ABG-VBG Analysis

Brian Locke, Anila Mehta

Table of contents

1 TODO

☐ Need to finish comment style and labeling throughout the new code.

☐ **VBG winsorization bug -** In your VBG IPW block you compute cut <- quantile(w, c(0.01, 1), na.rm = TRUE) and then clamp to [cut[1], cut[2]]. Because cut[2] is the *maximum* weight, there is no upper-tail trimming. This is correct (because it is inverse propensity of sampling weighting - so only those not sampled are up-weighted. This should be fixed.

Mix of standardized and raw differences in balance plots. The love plot warns that "Standardized mean differences and raw mean differences are present in the same plot." This happens when bal.tab uses raw diffs for binary variables and SMD for continuous ones. Set binary = "std" (and, commonly, s.d.denom = "pooled") in your bal.tab() call so *all* displayed metrics are SMDs and the warning disappears. Evidence: the warning text appears right under the love plot code. Where you compute balance, change:

```
cobalt::bal.tab(..., un = TRUE, m.threshold = 0.1)
```

to:

```
cobalt::bal.tab(..., un = TRUE, m.threshold = 0.1,
                binary = "std", s.d.denom = "pooled")
```

☐ **Document still contains the "Calculated ABG (Farkas)" section. All the calculated sections should be removed - not being included in the final analysis.** You mentioned removing it; the PDF still shows the section and models. Evidence: heading and code for calc_abg and hypercapnia_calc.

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.

Table title mismatch. The gt title says *"Odds Ratios for VBG Hypercapnia (>45 mmHg)'s association with.."* immediately above the Calculated-ABG section. That label is misleading (looks like a copy/paste). Evidence: the title right before the calc-ABG heading.

The three category PaCO2s in the imputed version earlier should be changed to mirror the ones used earlier - hypocapnia, eucapnia, and hypercapnia

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.

Unit tests to add for Data hygiene / preprocessing, Propensity (WeightIt + GBM), Outcome modeling (svyglm), Imputation (mice), Explainability (fastshap + shapviz)

- **Type locks** for key fields (sex, encounter_type, race_ethnicity, outcomes). Assert intended classes and 2 levels after coercion and *before* MI and modeling.

☐ **Missingness invariants.** Assert the count of non-missing rows for each endpoint × exposure before/after MI; fail fast if massive drops occur.

Deterministic sanity pass. With m=1 and maxit=1, check that predictors/blocks match expectations (no outcomes in predictors; treatment not imputed).

Weight diagnostics. Assert: (i) mean(w)=1, (ii) fraction of weights at each winsor cutoff $< 2\%$, (iii) effective sample size (ESS) $> 20\%$ of N after trimming.

Balance targets. Assert max(|SMD|) < 0.1 across imputations; fail with a concise table of offenders if violated. (Use binary="std".)

Common support. For each imputation, assert predicted propensity in [, 1−] (e.g., =.01) with <1% violations.

☐ **Coefficient extraction guard.** Your helper already checks for finite coef/vcov. Keep it. Add a check that the treatment column is present and non-aliased (no 1-level factor in any imputation).

Alignment test. Before running SHAP, verify setequal(colnames(X_used), gbm$var.names) and that factor levels in X_used are a subset of training levels; stop with a helpful message otherwise.

Seed escrow. Echo seeds used for MI, GBM, and SHAP into the PDF.

Regression calibration checks (new)

- **Rename the gt title** above the calc-ABG section (or remove the section) to prevent confusion in review copies.

1.1 Data Pre-processing

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

```
# put this in your first R chunk
if (!requireNamespace("kableExtra", quietly = TRUE)) install.packages("kableExtra")
library(kableExtra)
library(gtsummary)
library(purrr)        # functional programming

# globally tighten gtsummary/gt tables (smaller font + tighter padding)
gtsummary::theme_gtsummary_compact()
```

```
Setting theme "Compact"
```

```r
# helper: turn any gtsummary table into a PDF-safe, auto-scaling LaTeX table
to_pdf_table <- function(tbl, font_size = 8, landscape = FALSE,
                         label_col_width = NULL) {
  kbl <- gtsummary::as_kable(
    tbl,
    format    = "latex",
    booktabs  = TRUE,
    longtable = TRUE # allows multipage tables; repeats header with kableExtra option below
  )

  # optional: set a fixed width for the first (label) column to encourage wrapping
  if (!is.null(label_col_width)) {
    kbl <- kableExtra::column_spec(kbl, 1, width = label_col_width)
  }

  kbl <- kableExtra::kable_styling(
    kbl,
    latex_options = c("repeat_header", "hold_position", "scale_down"),
    font_size     = font_size
  )

  if (landscape) kbl <- kableExtra::landscape(kbl) # needs pdflscape (enabled above)
  kbl
}
```

```r
# Consolidated package management ------------------------------------------------
required_pkgs <- c(
  "WeightIt", "broom", "cobalt", "codebookr", "dplyr", "flextable", "parallel",
  "gbm", "ggplot2", "gt", "gtsummary", "haven", "labelled", "scales",
  "modelsummary", "officer", "patchwork", "rms", "survey", "tibble", "lubridate", "sensitivitymw"
)

# Install any missing packages (with dependencies)
missing_pkgs <- setdiff(required_pkgs, rownames(installed.packages()))
if (length(missing_pkgs)) {
  install.packages(missing_pkgs, dependencies = TRUE)
```

```
}

# Load (or attach) all required packages
invisible(lapply(required_pkgs, require, character.only = TRUE))
```

```
Loading required package: WeightIt

Loading required package: broom

Warning: package 'broom' was built under R version 4.5.2

Loading required package: cobalt

 cobalt (Version 4.6.1, Build Date: 2025-08-20)

Loading required package: codebookr

Loading required package: dplyr


Attaching package: 'dplyr'

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

    group_rows

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

```
Loading required package: flextable


Attaching package: 'flextable'

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

    compose

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

    continuous_summary

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

    as_image, footnote

Loading required package: parallel

Loading required package: gbm

Loaded gbm 2.2.2

This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

Loading required package: ggplot2

Loading required package: gt

Loading required package: haven

Loading required package: labelled
```

```
Loading required package: scales


Attaching package: 'scales'

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

    discard

Loading required package: modelsummary

Loading required package: officer

Loading required package: patchwork

Loading required package: rms

Loading required package: Hmisc


Attaching package: 'Hmisc'

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

    Mean

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

    html, latex

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

    src, summarize
```

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

    format.pval, units


Attaching package: 'rms'

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

    calibrate

Loading required package: survey

Loading required package: grid

Loading required package: Matrix

Loading required package: survival


Attaching package: 'survey'

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

    calibrate

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

    deff

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

    calibrate
```

```
The following object is masked from 'package:graphics':

    dotchart

Loading required package: tibble

Loading required package: lubridate


Attaching package: 'lubridate'

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

    date, intersect, setdiff, union

Loading required package: sensitivitymw
```

```r
# ensure predictable, writable figure path + robust PNG device
knitr::opts_chunk$set(
  fig.path = "figs/",   # short local dir for figures
  dev      = "png",
  dpi      = 144
)
dir.create("figs", showWarnings = FALSE, recursive = TRUE)
# on macOS and some setups this prevents device headaches
options(bitmapType = "cairo")

if (!requireNamespace("shapviz", quietly = TRUE) ||
    packageVersion("shapviz") < "0.2.0") {
  install.packages("shapviz")  # or: remotes::install_github("ModelOriented/shapviz")
}

if (interactive() && !requireNamespace("fastshap", quietly = TRUE)) {
  options(repos = c(CRAN = "https://cran.rstudio.com/"))
```

```
  install.packages("fastshap")
}

if (interactive() && !requireNamespace("fastshap", quietly = TRUE)) {
  options(repos = c(CRAN = "https://cran.rstudio.com/"))
  install.packages("DALEX")
}

if (interactive() && !requireNamespace("fastshap", quietly = TRUE)) {
  options(repos = c(CRAN = "https://cran.rstudio.com/"))
  install.packages("shapviz")
}
```

```
# Make gt tables robust in PDF: full width, caption, small font
gt_pdf <- function(x, title = NULL, subtitle = NULL) {
  out <- x |>
    gt::tab_options(
      table.width              = gt::pct(100),
      table.align              = "left",
      table.font.size          = gt::px(9),
      data_row.padding         = gt::px(1),
      column_labels.font.size  = gt::px(9),
      heading.title.font.size  = gt::px(10),
      heading.subtitle.font.size = gt::px(9)
    ) |>
    gt::opt_align_table_header(align = "left")
  if (!is.null(title))    out <- out |> gt::tab_caption(title)
  if (!is.null(subtitle)) out <- out |> gt::tab_source_note(subtitle)
  out
}
```

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'
```

```r
rdata_file <- file.path(data_dir_name, "full_trinetx.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.
```

```r
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'.
```

Creating subset_data

```r
set.seed(123)
rows_to_keep <- round(nrow(stata_data) * 0.005) #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
 862 1629
```

```r
dim(subset_data)
```

```
[1] 2491  546
```

Generating Codebook for the Full Dataset

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

```
Generating codebook for the dataset...
```

```r
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.
```

New Variable - Death at 60 days

```r
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)
  )
```

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

```
   0    1
2221  270
```

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

```
        0         1
0.8916098 0.1083902
```

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.1084  0.0000  1.0000
```

1.2 Baseline Tables

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   ~ "VBG_NoHypercapnia",
      has_vbg == 1 & hypercap_on_vbg == 1   ~ "VBG_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", "VBG_NoHypercapnia", "VBG_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
```

Generating Word Doc for Table 1A & 1B

Variable	**No ABG** N = 1,630[1]	**ABG_NoHypercapnia** N = 605[1]	**ABG_Hypercapnia** N = 256[1]
Age (years)	57.9 ± 17.9; 0.0/1,630.0 missing (0.0%)	60.4 ± 17.0; 0.0/605.0 missing (0.0%)	60.6 ± 16.7; 0.0/256.0 missing (0.0%)
Current BMI kg/m2	32.6 ± 8.8; 903.0/1,630.0 missing (55.4%)	28.3 ± 7.1; 347.0/605.0 missing (57.4%)	29.0 ± 7.9; 159.0/256.0 missing (62.1%)
sex_label			
Female	865 (53%)	267 (44%)	111 (43%)
Male	765 (47%)	338 (56%)	145 (57%)
race_ethnicity_label			
White	983 (60%)	341 (56%)	163 (64%)
Black or African American	317 (19%)	106 (18%)	37 (14%)
Hispanic	112 (6.9%)	36 (6.0%)	17 (6.6%)
Asian	20 (1.2%)	14 (2.3%)	1 (0.4%)
American Indian	12 (0.7%)	7 (1.2%)	2 (0.8%)
Pacific Islander	2 (0.1%)	1 (0.2%)	1 (0.4%)
Unknown	184 (11%)	100 (17%)	35 (14%)
location_label			
South	679 (42%)	337 (56%)	140 (55%)
Northeast	463 (28%)	106 (18%)	62 (24%)
Midwest	119 (7.3%)	36 (6.0%)	20 (7.8%)
West	369 (23%)	126 (21%)	34 (13%)
osa_label	297 (18%)	75 (12%)	48 (19%)
asthma_label	214 (13%)	57 (9.4%)	29 (11%)
copd_label	296 (18%)	91 (15%)	76 (30%)
chf_label	281 (17%)	133 (22%)	68 (27%)
nmd_label	66 (4.0%)	29 (4.8%)	7 (2.7%)
phtn_label	105 (6.4%)	49 (8.1%)	20 (7.8%)
ckd_label	300 (18%)	127 (21%)	55 (21%)
diabetes_label	483 (30%)	181 (30%)	72 (28%)
encounter_type_label			
Emergency	720 (44%)	94 (16%)	48 (19%)
Inpatient	910 (56%)	511 (84%)	208 (81%)
VBG PCO2	45.1 ± 11.1; 1,185.0/1,630.0 missing (72.7%)	43.2 ± 11.9; 417.0/605.0 missing (68.9%)	57.7 ± 19.0; 176.0/256.0 missing (68.8%)
Arterial PCO2	NA ± NA; 1,630.0/1,630.0 missing (100.0%)	35.3 ± 6.2; 0.0/605.0 missing (0.0%)	60.3 ± 25.4; 0.0/256.0 missing (0.0%)

[1]Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

Variable	**No VBG** N = 1,778[1]	**VBG_NoHypercapnia** N = 516[1]	**VBG_Hypercapnia** N = 197[1]
Age (years)	59.0 ± 17.8; 0.0/1,778.0 missing (0.0%)	57.9 ± 16.7; 0.0/516.0 missing (0.0%)	58.8 ± 17.3; 0.0/197.0 missing (0.0%)
Current BMI kg/m2	32.2 ± 8.7; 927.0/1,778.0 missing (52.1%)	27.8 ± 7.1; 339.0/516.0 missing (65.7%)	27.8 ± 8.1; 143.0/197.0 missing (72.6%)
sex_label			
Female	898 (51%)	255 (49%)	90 (46%)
Male	880 (49%)	261 (51%)	107 (54%)
race_ethnicity_label			
White	1,145 (64%)	235 (46%)	107 (54%)
Black or African American	309 (17%)	108 (21%)	43 (22%)
Hispanic	116 (6.5%)	38 (7.4%)	11 (5.6%)
Asian	19 (1.1%)	13 (2.5%)	3 (1.5%)
American Indian	9 (0.5%)	11 (2.1%)	1 (0.5%)
Pacific Islander	3 (0.2%)	1 (0.2%)	0 (0%)
Unknown	177 (10.0%)	110 (21%)	32 (16%)
location_label			
South	955 (54%)	130 (25%)	71 (36%)
Northeast	329 (19%)	220 (43%)	82 (42%)
Midwest	108 (6.1%)	47 (9.1%)	20 (10%)
West	386 (22%)	119 (23%)	24 (12%)
osa_label	303 (17%)	73 (14%)	44 (22%)
asthma_label	211 (12%)	62 (12%)	27 (14%)
copd_label	339 (19%)	66 (13%)	58 (29%)
chf_label	340 (19%)	90 (17%)	52 (26%)
nmd_label	79 (4.4%)	14 (2.7%)	9 (4.6%)
phtn_label	126 (7.1%)	30 (5.8%)	18 (9.1%)
ckd_label	341 (19%)	104 (20%)	37 (19%)
diabetes_label	511 (29%)	163 (32%)	62 (31%)
encounter_type_label			
Emergency	611 (34%)	182 (35%)	69 (35%)
Inpatient	1,167 (66%)	334 (65%)	128 (65%)
VBG PCO2	NA ± NA; 1,778.0/1,778.0 missing (100.0%)	40.1 ± 6.5; 0.0/516.0 missing (0.0%)	61.3 ± 13.7; 0.0/197.0 missing (0.0%)
Arterial PCO2	42.4 ± 19.2; 1,185.0/1,778.0 missing (66.6%)	39.1 ± 11.4; 336.0/516.0 missing (65.1%)	52.1 ± 23.8; 109.0/197.0 missing (55.3%)

[1]Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

```r
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")
```

1.2.1 Table 1 (Overall ABG/VBG status)

NEW Table 1

```r
# 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}%)"
    ),
    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
```

1.2.2 Table 2 (Hypercapnia within cohorts)

NEW Table 2

Table 1

Variable	Everyone[1]	**Did not get ABG** N = 1,630[1]	**Did get ABG** N = 861[1]
Age (years)	58.8 ± 17.6; 0.0/2,491.0 missing (0.0%)	57.9 ± 17.9; 0.0/1,630.0 missing (0.0%)	60.4 ± 16.9; 0.0/861.0 missing (0.0%)
Current BMI kg/m2	31.2 ± 8.6; 1,409.0/2,491.0 missing (56.6%)	32.6 ± 8.8; 903.0/1,630.0 missing (55.4%)	28.5 ± 7.3; 506.0/861.0 missing (58.8%
sex_label			
Female	1,243 (50%)	865 (53%)	378 (44%)
Male	1,248 (50%)	765 (47%)	483 (56%)
race_ethnicity_label			
White	1,487 (60%)	983 (60%)	504 (59%)
Black or African American	460 (18%)	317 (19%)	143 (17%)
Hispanic	165 (6.6%)	112 (6.9%)	53 (6.2%)
Asian	35 (1.4%)	20 (1.2%)	15 (1.7%)
American Indian	21 (0.8%)	12 (0.7%)	9 (1.0%)
Pacific Islander	4 (0.2%)	2 (0.1%)	2 (0.2%)
Unknown	319 (13%)	184 (11%)	135 (16%)
location_label			
South	1,156 (46%)	679 (42%)	477 (55%)
Northeast	631 (25%)	463 (28%)	168 (20%)
Midwest	175 (7.0%)	119 (7.3%)	56 (6.5%)
West	529 (21%)	369 (23%)	160 (19%)
osa_label	420 (17%)	297 (18%)	123 (14%)
asthma_label	300 (12%)	214 (13%)	86 (10.0%)
copd_label	463 (19%)	296 (18%)	167 (19%)
chf_label	482 (19%)	281 (17%)	201 (23%)
nmd_label	102 (4.1%)	66 (4.0%)	36 (4.2%)
phtn_label	174 (7.0%)	105 (6.4%)	69 (8.0%)
ckd_label	482 (19%)	300 (18%)	182 (21%)
diabetes_label	736 (30%)	483 (30%)	253 (29%)
encounter_type_label			
Emergency	862 (35%)	720 (44%)	142 (16%)
Inpatient	1,629 (65%)	910 (56%)	719 (84%)
VBG PCO2	46.0 ± 13.1; 1,778.0/2,491.0 missing (71.4%)	45.1 ± 11.1; 1,185.0/1,630.0 missing (72.7%)	47.5 ± 15.8; 593.0/861.0 missing (68.9%
Arterial PCO2	42.7 ± 18.7; 1,630.0/2,491.0 missing (65.4%)	NA ± NA; 1,630.0/1,630.0 missing (100.0%)	42.7 ± 18.7; 0.0/861.0 missing (0.0%)

[1]Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

```r
# 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(
    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
```

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") %>%
```

	Table 1		Table 2	
Variable	**Got ABG & Hypercapnia[1]**	**Got ABG & No hypercapnia[1]**	**Got VBG & Hypercapnia[1]**	
Age (years)	60.6 ± 16.7; 0.0/256.0 missing (0.0%)	60.4 ± 17.0; 0.0/605.0 missing (0.0%)	58.8 ± 17.3; 0.0/197.0 missing (0.0%)	5
Current BMI kg/m2	29.0 ± 7.9; 159.0/256.0 missing (62.1%)	28.3 ± 7.1; 347.0/605.0 missing (57.4%)	27.8 ± 8.1; 143.0/197.0 missing (72.6%)	27
sex_label				
Female	111 (43%)	267 (44%)	90 (46%)	
Male	145 (57%)	338 (56%)	107 (54%)	
race_ethnicity_label				
White	163 (64%)	341 (56%)	107 (54%)	
Black or African American	37 (14%)	106 (18%)	43 (22%)	
Hispanic	17 (6.6%)	36 (6.0%)	11 (5.6%)	
Asian	1 (0.4%)	14 (2.3%)	3 (1.5%)	
American Indian	2 (0.8%)	7 (1.2%)	1 (0.5%)	
Pacific Islander	1 (0.4%)	1 (0.2%)	0 (0%)	
Unknown	35 (14%)	100 (17%)	32 (16%)	
location_label				
South	140 (55%)	337 (56%)	71 (36%)	
Northeast	62 (24%)	106 (18%)	82 (42%)	
Midwest	20 (7.8%)	36 (6.0%)	20 (10%)	
West	34 (13%)	126 (21%)	24 (12%)	
osa_label	48 (19%)	75 (12%)	44 (22%)	
asthma_label	29 (11%)	57 (9.4%)	27 (14%)	
copd_label	76 (30%)	91 (15%)	58 (29%)	
chf_label	68 (27%)	133 (22%)	52 (26%)	
nmd_label	7 (2.7%)	29 (4.8%)	9 (4.6%)	
phtn_label	20 (7.8%)	49 (8.1%)	18 (9.1%)	
ckd_label	55 (21%)	127 (21%)	37 (19%)	
diabetes_label	72 (28%)	181 (30%)	62 (31%)	
encounter_type_label				
Emergency	48 (19%)	94 (16%)	69 (35%)	
Inpatient	208 (81%)	511 (84%)	128 (65%)	
VBG PCO2	57.7 ± 19.0; 176.0/256.0 missing (68.8%)	43.2 ± 11.9; 417.0/605.0 missing (68.9%)	61.3 ± 13.7; 0.0/197.0 missing (0.0%)	4
Arterial PCO2	60.3 ± 25.4; 0.0/256.0 missing (0.0%)	35.3 ± 6.2; 0.0/605.0 missing (0.0%)	52.1 ± 23.8; 109.0/197.0 missing (55.3%)	39.

