness summaries/plots
```

Attaching package: 'naniar'

The following object is masked from 'package:miceadds':

prop\_miss

```
library(visdat)      # quick type/missingness viz
library(skimr)       # data skim for large frames
```

Attaching package: 'skimr'

The following object is masked from 'package:naniar':

```
n_complete
```

```
# Modeling
library(WeightIt)    # GBM propensity with weights
library(gbm)          # underlying GBM engine
library(survey)        # svyglm outcome models
library(cobalt)        # balance diagnostics
library(broom)         # tidy model outputs
library(dplyr)         # data manipulation
library(ggplot2)

# Pooling and MI bookkeeping
library(mitoools)     # MIcombine for pooling (generic)
library(parallel)       # basic parallel where helpful

# Parallel + progress setup
library(future)
```

Attaching package: 'future'

The following object is masked from 'package:survival':

```
cluster
```

```

# setup
library(future.apply)
library(progressr)

workers <- max(1L, future::availableCores() - 1L)
future::plan(multisession, workers = workers)
on.exit(future::plan("sequential"), add = TRUE)

# choose a handler, but DO NOT make it global inside a knitted document
progressr::handlers(progressr::handler_rstudio)    # or handler_txtprogressbar
options(future.rng.onMisuse = "error")              # safer RNG with futures

set.seed(20251206)

# ensure a writable figure dir + stable device on macOS
if (!dir.exists("figs")) dir.create("figs", recursive = TRUE, showWarnings = FALSE)
knitr::opts_chunk$set(fig.path = "figs/", dev = "png", dpi = 144)
options(bitmapType = "cairo")  # prevents device issues on macOS

```

*Chunk mi-packages runtime: 1.55 s*

### 5.1.1 7.1 Missingness audit (what, where, how much)

```

# --- Lean missingness audit (memory-safe) -----
library(dplyr)
library(ggplot2)
library(naniar)

# 1) Tabular summary (cheap)
miss_tbl <- naniar::miss_var_summary(subset_data) %>% arrange(desc(pct_miss))
print(utils::head(miss_tbl, 40))  # show top 40 in the report

# A tibble: 40 x 3

```

```

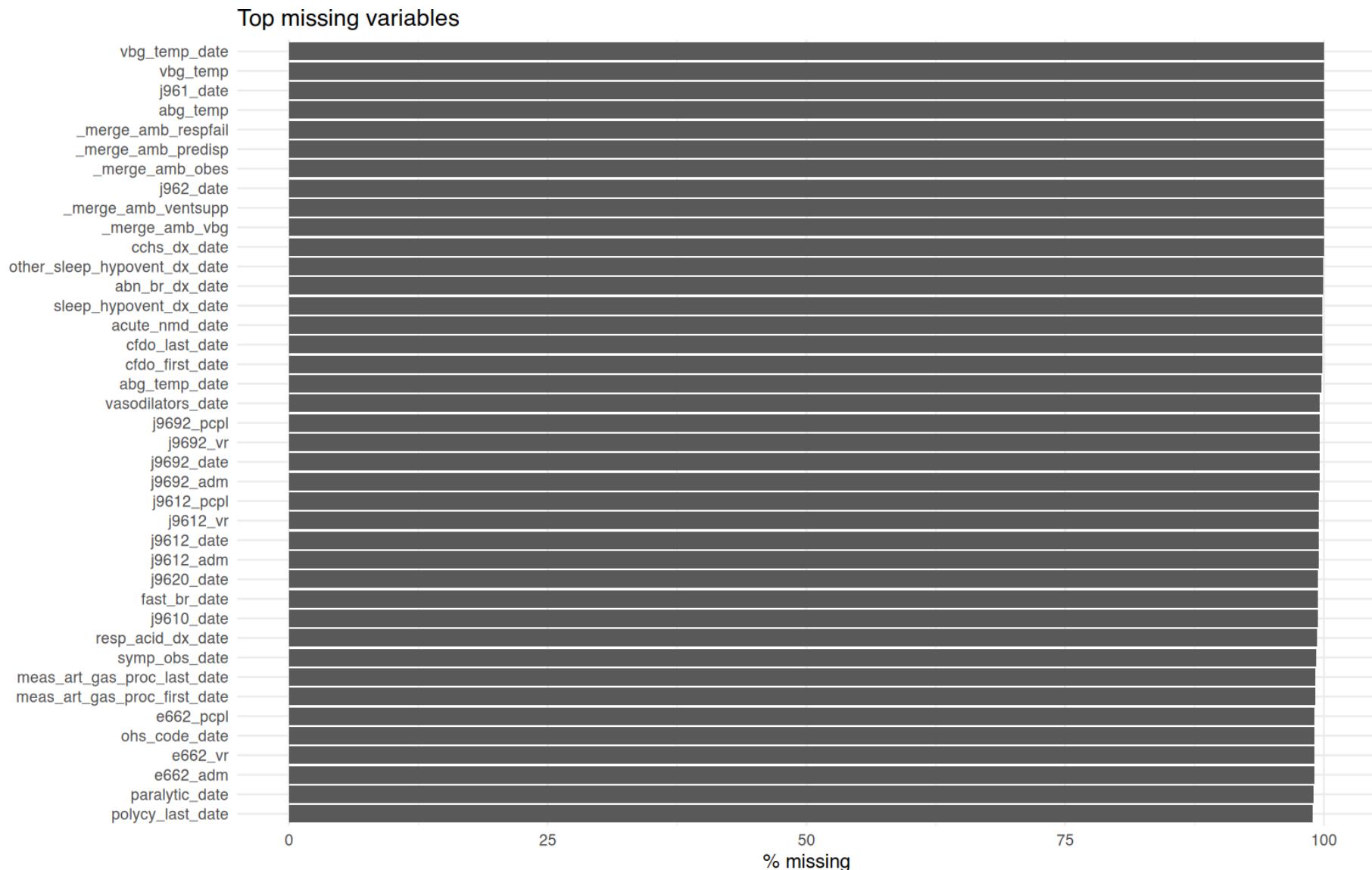
variable          n_miss pct_miss
<chr>            <int>   <num>
1 vbg_temp        515286 100
2 abg_temp        515286 100
3 vbg_temp_date  515286 100
4 j961_date       515286 100
5 _merge_amb_obes 515286 100
6 _merge_amb_predisp 515286 100
7 _merge_amb_respfail 515286 100
8 j962_date       515285 100.0
9 _merge_amb_vbg  515285 100.0
10 _merge_amb_ventsupp 515285 100.0
# i 30 more rows

```

```

# 2) Bar plot of top-K missing variables (avoid long-form of all columns)
K <- 40
top_vars <- miss_tbl$variable[seq_len(min(K, nrow(miss_tbl)))]
p_top <- ggplot(miss_tbl[miss_tbl$variable %in% top_vars, ],
                 aes(x = reorder(variable, pct_miss), y = pct_miss)) +
  geom_col() +
  coord_flip() +
  labs(title = "Top missing variables", x = NULL, y = "% missing") +
  theme_minimal()
print(p_top)

```



```
# 3) Small "type+NA heatmap" on a subsample (avoid full vis_dat on 500+ cols)
#   limit to first M columns with highest missingness and sample rows
M <- 60
R <- min(1500, nrow(subset_data))
```

```
cols_heat <- head(top_vars, M)
rows_heat <- dplyr::slice_sample(subset_data, n = R)

# Use naniar::vis_miss on a subset - much cheaper than vis_dat on all cols
p_heat <- naniar::vis_miss(rows_heat[, cols_heat, drop = FALSE]) +
  labs(title = sprintf("Missingness heatmap on %d rows x %d cols", R, length(cols_heat)))
print(p_heat)
```

Missingness heatmap on 1500 rows x 40 cols



```
# 4) Optional, but often heavy: UpSet of co-missingness - skip by default.  
# If you really want it, do it for top 6-10 variables only:  
# naniar::gg_miss_upset(subset_data[, head(top_vars, 8)], drop = FALSE)
```

*Chunk mi-missing-audit runtime: 3.44 s*

## 5.2 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.

```
# Ensure intended factor/numeric types for imputation
# --- Inspect current encounter_type -----
cat("encounter_type class:", paste(class(subset_data$encounter_type), collapse = ", "), "\n")
```

```
encounter_type class: factor
```

```
print(utils::head(unique(subset_data$encounter_type), 20))
```

```
[1] Emergency Inpatient
Levels: Emergency Inpatient
```

```
# Keep a raw copy for debugging if mapping fails
encounter_type_raw <- subset_data$encounter_type

# --- Helpers -----

# Map various encodings to strict 0/1 integer (for sex + 0/1 indicators)
to01 <- function(x) {
  if (is.logical(x)) return(as.integer(x))
  if (is.factor(x)) x <- as.character(x)

  out <- rep(NA_integer_, length(x))
  xs <- suppressWarnings(as.numeric(x))
  is_num <- !is.na(xs)

  # numeric 0/1
  out[is_num & xs %in% c(0, 1)] <- as.integer(xs[is_num & xs %in% c(0, 1)])
}
```

```

# character encodings (case/space-insensitive)
if (any(!is_num)) {
  s <- trimws(tolower(as.character(x[!is_num])))
  out[!is_num][s %in% c("0","no","false","female","f")] <- 0L
  out[!is_num][s %in% c("1","yes","true","male","m")] <- 1L
}
out
}

# Robust normalizer for encounter_type:
# - accepts numeric 2/3, digit-strings "2", "3", "2 - ED", etc.
# - accepts common synonyms with word-boundary protection
normalize_encounter_type <- function(x) {
  # to character once
  s_chr <- trimws(tolower(as.character(x)))

  # try to pull numeric code from any digits present
  num_from_text <- suppressWarnings(as.numeric(gsub("[^0-9]+", "", s_chr)))

  lab <- rep(NA_character_, length(s_chr))

  # numeric path
  lab[!is.na(num_from_text) & num_from_text == 2] <- "Emergency"
  lab[!is.na(num_from_text) & num_from_text == 3] <- "Inpatient"

  # synonym path (fill only still-NA)
  is_na <- is.na(lab)

  # Emergency synonyms: "emergency", "emerg", exact "ed", "a&e", "emergency dept"
  is_em <- grepl("\bemerg(?:ency)?\b", s_chr) |
    grepl("(^|[^a-z])ed([a-z]|$)", s_chr) |
    grepl("\ba&e\b", s_chr) |
    grepl("\bemergency\s+dept\b", s_chr)

  # Inpatient synonyms: "inpatient", "inpt", "inpat", exact "ip"
  is_ip <- grepl("\binpatient\b", s_chr) |

```

```

    grepl("\\binpt\\b", s_chr) |
    grepl("\\binpat\\b", s_chr) |
    grepl("(^|[^a-z])ip([a-z]|\$)", s_chr)

lab[is_na & is_em] <- "Emergency"
lab[is_na & is_ip] <- "Inpatient"

factor(lab, levels = c("Emergency", "Inpatient"))
}

# --- Coerce analysis types (including encounter_type) -----
subset_data <- subset_data |>
  mutate(
    sex                  = factor(to01(sex), levels = c(0L, 1L), labels = c("Female", "Male")),
    race_ethnicity       = if (is.factor(race_ethnicity)) race_ethnicity else factor(race_ethnicity),
    location             = if (is.factor(location))      location      else factor(location),
    encounter_type       = normalize_encounter_type(encounter_type),
    has_abg              = to01(has_abg),
    has_vbg              = to01(has_vbg),
    hypercap_on_abg     = to01(hypercap_on_abg),
    hypercap_on_vbg     = to01(hypercap_on_vbg)
  )

# immediately drop unused levels and assert exactly two levels in the observed data used by MI
subset_data$encounter_type <- droplevels(subset_data$encounter_type)
stopifnot(nlevels(subset_data$encounter_type) == 2L)

# --- Diagnostics for encounter_type -----
tab_enc <- table(subset_data$encounter_type, useNA = "ifany")
print(tab_enc)

```

Emergency Inpatient  
 171727 343559

```

if (sum(!is.na(subset_data$encounter_type)) == 0) {
  message("All encounter_type values are NA after normalization. Showing top raw values:")
  s_raw <- trimws(tolower(as.character(encounter_type_raw)))
  print(utils::head(sort(table(s_raw), decreasing = TRUE), 20))
  stop("normalize_encounter_type produced all NA; extend the synonym map to your raw values.")
}

# Must have at least one observed value and (after droplevels) exactly two levels
stopifnot(sum(!is.na(subset_data$encounter_type)) > 0)
stopifnot(nlevels(droplevels(subset_data$encounter_type)) == 2)

```

*Chunk mi-prep runtime: 2.76 s*

*Chunk type-invariants runtime: 0.24 s*

## 5.3 9) Imputation model specification (MICE)

### 5.3.1 9.1 Predictor matrix & methods. Run MICE (moderate settings for scale)

```

# --- variables for GBM propensity (kept identical to main analysis) ---
# ----- MICE setup (Option A: include PaCO2 for ABG + keep VBG CO2/O2Sat) -----
library(mice)
library(dplyr)

# --- add analysis targets and CO2 measures explicitly -----
co2_vars <- c("paco2", "vbg_co2", "vbg_o2sat")

mi_vars <- unique(c(
  covars_gbm,
  "has_abg", "has_vbg",                                # treatments (NOT imputed)
  "inv_proc", "niv_proc", "death_60d", "hypercap_resp_failure", # outcomes (NOT imputed)
  co2_vars
))

```

```

mi_df <- subset_data[, mi_vars, drop = FALSE]

# Make binary comorbid factors so "logreg" is used (and stays binary)
bin_covars <- c("copd", "asthma", "osa", "chf", "acute_nmd", "phtn", "ckd", "dm")
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"))
})

# Force CO2 variables to be numeric BEFORE we convert any leftover characters to factor
coerce_num <- function(x) suppressWarnings(as.numeric(as.character(x)))
for (nm in intersect(co2_vars, names(mi_df))) mi_df[[nm]] <- coerce_num(mi_df[[nm]])

# For MICE: convert any remaining characters → factors (after CO2 numeric coercion)
mi_df <- dplyr::mutate(mi_df, across(where(is.character), ~ factor(.x)))

# --- 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 or outcomes
no_imp <- c("has_abg", "has_vbg", "imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")
meth[intersect(names(meth), no_imp)] <- ""

# predictor matrix; forbid treatments/outcomes as predictors
pred <- mice::quickpred(mi_df, mincor = 0.05, minpuc = 0.25)

```

```

pred[, intersect(colnames(pred), no_imp)] <- 0
pred[intersect(rownames(pred), no_imp), ] <- 0

# MI integrity: treatments/outcomes excluded; binaries use logreg
stopifnot(all(meth[no_imp] == ""))
stopifnot(all(colSums(pred[, intersect(colnames(pred), no_imp), drop = FALSE]) == 0))
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))))
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 -----
set.seed(20251206)
imp <- mice::mice(
  data          = mi_df,
  m             = 5,
  maxit         = 10,
  predictorMatrix = pred,
  method        = meth,
  printFlag     = TRUE,
  seed          = 20251206
)

```

iter	imp	variable	curr_bmi	temp_new	sbp	dbp	hr	spo2	sodium	serum_cr	serum_hco3	serum_cl	serum_lac	serum_k	wbc	plt	bnp	seru
1	1																	
1	2																	
1	3																	
1	4																	



```

9  1 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
9  2 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
9  3 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
9  4 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
9  5 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
10 1 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
10 2 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
10 3 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
10 4 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum
10 5 curr_bmi temp_new sbp dbp hr spo2 sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp serum

```

```

saveRDS(imp, file = "mi_abg_vbg_mids.rds")

# quick sanity: these must exist and be numeric in completed data
dlist <- mice:::complete(imp, action = "all")
stopifnot(all(c("paco2","vbg_co2") %in% names(dlist[[1]])))
stopifnot(is.numeric(dlist[[1]]$paco2), is.numeric(dlist[[1]]$vbg_co2))

```

*Chunk mi-exec runtime: 2806.18 s*

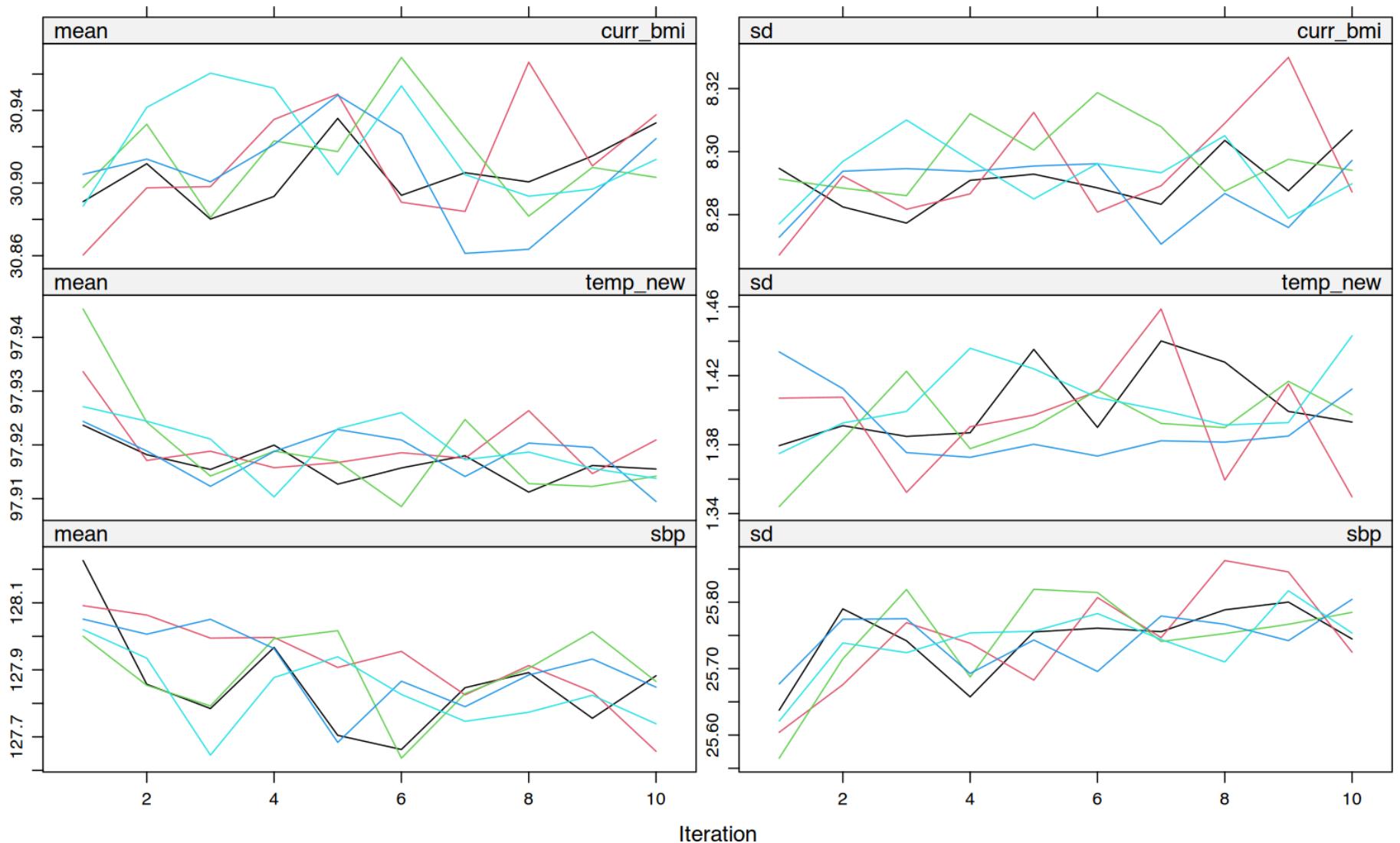
### 5.3.2 9.2 Convergence & plausibility checks

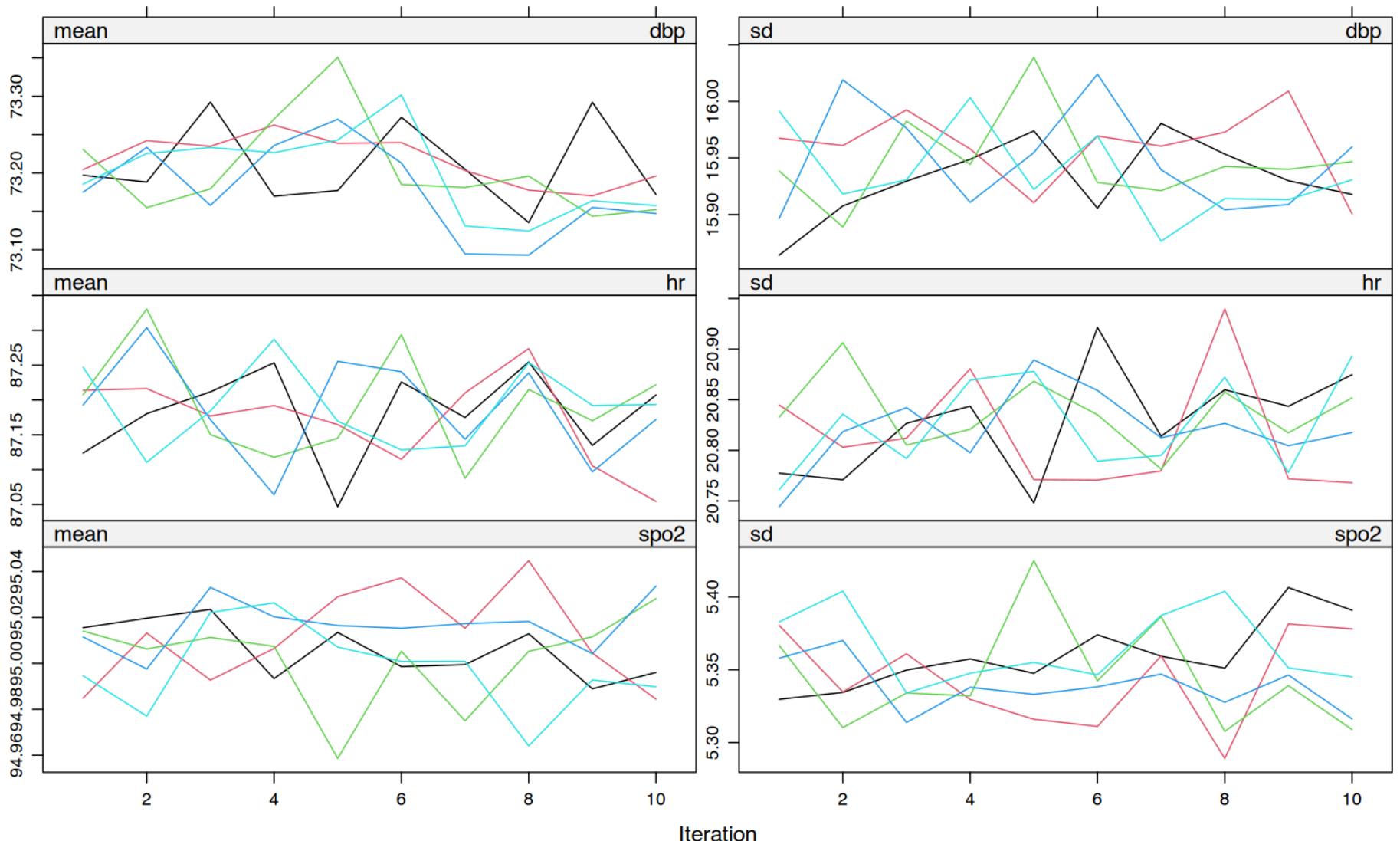
```

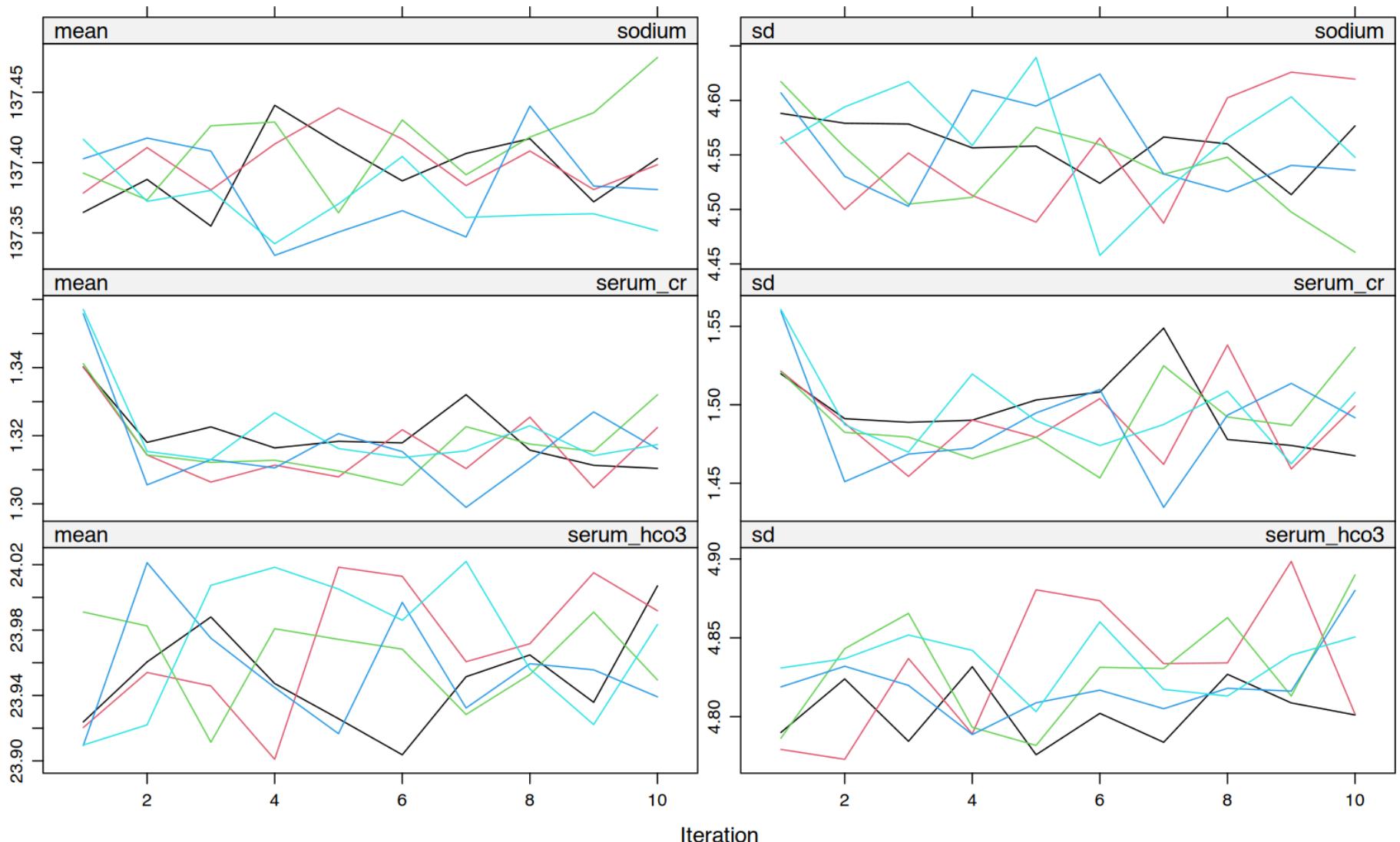
imp <- readRDS("mi_abg_vbg_mids.rds")

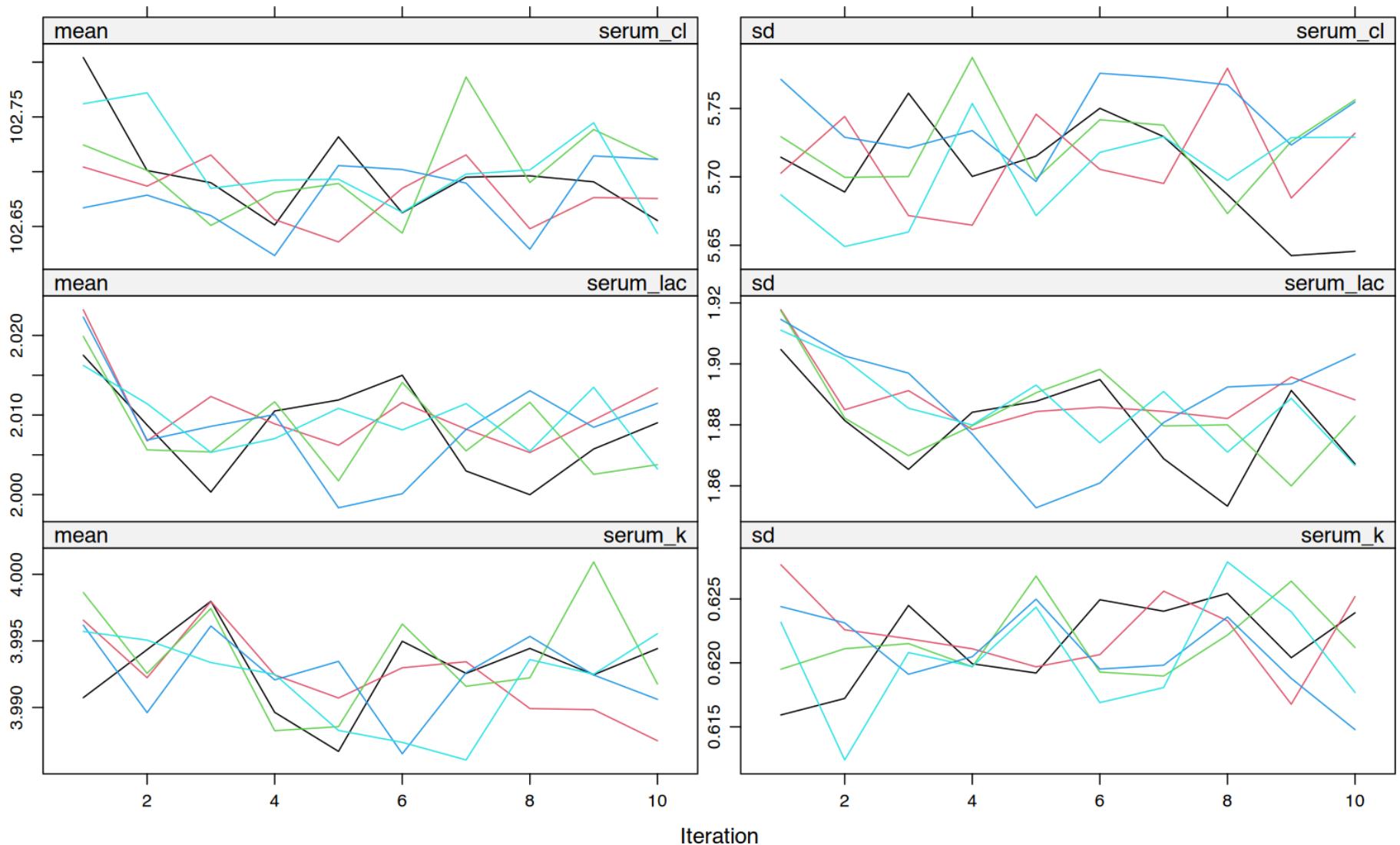
# Throttle diagnostics to avoid memory blow-up
plot(imp)                                # trace plots (m=5)