[1]Mean ± SD; N Missing/No. obs. missing (% Missing); n (%)

```
  body_add_flextable(ft2)

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

1.3 Unweighted Binary Logistic Regressions

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

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

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)     -2.28209    0.07297 -31.276   <2e-16 ***
hypercap_on_abg  1.47538    0.15373   9.597   <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: 1775.9  on 2490  degrees of freedom
Residual deviance: 1695.6  on 2489  degrees of freedom
AIC: 1699.6

Number of Fisher Scoring iterations: 5
```

```r
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)	0.102071	0.0729658	-31.27609	0	0.0882171	0.1174451
hypercap_on_abg	4.372718	0.1537269	9.59744	0	3.2247983	5.8953649

```r
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)     -2.88707    0.09459 -30.523  < 2e-16 ***
hypercap_on_abg  0.86773    0.21611   4.015 5.94e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1122.7  on 2490  degrees of freedom
Residual deviance: 1108.8  on 2489  degrees of freedom
AIC: 1112.8

Number of Fisher Scoring iterations: 5
```

```r
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)	0.0557393	0.0945852	-30.523493	0.00e+00	0.0460622	0.0667603
hypercap_on_abg	2.3815059	0.2161122	4.015197	5.94e-05	1.5352081	3.5918457

```r
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)     -2.18979    0.07029 -31.152  < 2e-16 ***
hypercap_on_abg  0.64459    0.17861   3.609 0.000308 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1709.5  on 2490  degrees of freedom
Residual deviance: 1697.7  on 2489  degrees of freedom
AIC: 1701.7

Number of Fisher Scoring iterations: 4
```

```r
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)	0.1119403	0.0702937	-31.152025	0.0000000	0.0972791	0.1281605

term	estimate	std.error	statistic	p.value	conf.low	conf.high
hypercap_on_abg	1.9052132	0.1786134	3.608877	0.0003075	1.3292820	2.6813577

```r
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)      -3.3604     0.1175  -28.61   <2e-16 ***
hypercap_on_abg   2.2618     0.1861   12.15   <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: 1072.39  on 2490  degrees of freedom
Residual deviance:  944.55  on 2489  degrees of freedom
AIC: 948.55

Number of Fisher Scoring iterations: 6
```

```r
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)	0.0347222	0.1174576	-28.60927	0	0.0273463	0.0433655
hypercap_on_abg	9.6000000	0.1860903	12.15411	0	6.6570448	13.8235066

	Intubated	NIV	Death	ICD Hyper
hypercap_on_abg	4.37 (3.22, 5.90)	2.38 (1.54, 3.59)	1.91 (1.33, 2.68)	9.60 (6.66, 13.82)

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

```
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     = ".*",                    # drop all goodness-of-fit rows
  fmt          = 2,                       # 2 decimal places everywhere
  output       = "gt"
) |>
  gt_pdf(title = "Odds Ratios for ABG Hypercapnia (>45 mmHg)'s association with...")
```

Unweighted VBG Group

1.3.2 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)     -2.13302    0.06784 -31.442  < 2e-16 ***
hypercap_on_vbg  0.85727    0.18534   4.625 3.74e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1775.9  on 2490  degrees of freedom
Residual deviance: 1757.1  on 2489  degrees of freedom
AIC: 1761.1

Number of Fisher Scoring iterations: 4
```

```r
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)	0.1184788	0.0678409	-31.441510	0.0e+00	0.1034798	0.135022
hypercap_on_vbg	2.3567153	0.1853421	4.625333	3.7e-06	1.6227791	3.361327

```r
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)     -2.87076    0.09268 -30.974  < 2e-16 ***
```

```
hypercap_on_vbg  0.94214    0.23325   4.039 5.36e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1122.7  on 2490  degrees of freedom
Residual deviance: 1108.9  on 2489  degrees of freedom
AIC: 1112.9

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)	0.0566559	0.0926834	-30.973825	0.00e+00	0.0470051	0.0676195
hypercap_on_vbg	2.5654661	0.2332470	4.039238	5.36e-05	1.5926070	3.9883008

```
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)     -2.16566    0.06872 -31.514  < 2e-16 ***
hypercap_on_vbg  0.59827    0.20067   2.981  0.00287 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1709.5  on 2490  degrees of freedom
Residual deviance: 1701.5  on 2489  degrees of freedom
AIC: 1705.5

Number of Fisher Scoring iterations: 4
```

```r
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)	0.1146744	0.0687213	-31.513619	0.0000000	0.0999759	0.1309024
hypercap_on_vbg	1.8189664	0.2006723	2.981321	0.0028701	1.2102830	2.6639180

```r
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)      -3.1867     0.1070  -29.79   <2e-16 ***
hypercap_on_vbg   2.0540     0.1975   10.40   <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: 1072.39  on 2490  degrees of freedom
Residual deviance:  984.46  on 2489  degrees of freedom
AIC: 988.46

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)	0.0413073	0.1069717	-29.79028	0	0.0332606	0.050612
hypercap_on_vbg	7.7988052	0.1974530	10.40233	0	5.2685351	11.442123

Display model coefficients for binary hypercapnia on VBG logistic regression

```
modelsummary(
  list("Intubated" = logit_intubated_vbg,
       "NIV"       = logit_niv_vbg,
       "Death"     = logit_death_vbg,
       "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)'s association with...")
```

Calculated ABG from VBG Using Farkas equation - binary predictor

	Intubated	NIV	Death	ICD Hyper
hypercap_on_vbg	2.36 (1.62, 3.36)	2.57 (1.59, 3.99)	1.82 (1.21, 2.66)	7.80 (5.27, 11.44)

1.3.3 Calculated ABG (Farkas) binary hypercapnia models

```
subset_data <- subset_data %>%
  mutate(
    calc_abg = vbg_co2 - (0.22 * (93 - vbg_o2sat))
  )
subset_data <- subset_data %>%
  mutate(
    hypercapnia_calc = ifelse(calc_abg > 45, 1, 0)
  )
with(subset_data, table(hypercapnia_calc,niv_proc))
```

```
               niv_proc
hypercapnia_calc   0   1
               0 193  13
               1  72  13
```

```
logit_intubated_calc <- glm(imv_proc ~ hypercapnia_calc, data = subset_data, family = binomial)
summary(logit_intubated_calc)
```

```
Call:
glm(formula = imv_proc ~ hypercapnia_calc, family = binomial,
    data = subset_data)

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -1.9796     0.2134  -9.278   <2e-16 ***
hypercapnia_calc  0.5181     0.3500   1.480    0.139
---
```

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

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 236.63  on 290  degrees of freedom
Residual deviance: 234.51  on 289  degrees of freedom
  (2200 observations deleted due to missingness)
AIC: 238.51

Number of Fisher Scoring iterations: 4
```

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

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1381215	0.2133642	-9.278134	0.0000000	0.0887614	0.2056083
hypercapnia_calc	1.6788406	0.3500243	1.480193	0.1388218	0.8323508	3.3103225

```r
logit_niv_calc <- glm(niv_proc ~ hypercapnia_calc, data = subset_data, family = binomial)
summary(logit_niv_calc)
```

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

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.6977     0.2865  -9.415   <2e-16 ***
hypercapnia_calc  0.9860     0.4158   2.371   0.0177 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 175.20  on 290  degrees of freedom
Residual deviance: 169.72  on 289  degrees of freedom
  (2200 observations deleted due to missingness)
AIC: 173.72

Number of Fisher Scoring iterations: 5
```

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

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0673575	0.286537	-9.414982	0.0000000	0.0365045	0.1132877
hypercapnia_calc	2.6805556	0.415831	2.371214	0.0177298	1.1776471	6.1085641

```r
logit_death_calc <- glm(death_60d ~ hypercapnia_calc, data = subset_data, family = binomial)
summary(logit_death_calc)
```

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

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -1.7693     0.1975  -8.957   <2e-16 ***
hypercapnia_calc  0.3078     0.3406   0.904    0.366
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 254.02  on 290  degrees of freedom
Residual deviance: 253.22  on 289  degrees of freedom
  (2200 observations deleted due to missingness)
AIC: 257.22

Number of Fisher Scoring iterations: 4
```

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

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1704545	0.1975225	-8.9573918	0.0000000	0.1135396	0.2469652
hypercapnia_calc	1.3603865	0.3405993	0.9036097	0.3662024	0.6849459	2.6236295

```r
logit_icd_calc <- glm(hypercap_resp_failure ~ hypercapnia_calc, data = subset_data, family = binomial)
summary(logit_icd_calc)
```

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

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       -3.3474     0.3846  -8.705  < 2e-16 ***
hypercapnia_calc   2.5828     0.4496   5.744 9.23e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 209.86  on 290  degrees of freedom
Residual deviance: 167.37  on 289  degrees of freedom
```

```
  (2200 observations deleted due to missingness)
AIC: 171.37

Number of Fisher Scoring iterations: 6
```

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

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0351759	0.3845544	-8.704605	0	0.0149647	0.0691403
hypercapnia_calc	13.2339901	0.4496227	5.744347	0	5.7654707	34.4230562

Display regression coefficients for binary Farkas adjustment (hypercapnia yes/no as predictor)

```
modelsummary(
  list("Intubated" = logit_intubated_calc,
       "NIV"       = logit_niv_calc,
       "Death"     = logit_death_calc,
       "ICD Hyper" = logit_icd_calc),
  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 Calculated Hypercapnia (>45 mmHg)'s association with...")
```

1.3.4 Combined OR visualizations (binary hypercapnia)

Odds Ratio Graph of all 3 simple, binary-predictor logistic regressions

	Intubated	NIV	Death	ICD Hyper
hypercapnia_calc	1.68 (0.83, 3.31)	2.68 (1.18, 6.11)	1.36 (0.68, 2.62)	13.23 (5.77, 34.42)

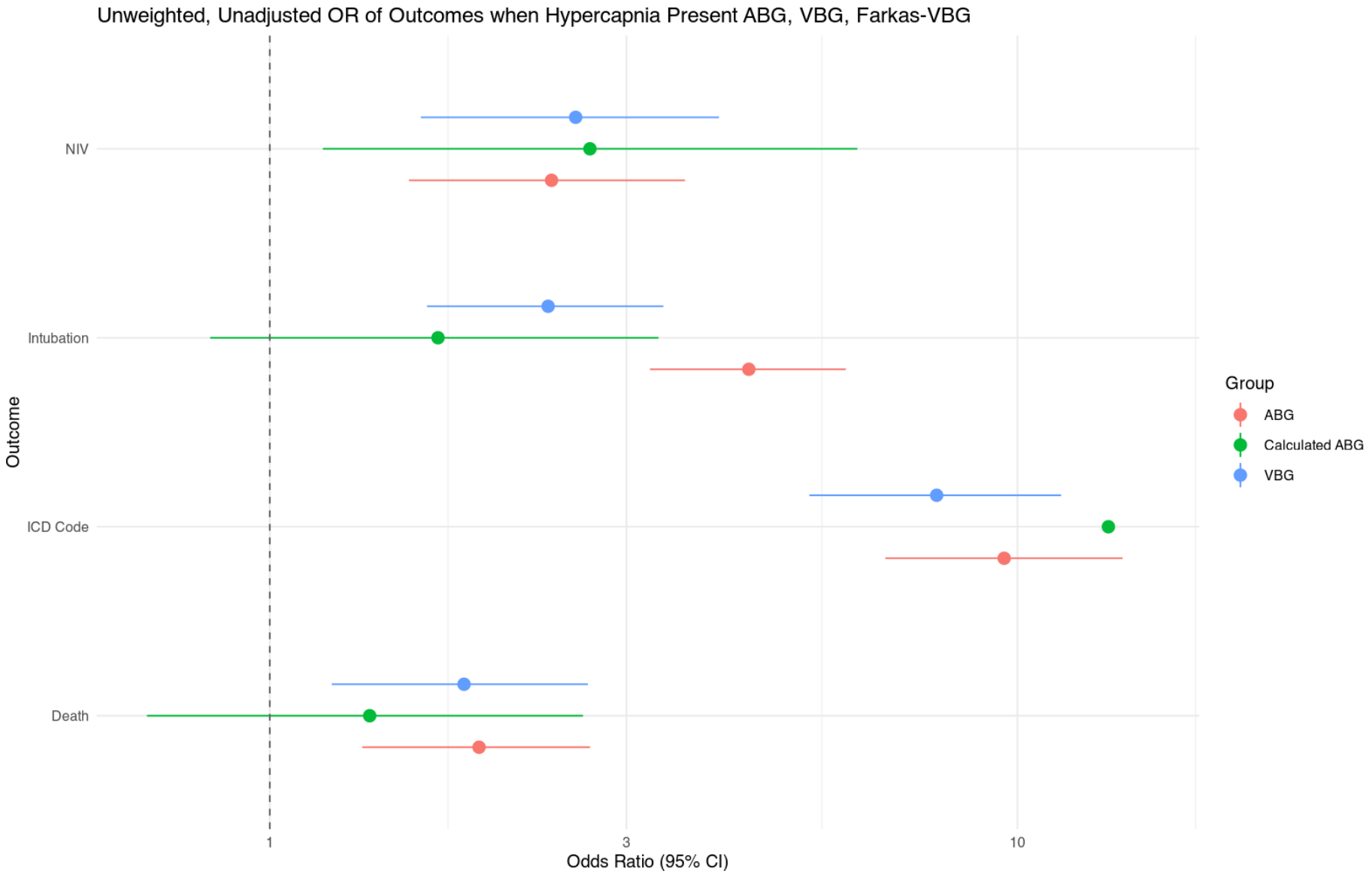
```r
tidy_with_labels <- function(model, group_label, outcome_label) {
  tidy(model, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term == "hypercap_on_abg" | term == "hypercap_on_vbg" | term == "hypercapnia_calc") %>%
    mutate(
      group = group_label,
      outcome = outcome_label
    )
}

# --- ABG Models ---
abg_intub <- tidy_with_labels(glm(imv_proc ~ hypercap_on_abg, data = subset_data, family = binomial), "ABG", "Intubation")
abg_niv   <- tidy_with_labels(glm(niv_proc ~ hypercap_on_abg, data = subset_data, family = binomial), "ABG", "NIV")
abg_death <- tidy_with_labels(glm(death_60d ~ hypercap_on_abg, data = subset_data, family = binomial), "ABG", "Death")
abg_icd   <- tidy_with_labels(glm(hypercap_resp_failure ~ hypercap_on_abg, data = subset_data, family = binomial), "ABG", "ICD C

# --- VBG Models ---
vbg_intub <- tidy_with_labels(glm(imv_proc ~ hypercap_on_vbg, data = subset_data, family = binomial), "VBG", "Intubation")
vbg_niv   <- tidy_with_labels(glm(niv_proc ~ hypercap_on_vbg, data = subset_data, family = binomial), "VBG", "NIV")
vbg_death <- tidy_with_labels(glm(death_60d ~ hypercap_on_vbg, data = subset_data, family = binomial), "VBG", "Death")
vbg_icd   <- tidy_with_labels(glm(hypercap_resp_failure ~ hypercap_on_vbg, data = subset_data, family = binomial), "VBG", "ICD C

# --- Calculated ABG Models ---
calc_intub <- tidy_with_labels(glm(imv_proc ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculated ABG", "Intuba
calc_niv   <- tidy_with_labels(glm(niv_proc ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculated ABG", "NIV")
calc_death <- tidy_with_labels(glm(death_60d ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculated ABG", "Death
calc_icd   <- tidy_with_labels(glm(hypercap_resp_failure ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculated

# --- Combine all model results ---
combined_or_df <- bind_rows(
  abg_intub, abg_niv, abg_death, abg_icd,
  vbg_intub, vbg_niv, vbg_death, vbg_icd,
  calc_intub, calc_niv, calc_death, calc_icd
```

```
)
```

```
ggplot(combined_or_df, aes(x = outcome, y = estimate, ymin = conf.low, ymax = conf.high, color = group)) +
  geom_pointrange(position = position_dodge(width = 0.5), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "gray40") +
  coord_flip() +
  labs(
    title = "Unweighted, Unadjusted OR of Outcomes when Hypercapnia Present ABG, VBG, Farkas-VBG ",
    x = "Outcome",
    y = "Odds Ratio (95% CI)",
    color = "Group"
  ) +
  scale_y_log10(limits = c(-0.5, 15)) +  # optional log scale for better spacing
  theme_minimal(base_size = 10)
```

```
Warning in transform$transform(limits): NaNs produced
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_segment()`).
```

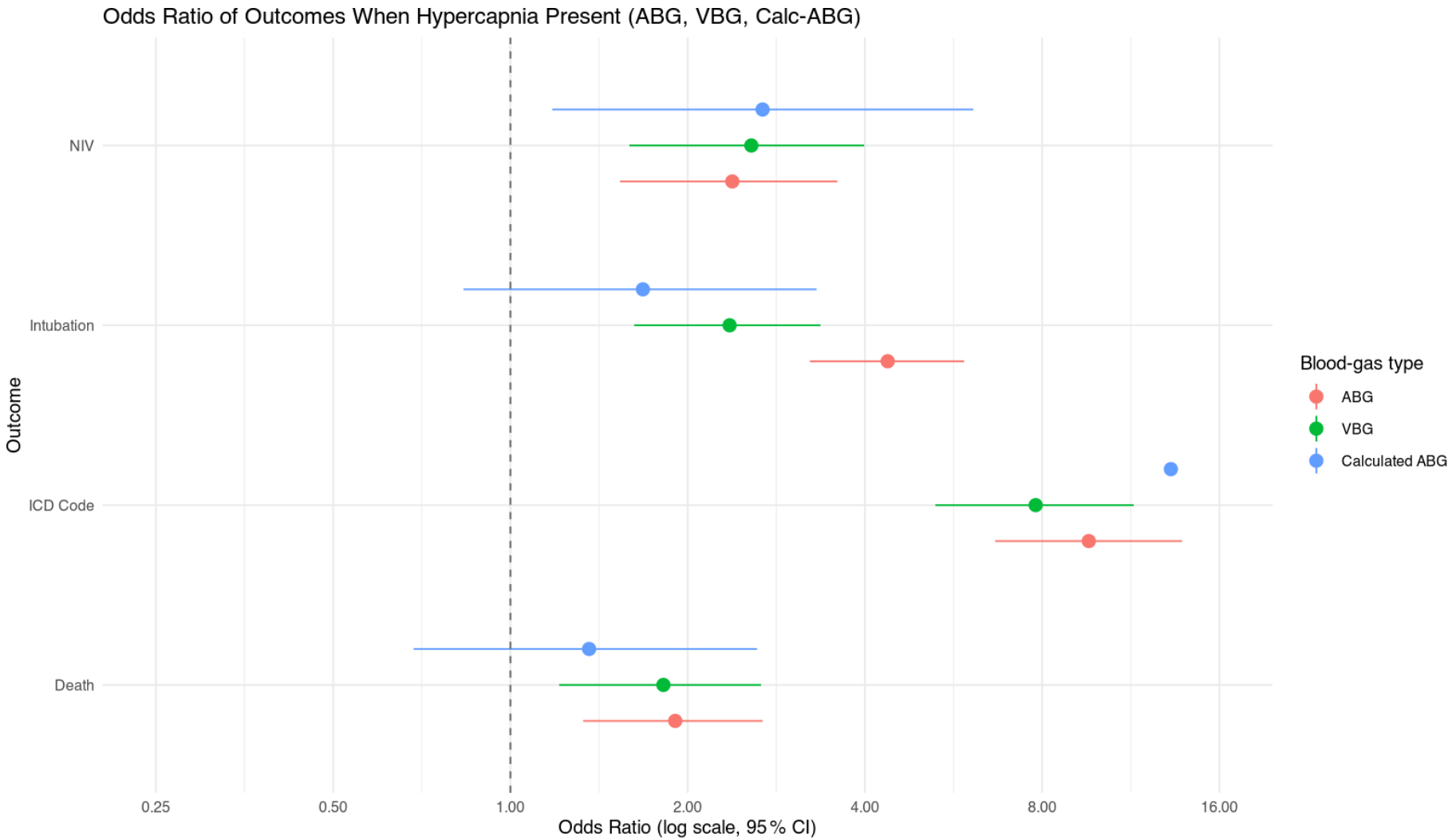



```
combined_or_df$group <- factor(combined_or_df$group,
  levels = c("ABG", "VBG", "Calculated ABG"))
```

```r
#   prerequisites

# order groups before plotting
combined_or_df$group <- factor(
  combined_or_df$group,
  levels = c("ABG", "VBG", "Calculated ABG")
)

#   plot
ggplot(
  combined_or_df,
  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") +
  ## NOTE:  scale_y_log10 applies to the axis that *becomes horizontal* after coord_flip()
  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 Ratio of Outcomes When Hypercapnia Present (ABG, VBG, Calc-ABG)",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95 % CI)",
    color  = "Blood-gas type",
```

```
    caption = paste(
      "Odds ratios are computed *within* each blood-gas cohort.",
      "Numerator = patients who received that blood-gas and **had** hypercapnia;",
      "denominator = patients who received the same blood-gas and **did not** have hypercapnia.",
      "Because the underlying cohorts differ (ABG, VBG, Calculated ABG),",
      "denominators are not identical across groups.",
      sep = "\n"
    )
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_segment()`).
```



1.3.5 Three-level PCO categories (unweighted)

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

```r
subset_data <- subset_data %>%
  mutate(
    pco2_cat_abg = case_when(
      !is.na(paco2) & paco2 < 35 ~ "Below normal",
      !is.na(paco2) & paco2 > 45 ~ "Above normal",
      !is.na(paco2)              ~ "Normal"
    ),
    pco2_cat_vbg = case_when(
      !is.na(vbg_co2) & vbg_co2 < 40 ~ "Below normal",
      !is.na(vbg_co2) & vbg_co2 > 50 ~ "Above normal",
      !is.na(vbg_co2)                ~ "Normal"
    ),
    pco2_cat_calc = case_when(
      !is.na(calc_abg) & calc_abg < 35 ~ "Below normal",
      !is.na(calc_abg) & calc_abg > 45 ~ "Above normal",
      !is.na(calc_abg)                 ~ "Normal"
    )
  ) %>%
  mutate(
    across(starts_with("pco2_cat"),
           ~factor(.x, levels = c("Normal", "Below normal", "Above normal")))
  )

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
    )
}
```

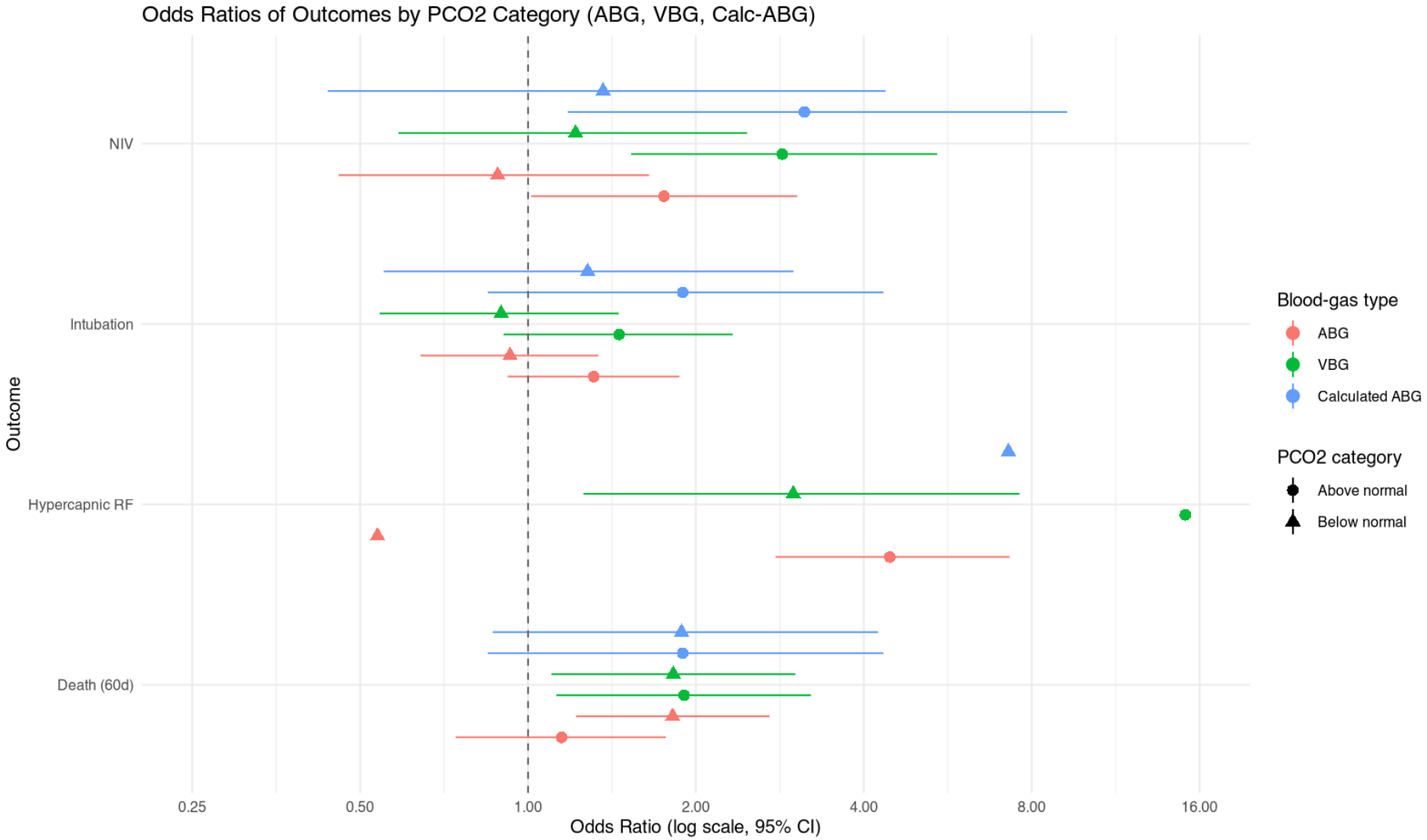
```r
outcomes <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

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")),
  lapply(outcomes, function(o) run_logit(subset_data, o, "pco2_cat_calc", "Calculated ABG"))
)

combined_or_df <- results %>%
  mutate(
    exposure = recode(term,
                      "pco2_cat_abgBelow normal"   = "Below normal",
                      "pco2_cat_abgAbove normal"   = "Above normal",
                      "pco2_cat_vbgBelow normal"   = "Below normal",
                      "pco2_cat_vbgAbove normal"   = "Above normal",
                      "pco2_cat_calcBelow normal"  = "Below normal",
                      "pco2_cat_calcAbove normal"  = "Above normal"),
    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)
```

```r
library(scales)

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

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.",
      "Below normal: <35 mmHg. Above normal: >45 mmHg (ABG, Calc-ABG) or >50 mmHg (VBG).",
      "Because the underlying cohorts differ (ABG, VBG, Calculated ABG), denominators are not identical across groups.",
      sep = "\n"
    )
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_pointrange()`).
```

```
Warning: Removed 3 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



Odds ratios are computed within each blood-gas cohort.
Reference = patients in the normal PCO2 range.
Below normal: <35 mmHg. Above normal: >45 mmHg (ABG, Calc-ABG) or >50 mmHg (VBG).
Because the underlying cohorts differ (ABG, VBG, Calculated ABG), denominators are not identical across groups.

1.4 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")
```

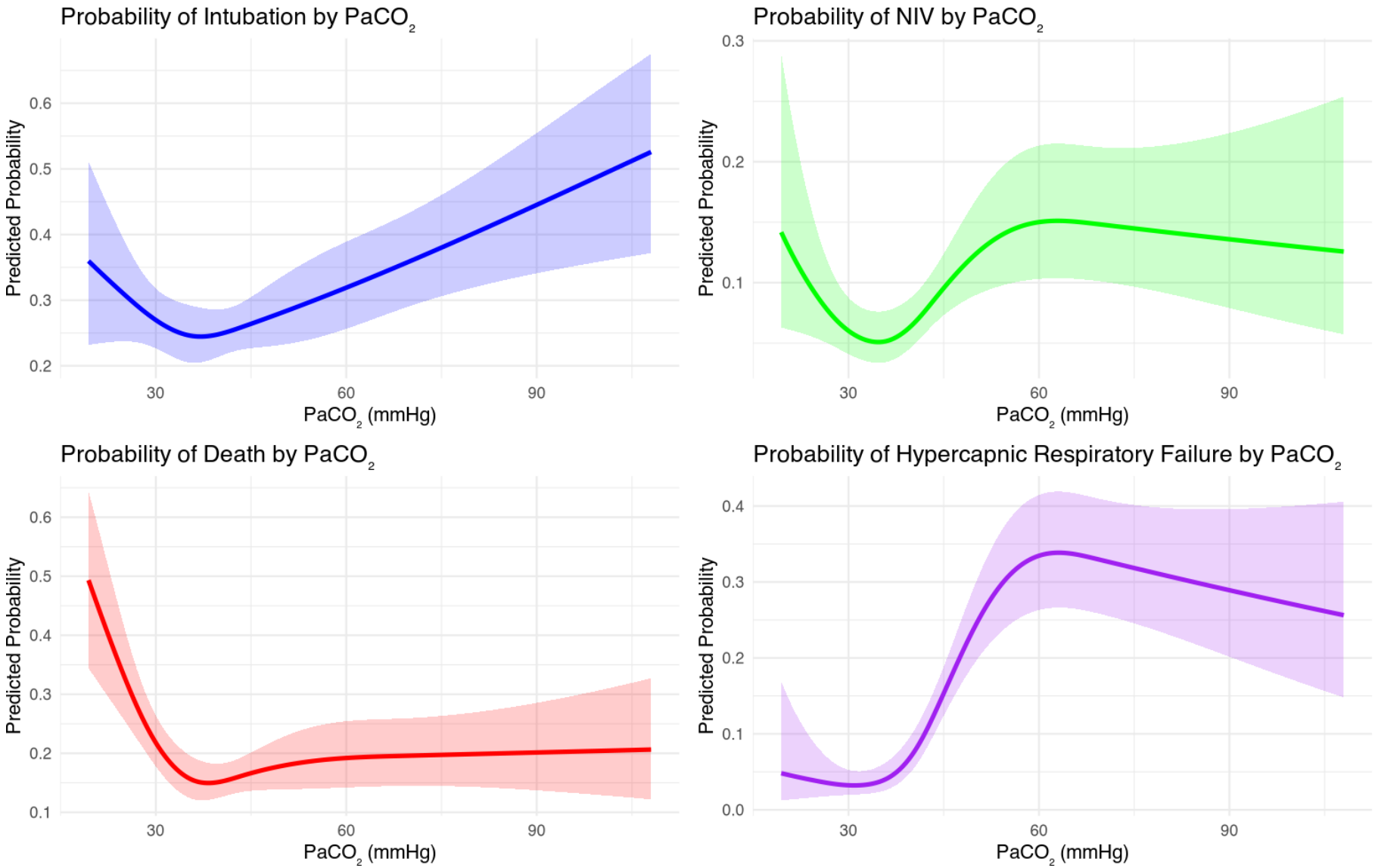
Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```
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 PaCO ",
       x = "PaCO  (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 PaCO ",
       x = "PaCO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()
```

```r
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 PaCO ",
       x = "PaCO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

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

plot_hcrf <- ggplot(pred_hcrf, 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 PaCO ",
       x = "PaCO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

(plot_imv | plot_niv) / (plot_death | plot_hcrf)
```



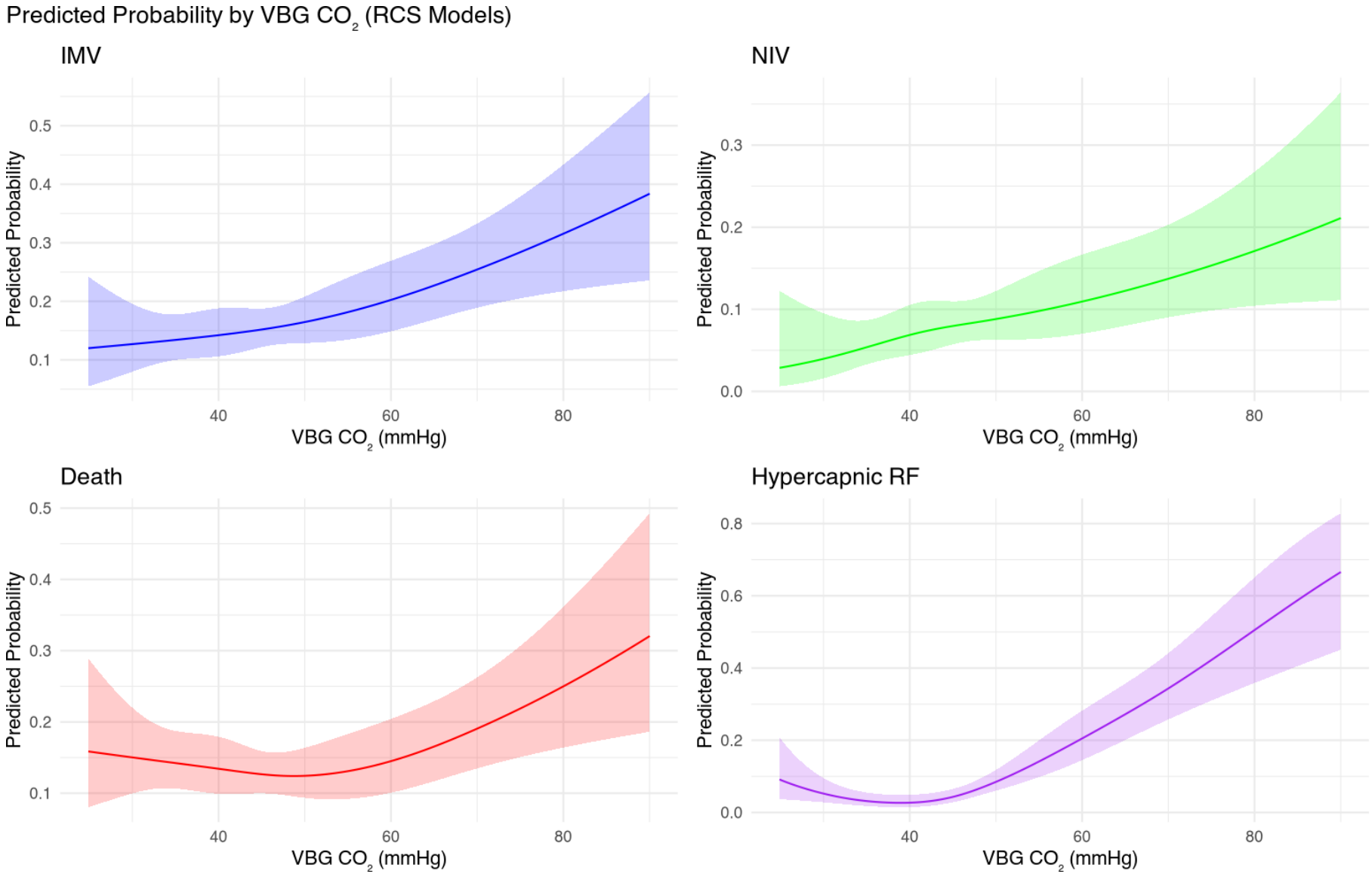
Unweighted, Restricted Cubic Spline - VBG

```
# --- VBG dataset ---
subset_data_vbg <- subset_data %>%
```

```r
  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")
```

```r
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_hcrf_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_hcrf_vbg <- as.data.frame(Predict(fit_hcrf_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 CO  (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 CO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()
```

```r
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 CO  (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 CO  (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 CO  (RCS Models)")
```