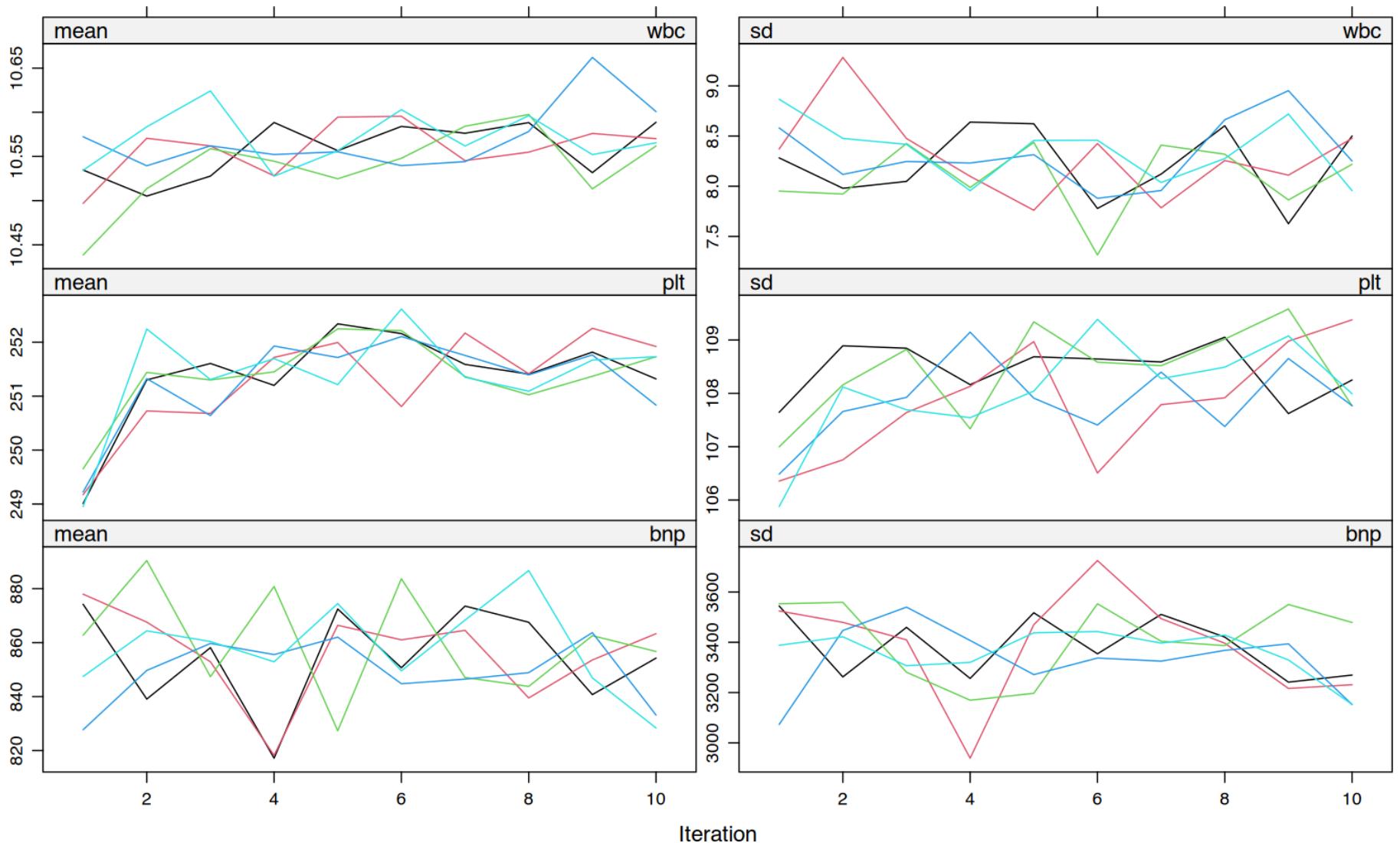
```

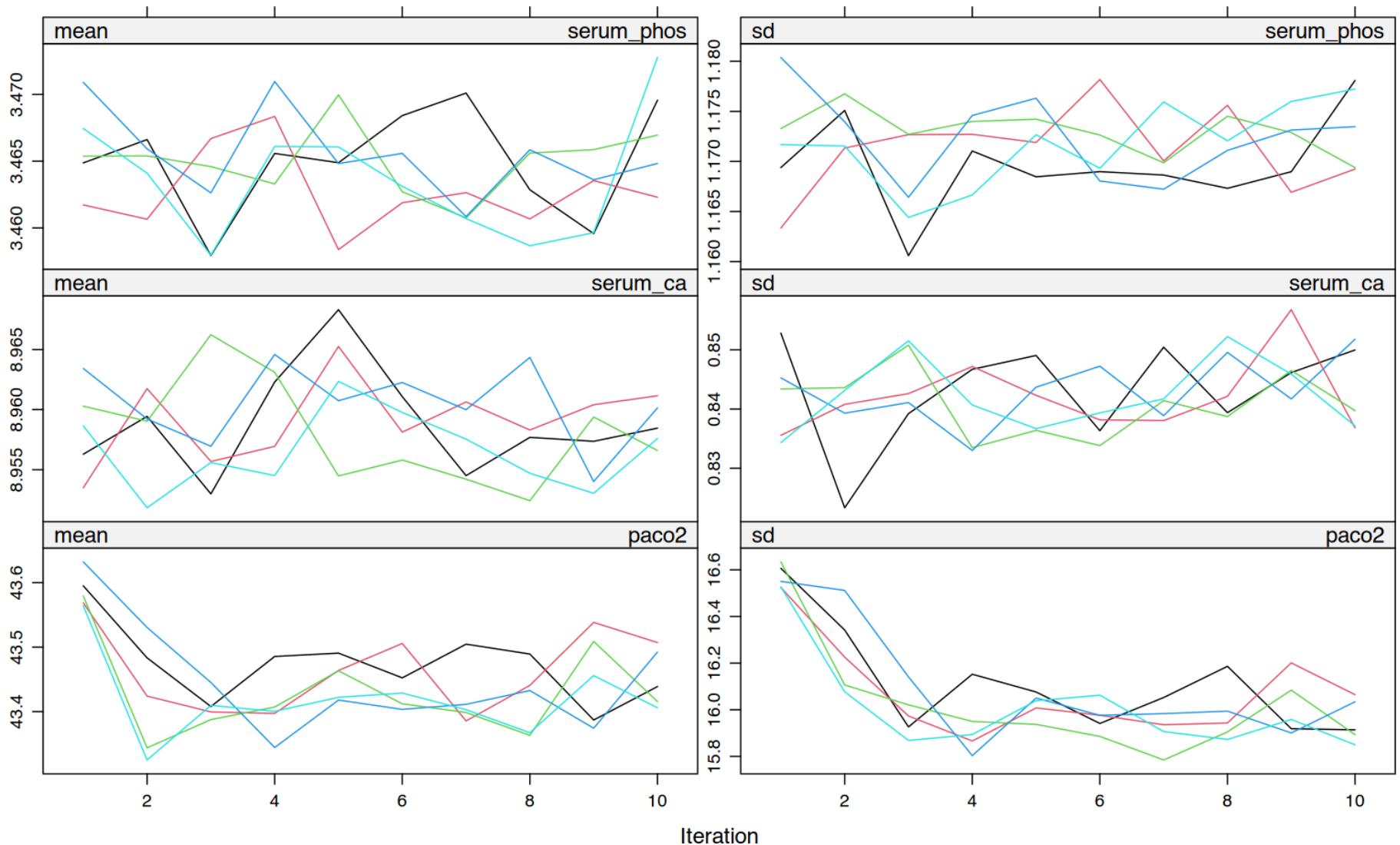


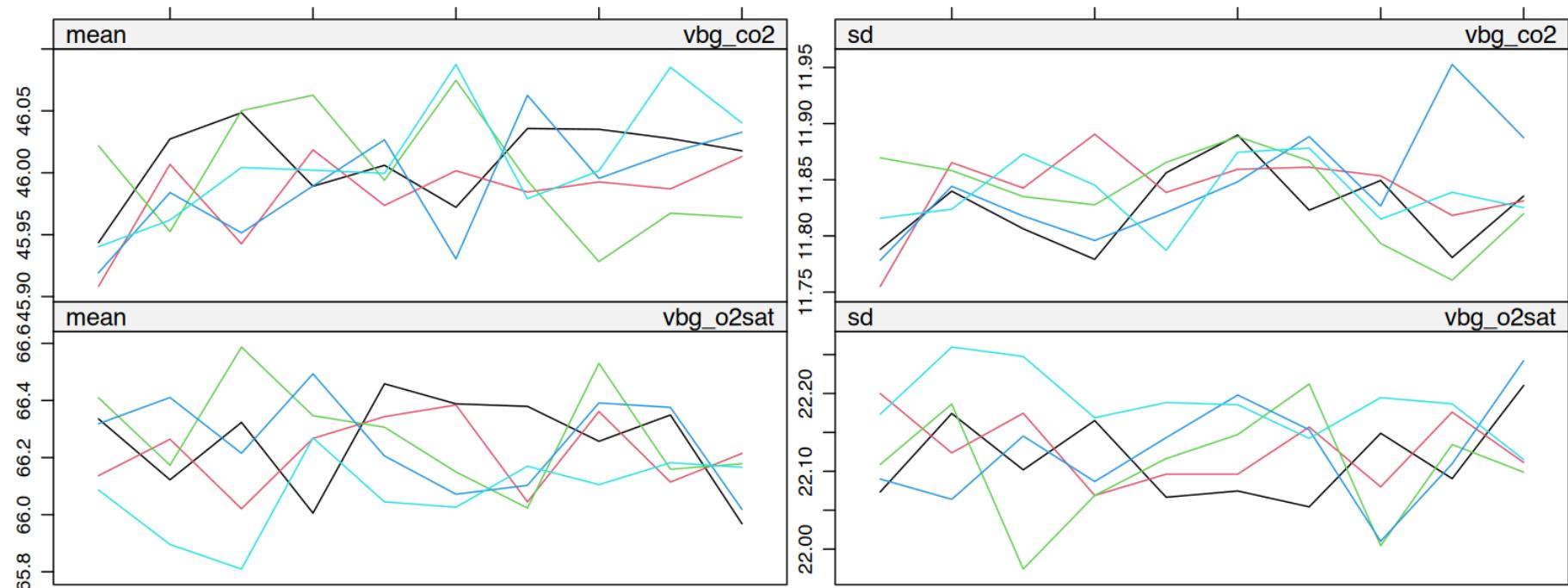






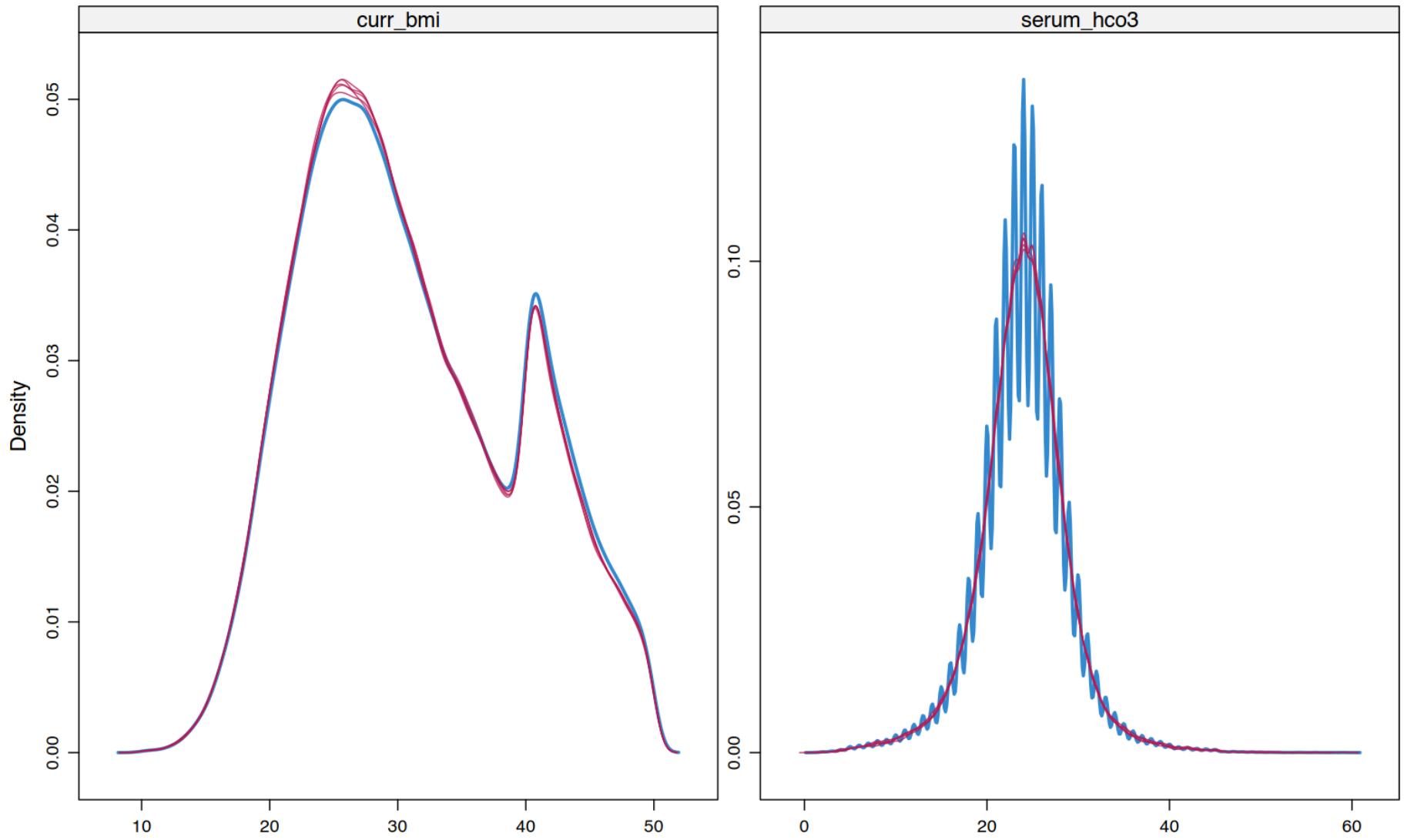




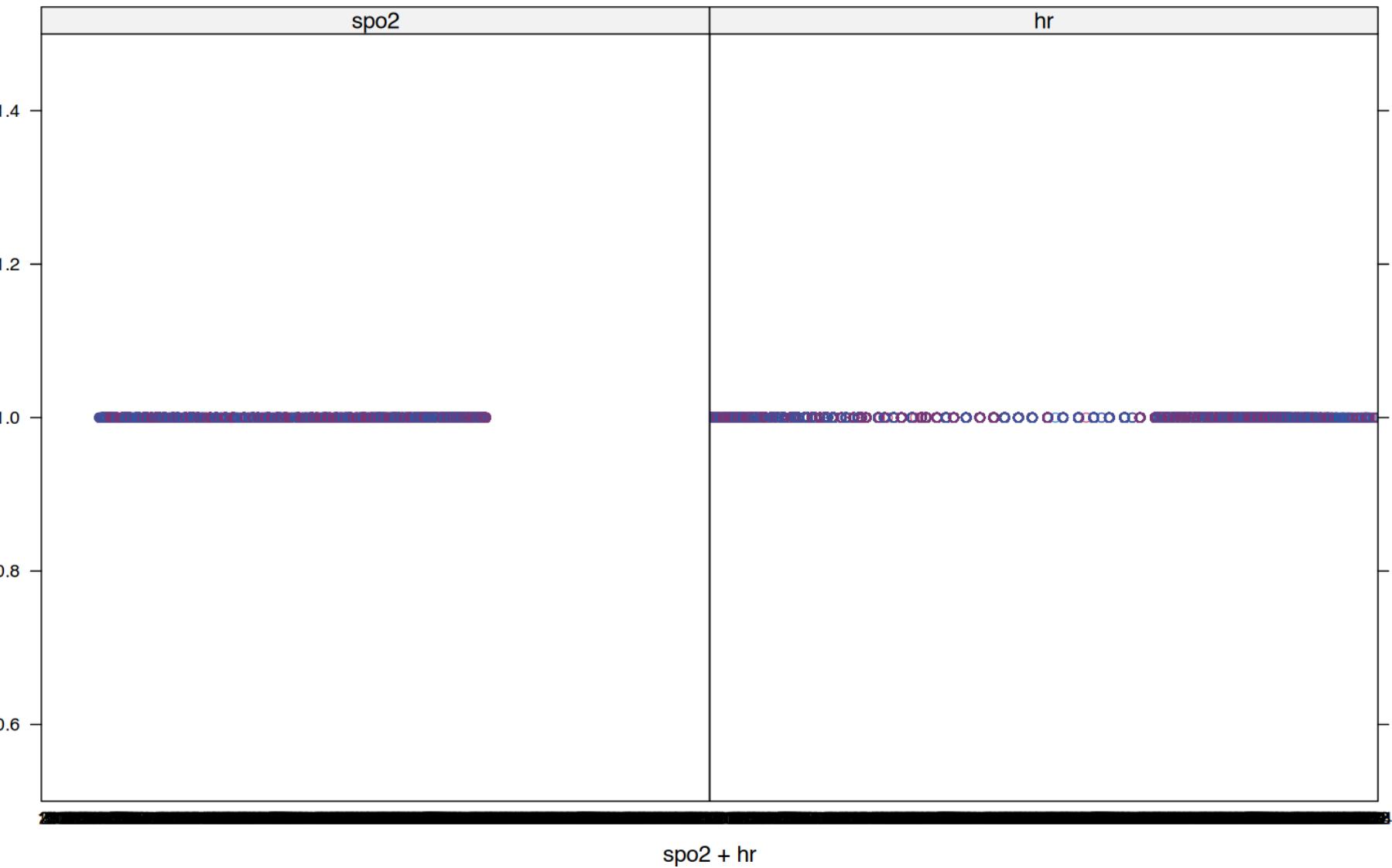


Iteration

```
densityplot(impt, ~ curr_bmi + serum_hco3) # imputed vs observed
```



```
stripplot(imp, ~ spo2 + hr, subset = .imp == 1)      # distribution overlap on first imp only
```



```
md.pattern(complete(imp, "long", include = FALSE)) # pattern after MI (compact)
```

/\ ^  
{ `---' }

```

{ 0 0 }
==> V <== No need for mice. This data set is completely observed.
\ \ | / /
`-----'

age_at_encounter sex_race_ethnicity curr_bmi copd asthma osa chf
2576430          1   1           1   1   1   1   1
                  0   0           0   0   0   0   0
acute_nmd phtn ckd dm location encounter_type temp_new sbp dbp hr spo2
2576430          1   1   1   1           1   1   1   1   1
                  0   0   0   0           0   0   0   0   0
sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp
2576430          1   1           1   1   1   1   1   1
                  0   0           0   0   0   0   0   0
serum_phos serum_ca has_abg has_vbg imv_proc niv_proc death_60d
2576430          1   1           1   1   1   1
                  0   0           0   0   0   0
hypercap_resp_failure paco2 vbg_co2 vbg_o2sat .imp .id
2576430          1   1           1   1   1   0
                  0   0           0   0   0   0

```

*Chunk mi-diagnostics runtime: 72.17 s*

## 5.4 10) Refit propensity models within each imputation

We keep the GBM recipe close to the non-MI run, but with lighter CV (cv.folds = 3) and slightly fewer trees.

### 5.4.1 10.1 ABG propensity (has\_abg)

```

# Create completed datasets
dlist <- mice::complete(imp, action = "all")

# Fit ABG propensity weights in each imputation
fit_abg_one <- function(d) {
  w <- weightit(
    formula_abg,
    data   = d[, c("has_abg", covars_gbm)],
    method = "gbm",
    estimand = "ATE",
    include.obj = TRUE,
    n.trees      = gbm_params$n.trees,
    interaction.depth = gbm_params$interaction.depth,
    shrinkage     = gbm_params$shrinkage,
    bag.fraction  = gbm_params$bag.fraction,
    cv.folds       = gbm_params$cv.folds,
    stop.method    = gbm_params$stop.method,
    n.cores        = gbm_params$n.cores
  )
  # stabilise + two-sided Winsorization
  ww <- w$weights; ww <- ww/mean(ww)
  cut <- stats::quantile(ww, c(.01,.99), na.rm = TRUE)
  ww <- pmin(pmax(ww, cut[1]), cut[2]); ww <- ww/mean(ww)
  w$weights <- ww
  w
}

with_progress({
  p <- progressor(along = seq_along(dlist))

  # Avoid hitting future.globals.maxSize with large dlist
  options(future.globals.maxSize = +Inf)
  if (utils::object.size(dlist) > 4e8) future::plan(sequential)

  W_abg_list <- future_lapply(

```

```

X = seq_along(dlist),
FUN = function(i) {
  p(sprintf("Fitting ABG on imputation %d", i))
  set.seed(20251206 + i)           # per-imputation seed for reproducibility
  fit_abg_one(dlist[[i]])
},
future.seed = TRUE                 # reproducible RNG across workers
)
})

saveRDS(W_abg_list, "mi_W_abg_list.rds")

```

*Chunk mi-propensity-abg runtime: 5029.90 s*

	n	min	p99.99%	max	ess
[1,]	515286	0.565	3.343	3.343	420233.2
[2,]	515286	0.565	3.353	3.353	419905.9
[3,]	515286	0.565	3.350	3.350	420423.3
[4,]	515286	0.565	3.359	3.359	419865.0
[5,]	515286	0.565	3.349	3.349	420180.0

*Chunk mi-weight-diagnostics-abg runtime: 4.41 s*

#### 5.4.2 10.2 Balance diagnostics across imputations

```

# Vars you intended to use (from your earlier code)
vars0 <- covars_gbm

# Which factors collapse to 1 level AFTER complete-case filtering (per imputation, per arm)?
find_offenders_post_cc <- function(d, treat_var, vars) {
  keep <- c(treat_var, vars)
  dd   <- d[, keep, drop = FALSE]
  dd   <- dd[stats::complete.cases(dd), , drop = FALSE]  # mimic cobalt's CC
}

```

```

if (!nrow(dd)) return(character(0))

# factor with <2 levels in either arm
bad <- vapply(vars, function(v) {
  x <- dd[[v]]
  if (!is.factor(x)) return(FALSE)
  by_arm <- tapply(x, dd[[treat_var]], function(z) nlevels(droplevels(z)))
  any(is.na(by_arm)) || any(by_arm < 2)
}, logical(1))

names(bad)[bad]
}

off_by_imp <- lapply(dlist, find_offenders_post_cc, treat_var = "has_abg", vars = vars0)
to_drop    <- Reduce(union, off_by_imp) # union across imputations
message("Offenders (post CC): ", if (length(to_drop)) paste(to_drop, collapse = ", ") else "<none>")

```

Offenders (post CC): <none>

```

# Keep only variables that never collapse post-CC
vars_keep2 <- setdiff(vars0, to_drop)
stopifnot(length(vars_keep2) > 0)

```

*Chunk unnamed-chunk-1 runtime: 111.67 s*