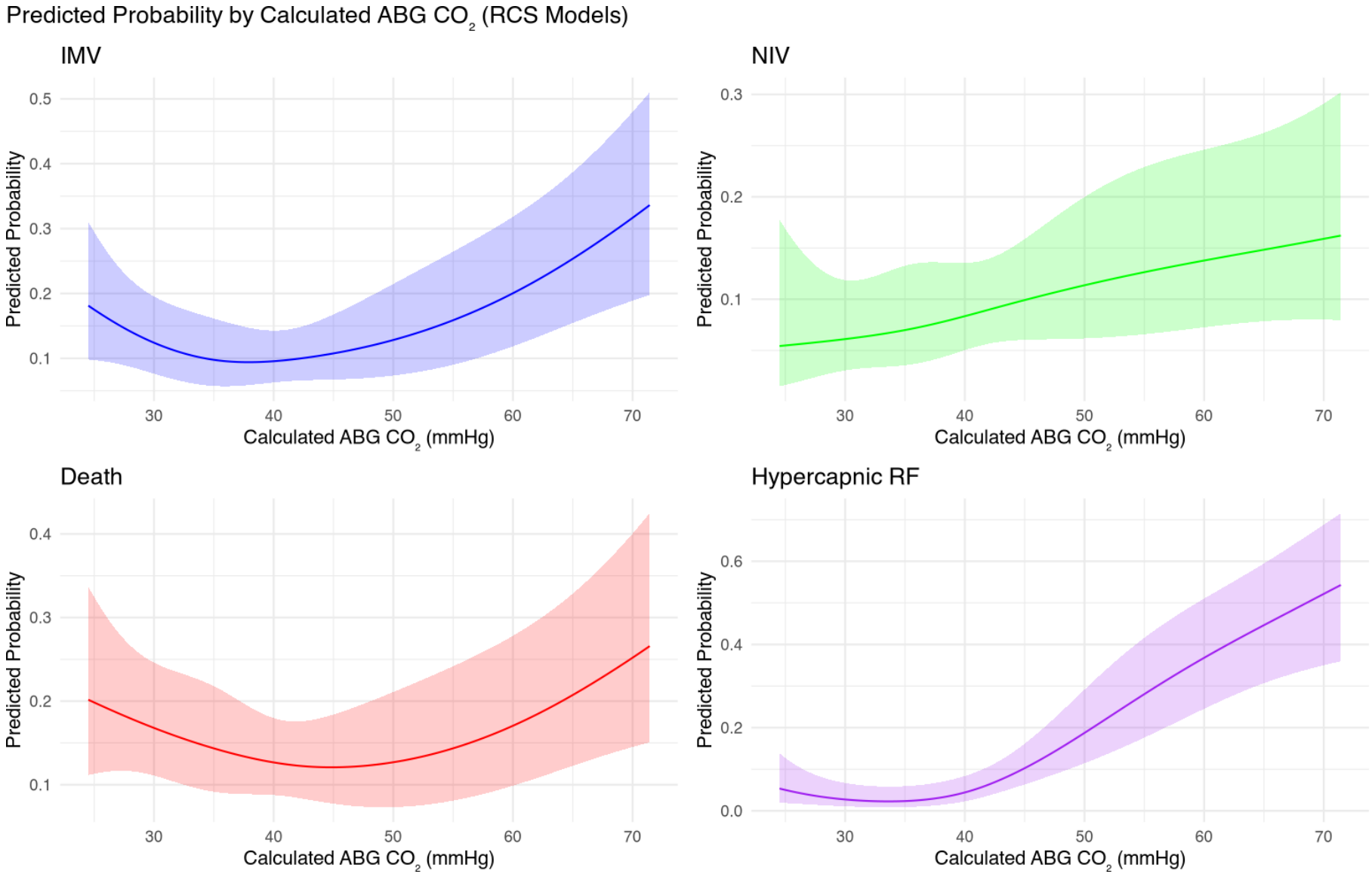


Unweighted, Restricted Cubic Spline Logistic Regressio - Calculated VBG to ABG (Farkas VBG Adjustment)

```
subset_data_calc <- subset_data %>%
  select(calc_abg, imv_proc, niv_proc, death_60d, hypercap_resp_failure) %>%
```

```r
  filter(!is.na(calc_abg) & complete.cases(.))

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

fit_imv_abg   <- lrm(imv_proc ~ rcs(calc_abg, 4), data = subset_data_calc)
fit_niv_abg   <- lrm(niv_proc ~ rcs(calc_abg, 4), data = subset_data_calc)
fit_death_abg  <- lrm(death_60d ~ rcs(calc_abg, 4), data = subset_data_calc)
fit_hcrf_abg   <- lrm(hypercap_resp_failure ~ rcs(calc_abg, 4), data = subset_data_calc)

pred_imv_abg   <- as.data.frame(Predict(fit_imv_abg, calc_abg, fun = plogis))
pred_niv_abg   <- as.data.frame(Predict(fit_niv_abg, calc_abg, fun = plogis))
pred_death_abg <- as.data.frame(Predict(fit_death_abg, calc_abg, fun = plogis))
pred_hcrf_abg  <- as.data.frame(Predict(fit_hcrf_abg, calc_abg, fun = plogis))

plot_imv_abg <- ggplot(pred_imv_abg, aes(x = calc_abg, y = yhat)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "blue", alpha = 0.2) +
  labs(title = "IMV", x = "Calculated ABG CO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

plot_niv_abg <- ggplot(pred_niv_abg, aes(x = calc_abg, y = yhat)) +
  geom_line(color = "green") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "green", alpha = 0.2) +
  labs(title = "NIV", x = "Calculated ABG CO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

plot_death_abg <- ggplot(pred_death_abg, aes(x = calc_abg, y = yhat)) +
  geom_line(color = "red") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.2) +
  labs(title = "Death", x = "Calculated ABG CO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

plot_hcrf_abg <- ggplot(pred_hcrf_abg, aes(x = calc_abg, y = yhat)) +
  geom_line(color = "purple") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.2) +
```

```
  labs(title = "Hypercapnic RF", x = "Calculated ABG CO  (mmHg)", y = "Predicted Probability") +
  theme_minimal()

((plot_imv_abg | plot_niv_abg) /
 (plot_death_abg | plot_hcrf_abg)) +
plot_annotation(title = "Predicted Probability by Calculated ABG CO  (RCS Models)")
```


Predicted Probability by Calculated ABG CO_2 (RCS Models)
IMV
NIV
Death
Hypercapnic RF
Predicted Probability
Calculated ABG CO_2 (mmHg)
0.1
0.2
0.3
0.4
0.5
0.6
0.0
30
40
50
60
70

1.5 Inverse Propensity Weighting

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

Current propensity score uses **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.

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"))
```

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

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

weight_model <- weightit(
  has_abg ~ age_at_encounter + sex + factor(race_ethnicity) + curr_bmi + copd + asthma + osa + chf + acute_nmd + phtn + ckd + dm
```

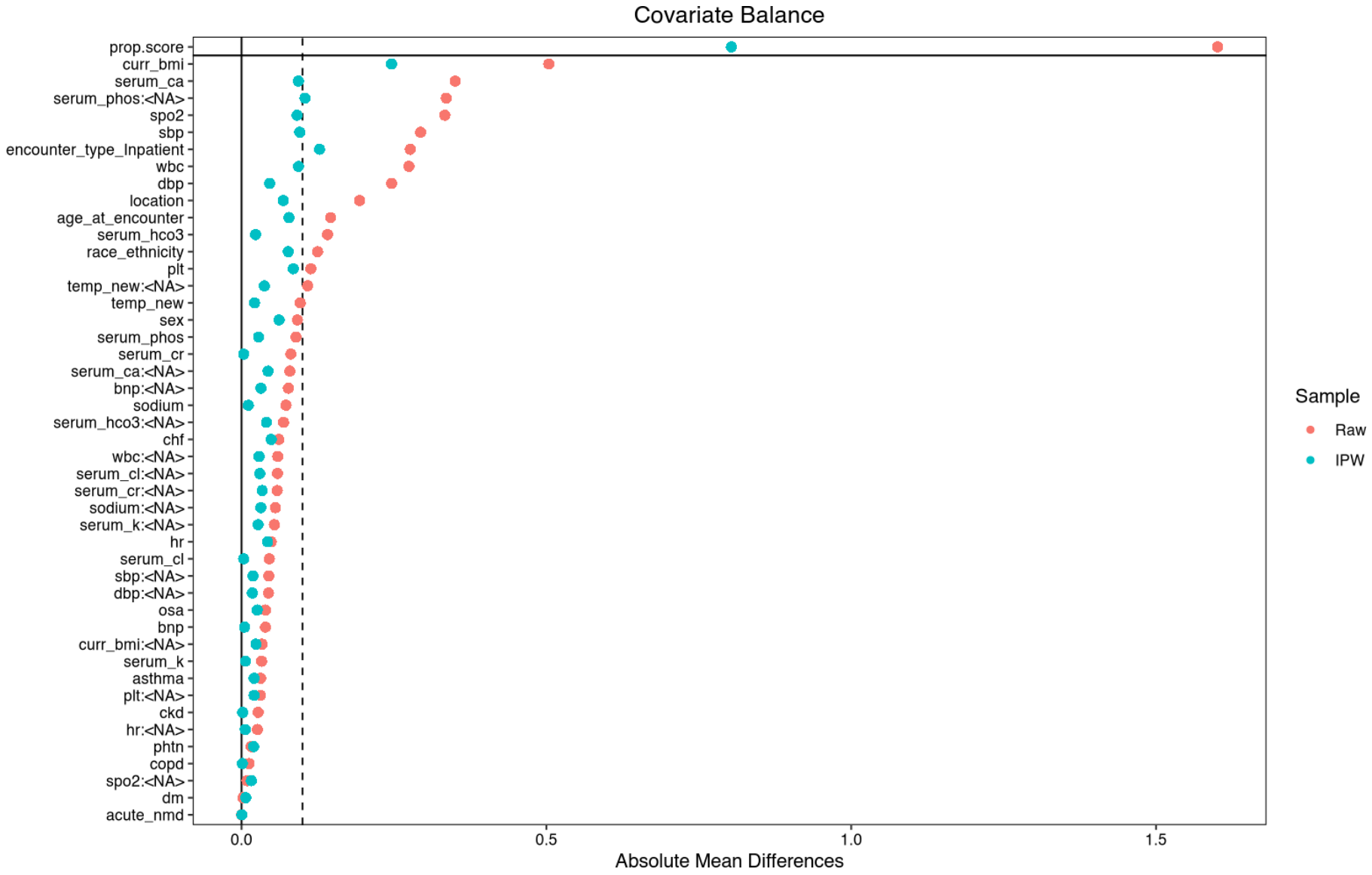
```
  data        = subset_data,
  method      = "gbm",
  estimand    = "ATE",
  missing    = "ind",
  include.obj = TRUE,        # ← REQUIRED for importance/SHAP
  n.trees     = 1500,        #decreased trees from 3000 to 1500
  interaction.depth = 3,
  shrinkage   = 0.01,
  bag.fraction= 0.8,   #increased bagging 0.6 to 0.8 - less overfit extremes
  cv.folds    = 5,
  stop.method = "es.mean",
  n.cores     = parallel::detectCores()
)
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, 1), 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)
```

```
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")
)
```

```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```




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

#   5. outcome models (examples)
```

```r
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
  8.793272   6.121617  12.630916

$NIV
OR.has_abg        LCL        UCL
  1.645595   1.144064   2.366984

$Death
OR.has_abg        LCL        UCL
  2.312599   1.742440   3.069325

$ICD
OR.has_abg        LCL        UCL
  4.158374   2.720780   6.355558
```

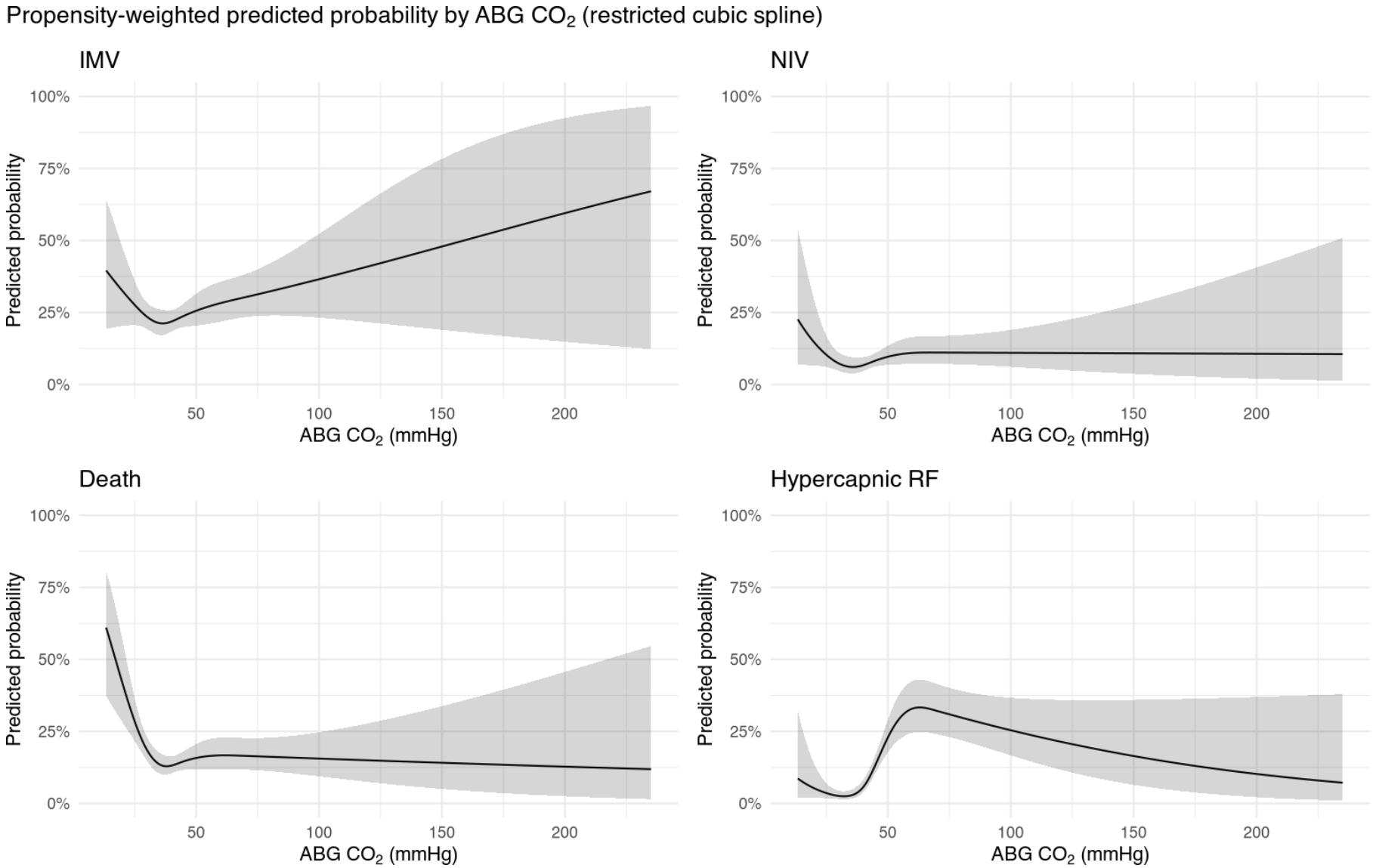
Inverse Propensity-Weighted Logistic Regressions with CO2 predictor represented as a restricted cubic spline.

1.5.2 ABG IPW spline models

```r
# 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 PaCO
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 PaCO  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_hcrf_abg  <- mkpred(fit_hcrf_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_hcrf_abg,  "Hypercapnic RF"),
    ncol = 2
  )
) +
  plot_annotation(
    title = expression(
      paste("Propensity-weighted predicted probability by ABG CO"[2],
            " (restricted cubic spline)")
    )
)
```



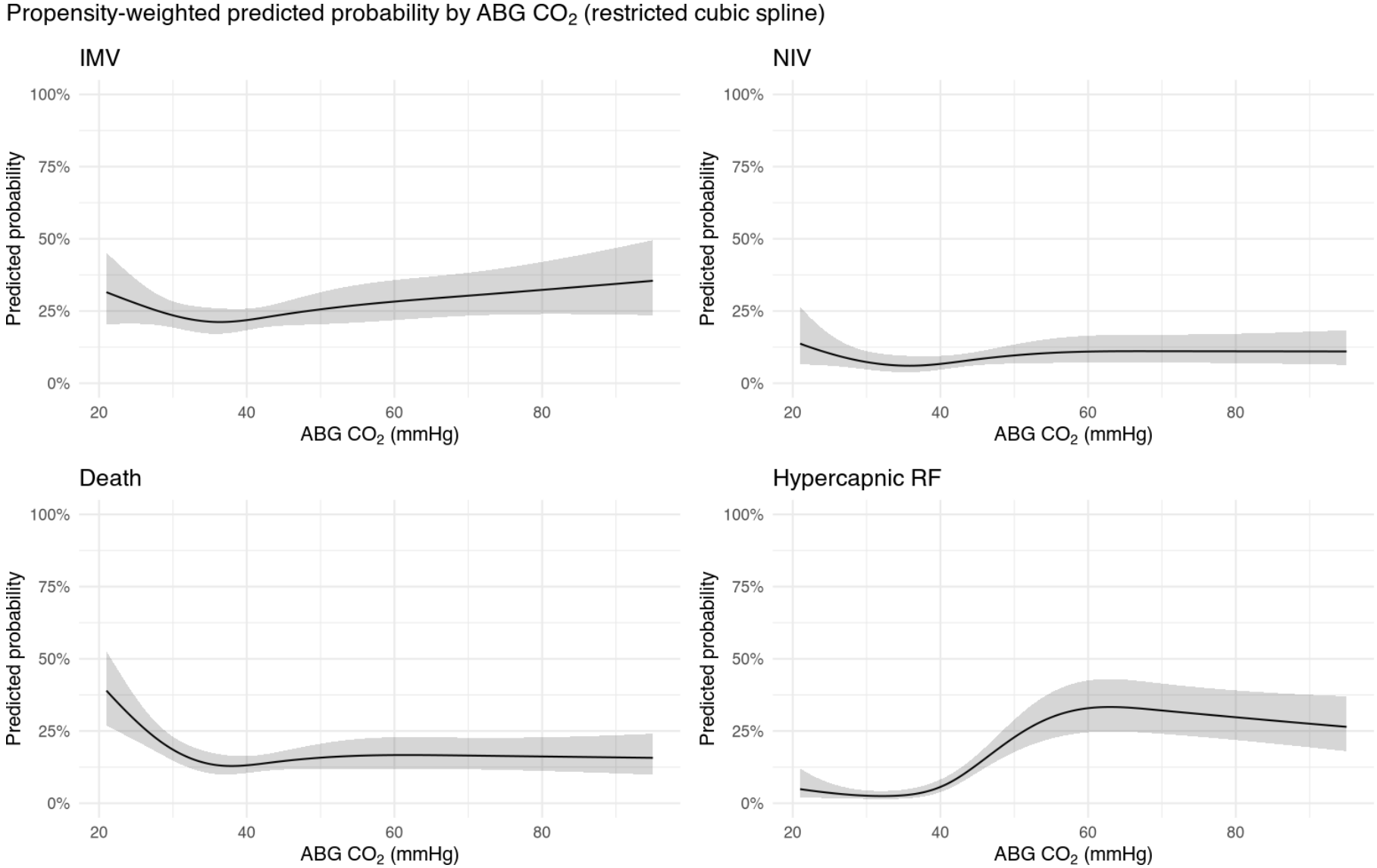
Restricting plots bewtween 0.02 and 0.98

1.5.3 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 PaCO  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)
```

```r
  # 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_hcrf_abg  <- mkpred(fit_hcrf_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_hcrf_abg,  "Hypercapnic RF"),
    ncol = 2
```

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



VBG - changed trees and bag fraction

1.5.4 VBG IPW weighting and spline models

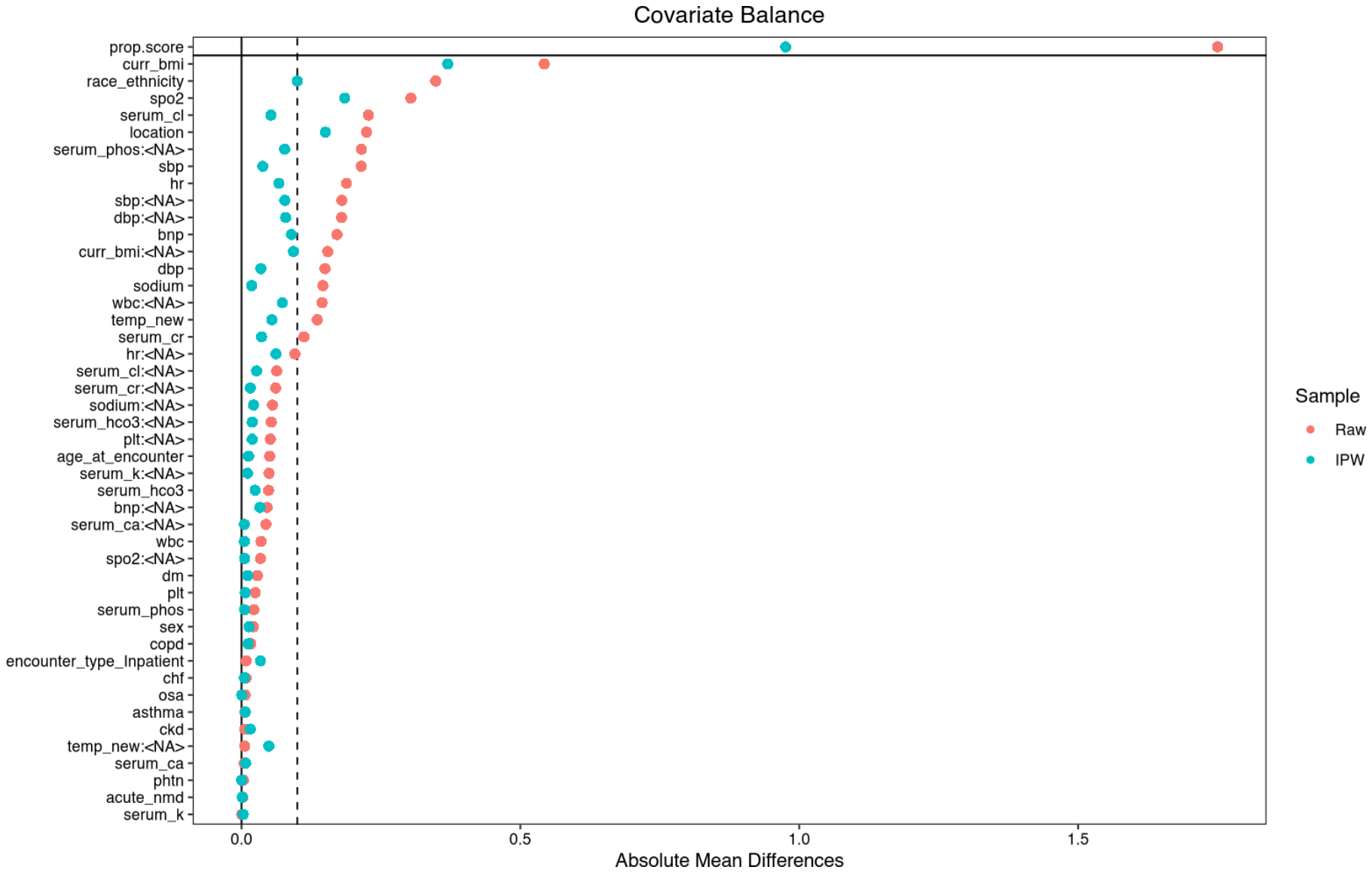
```
#  Inverse-propensity weighting & outcome modelling for **VBG** cohort
#    - mirrored 1-to-1 to the validated ABG workflow

set.seed(42)

# 1. IPW for VBG ---------------------------------------------------------------
w_vbg <- weightit(
  has_vbg ~ age_at_encounter + sex + factor(race_ethnicity) + curr_bmi +
    copd + asthma + osa + chf + acute_nmd + phtn + ckd + dm +
    factor(location) + factor(encounter_type) + temp_new + sbp + dbp + hr + spo2 + sodium + serum_cr + serum_hco3 + serum_cl +
    serum_k + wbc + plt + bnp + serum_phos + serum_ca,
  data        = subset_data,
  method      = "gbm",
  estimand    = "ATE",
  missing    = "ind",
  include.obj = TRUE,        # ← REQUIRED for importance/SHAP
  n.trees     = 1500,
  interaction.depth = 3,
  shrinkage   = 0.01,
  bag.fraction= 0.8,
  cv.folds    = 5,
  stop.method = "es.mean",
  n.cores     = parallel::detectCores()
)

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

w_vbg$weights   <- w
```

```r
subset_data$w_vbg <- w

v_bal <- bal.tab(w_vbg, un = TRUE, m.threshold = 0.1)
```

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

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

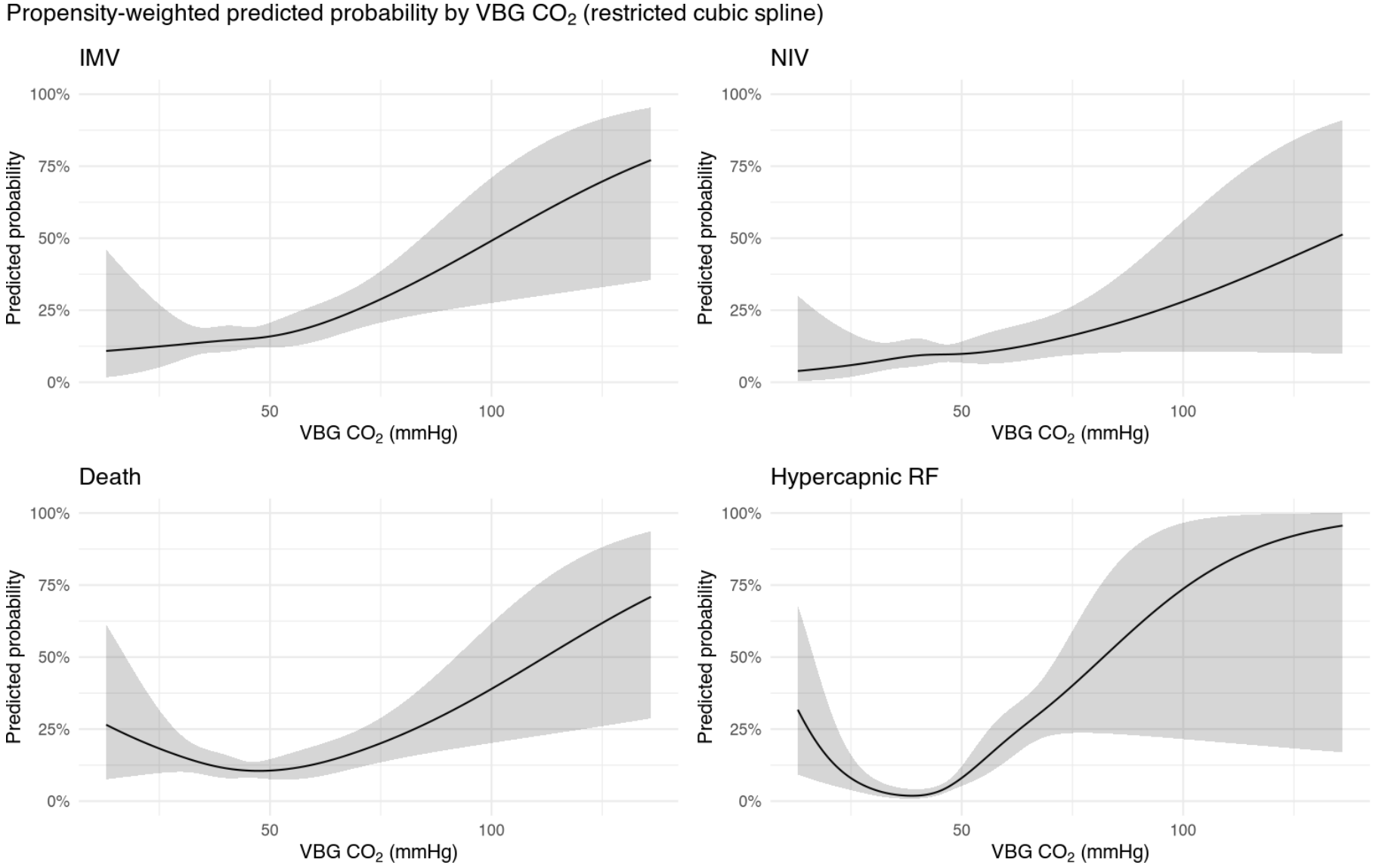
```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```




```
# 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)")
    )
)
```



Calculated VBG to ABG / Farkas

1.5.5 Calculated ABG IPW weighting and spline models

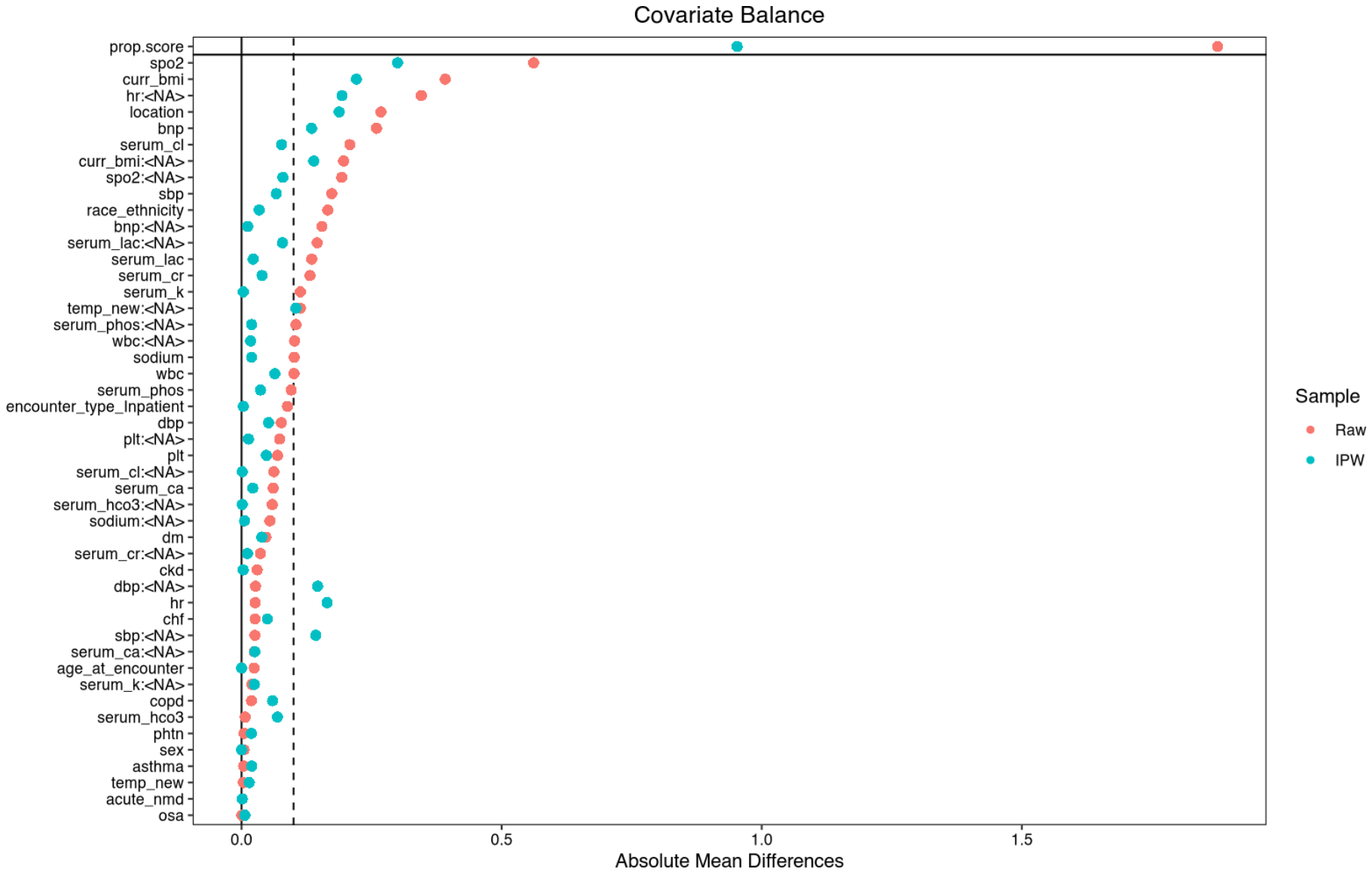
```
#  Propensity-weighted spline models for **Calculated ABG CO **
#    (weights still derive from propensity to receive a VBG)

#   1. define the new treatment variable ---------------------------------------
subset_data <- subset_data %>%
  mutate(
    has_vbg_co2_o2_sat = if_else(
      !is.na(vbg_co2)  & vbg_co2  != 0 &
      !is.na(vbg_o2sat) & vbg_o2sat != 0,
      1, 0
    )
  )

# quick sanity check
# table(subset_data$has_vbg_co2_o2_sat, useNA = "ifany")

#   2. fit the GBM propensity model --------------------------------------------
set.seed(42)

w_vbg_calc <- weightit(
  has_vbg_co2_o2_sat ~ age_at_encounter + sex + factor(race_ethnicity) + curr_bmi + copd + asthma + osa + chf + acute_nmd + phtn
  data        = subset_data,
  method      = "gbm",
  estimand    = "ATE",
  missing    = "ind",
  include.obj = TRUE,
  n.trees     = 3000,
  interaction.depth = 3,
  shrinkage   = 0.01,
  bag.fraction= 0.6,
  cv.folds    = 5,
  stop.method = "es.mean",
  n.cores     = parallel::detectCores()
```

```
)

#   3. (optional) stabilise + two-sided Winsorisation -------------------------
w <- w_vbg_calc$weights
w <- w / mean(w)

cut <- quantile(w, c(0.01, 1), na.rm = TRUE)
w   <- pmin(pmax(w, cut[1]), cut[2])
w   <- w / mean(w)

subset_data$w_vbg_calc <- w          # attach to data frame
w_vbg_calc$weights     <- w          # overwrite inside object for diagnostics

v_calc_bal <- bal.tab(w_vbg_calc, un = TRUE, m.threshold = 0.1)   # inspect balance
```

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

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

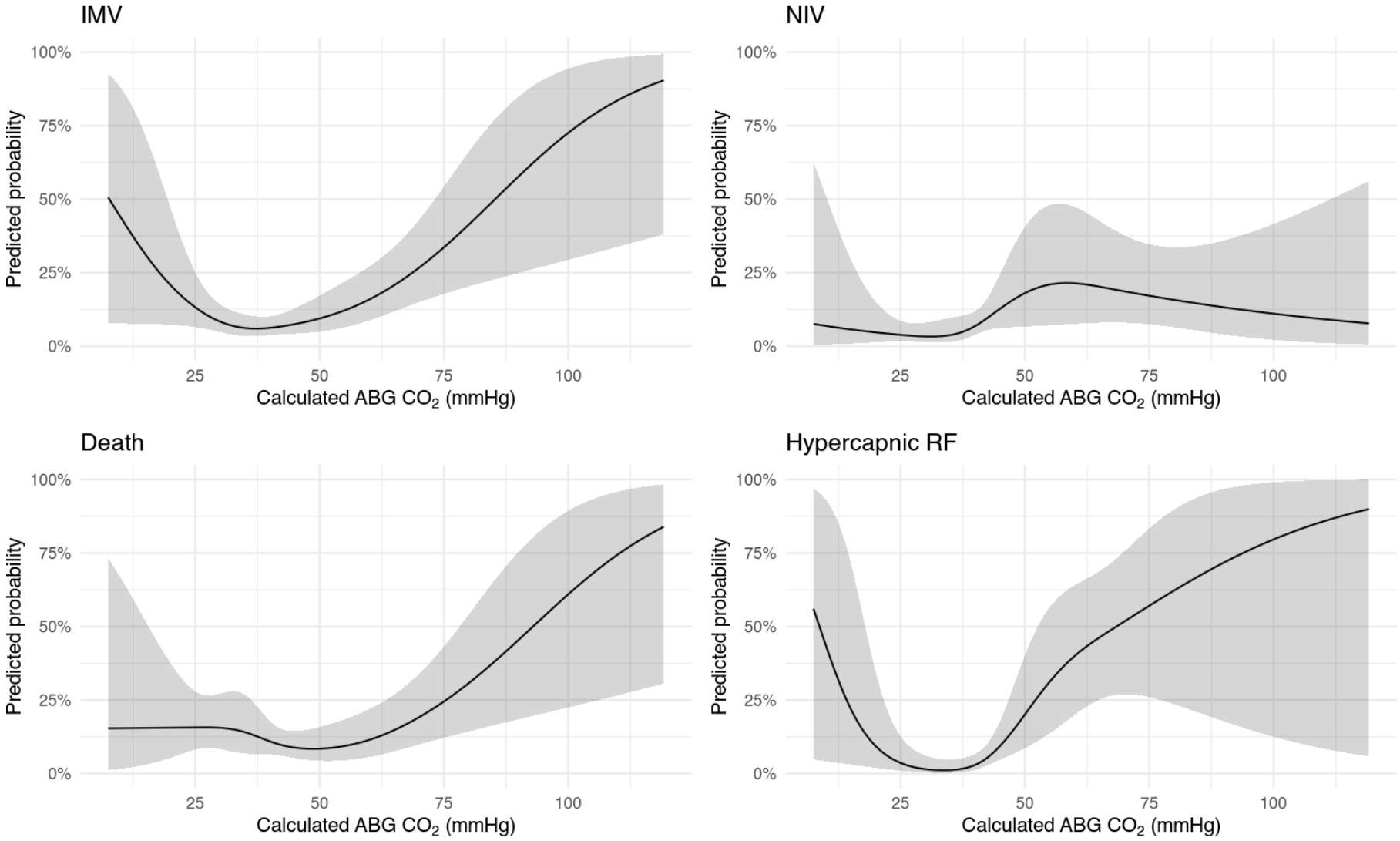
```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```




```
# 2. Analysis sample: rows with a calculated ABG CO ---------------------------
subset_data_calc <- subset_data %>%
  filter(!is.na(calc_abg)) %>%                       # implies has_vbg == 1
  select(calc_abg, imv_proc, niv_proc, death_60d,
```

```
         hypercap_resp_failure, w_vbg_calc) %>%
  filter(complete.cases(.))