```

# Build a variable set that has 2 levels in *every* imputation (prevents contrasts errors)
vary_ok <- function(z) {
  nz <- z[!is.na(z)]
  if (is.factor(nz)) nlevels(droplevels(nz)) > 1 else dplyr::n_distinct(nz) > 1
}
vars_keep <- Reduce(intersect, lapply(dlist, function(d) {
  keep <- vapply(d[, covars_gbm, drop = FALSE], vary_ok, logical(1))
  names(keep)[keep]
})))

```

```

# Long data for cobalt with weights and imputation id
make_long_for_cobalt <- function(dlist, W_list, treat_var, covars) {
  stopifnot(length(dlist) == length(W_list))
  do.call(rbind, lapply(seq_along(dlist), function(i) {
    di <- dlist[[i]][, c(treat_var, covars), drop = FALSE]
    di$.imp <- i
    di$.w   <- W_list[[i]]$weights
    di
  }))
}
dlong_abg <- make_long_for_cobalt(dlist, W_abg_list, "has_abg", vars_keep)

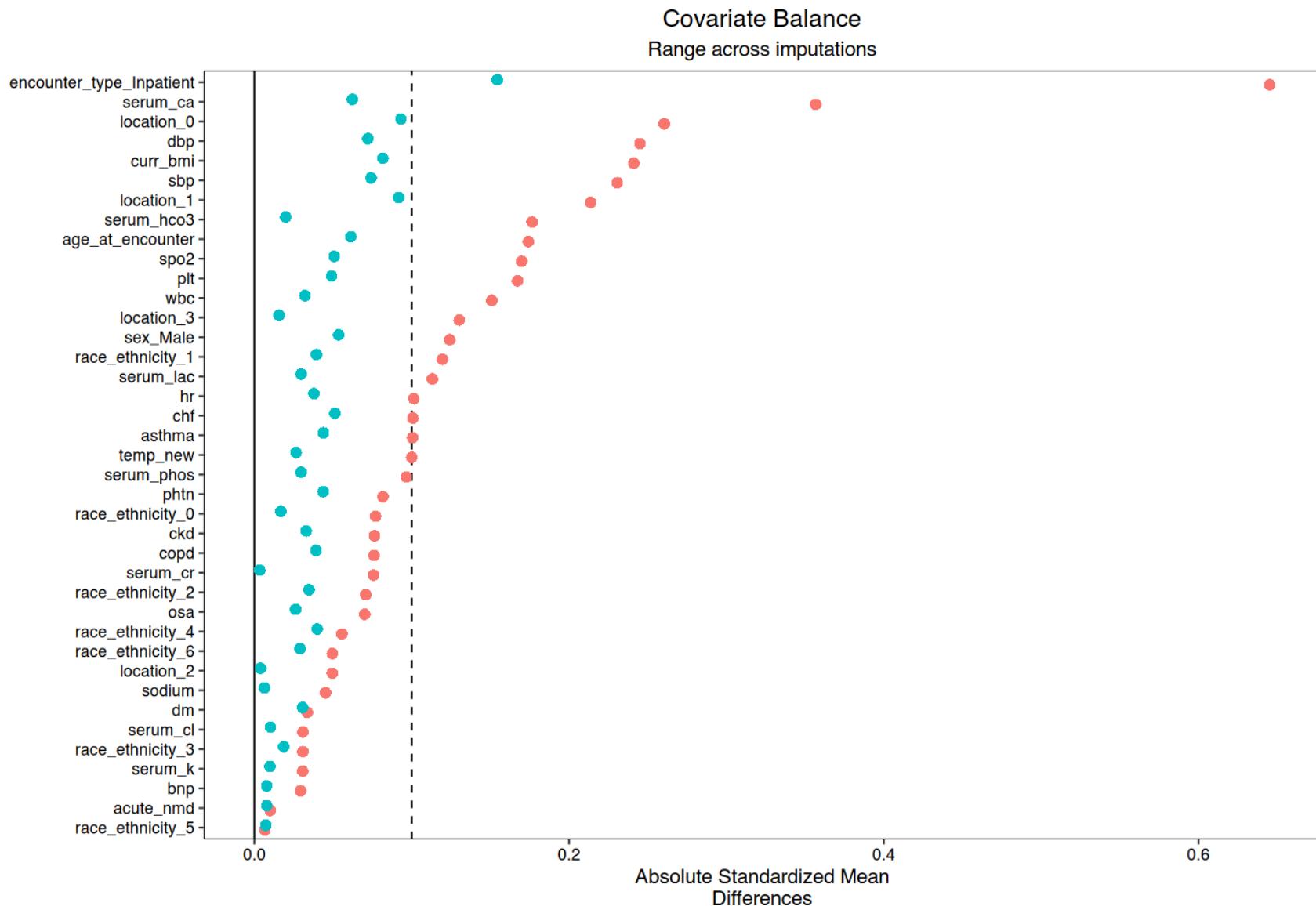
# removes empty levels introduced by the per-imputation slicing and prevents spurious contrast errors.
dlong_abg <- droplevels(dlong_abg)

# Final guard: drop any factor that is 1-level in the long frame (should be none after vars_keep)
one_level_factors <- names(Filter(function(x) is.factor(x) && nlevels(droplevels(x)) < 2,
                                      dlong_abg[vars_keep]))
if (length(one_level_factors)) {
  message("Dropping 1-level factors in long data: ", paste(one_level_factors, collapse = ", "))
  vars_keep <- setdiff(vars_keep, one_level_factors)
}

# Balance with imputation identifiers
fml_abg <- reformulate(termlabels = vars_keep, response = "has_abg")
bal_abg <- cobalt::bal.tab(
  fml_abg,
  data      = dlong_abg,
  weights   = dlong_abg$.w,
  imp       = dlong_abg$.imp,      # vector of imputation IDs
  estimand  = "ATE",
  un        = TRUE,
  m.threshold = 0.1,
  binary    = "std",
  s.d.denom = "pooled"
)

```

```
# Optional plot
cobalt::love.plot(
  bal_abg,
  var.order      = "unadjusted",
  thresholds    = c(m = .1),
  sample.names  = c("Raw", "IPW"),
  abs            = TRUE
)
```



Chunk mi-balance-abg runtime: 44.57 s

### 5.4.3 10.3 VBG propensity (has\_vbg)

```
fit_vbg_one <- function(d) {
  w <- weightit(
    formula_vbg,
    data    = d[, c("has_vbg", covars_gbm)],
    method  = "gbm",
    estimand = "ATE",
    include.obj = TRUE,
    n.trees       = gbm_params$n.trees,
    interaction.depth = gbm_params$interaction.depth,
    shrinkage     = gbm_params$shrinkage,
    bag.fraction   = gbm_params$bag.fraction,
    cv.folds       = gbm_params$cv.folds,
    stop.method    = gbm_params$stop.method,
    n.cores        = gbm_params$n.cores
  )
  ww <- w$weights; ww <- ww/mean(ww); cut <- stats::quantile(ww, c(.01,.99), na.rm=TRUE)
  ww <- pmin(pmax(ww, cut[1]), cut[2]); ww <- ww/mean(ww)
  w$weights <- ww
  w
}

with_progress({
  p <- progressor(along = seq_along(dlist))
  options(future.globals.maxSize = +Inf)
  if (utils::object.size(dlist) > 4e8) future::plan(sequential)

  W_vbg_list <- future_lapply(
    X = seq_along(dlist),
    FUN = function(i) {
      p(sprintf("Fitting VBG on imputation %d", i))
      set.seed(30251206 + i)
      fit_vbg_one(dlist[[i]])
    },
  )
})
```

```

    future.seed = TRUE
  )
})

saveRDS(W_vbg_list, "mi_W_vbg_list.rds")

```

*Chunk mi-propensity-vbg runtime: 10811.26 s*

	n	min	p99.99%	max	ess
[1,]	515286	0.573	3.802	3.802	360068.4
[2,]	515286	0.572	3.811	3.811	359415.2
[3,]	515286	0.572	3.806	3.806	360004.1
[4,]	515286	0.572	3.815	3.815	359809.4
[5,]	515286	0.572	3.828	3.828	359000.1

*Chunk mi-weight-diagnostics-vbg runtime: 0.07 s*

#### 5.4.4 10.4 VBG balance

```

# --- VBG: balance across imputations (fast per-imputation SMD pooling) -----

stopifnot(length(dlist) == length(W_vbg_list))

vary_ok <- function(z) {
  nz <- z[!is.na(z)]
  if (is.factor(nz)) nlevels(droplevels(nz)) > 1 else dplyr::n_distinct(nz) > 1
}

# 1) Keep only covariates that vary (2 levels for factors) in every imputation
vars_keep_vbg <- Reduce(intersect, lapply(dlist, function(d) {
  keep <- vapply(d[, covars_gbm, drop = FALSE], vary_ok, logical(1))
  names(keep)[keep]
})))

```

```

# 2) Build per-imputation lists; align weights and drop non-finite rows
X_list_vbg <- vector("list", length(dlist))
t_list_vbg <- vector("list", length(dlist))
w_list_vbg <- vector("list", length(dlist))
for (i in seq_along(dlist)) {
  di <- dlist[[i]][, vars_keep_vbg, drop = FALSE]
  ids <- rownames(di)
  ti <- dlist[[i]][["has_vbg"]]
  wi <- W_vbg_list[[i]]$weights
  if (!is.null(names(wi))) {
    wi <- wi[ids]
  } else if (length(wi) != length(ids)) {
    stop("Length mismatch weights vs data in VBG imp ", i)
  }
  keep <- is.finite(wi)
  if (!all(keep)) {
    di <- di[keep, , drop = FALSE]
    ti <- ti[keep]
    wi <- wi[keep]
  }
  X_list_vbg[[i]] <- di
  t_list_vbg[[i]] <- ti
  w_list_vbg[[i]] <- wi
}

# 3) Per-imputation SMDs (mean diffs only; quick path)
one_imp_bal_vbg <- function(i) {
  b <- cobalt::bal.tab(
    x           = X_list_vbg[[i]],
    treat       = t_list_vbg[[i]],
    weights     = w_list_vbg[[i]],
    stats       = "m",          # mean diffs only
    s.d.denom   = "pooled",
    quick       = TRUE,
    int         = FALSE,

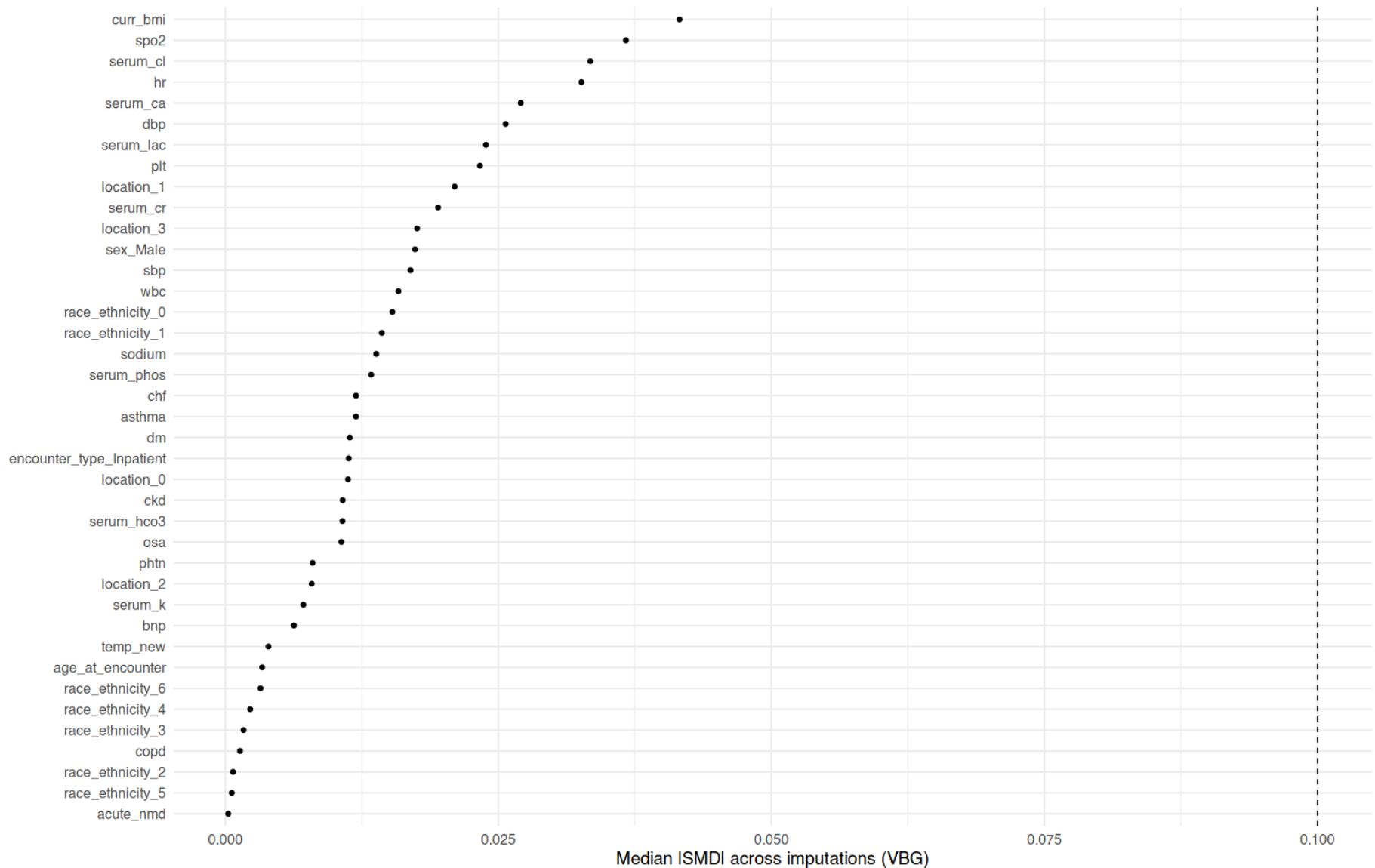
```

```

    disp.v.ratio = FALSE,
    disp.ks      = FALSE,
    un          = TRUE
)
S <- as.data.frame(b$Balance)
setNames(S[["Diff.Adj"]], rownames(S))
}

smd_list_vbg <- lapply(seq_along(dlist), one_imp_bal_vbg)
all_rows_vbg <- Reduce(union, lapply(smd_list_vbg, names))
SMD_mat_vbg  <- do.call(cbind, lapply(smd_list_vbg, function(v) v[match(all_rows_vbg, names(v))]))
smd_abs_vbg  <- abs(SMD_mat_vbg)
smd_pooled_vbg <- data.frame(
  covariate = all_rows_vbg,
  smd_med   = apply(smd_abs_vbg, 1, median, na.rm = TRUE),
  smd_mean  = rowMeans(smd_abs_vbg, na.rm = TRUE),
  smd_max   = apply(smd_abs_vbg, 1, max,     na.rm = TRUE),
  stringsAsFactors = FALSE
)
# Optional quick plot of pooled median |SMD|
ggplot(smd_pooled_vbg, aes(x = reorder(covariate, smd_med), y = smd_med)) +
  geom_hline(yintercept = 0.1, linetype = 2, linewidth = 0.3) +
  geom_point(size = 1) +
  coord_flip() +
  labs(x = NULL, y = "Median |SMD| across imputations (VBG)") +
  theme_minimal(base_size = 10)

```



```
message("VBG balance pooled from per-imputation SMDs; Love plot omitted to save memory.")
```

VBG balance pooled from per-imputation SMDs; Love plot omitted to save memory.

Chunk mi-balance-vbg runtime: 34.42 s

## 5.5 11) Weighted outcome models within each imputation + pooling

Pool via Rubin's rules using mitools::MIcombine. Extract the coefficient and its variance for the treatment effect from each imputation.

### 5.5.1 11.1 Helper: fit + extract log-OR and SE from svyglm

```
# --- Robust coefficient extraction from svyglm (handles factor-coded terms) ---
coef_var_from_svy <- function(fit, treat_name) {
  cn <- names(coef(fit))
  idx <- match(treat_name, cn)
  if (is.na(idx)) {
    # handle factor encodings like has_abg1, has_abgYes, etc.
    pat <- paste0("^", gsub("[\\W]", "\\\\$1", treat_name), "(1|Yes|TRUE|Male)?$")
    idx <- grep(pat, cn, perl = TRUE)[1]
  }
  if (is.na(idx)) stop("Treatment coefficient '", treat_name, "' not found in model.")
  b <- unname(coef(fit)[idx])
  V <- unname(vcov(fit)[idx, idx, drop = TRUE])
  if (!is.finite(b) || !is.finite(V)) stop("Non-finite estimate/variance.")
  list(b = b, V = V)
}

# --- Fit one weighted GLM on one completed dataset and extract log-OR + var ---
fit_and_extract <- function(data, weights, formula, treat_name) {
  stopifnot(length(weights) == nrow(data))
  # basic guards: variation in outcome and treatment
  yname <- all.vars(formula)[1L]
  if (dplyr::n_distinct(na.omit(data[[yname]])) < 2L) stop("Outcome '", yname, "' has one level.")
  if (dplyr::n_distinct(na.omit(data[[treat_name]])) < 2L) stop("Treatment '", treat_name, "' has one level.")
  # design + fit
  data$w <- as.numeric(weights)
  des <- survey::svydesign(ids = ~1, weights = ~w, data = data)
```

```

fit <- survey::svyglm(formula, design = des, family = quasibinomial())
cv  <- coef_var_from_svy(fit, treat_name)
list(coef = cv$b, vcov = cv$V)
}

# --- Pool across imputations; tolerate failures; report how many used -----
pool_logOR <- function(est_list, term) {
  ok <- vapply(est_list, function(x) is.list(x) && is.finite(x$coef) && is.finite(x$vcov), logical(1))
  est_list <- est_list[ok]
  m_ok  <- length(est_list); m_tot <- length(ok)
  if (m_ok == 0L) {
    return(data.frame(term = term, logOR = NA_real_, SE = NA_real_, OR = NA_real_,
                      LCL = NA_real_, UCL = NA_real_, m_used = 0L, m_total = m_tot))
  }
  results  <- lapply(est_list, function(x) setNames(c(x$coef), term))
  variances <- lapply(est_list, function(x) { M <- matrix(x$vcov, 1, 1); dimnames(M) <- list(term, term); M })
  pooled <- mitools::MIcombine(results = results, variances = variances)
  est <- as.numeric(coef(pooled))
  se  <- sqrt(diag(pooled$variance))
  data.frame(term = term, logOR = est, SE = se, OR = exp(est),
             LCL = exp(est - 1.96 * se), UCL = exp(est + 1.96 * se),
             m_used = m_ok, m_total = m_tot, row.names = NULL)
}

```

*Chunk mi-pool-helpers runtime: 0.00 s*

### 5.5.2 11.2 ABG: outcomes = IMV, NIV, Death(60d), Hypercapnic RF

```

# Parallel ABG outcome fits and pooling across imputations
library(future.apply)
library(progressr)

# Inputs assumed present: imp, W_abg_list
stopifnot(exists("imp"), exists("W_abg_list"))

```

```

dlist <- mice::complete(imp, action = "all")
stopifnot(length(W_abg_list) == length(dlist))

outcomes_abg <- list(
  imv_proc = imv_proc ~ has_abg,
  niv_proc = niv_proc ~ has_abg,
  death     = death_60d ~ has_abg,
  hcrf      = hypercap_resp_failure ~ has_abg
)

old_plan <- future::plan()
workers  <- max(1L, as.integer(future::availableCores() - 1L))
future::plan(multisession, workers = workers)
on.exit(future::plan(old_plan), add = TRUE)

abg_results <- NULL
progressr::with_progress({
  p <- progressr::progressor(steps = length(outcomes_abg) * length(dlist))
  abg_results <- lapply(names(outcomes_abg), function(nm) {
    fml <- outcomes_abg[[nm]]
    ests <- future.apply::future_lapply(
      X = seq_along(dlist),
      FUN = function(i) {
        p(sprintf("ABG: %s (imp %d)", nm, i))
        tryCatch(
          fit_and_extract(dlist[[i]], W_abg_list[[i]]$weights, fml, "has_abg"),
          error = function(e) NULL
        )
      },
      future.seed = TRUE
    )
    out <- pool_logOR(ests, term = "has_abg")
    out$outcome <- nm
    out
  })
})

```

```
abg_results <- dplyr::bind_rows(abg_results)[, c("outcome", "term", "logOR", "SE", "OR", "LCL", "UCL", "m_used", "m_total")]
abg_results
```

outcome	term	logOR	SE	OR	LCL	UCL	m_used	m_total
imv_proc	has_abg	1.9608779	0.0124707	7.105562	6.933990	7.281380	5	5
niv_proc	has_abg	0.6457106	0.0121964	1.907342	1.862288	1.953486	5	5
death	has_abg	0.7878883	0.0101760	2.198748	2.155329	2.243043	5	5
hcrf	has_abg	1.2395945	0.0140507	3.454213	3.360384	3.550661	5	5

Chunk mi-abg-outcomes runtime: 35.37 s

### 5.5.3 11.3 Repeat for VBG

```
# --- VBG outcomes -----
# Parallel VBG outcome fits and pooling across imputations
library(future.apply)
library(progressr)

# Inputs assumed present: imp, W_vbg_list
stopifnot(exists("imp"), exists("W_vbg_list"))
dlist <- mice::complete(imp, action = "all")
stopifnot(length(W_vbg_list) == length(dlist))

outcomes_vbg <- list(
  imv_proc = imv_proc ~ has_vbg,
  niv_proc = niv_proc ~ has_vbg,
  death    = death_60d ~ has_vbg,
  hcrf     = hypercap_resp_failure ~ has_vbg
)

old_plan <- future::plan()
workers <- max(1L, as.integer(future::availableCores() - 1L))
```

```

future::plan(multisession, workers = workers)
on.exit(future::plan(old_plan), add = TRUE)

vbg_results <- NULL
progressr::with_progress({
  p <- progressr::progressor(steps = length(outcomes_vbg) * length(dlist))
  vbg_results <- lapply(names(outcomes_vbg), function(nm) {
    fml <- outcomes_vbg[[nm]]
    ests <- future.apply::future_lapply(
      X = seq_along(dlist),
      FUN = function(i) {
        p(sprintf("VBG: %s (imp %d)", nm, i))
        tryCatch(
          fit_and_extract(dlist[[i]], W_vbg_list[[i]]$weights, fml, "has_vbg"),
          error = function(e) NULL
        )
      },
      future.seed = TRUE
    )
    out <- pool_logOR(ests, term = "has_vbg")
    out$outcome <- nm
    out
  })
})
vbg_results <- dplyr::bind_rows(vbg_results)[, c("outcome", "term", "logOR", "SE", "OR", "LCL", "UCL", "m_used", "m_total")]
vbg_results

```

outcome	term	logOR	SE	OR	LCL	UCL	m_used	m_total
imv_proc	has_vbg	0.4850977	0.0100369	1.624334	1.592692	1.656605	5	5
niv_proc	has_vbg	0.3728752	0.0130719	1.451903	1.415177	1.489583	5	5
death	has_vbg	0.2730692	0.0104904	1.313991	1.287250	1.341288	5	5
hcrf	has_vbg	0.7277423	0.0132420	2.070401	2.017356	2.124840	5	5

Chunk mi-vbg-outcomes runtime: 34.73 s

## 5.6 12) Explainability on one representative imputation

To manage runtime, compute SHAP/PDP/ALE on the first imputed dataset and its fitted GBM(s).

*Chunk shap-axis-labels runtime: 0.00 s*

```
# --- Fast SHAP for WeightIt GBM fits (works with MI) -----
library(fastshap)
```

Attaching package: 'fastshap'

The following object is masked from 'package:dplyr':

explain

```
library(shapviz)
# --- Robust SHAP backend for WeightIt GBM fits -----
# Works whether gbm$var.names are raw feature names (factors ok) or one-hot names
# like "sexFemale", "race_ethnicity3", "encounter_typeInpatient", etc.

# Map of factor levels observed at training time (for raw-factor path)
.train_levels <- function(df) {
  f <- vapply(df, is.factor, logical(1))
  lapply(df[f], levels)
}

# Align factors in 'df' to a stored levels map (keeps order, avoids new/ambiguous levels)
.align_to_levels <- function(df, levels_map) {
  out <- as.data.frame(df, stringsAsFactors = FALSE)
  for (nm in intersect(names(levels_map), names(out))) {
    out[[nm]] <- factor(as.character(out[[nm]]), levels = levels_map[[nm]])
  }
  out
}
```

```

# Build a design with columns EXACTLY equal to 'varnames' by deriving indicators
# from the raw covariate frame 'X_raw'. This covers one-hot names like "sexMale",
# "race_ethnicity2", "encounter_typeInpatient", etc.
.design_from_varnames <- function(X_raw, varnames) {
  X_raw <- as.data.frame(X_raw, stringsAsFactors = FALSE)

  # normalize odd classes; turn characters into factors (stable level strings)
  for (nm in names(X_raw)) {
    if (inherits(X_raw[[nm]], "haven_labelled")) X_raw[[nm]] <- as.character(X_raw[[nm]])
  }
  X_raw[] <- lapply(X_raw, function(z) if (is.character(z)) factor(z) else z)

  cn      <- colnames(X_raw)
  cn_s    <- make.names(cn)
  vn      <- varnames
  vn_s    <- make.names(vn)

  out <- matrix(NA_real_, nrow = nrow(X_raw), ncol = length(vn))
  colnames(out) <- vn

  for (i in seq_along(vn)) {
    v     <- vn[i]
    v_s   <- vn_s[i]

    # Case 1: exact column present
    if (v %in% cn) {
      z <- X_raw[[v]]
      out[, i] <- if (is.factor(z)) as.numeric(z) else as.numeric(z)
      next
    }

    # Case 2: derive dummy = 1{ base == level } from a base column prefix
    # Find the longest raw name that is a prefix of v (sanitized comparison)
    cand <- which(startsWith(v_s, cn_s))
    if (length(cand)) {

```

```

j <- cand[which.max(nchar(cn_s[cand]))]
base <- cn[j]
# level is the suffix of v after the base name (unsanitized; preserves case)
lev <- sub(paste0("^", base), "", v)
x <- X_raw[[base]]

# Compare as strings to match labels like "Female", "Inpatient", "0","1',...
x_chr <- if (is.factor(x)) as.character(x) else as.character(x)
out[, i] <- as.numeric(x_chr == lev)
out[is.na(x_chr), i] <- NA_real_
next
}

stop("Cannot construct design column for '", v, "'.
      "No matching base variable found in raw covariates.")
}

as.data.frame(out, check.names = FALSE)
}

# Unified backend:
# - If gbm$var.names are raw feature names, pass factors directly (aligning levels).
# - Otherwise, build a one-hot design with those exact column names and predict on it.
make_shap_backend_any <- function(W) {
  gbm_fit <- if (!is.null(W$obj)) W$obj else if (!is.null(W$info$obj)) W$info$obj else W$info$model.obj
  stopifnot(inherits(gbm_fit, "gbm"))
  best_tree <- if (!is.null(W$info$best.tree)) W$info$best.tree else gbm_fit$n.trees

  X_raw <- as.data.frame(W$covs, stringsAsFactors = FALSE)
  # tidy odd classes; keep factors where they already are
  for (nm in names(X_raw)) {
    if (inherits(X_raw[[nm]], "haven_labelled")) X_raw[[nm]] <- as.character(X_raw[[nm]])
  }
  X_raw[] <- lapply(X_raw, function(z) if (is.character(z)) factor(z) else z)
  X_raw[] <- lapply(X_raw, function(z) if (is.factor(z)) droplevels(z) else z)
}

```

```

vn <- gbm_fit$var.names
levels_map <- .train_levels(X_raw)

# Path A: raw-factor training (names line up directly)
if (all(vn %in% colnames(X_raw))) {
  X_use <- .align_to_levels(X_raw[, vn, drop = FALSE], levels_map)
  pred <- function(object, newdata) {
    nd <- .align_to_levels(newdata, levels_map)
    predict(object, newdata = nd, n.trees = best_tree, type = "link")
  }
  return(list(gbm = gbm_fit, X = X_use, pred = pred, best_tree = best_tree))
}

# Path B: dummy-coded training (var.names are one-hot)
X_use <- .design_from_varnames(X_raw, vn)
pred <- function(object, newdata) {
  # If caller already supplies the dummy design, use it; else derive it
  if (all(vn %in% colnames(newdata))) {
    nd <- as.data.frame(newdata)[, vn, drop = FALSE]
  } else {
    nd <- .design_from_varnames(newdata, vn)
  }
  predict(object, newdata = nd, n.trees = best_tree, type = "link")
}

list(gbm = gbm_fit, X = X_use, pred = pred, best_tree = best_tree)
}

```

*Chunk unnamed-chunk-2 runtime: 0.05 s*

```

# Choose one completed dataset's fit
# Device safety (once per doc)
if (!dir.exists("figs")) dir.create("figs", recursive = TRUE, showWarnings = FALSE)
knitr::opts_chunk$set(fig.path = "figs/", dev = "ragg_png", dpi = 200)

# SHAP throttle knobs (single imputation, subsample rows, fewer sims)

```

```

shap_cfg <- list(
  frac_rows = 0.15,
  nsim      = 8,
  top_k     = 20,
  seed       = 123
)

# --- ABG explainability on imputation 1 -----
stopifnot(exists("W_abg_list"), length(W_abg_list) >= 1)
W1_abg <- W_abg_list[[1]]
bk_abg <- make_shap_backend_any(W1_abg)

# Equivalence: wrapper vs direct gbm predict on the same design matrix
bk <- make_shap_backend_any(W_abg_list[[1]])
p_wrap <- bk$pred(bk$gbm, bk$X)
p_direct <- predict(
  bk$gbm,
  newdata = bk$X[, bk$gbm$var.names, drop = FALSE],
  n.trees = bk$best_tree,
  type    = "link"
)
stopifnot(mean(abs(p_wrap - p_direct), na.rm = TRUE) < 1e-8)

# Subsample rows for SHAP speed
set.seed(shap_cfg$seed)
ix_abg <- sample.int(nrow(bk_abg$X), max(50L, floor(shap_cfg$frac_rows * nrow(bk_abg$X))))
X_abg <- bk_abg$X[ix_abg, , drop = FALSE]

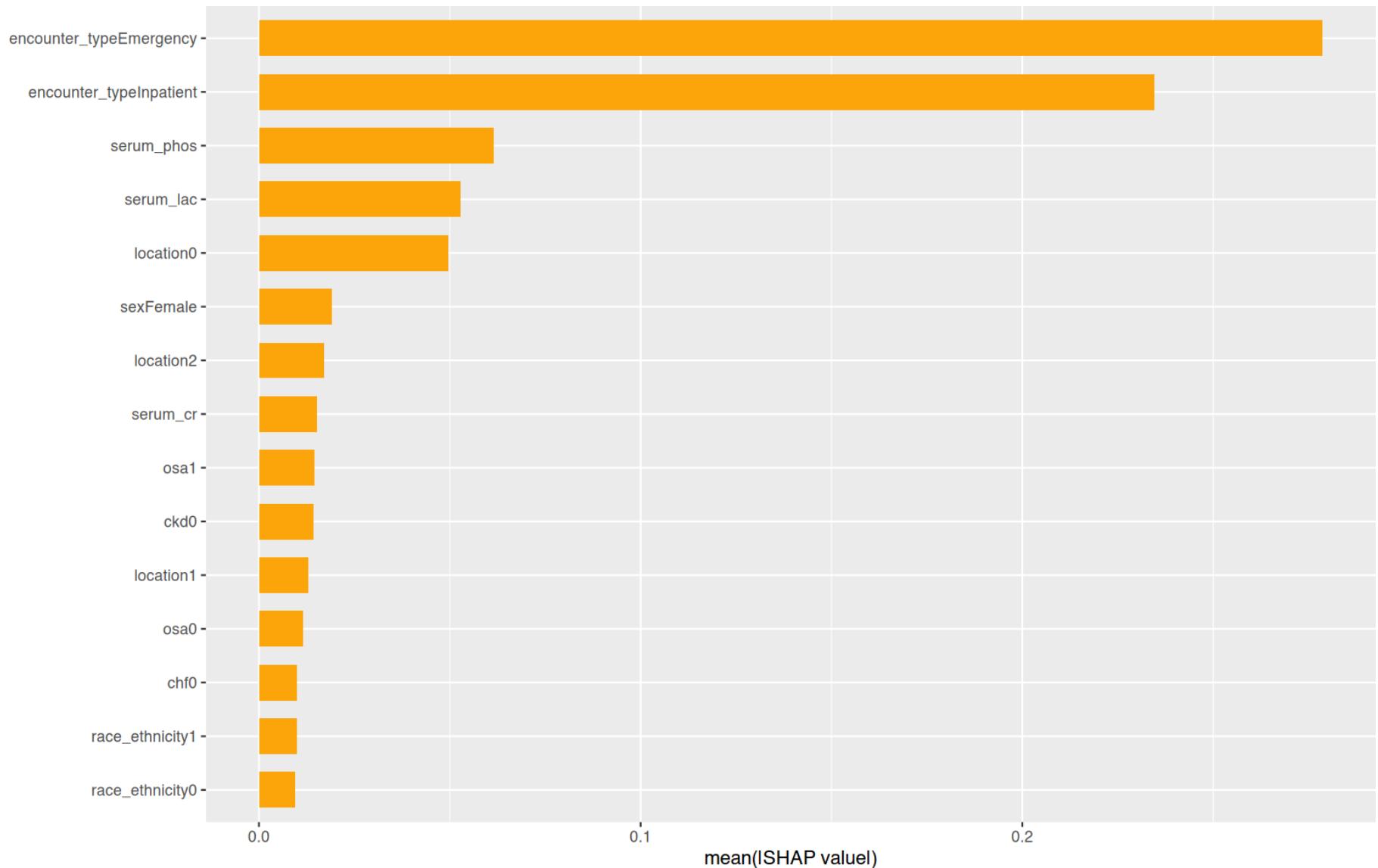
# Fast SHAP on link (logit) scale
S_abg <- fastshap::explain(
  object      = bk_abg$gbm,
  X           = X_abg,
  pred_wrapper = bk_abg$pred,
  nsim        = shap_cfg$nsim,
  adjust      = FALSE
)

```

```
# shapviz object (X can be data.frame or matrix; keep column names)
sv_abg <- shapviz::shapviz(as.matrix(S_abg), X = as.matrix(X_abg))