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

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

fit_imv_calc   <- fitfun(imv_proc              ~ rcs(calc_abg, 4))
fit_niv_calc   <- fitfun(niv_proc              ~ rcs(calc_abg, 4))
fit_death_calc <- fitfun(death_60d                  ~ rcs(calc_abg, 4))
fit_hcrf_calc  <- fitfun(hypercap_resp_failure ~ rcs(calc_abg, 4))

# 4. Prediction helper ---------------------------------------------------------
mkpred <- function(fit, data_ref) {
  newd <- data.frame(
    calc_abg = seq(min(data_ref$calc_abg, na.rm = TRUE),
                   max(data_ref$calc_abg, 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)
  )
}
```

```r
pred_imv_calc   <- mkpred(fit_imv_calc,   subset_data_calc)
pred_niv_calc   <- mkpred(fit_niv_calc,   subset_data_calc)
pred_death_calc <- mkpred(fit_death_calc, subset_data_calc)
pred_hcrf_calc  <- mkpred(fit_hcrf_calc,  subset_data_calc)

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

plt <- function(dat, title)
  ggplot(dat, aes(calc_abg, 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_calc,   "IMV"),
    plt(pred_niv_calc,   "NIV"),
    plt(pred_death_calc, "Death"),
    plt(pred_hcrf_calc,  "Hypercapnic RF"),
    ncol = 2
  )
) +
  plot_annotation(
    title = expression(
      paste("Propensity-weighted predicted probability by Calculated ABG CO"[2],
            " (restricted cubic spline)")
    )
)
```



Superimposing ABG and VBG weighted restricted cubic splines

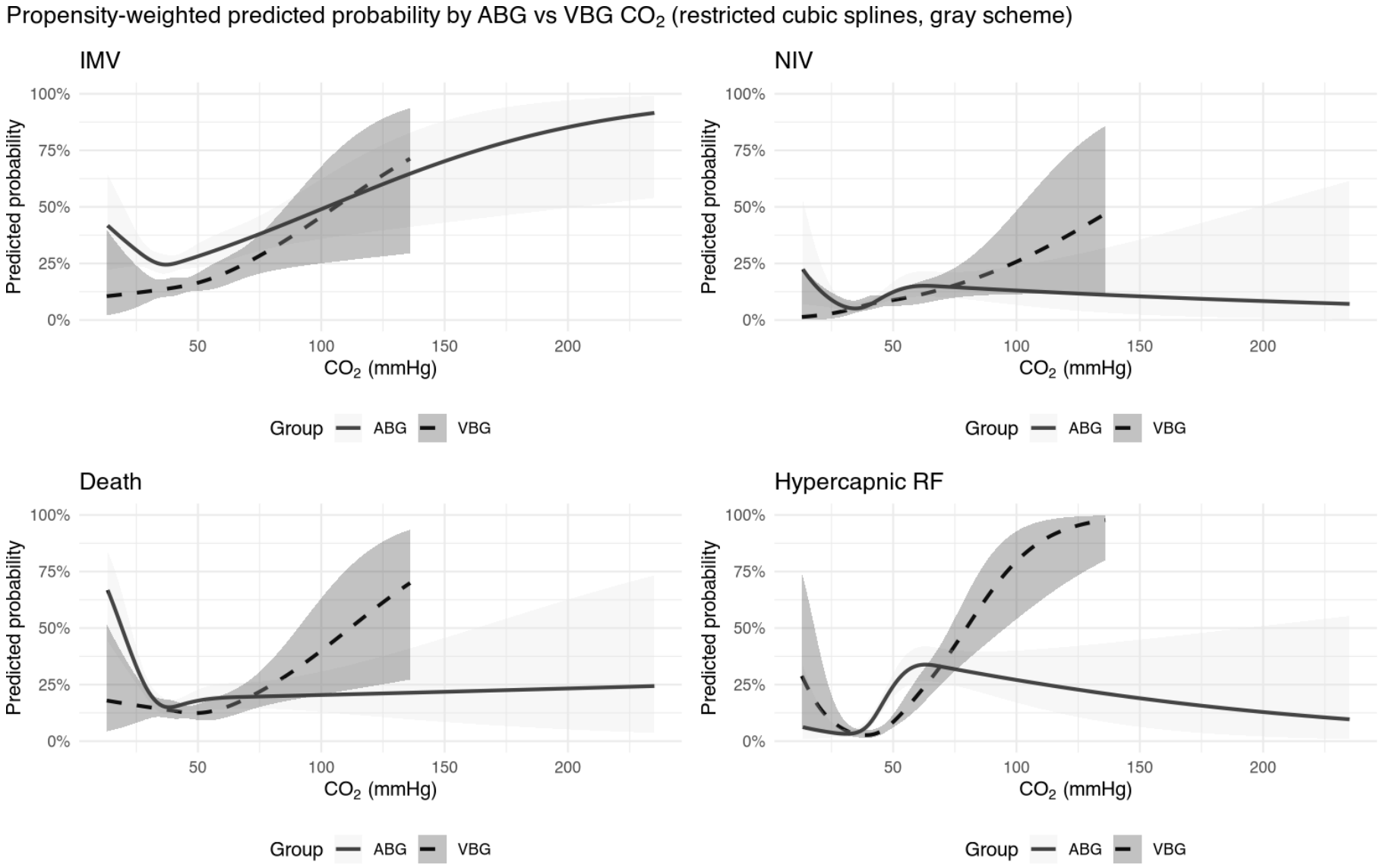
1.5.6 Overlay: ABG vs VBG IPW spline predictions

```
library(dplyr)
library(ggplot2)
library(patchwork)
library(scales)
library(rms)

# ABG spline fits (unweighted, rms::lrm)
fit_imv_abg   <- lrm(imv_proc              ~ rcs(paco2, 4), data = subset_data_abg)
fit_niv_abg   <- lrm(niv_proc              ~ rcs(paco2, 4), data = subset_data_abg)
fit_death_abg <- lrm(death_60d             ~ rcs(paco2, 4), data = subset_data_abg)
fit_hcrf_abg  <- lrm(hypercap_resp_failure ~ rcs(paco2, 4), data = subset_data_abg)

# VBG spline fits (mirror pattern)
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_hcrf_vbg  <- lrm(hypercap_resp_failure ~ rcs(vbg_co2, 4), data = subset_data_vbg)

library(rms)  # ensure lrm() and Predict() are available

# Helper to make predictions with standardized columns: co2, yhat, lower, upper, group
mkpred <- function(fit, data_ref, xvar, group_label, n = 200) {
  stopifnot(is.character(xvar), length(xvar) == 1, xvar %in% names(data_ref))
  xseq <- seq(min(data_ref[[xvar]], na.rm = TRUE),
              max(data_ref[[xvar]], na.rm = TRUE),
              length.out = n)

  if (inherits(fit, "lrm")) {
    # Predict() needs a datadist object visible by name set in options(datadist=)
    dd <- rms::datadist(data_ref)
    old <- options(datadist = "dd")
    on.exit(options(old), add = TRUE)
    assign("dd", dd, envir = .GlobalEnv)
```

```
    # IMPORTANT: name the model argument 'object', not 'fit'
    args <- c(list(object = fit, fun = plogis),
              stats::setNames(list(xseq), xvar))
    p <- do.call(rms::Predict, args)
    out <- as.data.frame(p)
    # standardize column names used by plotting code
    names(out)[names(out) == xvar] <- "co2"
    out$group <- group_label
    out[, c("co2", "yhat", "lower", "upper", "group")]
  } else {
    # glm/svyglm path
    newd <- stats::setNames(data.frame(xseq), xvar)
    X    <- stats::model.matrix(stats::delete.response(stats::terms(fit)), newd)
    beta <- stats::coef(fit)
    eta  <- drop(X %*% beta)
    V    <- stats::vcov(fit)
    se   <- sqrt(rowSums((X %*% V) * X))
    data.frame(
      co2    = xseq,
      yhat   = plogis(eta),
      lower  = plogis(eta - 1.96 * se),
      upper  = plogis(eta + 1.96 * se),
      group  = group_label,
      check.names = FALSE
    )
  }
}

#   Generate predictions
# VBG
pred_imv_vbg   <- mkpred(fit_imv_vbg,   subset_data_vbg, "vbg_co2", "VBG")
pred_niv_vbg   <- mkpred(fit_niv_vbg,   subset_data_vbg, "vbg_co2", "VBG")
pred_death_vbg <- mkpred(fit_death_vbg, subset_data_vbg, "vbg_co2", "VBG")
pred_hcrf_vbg  <- mkpred(fit_hcrf_vbg,  subset_data_vbg, "vbg_co2", "VBG")
```

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

# Combine
pred_imv   <- bind_rows(pred_imv_vbg,   pred_imv_abg)
pred_niv   <- bind_rows(pred_niv_vbg,   pred_niv_abg)
pred_death <- bind_rows(pred_death_vbg, pred_death_abg)
pred_hcrf  <- bind_rows(pred_hcrf_vbg,  pred_hcrf_abg)

#   Plotting function in grayscale with distinguishable ribbons
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")) +  # different gray shades
    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() +
    theme(legend.position = "bottom")
}

#   Patchwork layout with gray shades
(patchwork::wrap_plots(
    plt_gray(pred_imv,   "IMV"),
    plt_gray(pred_niv,   "NIV"),
    plt_gray(pred_death, "Death"),
```

```
    plt_gray(pred_hcrf,  "Hypercapnic RF"),
    ncol = 2
  )
) +
  plot_annotation(
    title = expression(
      paste("Propensity-weighted predicted probability by ABG vs VBG CO"[2],
            " (restricted cubic splines, gray scheme)")
    )
  )
```



Restricting the plot to 0.02 to 0.99 (since this puts it at about 100mmHg for CO2)

1.5.7 Overlay: ABG vs VBG IPW spline predictions (2–98th percentile)

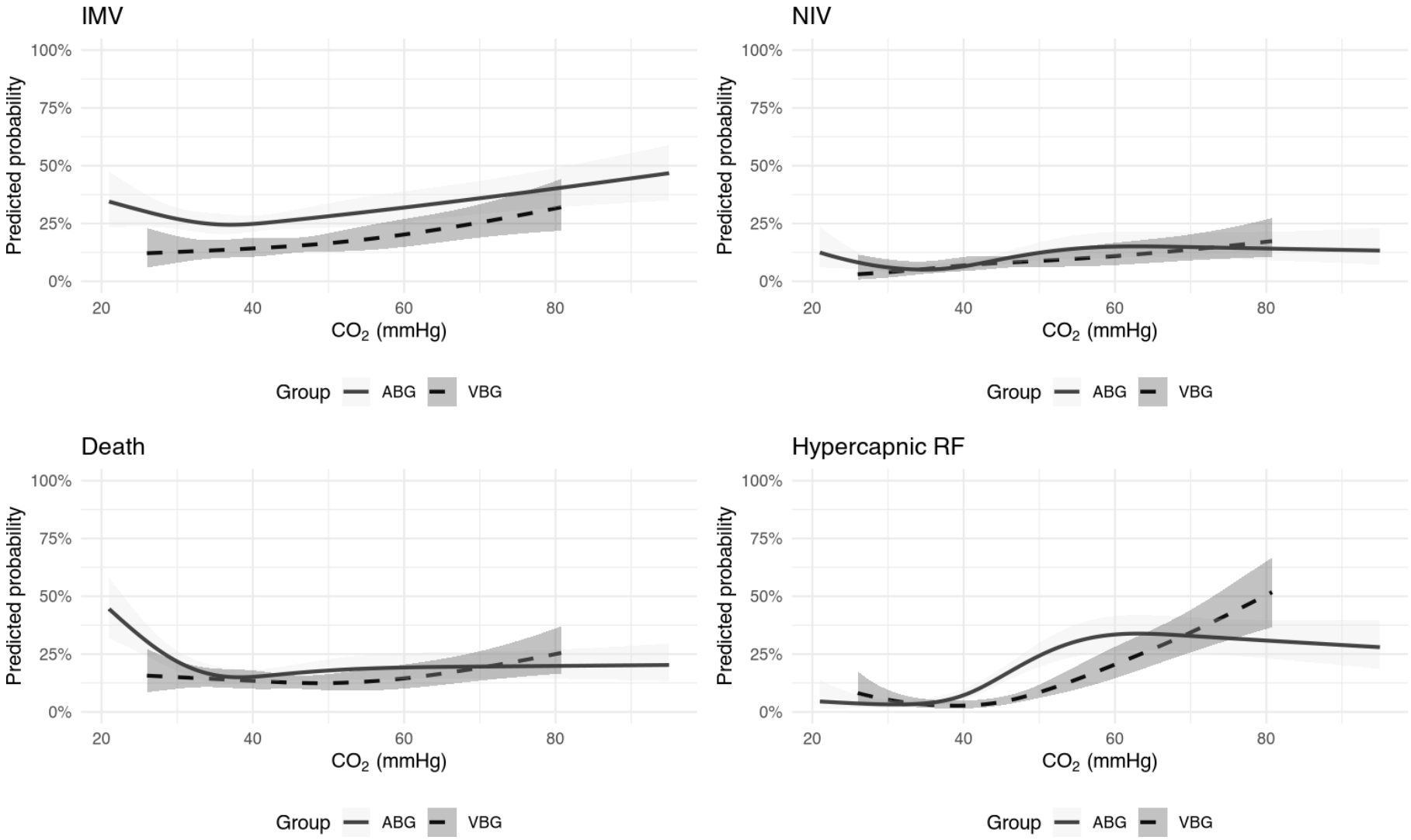
```
library(dplyr)
library(ggplot2)
library(patchwork)
library(scales)
library(rms)

# ABG spline fits (unweighted, rms::lrm)
fit_imv_abg   <- lrm(imv_proc              ~ rcs(paco2, 4), data = subset_data_abg)
fit_niv_abg   <- lrm(niv_proc              ~ rcs(paco2, 4), data = subset_data_abg)
fit_death_abg <- lrm(death_60d             ~ rcs(paco2, 4), data = subset_data_abg)
fit_hcrf_abg  <- lrm(hypercap_resp_failure ~ rcs(paco2, 4), data = subset_data_abg)

# VBG spline fits (mirror pattern)
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_hcrf_vbg  <- lrm(hypercap_resp_failure ~ rcs(vbg_co2, 4), data = subset_data_vbg)

library(rms)  # ensure lrm() and Predict() are available

# Helper to make predictions with standardized columns: co2, yhat, lower, upper, group
mkpred <- function(fit, data_ref, xvar, group_label, n = 200) {
  stopifnot(is.character(xvar), length(xvar) == 1, xvar %in% names(data_ref))

  # Restrict to 2nd and 98th percentiles
  rng <- quantile(data_ref[[xvar]], probs = c(0.02, 0.98), na.rm = TRUE)
  xseq <- seq(rng[1], rng[2], length.out = n)

  if (inherits(fit, "lrm")) {
    dd <- rms::datadist(data_ref)
    old <- options(datadist = "dd")
    on.exit(options(old), add = TRUE)
    assign("dd", dd, envir = .GlobalEnv)
```

```
    args <- c(list(object = fit, fun = plogis),
              stats::setNames(list(xseq), xvar))
    p <- do.call(rms::Predict, args)
    out <- as.data.frame(p)
    names(out)[names(out) == xvar] <- "co2"
    out$group <- group_label
    out[, c("co2", "yhat", "lower", "upper", "group")]
  } else {
    newd <- stats::setNames(data.frame(xseq), xvar)
    X    <- stats::model.matrix(stats::delete.response(stats::terms(fit)), newd)
    beta <- stats::coef(fit)
    eta  <- drop(X %*% beta)
    V    <- stats::vcov(fit)
    se   <- sqrt(rowSums((X %*% V) * X))
    data.frame(
      co2    = xseq,
      yhat   = plogis(eta),
      lower  = plogis(eta - 1.96 * se),
      upper  = plogis(eta + 1.96 * se),
      group  = group_label,
      check.names = FALSE
    )
  }
}

#   Generate predictions
# VBG
pred_imv_vbg   <- mkpred(fit_imv_vbg,   subset_data_vbg, "vbg_co2", "VBG")
pred_niv_vbg   <- mkpred(fit_niv_vbg,   subset_data_vbg, "vbg_co2", "VBG")
pred_death_vbg <- mkpred(fit_death_vbg, subset_data_vbg, "vbg_co2", "VBG")
pred_hcrf_vbg  <- mkpred(fit_hcrf_vbg,  subset_data_vbg, "vbg_co2", "VBG")

# ABG
pred_imv_abg   <- mkpred(fit_imv_abg,   subset_data_abg, "paco2", "ABG")
pred_niv_abg   <- mkpred(fit_niv_abg,   subset_data_abg, "paco2", "ABG")
```

```r
pred_death_abg <- mkpred(fit_death_abg, subset_data_abg, "paco2", "ABG")
pred_hcrf_abg  <- mkpred(fit_hcrf_abg,  subset_data_abg, "paco2", "ABG")

# Combine
pred_imv   <- bind_rows(pred_imv_vbg,   pred_imv_abg)
pred_niv   <- bind_rows(pred_niv_vbg,   pred_niv_abg)
pred_death <- bind_rows(pred_death_vbg, pred_death_abg)
pred_hcrf  <- bind_rows(pred_hcrf_vbg,  pred_hcrf_abg)

#   Plotting function in grayscale with distinguishable ribbons
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")) +  # different gray shades
    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() +
    theme(legend.position = "bottom")
}

#   Patchwork layout with gray shades
(patchwork::wrap_plots(
    plt_gray(pred_imv,   "IMV"),
    plt_gray(pred_niv,   "NIV"),
    plt_gray(pred_death, "Death"),
    plt_gray(pred_hcrf,  "Hypercapnic RF"),
    ncol = 2
  )
```

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

Propensity-weighted predicted probability by ABG vs VBG CO_2 (restricted cubic splines, gray scheme)



1.6 Explainability (propensity models)

Feature importance: *global* contribution of a feature to the model's predictive performance on the training distribution. Quick global triage—*which variables the model leaned on to fit propensity.* Good for model debugging, feature pruning, and tracking drift across refits (qualitatively).

SHAP: a *local*, signed attribution for that subject: "by how much did feature j push this person's log-odds of receiving the test up or down vs baseline?, then global shap is mean absolute SHAP across subjects—i.e., the **typical magnitude** of a feature's contribution to predictions in your population. Good for auditability and **directional insight**—*who is assigned higher/lower propensity by which features*, spot proxies, and communicate fairness/operational drivers. Aggregate with mean |SHAP| for a **global ranking with direction available** when needed.

TODO: **Can label y-axis in plots: contribution to the log odds of receiving an ABG or VBG for the SHAP values.**

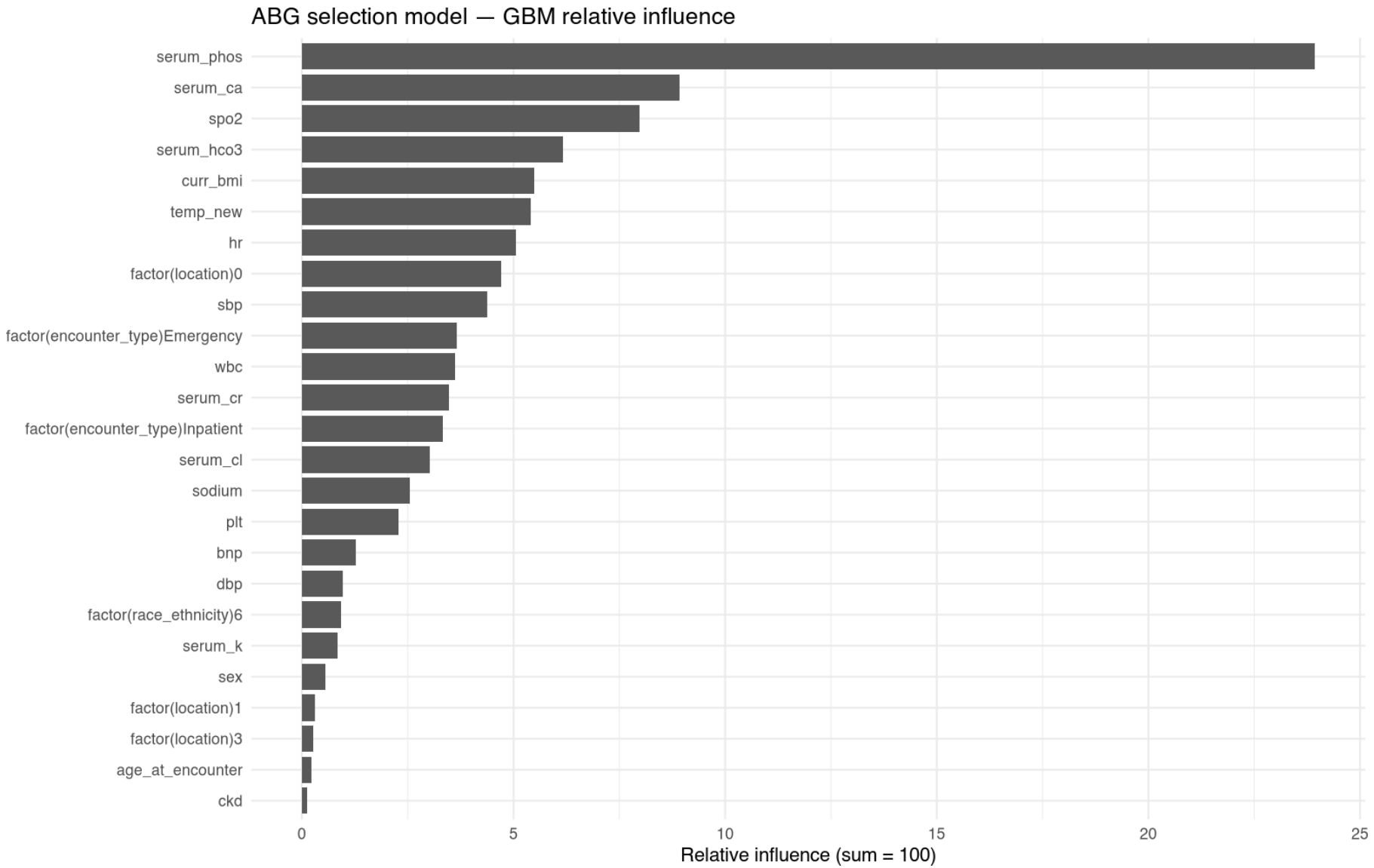
1.6.1 ABG explainability

For the top SHAP-ranked predictors we computed partial- and accumulated-local-effects (ALE) to estimate the marginal change in predicted risk across clinically relevant ranges, robust to covariate correlation. We complemented this with SHAP dependence plots (colored by plausible interactions) and fitted transparent spline-logistic models to identify turn-points ('knees') where marginal log-odds slope changed.

```
# --- deps ------------------------------------------------------------------
library(WeightIt)
library(gbm)
library(dplyr)
library(ggplot2)
library(fastshap)

# --- 0) Canonicalize object name -------------------------------------------
# Your 9-26.qmd labeled the ABG propensity object `weight_model`.
if (!exists("w_abg", inherits = TRUE) && exists("weight_model", inherits = TRUE)) {
  w_abg <- weight_model
}
stopifnot(exists("w_abg", inherits = TRUE))

# --- 1) Ensure the WeightIt object stores the GBM + covariate matrix -----------
ensure_gbm_obj <- function(W) {
  stopifnot(inherits(W, "weightit"))
  has_obj <- !is.null(W$obj) || !is.null(W$info$obj) || !is.null(W$info$model.obj)
```

```r
  has_cov <- !is.null(W$covs)
  if (has_obj && has_cov) return(W)

  cl <- as.list(W$call); cl[[1]] <- WeightIt::weightit
  if (!is.null(cl[["missing."]])) { cl$missing <- cl[["missing."]]; cl[["missing."]] <- NULL }
  if (is.null(cl$missing))        cl$missing <- "ind"
  cl$include.obj <- TRUE
  if (is.null(cl$method))         cl$method <- "gbm"
  if (is.language(cl$formula)) cl$formula <- eval(cl$formula, envir = .GlobalEnv)
  if (is.language(cl$data))    cl$data    <- eval(cl$data,    envir = .GlobalEnv)
  do.call(WeightIt::weightit, cl[-1])
}
w_abg <- ensure_gbm_obj(w_abg)

# --- 2) Helpers: design alignment, importance, fast SHAP (logit scale) ---------
prep_design <- function(W) {
  stopifnot(inherits(W, "weightit"))
  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"))
  stopifnot(!is.null(W$covs))

  X <- W$covs
  if (inherits(X, "tbl"))    X <- as.data.frame(X)
  if (inherits(X, "Matrix")) X <- as.matrix(X)
  X <- as.data.frame(X, stringsAsFactors = FALSE)

  # conservative coercion: only numeric-like strings → numeric
  for (nm in names(X)) {
    if (is.factor(X[[nm]])) X[[nm]] <- as.character(X[[nm]])
    if (is.character(X[[nm]])) {
      ok <- grepl("^[-+]?[0-9.]+$", X[[nm]] %||% "")
      if (all(ok | is.na(X[[nm]]))) suppressWarnings(X[[nm]] <- as.numeric(X[[nm]]))
    }
  }

  vars <- gbm_fit$var.names
```

```r
  miss <- setdiff(vars, colnames(X))
  if (length(miss)) for (nm in miss) X[[nm]] <- 0
  X <- X[, vars, drop = FALSE]

  best_tree <- if (!is.null(W$info$best.tree)) W$info$best.tree else gbm_fit$n.trees
  list(X = X, gbm_fit = gbm_fit, best_tree = best_tree)
}

extract_gbm_importance <- function(W, top_n = 25) {
  mats <- prep_design(W)
  as.data.frame(summary(mats$gbm_fit, n.trees = mats$best_tree, plotit = FALSE)) |>
    arrange(desc(rel.inf)) |>
    slice_head(n = top_n)
}

plot_gbm_importance <- function(imp_df, title = "GBM variable importance (relative influence)") {
  ggplot(imp_df, aes(x = rel.inf, y = reorder(var, rel.inf))) +
    geom_col(width = 0.85) +
    labs(x = "Relative influence (sum = 100)", y = NULL, title = title) +
    theme_minimal(base_size = 11)
}

# --- 3) Run: ABG selection model - importance + fast SHAP ----------------------
imp_abg   <- extract_gbm_importance(w_abg, top_n = 25)
p_imp_abg <- plot_gbm_importance(imp_abg, "ABG selection model - GBM relative influence")
p_imp_abg
```




```
# --- Build shapviz object robustly -----------------------------------------
library(shapviz)

# Align newdata's factor levels to the GBM's training levels
```

```r
align_levels_to_gbm <- function(newdata, gbm_fit) {
  out <- as.data.frame(newdata, stringsAsFactors = FALSE)
  if (!is.null(gbm_fit$var.levels)) {
    for (nm in intersect(names(gbm_fit$var.levels), names(out))) {
      lev <- unique(as.character(gbm_fit$var.levels[[nm]]))
      out[[nm]] <- factor(as.character(out[[nm]]), levels = lev)
    }
  }
  out
}

# fast SHAP on LOGIT scale; prunes zero-variance features; subsamples rows
compute_shap_fast <- function(W, top_k = 30, nsim = 64, frac_rows = 0.50,
                              max_rows = 100000, seed = 123) {
  mats <- prep_design(W); X <- mats$X; gbm_fit <- mats$gbm_fit; best_tree <- mats$best_tree

  imp <- as.data.frame(summary(gbm_fit, n.trees = best_tree, plotit = FALSE))
  top_feats <- head(imp$var, min(top_k, nrow(imp)))
  top_feats <- intersect(top_feats, colnames(X))

  # drop constant features (avoid flat SHAP/plots)
  nzv <- sapply(X[, top_feats, drop = FALSE], function(z) sd(z, na.rm = TRUE) > 0)
  top_feats <- top_feats[nzv]
  if (!length(top_feats)) stop("All candidate features are near-constant in this subset.")

  n <- nrow(X); target_n <- min(n, max_rows, ceiling(frac_rows * n))
  set.seed(seed)
  Xsub <- if (target_n < n) X[sample.int(n, target_n), , drop = FALSE] else X

  # SHAP on logit scale for contrast/stability
  pfun <- function(object, newdata) {
    newd <- align_levels_to_gbm(newdata, object)
    predict(object, newdata = newd, n.trees = best_tree, type = "link")
  }

  fs_formals <- names(formals(fastshap::explain))
```

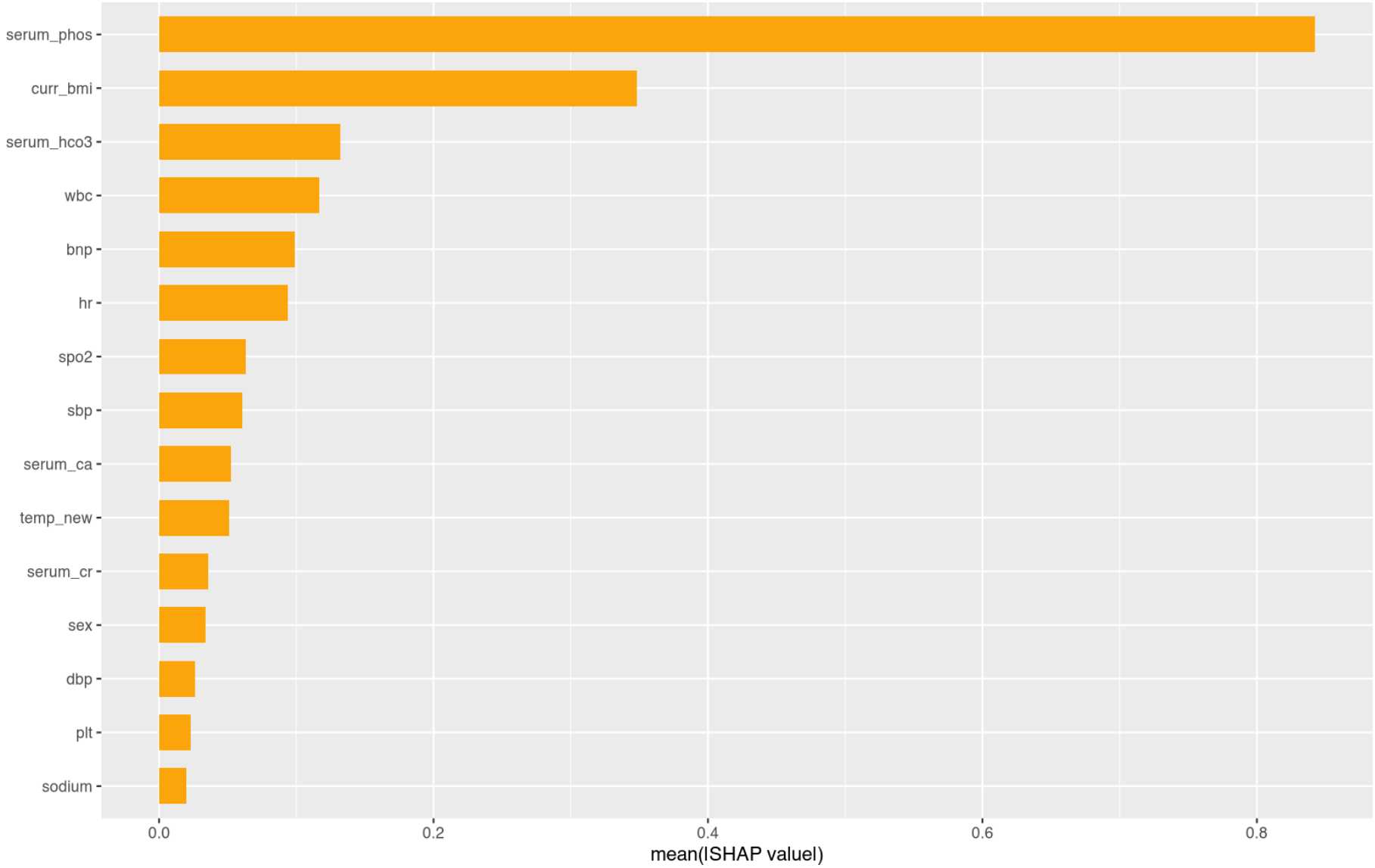
```
  args <- list(object = gbm_fit, X = Xsub, pred_wrapper = pfun, nsim = nsim, adjust = TRUE)
  if ("feature_names" %in% fs_formals) args$feature_names <- top_feats

  set.seed(seed)
  S <- do.call(fastshap::explain, args)  # matrix or data.frame of SHAP

  list(shap = S, X = Xsub, top_feats = top_feats, imp = imp)
}

t0 <- Sys.time()
sh_abg_fast <- compute_shap_fast(w_abg, top_k = 100, nsim = 32, frac_rows = 0.25, max_rows = 100000)
t1 <- Sys.time(); message(sprintf("[compute_shap_fast] %.2f s", as.numeric(difftime(t1, t0, units="secs"))))
```

```
[compute_shap_fast] 37.36 s
```

```
# 1) Take SHAP and design from your fast SHAP object
S <- as.matrix(sh_abg_fast$shap)   # n x p SHAP matrix
X <- as.data.frame(sh_abg_fast$X)  # matching rows, p columns

# 2) Make X numeric-only (handles factors / labelled types safely)
for (nm in names(X)) {
  if (inherits(X[[nm]], "haven_labelled")) {
    X[[nm]] <- labelled::to_factor(X[[nm]])
  }
  if (is.factor(X[[nm]]))   X[[nm]] <- as.character(X[[nm]])
  if (is.character(X[[nm]])) suppressWarnings(X[[nm]] <- as.numeric(X[[nm]]))
}

# 3) Align names/order between S and X (and give S names if missing)
if (is.null(colnames(S))) colnames(S) <- colnames(X)
S <- S[, intersect(colnames(S), colnames(X)), drop = FALSE]
X <- X[, colnames(S), drop = FALSE]

# 4) Construct shapviz object: PASS S POSITIONALLY (no name) to avoid dispatch bug
sv <- shapviz(S, X = as.matrix(X))
```

```r
# --- Examples ----------------------------------------------------------------
# Bar plot of top 30 (global |SHAP|)
ord  <- order(colMeans(abs(S), na.rm = TRUE), decreasing = TRUE)
topK <- colnames(S)[ord[1:min(30, ncol(S))]]
sv_importance(sv, kind = "bar", v = topK)
```

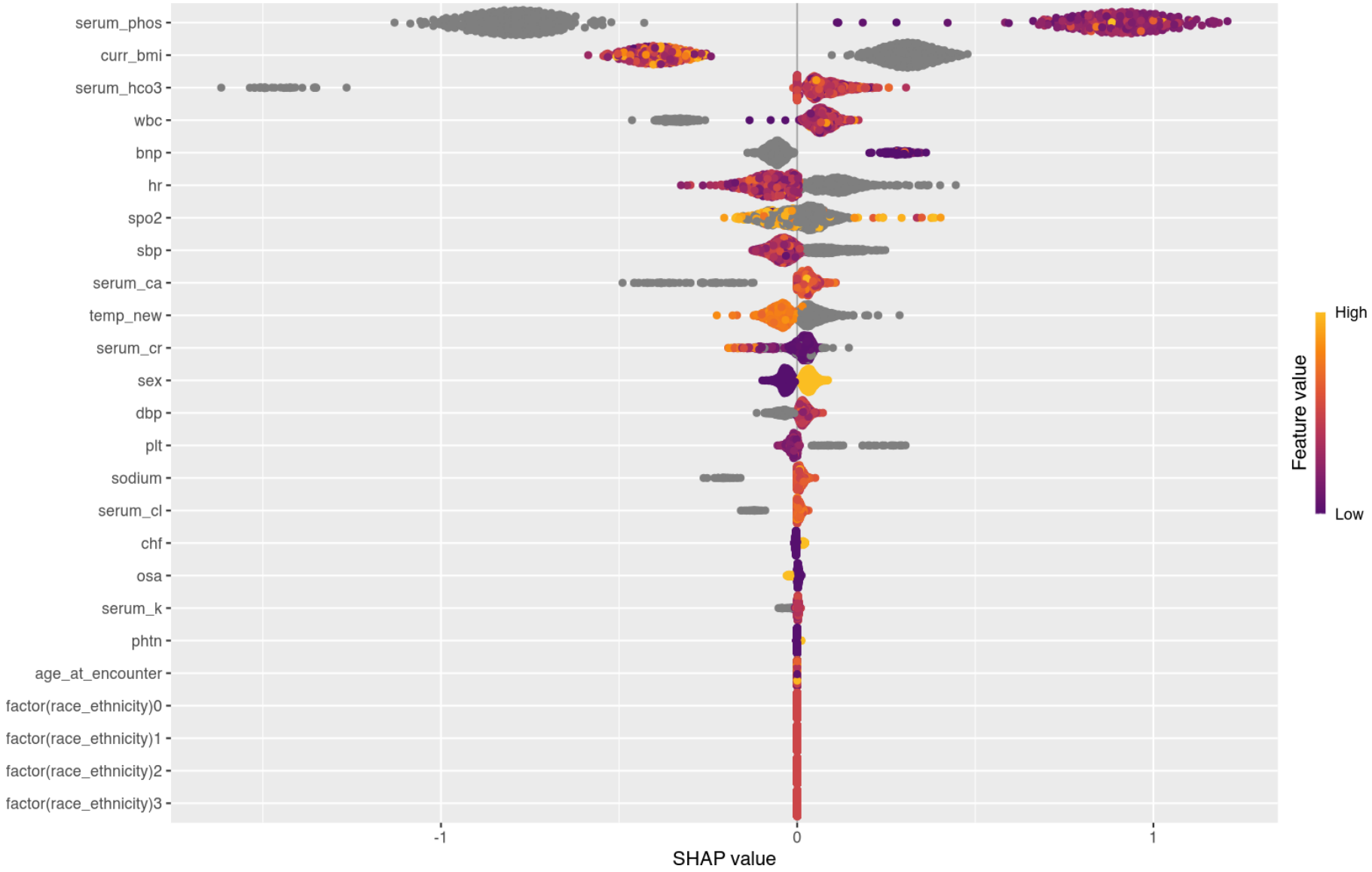
```
Warning: `label` cannot be a <ggplot2::element_blank> object.
```