# Bar importance (top K)
ord_abg   <- order(colMeans(abs(S_abg), na.rm = TRUE), decreasing = TRUE)
topK_abg  <- colnames(S_abg)[ord_abg[1:min(shap_cfg$top_k, ncol(S_abg))]]
p_bar_abg <- shapviz::sv_importance(sv_abg, kind = "bar", v = topK_abg)
p_bar_abg
```

Warning: `label` cannot be a `<ggplot2::element_blank>` object.

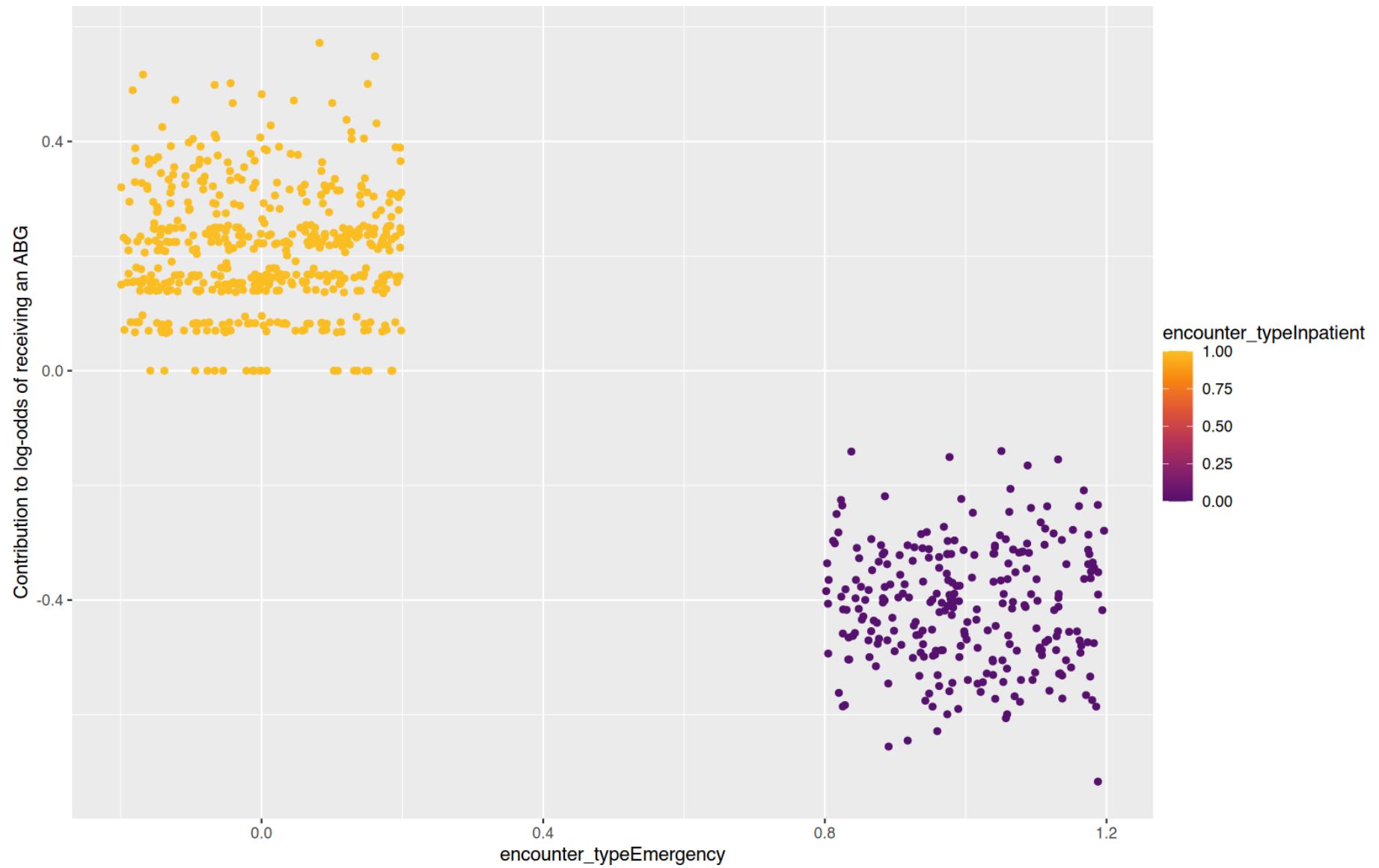


```
# Dependence: top feature colored by second (add smoother explicitly)
pri_abg <- topK_abg[1]
aux_abg <- topK_abg[2]
# dependence: replace loess with GAM to avoid near-singularity
```

```
p_dep_abg <- shapviz::sv_dependence(sv_abg, v = pri_abg, color_var = aux_abg) +
  ggplot2::geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs", k = 4),
    se = FALSE, linewidth = 0.5) +
  ggplot2::labs(y = shap_y_abg, x = pri_abg)

# for importance: don't set label = element_blank() on shapviz's plot; let it draw defaults
# if you see "aesthetics dropped: colour", avoid mapping `colour` in a stat
p_dep_abg
```

Warning: Failed to fit group -1.  
Caused by error in `smooth.construct.cr.smooth.spec()`:  
! x has insufficient unique values to support 4 knots: reduce k.



Chunk shap-abg-vbg runtime: 12.19 s

```
# Repeat for VBG
```

```

# --- VBG explainability on imputation 1 -----
stopifnot(exists("W_vbg_list"), length(W_vbg_list) >= 1)
W1_vbg <- W_vbg_list[[1]]

bk_vbg <- make_shap_backend_any(W1_vbg)

set.seed(shap_cfg$seed)
ix_vbg <- sample.int(nrow(bk_vbg$X), max(50L, floor(shap_cfg$frac_rows * nrow(bk_vbg$X))))
X_vbg <- bk_vbg$X[ix_vbg, , drop = FALSE]

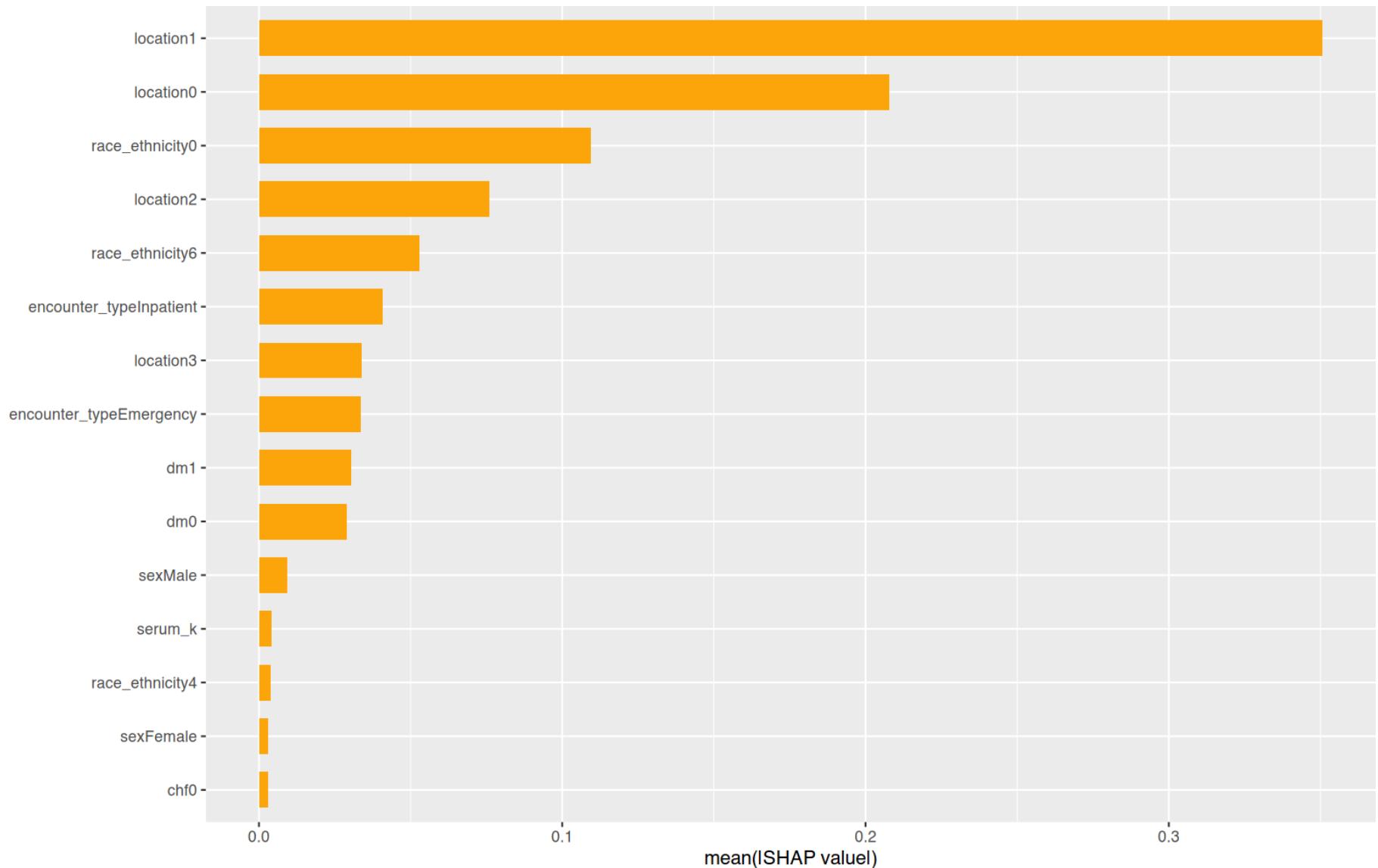
S_vbg <- fastshap::explain(
  object      = bk_vbg$gbm,
  X           = X_vbg,
  pred_wrapper = bk_vbg$pred,
  nsim        = shap_cfg$nsim,
  adjust       = FALSE
)

sv_vbg <- shapviz::shapviz(as.matrix(S_vbg), X = as.matrix(X_vbg))

ord_vbg   <- order(colMeans(abs(S_vbg), na.rm = TRUE), decreasing = TRUE)
topK_vbg <- colnames(S_vbg)[ord_vbg[1:min(shap_cfg$top_k, ncol(S_vbg))]]
p_bar_vbg <- shapviz::sv_importance(sv_vbg, kind = "bar", v = topK_vbg)
p_bar_vbg

```

Warning: `label` cannot be a `<ggplot2::element_blank>` object.



```

pri_vbg <- topK_vbg[1]
aux_vbg <- topK_vbg[2]
p_dep_vbg <- shapviz::sv_dependence(sv_vbg, v = pri_vbg, color_var = aux_vbg) +
  ggplot2::geom_smooth(se = FALSE, linewidth = 0.5, method = "loess", formula = y ~ x) +

```

```
ggplot2::labs(y = shap_y_vbg, x = pri_vbg)  
p_dep_vbg
```

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: pseudoinverse used at -0.005

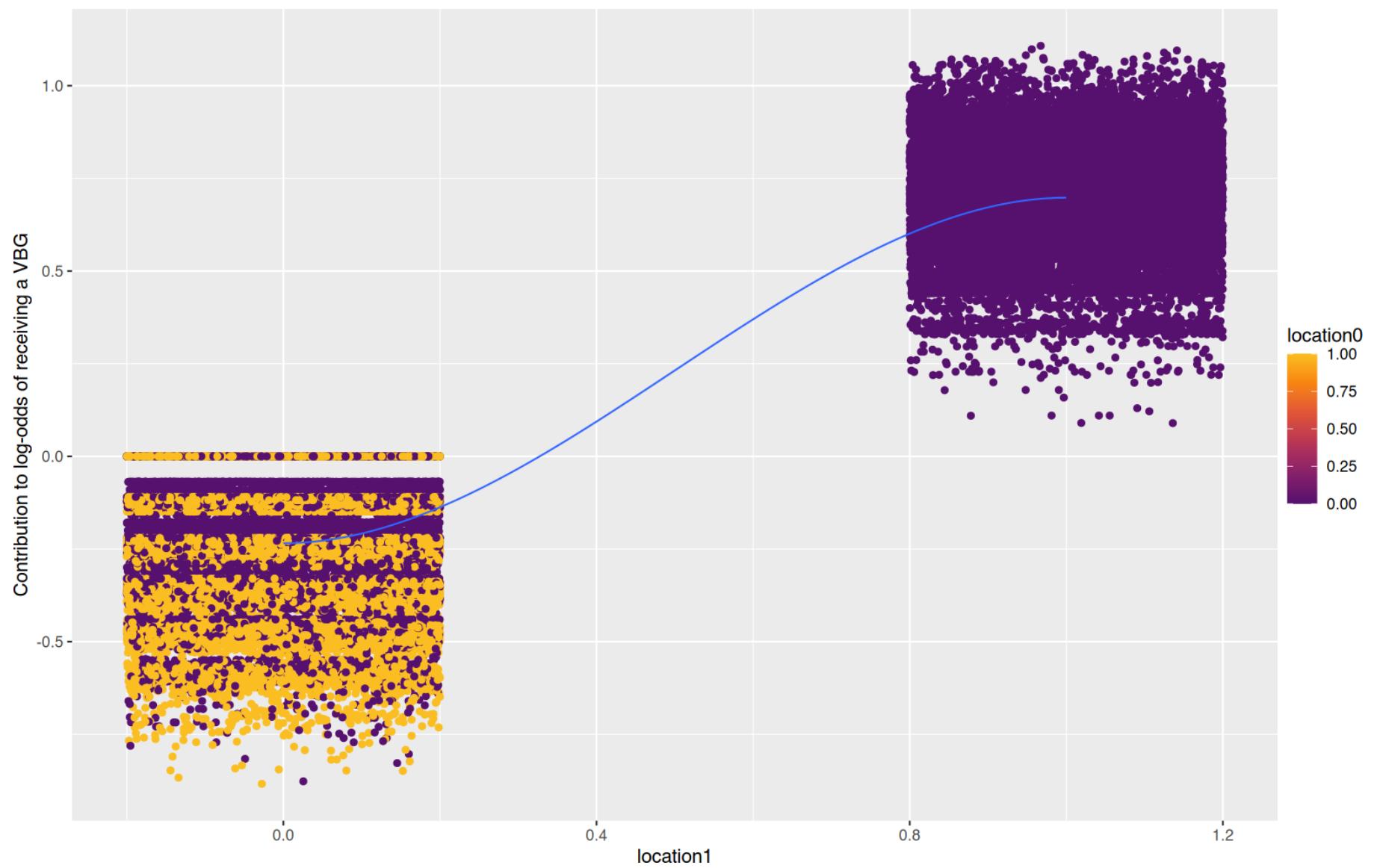
Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: neighborhood radius 1.005

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: reciprocal condition number 2.4157e-26

Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
: There are other near singularities as well. 1.01

Warning: The following aesthetics were dropped during statistical transformation:  
colour.

- i This can happen when ggplot fails to infer the correct grouping structure in  
the data.
- i Did you forget to specify a `group` aesthetic or to convert a numerical  
variable into a factor?



Chunk unnamed-chunk-3 runtime: 3426.53 s

## 5.7 13) Imputed, weighted, three-level PCO2 (ABG & VBG)

```
# --- helpers -----
library(splines)
library(mitoools)
library(survey)
library(dplyr)

# Pool (Rubin) any subset of coefficients across imputed fits (glm/svyglm)
pool_terms <- function(fits, term_prefix = NULL, term_pattern = NULL) {
  fits <- Filter(Negate(is.null), fits)
  if (!length(fits)) return(
    data.frame(term = character(), logOR = numeric(), SE = numeric(),
               OR = numeric(), LCL = numeric(), UCL = numeric())
  )

  coef_names <- lapply(fits, function(f) names(stats::coef(f)))
  common <- Reduce(intersect, coef_names)
  if (!is.null(term_prefix)) common <- common[startsWith(common, term_prefix)]
  if (!is.null(term_pattern)) common <- common[grep(term_pattern, common)]
  if (!length(common)) return(
    data.frame(term = character(), logOR = numeric(), SE = numeric(),
               OR = numeric(), LCL = numeric(), UCL = numeric())
  )

  rows <- lapply(common, function(tt) {
    results <- lapply(fits, function(f) setNames(c(stats::coef(f)[tt]), tt))
    variances <- lapply(fits, function(f) {
      v <- stats::vcov(f)[tt, tt, drop = TRUE]
      m <- matrix(v, 1, 1); dimnames(m) <- list(tt, tt); m
    })
    pooled <- mitools::MIcombine(results = results, variances = variances)
    est <- as.numeric(coef(pooled))
    se <- sqrt(diag(pooled$variance))
    data.frame(term = tt,
```

```

    logOR = est, SE = se,
    OR  = exp(est),
    LCL = exp(est - 1.96*se),
    UCL = exp(est + 1.96*se),
    row.names = NULL)
  })
  dplyr::bind_rows(rows)
}

# 3-level CO2 category maker; can use fixed clinical cutpoints or data-driven
# --- CO2 category helper (3-level) -----
make_co2_cat <- function(x,
                         fixed_breaks = NULL,
                         labels = c("Eucapnia", "Hypocapnia", "Hypercapnia"),
                         normal = NULL) {
  x <- suppressWarnings(as.numeric(x))
  x_ok <- x[is.finite(x)]
  if (length(x_ok) < 10) {
    return(factor(rep(NA_character_, length(x)), levels = labels))
  }

  brks <- NULL
  # priority: explicit breaks → "normal" window → quantile fallback
  if (!is.null(fixed_breaks)) {
    brks <- fixed_breaks
  } else if (!is.null(normal)) {
    n <- sort(unique(as.numeric(normal)))
    if (length(n) >= 2 && all(is.finite(n)) && (n[2] > n[1])) {
      brks <- c(-Inf, n[1], n[2], Inf)
    }
  }
  if (is.null(brks)) {
    brks <- stats::quantile(x_ok, probs = c(0, 1 / 3, 2 / 3, 1), na.rm = TRUE)
  }

  brks <- unique(brks)
}

```

```

if (length(brks) < 4 || any(diff(brks) <= 0)) {
  return(factor(rep(NA_character_, length(x)), levels = labels))
}
cut(x, breaks = brks, include.lowest = TRUE, labels = labels, right = TRUE)
}

# defaults (safe if you didn't predefine)
use_fixed_abg <- if (exists("use_fixed_abg")) isTRUE(use_fixed_abg) else FALSE
co2_breaks_abg <- if (exists("co2_breaks_abg")) co2_breaks_abg else NULL
co2_labels_abg <- if (exists("co2_labels_abg")) co2_labels_abg else c("Eucapnia", "Hypocapnia", "Hypercapnia")
ref_label_abg <- if (exists("ref_label_abg")) ref_label_abg else "Eucapnia"

# (Optional) explicit clinical cutpoints - override by setting use_fixed_* = TRUE
use_fixed_abg <- TRUE
co2_breaks_abg <- c(-Inf, 45, 55, Inf)
co2_labels_abg <- c("Eucapnia", "Hypocapnia", "Hypercapnia")
ref_label_abg <- "Eucapnia"

use_fixed_vbg <- TRUE
co2_breaks_vbg <- c(-Inf, 50, 60, Inf)
co2_labels_vbg <- c("Eucapnia", "Hypocapnia", "Hypercapnia")
ref_label_vbg <- "Eucapnia"

# Fixed spline knots & boundary knots (shared across imputations)
# Use 2-98th percentile boundaries; 2 internal knots ( df=4 total)
make_knots <- function(x) {
  x <- x[is.finite(x)]
  B <- stats::quantile(x, probs = c(0.02, 0.98), na.rm = TRUE)
  K <- stats::quantile(x[x >= B[1] & x <= B[2]], probs = c(1/3, 2/3), na.rm = TRUE)
  list(boundary = unname(B), knots = unname(K))
}

```

*Chunk mi\_pco2\_threellevel runtime: 0.01 s*

*Chunk mi-co2-cat-checks runtime: 0.85 s*

*Chunk pool-spline-helpers runtime: 0.00 s*

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

### 5.8.1 14.1 ABG: MI + IPW, three-level PCO2 outcomes

```
# --- ABG: outcome ~ CO2 category, IPW by W_abg_list -----
# Assumes: dlist, W_abg_list, make_co2_cat(), use_fixed_abg, co2_breaks_abg,
#           co2_labels_abg, ref_label_abg are defined.

fit_abg_cat <- function(outcome_var) {
  fits <- vector("list", length(dlist))
  for (i in seq_along(dlist)) {
    d <- dlist[[i]]
    if (!("paco2" %in% names(d))) { fits[[i]] <- NULL; next }
    d$paco2 <- suppressWarnings(as.numeric(d$paco2))

    g <- with(d, has_abg == 1 & is.finite(paco2))
    if (!any(g)) { fits[[i]] <- NULL; next }

    d2 <- d[g, , drop = FALSE]
    w <- W_abg_list[[i]]$weights[g]
    w[!is.finite(w)] <- NA_real_
    d2$co2_cat <- make_co2_cat(
      d2$paco2,
      fixed_breaks = if (isTRUE(use_fixed_abg)) co2_breaks_abg else NULL,
      labels       = co2_labels_abg,
      normal       = if (exists("co2_breaks_abg", inherits = TRUE)) co2_breaks_abg else c(35, 45)
    )
    d2$co2_cat <- base::droplevels(d2$co2_cat)
    d2$co2_cat <- stats::relevel(d2$co2_cat, ref = ref_label_abg)
    if (nlevels(d2$co2_cat) < 2) { fits[[i]] <- NULL; next }

    ok <- is.finite(w)
    if (!all(ok)) {
      d2 <- d2[ok, , drop = FALSE]
      w <- w[ok]
```

```

  if (nrow(d2) == 0L) { fits[[i]] <- NULL; next }
}

des <- survey::svydesign(ids = ~1, weights = ~w, data = d2)
fml <- stats::as.formula(sprintf("%s ~ co2_cat", outcome_var))
fits[[i]] <- survey::svyglm(fml, design = des, family = quasibinomial())
}
pool_terms(fits, term_prefix = "co2_cat")
}

abg_cat_results <- dplyr::bind_rows(
  dplyr::mutate(fit_abg_cat("imv_proc"), outcome = "IMV"),
  dplyr::mutate(fit_abg_cat("niv_proc"), outcome = "NIV"),
  dplyr::mutate(fit_abg_cat("death_60d"), outcome = "Death60d"),
  dplyr::mutate(fit_abg_cat("hypercap_resp_failure"), outcome = "HCRF")
) |>
  dplyr::relocate(outcome)

```

*Chunk unnamed-chunk-4 runtime: 16.07 s*

### 5.8.2 14.2 VBG: MI + IPW, three-level PCO2 outcomes

```

# Assumes: dlist, W_vbg_list, make_co2_cat(), use_fixed_vbg, co2_breaks_vbg,
#           co2_labels_vbg, ref_label_vbg are defined.

fit_vbg_cat <- function(outcome_var) {
  fits <- vector("list", length(dlist))
  for (i in seq_along(dlist)) {
    d <- dlist[[i]]
    if (!("vbg_co2" %in% names(d))) { fits[[i]] <- NULL; next }
    d$vbg_co2 <- suppressWarnings(as.numeric(d$vbg_co2))

    g <- with(d, has_vbg == 1 & is.finite(vbg_co2))
    if (!any(g)) { fits[[i]] <- NULL; next }
  }
}
```

```

d2 <- d[g, , drop = FALSE]
w <- W_vbg_list[[i]]$weights[g]
w[!is.finite(w)] <- NA_real_
d2$co2_cat <- make_co2_cat(
  d2$vbg_co2,
  fixed_breaks = if (isTRUE(use_fixed_vbg)) co2_breaks_vbg else NULL,
  labels       = co2_labels_vbg,
  normal       = if (exists("co2_breaks_vbg", inherits = TRUE)) co2_breaks_vbg else c(40, 50)
)
d2$co2_cat <- base::droplevels(d2$co2_cat)
d2$co2_cat <- stats::relevel(d2$co2_cat, ref = ref_label_vbg)
if (nlevels(d2$co2_cat) < 2) { fits[[i]] <- NULL; next }

ok <- is.finite(w)
if (!all(ok)) {
  d2 <- d2[ok, , drop = FALSE]
  w  <- w[ok]
  if (nrow(d2) == 0L) { fits[[i]] <- NULL; next }
}

des <- survey::svydesign(ids = ~1, weights = ~w, data = d2)
fml <- stats::as.formula(sprintf("%s ~ co2_cat", outcome_var))
fits[[i]] <- survey::svyglm(fml, design = des, family = quasibinomial())
}
pool_terms(fits, term_prefix = "co2_cat")
}

vbg_cat_results <- dplyr::bind_rows(
  dplyr::mutate(fit_vbg_cat("imv_proc"),           outcome = "IMV"),
  dplyr::mutate(fit_vbg_cat("niv_proc"),           outcome = "NIV"),
  dplyr::mutate(fit_vbg_cat("death_60d"),          outcome = "Death60d"),
  dplyr::mutate(fit_vbg_cat("hypercap_resp_failure"), outcome = "HCRF")
) |>
  dplyr::relocate(outcome)

```

*Chunk unnamed-chunk-5 runtime: 13.37 s*

```
# After re-running MICE:  
dlist <- mice::complete(imp, action = "all")  
  
# 1) must exist and be numeric  
stopifnot(all(c("paco2","vbg_co2") %in% names(dlist[[1]])))  
stopifnot(is.numeric(dlist[[1]]$paco2), is.numeric(dlist[[1]]$vbg_co2))  
  
# 2) confirm at least two PaCO2 levels among those with ABG in each imputation  
table(vapply(dlist, function(d) dplyr::n_distinct(d$paco2[d$has_abg == 1 & is.finite(d$paco2)]), integer(1)) > 1)
```

TRUE

5

```
# 3) smoke test the ABG category fit on the first imputation  
tmp <- fit_abg_cat("imv_proc"); print(tmp)
```

	term	logOR	SE	OR	LCL	UCL
1	co2_catHypocapnia	-0.08900192	0.01599330	0.9148438	0.8866111	0.9439755
2	co2_catHypercapnia	0.41642606	0.01738164	1.5165319	1.4657368	1.5690872

*Chunk unnamed-chunk-6 runtime: 6.31 s*

### 5.8.3 14.3 Visualization: pooled three-level ORs

```
library(dplyr)  
library(survey)  
library(ggplot2)  
library(scales)  
library(purrr)  
library(mitoools)
```

```

# --- Pre-flight -----
if (!exists("dlist")) dlist <- mice::complete(imp, action = "all")
stopifnot(length(W_abg_list) == length(dlist),
          length(W_vbg_list) == length(dlist))

# --- Pooling helper for term-level log-ORs across imputations -----
pool_terms <- function(fits, term_prefix) {
  fits <- Filter(Negate(is.null), fits)
  if (length(fits) == 0L) {
    return(tibble::tibble(term=character(), logOR=numeric(), SE=numeric(),
                          OR=numeric(), LCL=numeric(), UCL=numeric()))
  }
  terms_list <- lapply(fits, function(f) names(stats::coef(f)))
  common      <- Reduce(intersect, terms_list)
  keep_terms <- grep(paste0("^", term_prefix), common, value = TRUE)
  if (!length(keep_terms)) {
    return(tibble::tibble(term=character(), logOR=numeric(), SE=numeric(),
                          OR=numeric(), LCL=numeric(), UCL=numeric()))
  }

  purrr::map_dfr(keep_terms, function(term) {
    res <- lapply(fits, function(f) setNames(c(stats::coef(f)[term]), term))
    vars <- lapply(fits, function(f) {
      V <- stats::vcov(f)
      m <- matrix(V[term, term], 1, 1); dimnames(m) <- list(term, term); m
    })
    pooled <- mitools::MIcombine(results = res, variances = vars)
    est <- as.numeric(stats::coef(pooled))
    se  <- sqrt(diag(pooled$variance))
    tibble::tibble(
      term  = term,
      logOR = est,
      SE    = se,
      OR    = exp(est),
      LCL   = exp(est - 1.96 * se),

```

```

    UCL    = exp(est + 1.96 * se)
  )
}
}

# --- Per-group runner over imputations (ABG/VBG) -----
fit_cat_group <- function(group = c("ABG", "VBG"), outcome) {
  group <- match.arg(group)
  fits <- vector("list", length(dlist))

  for (i in seq_along(dlist)) {
    d <- dlist[[i]]

    if (group == "ABG") {
      if (!("paco2" %in% names(d))) { fits[[i]] <- NULL; next }
      d$paco2 <- suppressWarnings(as.numeric(d$paco2))
      g <- with(d, has_abg == 1 & is.finite(paco2))
      if (!any(g)) { fits[[i]] <- NULL; next }
      d2 <- d[g, , drop = FALSE]
      d2$co2_cat <- make_co2_cat(
        d2$paco2,
        fixed_breaks = if (exists("use_fixed_abg", inherits = TRUE) && isTRUE(use_fixed_abg)) co2_breaks_abg else NULL,
        labels       = if (exists("co2_labels_abg", inherits = TRUE)) co2_labels_abg else c("Eucapnia", "Hypocapnia", "Hypercapnia"),
        normal       = if (exists("co2_breaks_abg", inherits = TRUE)) co2_breaks_abg else c(35, 45)
      )
      w <- W_abg_list[[i]]$weights[g]
      w[!is.finite(w)] <- NA_real_
    } else {
      if (!("vbg_co2" %in% names(d))) { fits[[i]] <- NULL; next }
      d$vbg_co2 <- suppressWarnings(as.numeric(d$vbg_co2))
      g <- with(d, has_vbg == 1 & is.finite(vbg_co2))
      if (!any(g)) { fits[[i]] <- NULL; next }
      d2 <- d[g, , drop = FALSE]
      d2$co2_cat <- make_co2_cat(
        d2$vbg_co2,

```

```

fixed_breaks = if (exists("use_fixed_vbg", inherits = TRUE) && isTRUE(use_fixed_vbg)) co2_breaks_vbg else NULL,
labels       = if (exists("co2_labels_vbg", inherits = TRUE)) co2_labels_vbg else c("Eucapnia", "Hypocapnia", "Hypercapnia")
normal       = if (exists("co2_breaks_vbg", inherits = TRUE)) co2_breaks_vbg else c(40, 50)
)
w <- W_vbg_list[[i]]$weights[g]
w[!is.finite(w)] <- NA_real_
}

ref_lab <- if (group == "ABG") {
  if (exists("ref_label_abg", inherits = TRUE)) ref_label_abg else levels(d2$co2_cat)[1]
} else {
  if (exists("ref_label_vbg", inherits = TRUE)) ref_label_vbg else levels(d2$co2_cat)[1]
}
if (!ref_lab %in% levels(d2$co2_cat)) ref_lab <- levels(d2$co2_cat)[1]
d2$co2_cat <- stats::relevel(base::droplevels(d2$co2_cat), ref = ref_lab)
if (nlevels(d2$co2_cat) < 2) { fits[[i]] <- NULL; next }

okw <- is.finite(w)
if (!all(okw)) {
  d2 <- d2[okw, , drop = FALSE]
  w  <- w[okw]
  if (nrow(d2) == 0L) { fits[[i]] <- NULL; next }
}

d2$w <- w
des  <- survey::svydesign(ids = ~1, weights = ~w, data = d2)
fml   <- stats::as.formula(paste0(outcome, " ~ co2_cat"))
fit   <- try(survey::svyglm(fml, design = des, family = quasibinomial()), silent = TRUE)
fits[[i]] <- if (!inherits(fit, "try-error")) fit else NULL
}

out <- pool_terms(fits, term_prefix = "co2_cat")
out
}

# --- Run & plot -----

```

```

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

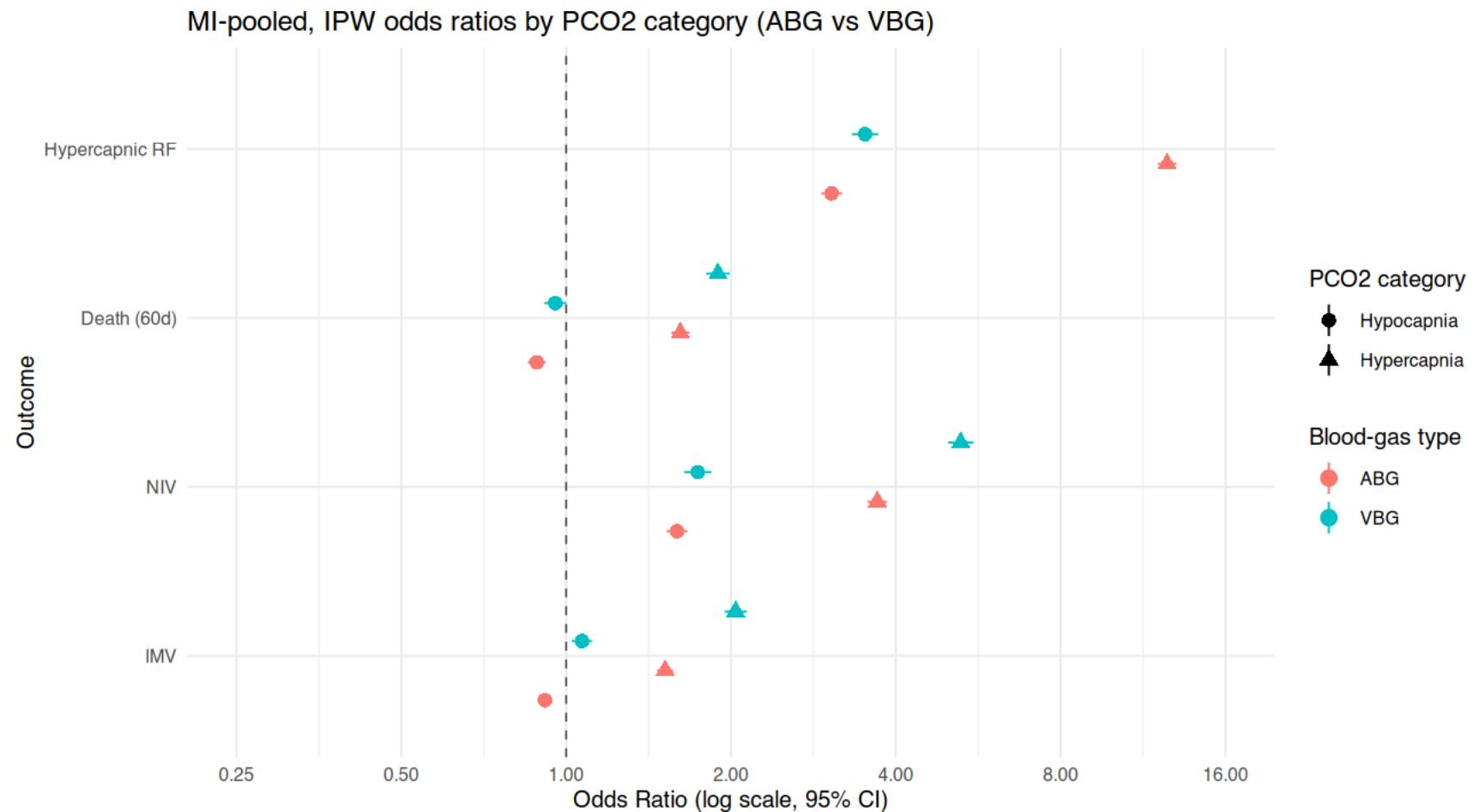
abg_df <- purrr::map_dfr(outs, ~ dplyr::mutate(fit_cat_group("ABG", .x),
                                                outcome = .x, group = "ABG"))
vbg_df <- purrr::map_dfr(outs, ~ dplyr::mutate(fit_cat_group("VBG", .x),
                                                outcome = .x, group = "VBG"))
combined <- dplyr::bind_rows(abg_df, vbg_df)

# decode "co2_cat<level>" → exposure
combined <- combined |>
  dplyr::mutate(exposure = gsub("^co2_cat", "", term),
                exposure = factor(exposure, levels = c("Eucapnia", "Hypocapnia", "Hypercapnia")),
                outcome = factor(outcome,
                                 levels = c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure"),
                                 labels = c("IMV", "NIV", "Death (60d)", "Hypercapnic RF")),
                group = factor(group, levels = c("ABG", "VBG")))

ggplot(
  combined,
  aes(x = outcome, y = OR, ymin = LCL, ymax = UCL, color = group, shape = exposure)
) +
  geom_pointrange(position = position_dodge(width = 0.7), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = scales::number_format(accuracy = 0.01)
  ) +
  coord_flip() +
  labs(
    title = "MI-pooled, IPW odds ratios by PCO2 category (ABG vs VBG)",
    x = "Outcome",
    y = "Odds Ratio (log scale, 95% CI)",
    color = "Blood-gas type",
    shape = "PCO2 category"
  )

```

```
theme_minimal(base_size = 10)
```



Chunk ipw-three-level-pco2-mi-abg-vbg runtime: 32.12 s

## 5.9 15) Imputed, weighted spline PCO2 (ABG & VBG)

### 5.9.1 15.1 ABG, imputed, weighted, spline outcome

```
# Use pooled per-group helpers defined above (fit_cat_group + pool_terms)
outs <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

abg_cat_results <- purrr::map_dfr(outs, ~ dplyr::mutate(fit_cat_group("ABG", .x),
                                                       outcome = .x, group = "ABG")) |>
  dplyr::relocate(outcome)
abg_cat_results
```

outcome	term	logOR	SE	OR	LCL	UCL	group
imv_proc	co2_catHypocapnia	-0.0890019	0.0159933	0.9148438	0.8866111	0.9439755	ABG
imv_proc	co2_catHypercapnia	0.4164261	0.0173816	1.5165319	1.4657368	1.5690872	ABG
niv_proc	co2_catHypocapnia	0.4658845	0.0224003	1.5934229	1.5249778	1.6649400	ABG
niv_proc	co2_catHypercapnia	1.3084795	0.0210550	3.7005427	3.5509376	3.8564509	ABG
death_60d	co2_catHypocapnia	-0.1234746	0.0190472	0.8838441	0.8514562	0.9174639	ABG
death_60d	co2_catHypercapnia	0.4801999	0.0193858	1.6163975	1.5561328	1.6789959	ABG
hypercap_resp_failure	co2_catHypocapnia	1.1163950	0.0221865	3.0538254	2.9238739	3.1895526	ABG
hypercap_resp_failure	co2_catHypercapnia	2.5270666	0.0202664	12.5167356	12.0292892	13.0239340	ABG

Chunk unnamed-chunk-7 runtime: 16.61 s

### 5.9.2 15.2 VBG, imputed, weighted, spline outcome

```
outs <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

vbg_cat_results <- purrr::map_dfr(outs, ~ dplyr::mutate(fit_cat_group("VBG", .x),
                                                       outcome = .x, group = "VBG")) |>
  dplyr::relocate(outcome)
vbg_cat_results
```

outcome	term	logOR	SE	OR	LCL	UCL	group
imv_proc	co2_catHypocapnia	0.0663405	0.0217299	1.0685905	1.0240342	1.115085	VBG
imv_proc	co2_catHypercapnia	0.7127940	0.0238266	2.0396821	1.9466187	2.137195	VBG
niv_proc	co2_catHypocapnia	0.5537299	0.0290045	1.7397299	1.6435870	1.841497	VBG
niv_proc	co2_catHypercapnia	1.6596402	0.0271958	5.2574188	4.9845169	5.545262	VBG
death_60d	co2_catHypocapnia	-0.0456934	0.0236120	0.9553349	0.9121299	1.000586	VBG
death_60d	co2_catHypercapnia	0.6381542	0.0253230	1.8929836	1.8013227	1.989309	VBG
hypercap_resp_failure	co2_catHypocapnia	1.2572216	0.0282544	3.5156399	3.3262414	3.715823	VBG
hypercap_resp_failure	co2_catHypercapnia	3.0457809	0.0262089	21.0264443	19.9736018	22.134784	VBG

Chunk unnamed-chunk-8 runtime: 14.24 s

### 5.9.3 15.3 Visualization

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

if (!exists("dlist")) dlist <- mice::complete(imp, action = "all")
stopifnot(length(W_abg_list) == length(dlist),
          length(W_vbg_list) == length(dlist))

# CO2 support sanity: ensure trimmed ranges exist for both groups
rng_abg <- quantile(
  unlist(lapply(dlist, function(d) as.numeric(d$paco2[d$has_abg == 1]))),
  c(.02, .98), na.rm = TRUE
)
rng_vbg <- quantile(
  unlist(lapply(dlist, function(d) as.numeric(d$vbg_co2[d$has_vbg == 1]))),
  c(.02, .98), na.rm = TRUE
```

```

)
stopifnot(rng_abg[1] < rng_abg[2], rng_vbg[1] < rng_vbg[2])

# Predict link for svyglm; fallback to X and delta method if predict() returns a vector
predict_link_svyglm <- function(fit, newdata) {
  pr <- try(stats::predict(fit, newdata = newdata, type = "link", se.fit = TRUE), silent = TRUE)
  if (!inherits(pr, "try-error") && is.list(pr) && !is.null(pr$fit)) {
    return(list(fit = as.numeric(pr$fit), se.fit = as.numeric(pr$se.fit)))
  }
  # manual:   = X ; Var( ) = X V X^T
  X <- stats::model.matrix(stats::delete.response(stats::terms(fit)), newdata)
  beta <- stats::coef(fit)
  eta <- drop(X %*% beta)
  V <- stats::vcov(fit)
  se <- sqrt(rowSums((X %*% V) * X))
  list(fit = eta, se.fit = se)
}

# Pool predicted curves on the link scale, then transform with logistic
pool_rcs_curve <- function(group = c("ABG", "VBG"), outcome,
                           df = 4, grid_n = 220, trim = c(0.02, 0.98)) {
  group <- match.arg(group)

  # global trimmed range for this group
  co2_all <- unlist(lapply(seq_along(dlist), function(i) {
    d <- dlist[[i]]
    if (group == "ABG") as.numeric(d$paco2[d$has_abg == 1])
    else as.numeric(d$vbg_co2[d$has_vbg == 1])
  }), use.names = FALSE)
  co2_all <- co2_all[is.finite(co2_all)]
  stopifnot(length(co2_all) > 0)
  rng <- as.numeric(stats::quantile(co2_all, probs = trim, na.rm = TRUE))
  if (!is.finite(rng[1]) || !is.finite(rng[2]) || rng[2] <= rng[1]) {
    med <- stats::median(co2_all); rng <- c(med - 1, med + 1)
  }
  xseq <- seq(rng[1], rng[2], length.out = grid_n)
}

```

```

eta_list <- list()
var_list <- list()

for (i in seq_along(dlist)) {
  d <- dlist[[i]]

  if (group == "ABG") {
    d$paco2 <- suppressWarnings(as.numeric(d$paco2))
    g <- with(d, has_abg == 1 & is.finite(paco2))
    if (!any(g)) next
    d2 <- d[g, , drop = FALSE]
    d2$co2 <- d2$paco2
    w <- W_abg_list[[i]]$weights[g]
  } else {
    d$vbg_co2 <- suppressWarnings(as.numeric(d$vbg_co2))
    g <- with(d, has_vbg == 1 & is.finite(vbg_co2))
    if (!any(g)) next
    d2 <- d[g, , drop = FALSE]
    d2$co2 <- d2$vbg_co2
    w <- W_vbg_list[[i]]$weights[g]
  }

  # drop NA/inf weights
  ok <- is.finite(w)
  if (!all(ok)) {
    d2 <- d2[ok, , drop = FALSE]
    w <- w[ok]
    if (nrow(d2) == 0L) next
  }

  d2$w <- w
  des <- survey::svydesign(ids = ~1, weights = ~w, data = d2)
  fml <- stats::as.formula(paste0(outcome, " ~ splines::ns(co2, ", df, ")"))
  fit <- try(survey::svyglm(fml, design = des, family = quasibinomial()), silent = TRUE)
  if (inherits(fit, "try-error")) next
}

```

```

newd <- data.frame(co2 = xseq)
pr   <- predict_link_svyglm(fit, newd)
eta_list[[length(eta_list) + 1L]] <- pr$fit
var_list[[length(var_list) + 1L]] <- pr$se.fit^2
}

m <- length(eta_list)
if (m == 0L) stop("No successful fits to pool for ", group, " / ", outcome)

ETA <- do.call(cbind, eta_list) # ngrid x m
VAR <- do.call(cbind, var_list) # ngrid x m

if (m == 1L) {
  etaBar <- as.numeric(ETA)
  Tvar   <- as.numeric(VAR)
} else {
  etaBar <- rowMeans(ETA)
  Wbar   <- rowMeans(VAR)
  B      <- apply(ETA, 1, stats::var)
  Tvar   <- Wbar + (1 + 1/m) * B
}
seBar <- sqrt(pmax(Tvar, 0))

tibble::tibble(
  co2    = xseq,
  yhat   = plogis(etaBar),
  lower  = plogis(etaBar - 1.96 * seBar),
  upper  = plogis(etaBar + 1.96 * seBar),
  group  = group
)
}

mk_curves <- function(outcome)
  dplyr::bind_rows(
    pool_rcs_curve("ABG", outcome, df = 4, grid_n = 220, trim = c(0.02, 0.98)),

```

```

    pool_rcs_curve("VBG", outcome, df = 4, grid_n = 220, trim = c(0.02, 0.98))
  )

cur_imv   <- mk_curves("imv_proc")
cur_niv   <- mk_curves("niv_proc")
cur_death <- mk_curves("death_60d")
cur_hcrcf <- mk_curves("hypercap_resp_failure")

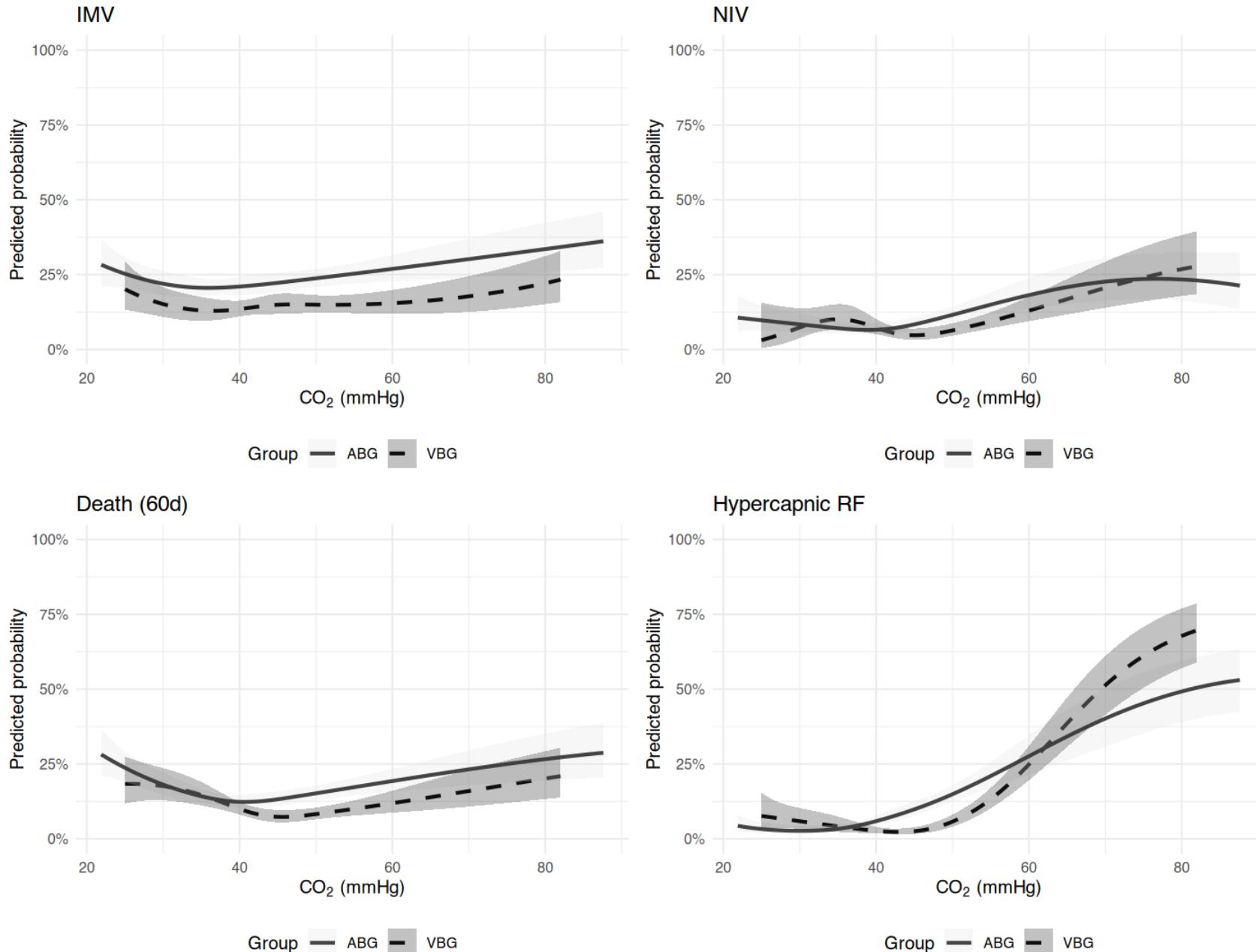
plt_gray <- function(dat, title) {
  ggplot(dat, aes(x = co2, y = yhat, linetype = group)) +
    geom_line(color = "black", linewidth = 1) +
    geom_ribbon(aes(ymin = lower, ymax = upper, fill = group), alpha = 0.3, color = NA) +
    scale_fill_manual(values = c("ABG" = "gray90", "VBG" = "gray20")) +
    scale_linetype_manual(values = c("ABG" = "solid", "VBG" = "dashed")) +
    scale_y_continuous(limits = c(0, 1),
                       labels = scales::percent_format(accuracy = 1)) +
    labs(title = title,
         x = expression(CO[2]~"(mmHg)"),
         y = "Predicted probability",
         fill = "Group", linetype = "Group") +
    theme_minimal(base_size = 10) +
    theme(legend.position = "bottom")
}

(patchwork::wrap_plots(
  plt_gray(cur_imv,    "IMV"),
  plt_gray(cur_niv,    "NIV"),
  plt_gray(cur_death,  "Death (60d)"),
  plt_gray(cur_hcrcf,  "Hypercapnic RF"),
  ncol = 2
) +
  plot_annotation(
    title = expression(
      paste("MI-pooled, propensity-weighted predicted probability: ABG vs VBG CO"[2],
            " (restricted cubic splines, 2-98% range)"))
)

```

)

MI-pooled, propensity-weighted predicted probability: ABG vs VBG CO<sub>2</sub> (restricted cubic splines, 2–98% range)



*Chunk ipw-rcs-overlay-mi-abg-vbg runtime: 1.60 s*

## 5.10 16) Save, export, and session info

```
saveRDS(list(abg = abg_results, vbg = vbg_results), "mi_pooled_results.rds")
```

*Chunk mi-save-exports runtime: 0.00 s*

```
sessionInfo()
```

```
R version 4.5.0 (2025-04-11)
Platform: aarch64-apple-darwin20
Running under: macOS 26.1
```

```
Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3.12.1
```

```
locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: America/Denver
tzcode source: internal
```

```
attached base packages:
[1] splines   grid      parallel  stats      graphics  grDevices utils
[8] datasets  methods   base
```

```
other attached packages:
[1] shapviz_0.10.2    fastshap_0.1.1    progressr_0.18.0
[4] future.apply_1.20.0 future_1.67.0    mitools_2.4
[7] skimr_2.2.1       visdat_0.6.0     naniar_1.1.0
[10] miceadds_3.18-36  mice_3.18.0     sensitivitymw_2.1
[13] lubridate_1.9.4    tibble_3.3.0     survey_4.4-8
```

```
[16] survival_3.8-3      Matrix_1.7-4       rms_8.0-0
[19] Hmisc_5.2-3        patchwork_1.3.2   officer_0.7.0
[22] modelsummary_2.5.0 scales_1.4.0       labelled_2.15.0
[25] haven_2.5.5        gt_1.1.0         ggplot2_4.0.0
[28] gbm_2.2.2          flextable_0.9.10 dplyr_1.1.4
[31] codebookkr_0.1.8   cobalt_4.6.1     broom_1.0.11
[34] WeightIt_1.5.0    purrr_1.2.0      gtsummary_2.4.0
[37] kableExtra_1.4.0
```

loaded via a namespace (and not attached):

```
[1] polspline_1.1.25      datawizard_1.2.0      rpart_4.1.24
[4] lifecycle_1.0.4        Rdpack_2.6.4        globals_0.18.0
[7] lattice_0.22-7        MASS_7.3-65        insight_1.4.2
[10] backports_1.5.0       magrittr_2.0.4      rmarkdown_2.30
[13] yaml_2.3.10          zip_2.3.3         askpass_1.2.1
[16] DBI_1.2.3            minqa_1.2.8       RColorBrewer_1.1-3
[19] multcomp_1.4-28       nnet_7.3-20        TH.data_1.1-4
[22] sandwich_3.1-1       gdtools_0.4.3      listenv_0.9.1
[25] cards_0.7.0           MatrixModels_0.5-4 performance_0.15.1
[28] parallelly_1.45.1     svglite_2.2.1      commonmark_2.0.0
[31] codetools_0.2-20      xml2_1.4.0        tidyselect_1.2.1
[34] shape_1.4.6.1         farver_2.1.2       lme4_1.1-38
[37] effectsize_1.0.1      base64enc_0.1-3    jsonlite_2.0.0
[40] mitml_0.4-5           Formula_1.2-5      iterators_1.0.14
[43] emmeans_1.11.2-8      systemfonts_1.2.3 foreach_1.5.2
[46] tools_4.5.0            ragg_1.5.0         Rcpp_1.1.0
[49] glue_1.8.0             gridExtra_2.3      pan_1.9
[52] mgcv_1.9-3            xfun_0.53         chk_0.10.0
[55] withr_3.0.2           fastmap_1.2.0     boot_1.3-32
[58] SparseM_1.84-2        openssl_2.3.3     litedown_0.7
[61] digest_0.6.37          timechange_0.3.0   R6_2.6.1
[64] estimability_1.5.1    textshaping_1.0.3  colorspace_2.1-2
[67] markdown_2.0            utf8_1.2.6         tidyverse_1.3.1
[70] generics_0.1.4          fontLiberation_0.1.0 data.table_1.17.8
[73] htmlwidgets_1.6.4      parameters_0.28.2 pkgconfig_2.0.3
[76] gtable_0.3.6           lmtest_0.9-40     S7_0.2.0
```

```
[79] htmltools_0.5.8.1      fontBitstreamVera_0.1.1 reformulas_0.4.2
[82] knitr_1.50              rstudioapi_0.17.1     uuid_1.2-1
[85] coda_0.19-4.1          checkmate_2.3.3      nlme_3.1-168
[88] nloptr_2.2.1           repr_1.1.7          zoo_1.8-14
[91] stringr_1.6.0          foreign_0.8-90      pillar_1.11.1
[94] vctrs_0.6.5            jomo_2.7-6          xtable_1.8-4
[97] cluster_2.1.8.1        htmlTable_2.4.3     evaluate_1.0.5
[100] tinytex_0.57           magick_2.9.0         mvtnorm_1.3-3
[103] cli_3.6.5              compiler_4.5.0      rlang_1.1.6
[106] crayon_1.5.3           labeling_0.4.3     forcats_1.0.1
[109] fs_1.6.6                stringi_1.8.7      viridisLite_0.4.2
[112] tables_0.9.31          glmnet_4.1-10       bayestestR_0.17.0
[115] quantreg_6.1            fontquiver_0.2.1    hms_1.1.4
[118] rbibutils_2.4           xgboost_1.7.11.1
```

*Chunk mi-session runtime: 0.07 s*