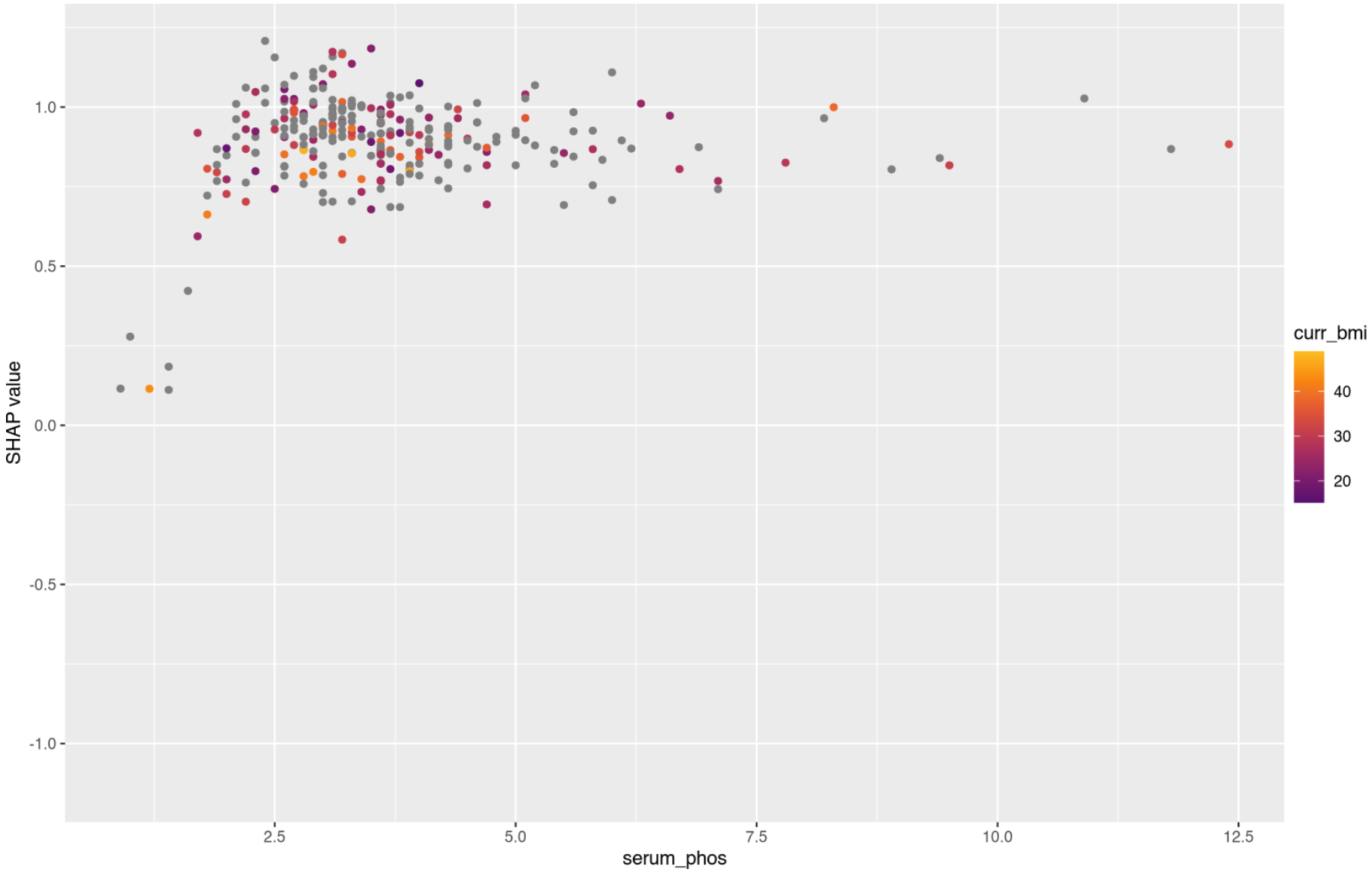

```r
library(shapviz)

S <- as.matrix(sh_abg_fast$shap)   # n x p SHAP values
X <- as.data.frame(sh_abg_fast$X)  # same rows, p columns (features)
```

```r
# Build shapviz object
sv <- shapviz(S, X = as.matrix(X))

# Beeswarm-style SHAP summary (like Python SHAP)
sv_importance(sv, kind = "beeswarm", max_display = 25)              # overall
```

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




```
# Primary = top feature; color by next feature
imp_order <- colnames(S)[ord]
sv_dependence(sv, v = imp_order[1], color_var = imp_order[2], smooth = TRUE)
```



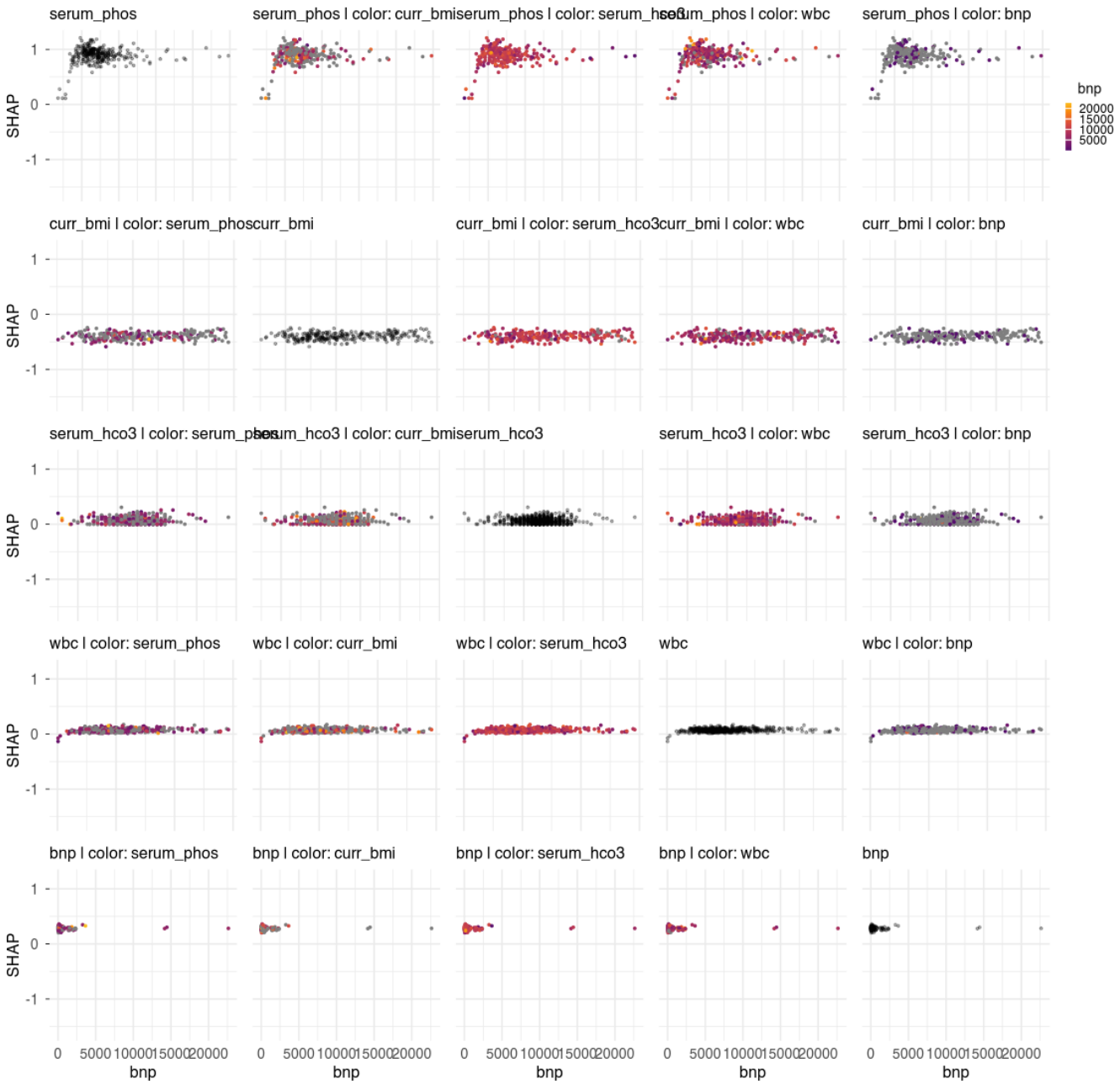

```
# --- Compact 5x5 SHAP dependence grid with shared axes and a single small legend ----
library(shapviz)
library(ggplot2)
library(patchwork)
```

```r
library(grid)   # for unit()

# If needed, recover S and X_sv from 'sv'
if (!exists("S"))  S  <- sv$S
if (!exists("X_sv")) X_sv <- as.data.frame(sv$X)
stopifnot(is.matrix(S), is.data.frame(X_sv))

# 1) Top-5 by global mean |SHAP|
ranked <- colnames(S)[order(colMeans(abs(S), na.rm = TRUE), decreasing = TRUE)]
top5   <- head(ranked, 5)

# 2) Shared y-range across all top-5 features
y_rng <- range(unlist(lapply(top5, function(v) S[, v])), finite = TRUE)

# 3) Small theme helpers
theme_axes_compact <- function(show_y = FALSE, show_x = FALSE, base = 8) {
  theme_minimal(base_size = base) +
    theme(
      axis.title.y   = if (show_y) element_text(size = base) else element_blank(),
      axis.text.y    = if (show_y) element_text(size = base - 1) else element_blank(),
      axis.ticks.y   = if (show_y) element_line(linewidth = 0.2) else element_blank(),
      axis.title.x   = if (show_x) element_text(size = base) else element_blank(),
      axis.text.x    = if (show_x) element_text(size = base - 1) else element_blank(),
      plot.title     = element_text(size = base, hjust = 0),
      legend.title   = element_text(size = base - 1),
      legend.text    = element_text(size = base - 2),
      legend.key.height = unit(22, "pt"),
      legend.key.width  = unit(3,  "pt"),
      legend.margin     = margin(0, 0, 0, 0, "pt"),
      legend.box.margin = margin(0, 0, 0, 0, "pt")
    )
}

# 4) One cell builder
cell_plot <- function(v_row, v_col, i, j, n) {
  show_y <- (j == 1)          # y-axis only on first column
```

```r
show_x <- (i == n)          # x-axis only on bottom row

if (identical(v_row, v_col)) {
  # diagonal: unshaded scatter (no legend)
  df <- data.frame(
    x    = as.numeric(X_sv[[v_row]]),
    shap = as.numeric(S[, v_row])
  )
  df <- df[is.finite(df$x) & is.finite(df$shap), , drop = FALSE]

  ggplot(df, aes(x = x, y = shap)) +
    geom_point(alpha = 0.30, size = 0.45, na.rm = TRUE) +
    scale_y_continuous(limits = y_rng) +
    labs(title = v_row, x = v_row, y = "SHAP") +
    theme_axes_compact(show_y = show_y, show_x = show_x, base = 8) +
    theme(legend.position = "none")
} else {
  # off-diagonal: color by partner feature
  p <- shapviz::sv_dependence(sv, v = v_row, color_var = v_col, size = 0.4) +
    scale_y_continuous(limits = y_rng) +
    labs(title = paste0(v_row, " | color: ", v_col),
         x = v_row, y = "SHAP")

  # keep a single, small legend on the top-right panel only
  keep_legend <- (i == 1 && j == length(top5))
  p +
    theme_axes_compact(show_y = show_y, show_x = show_x, base = 8) +
    guides(colour = guide_colorbar(
      barheight    = unit(24, "pt"),
      barwidth     = unit(3,  "pt"),
      title.position = "top",
      title.hjust    = 0.5,
      label.position = "right"
    )) +
    theme(legend.position = if (keep_legend) "right" else "none")
}
```

```
}

# 5) Build grid row-wise
n <- length(top5)
plots <- vector("list", n * n)
idx <- 1
for (i in seq_len(n)) {
  for (j in seq_len(n)) {
    vr <- top5[i]; vc <- top5[j]
    plots[[idx]] <- cell_plot(vr, vc, i, j, n)
    idx <- idx + 1
  }
}

# 6) Draw: 5 columns, shared layout; keep (not collect) legends so only the chosen one stays
patchwork::wrap_plots(plots, ncol = n, guides = "keep") +
  plot_annotation(title = "Top-5 SHAP dependence: interactions (off-diagonal) and main effects (diagonal)")
```

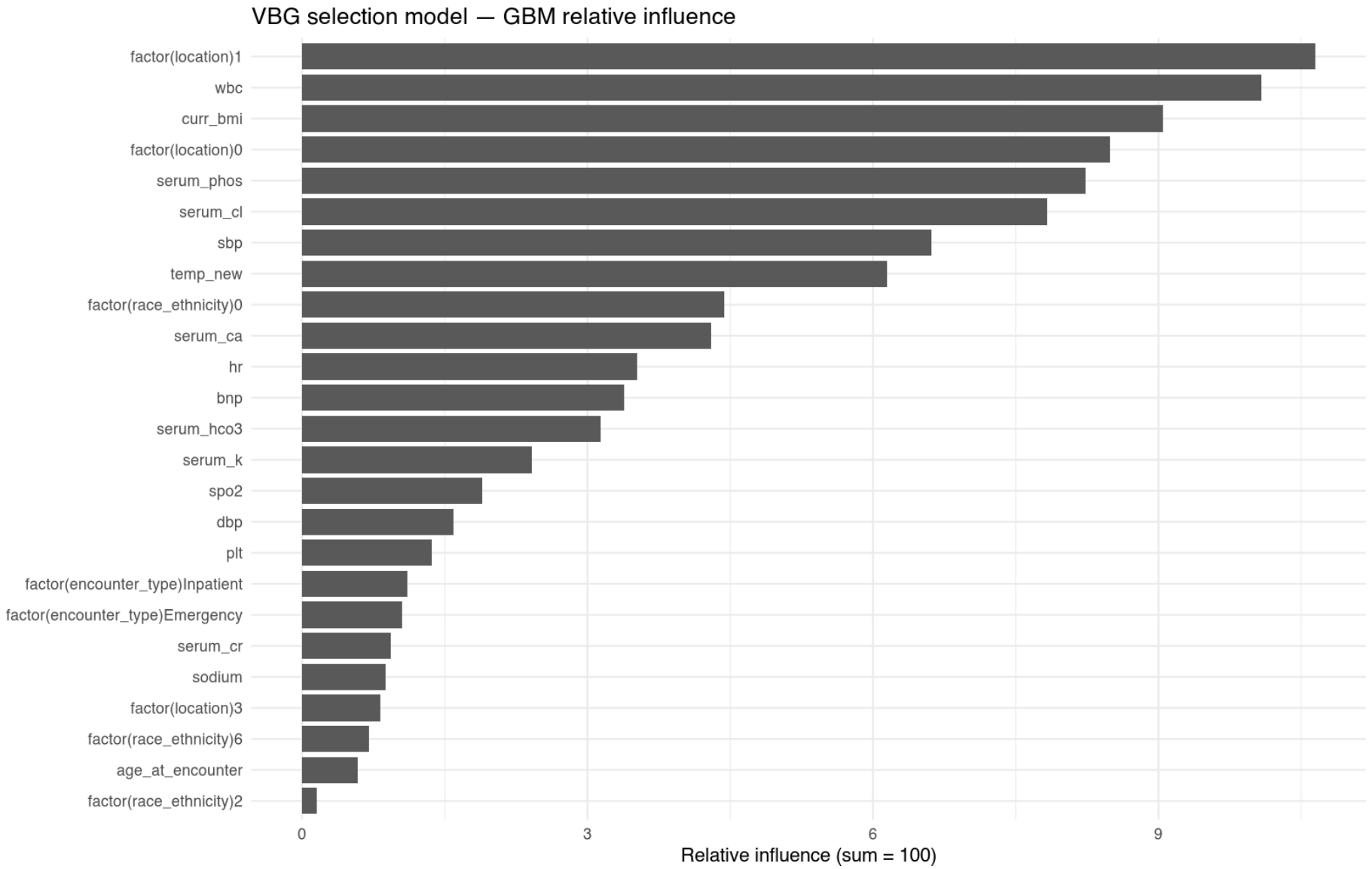
Top-5 SHAP dependence: interactions (off-diagonal) and main effects (diagonal)



1.6.2 VBG explainability

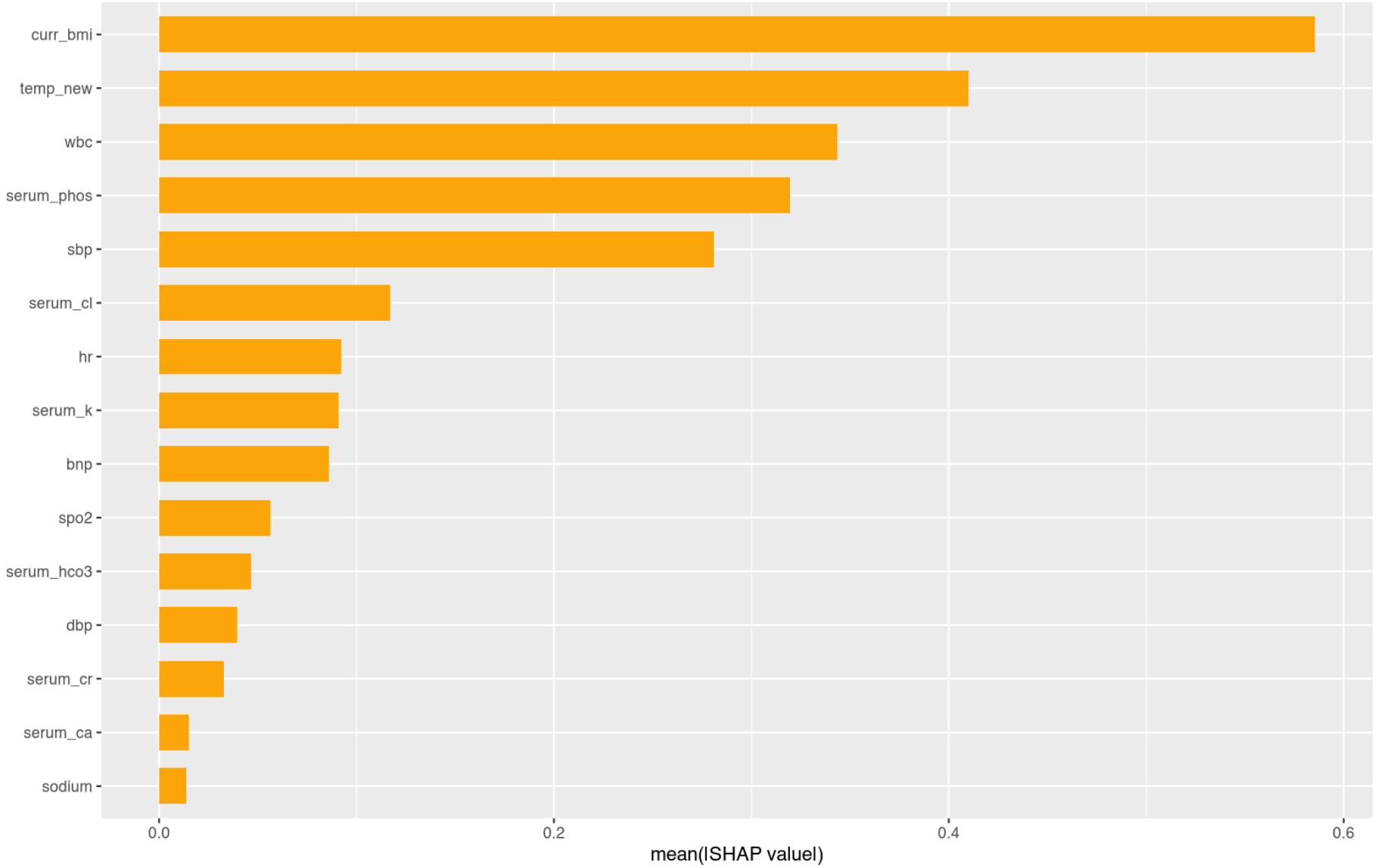
VBG Explainability

```
library(WeightIt); library(gbm); library(dplyr); library(ggplot2)

stopifnot(exists("w_vbg", inherits = TRUE))
w_vbg <- ensure_gbm_obj(w_vbg)

imp_vbg   <- extract_gbm_importance(w_vbg, top_n = 25)
p_imp_vbg <- plot_gbm_importance(imp_vbg, "VBG selection model - GBM relative influence")
p_imp_vbg
```

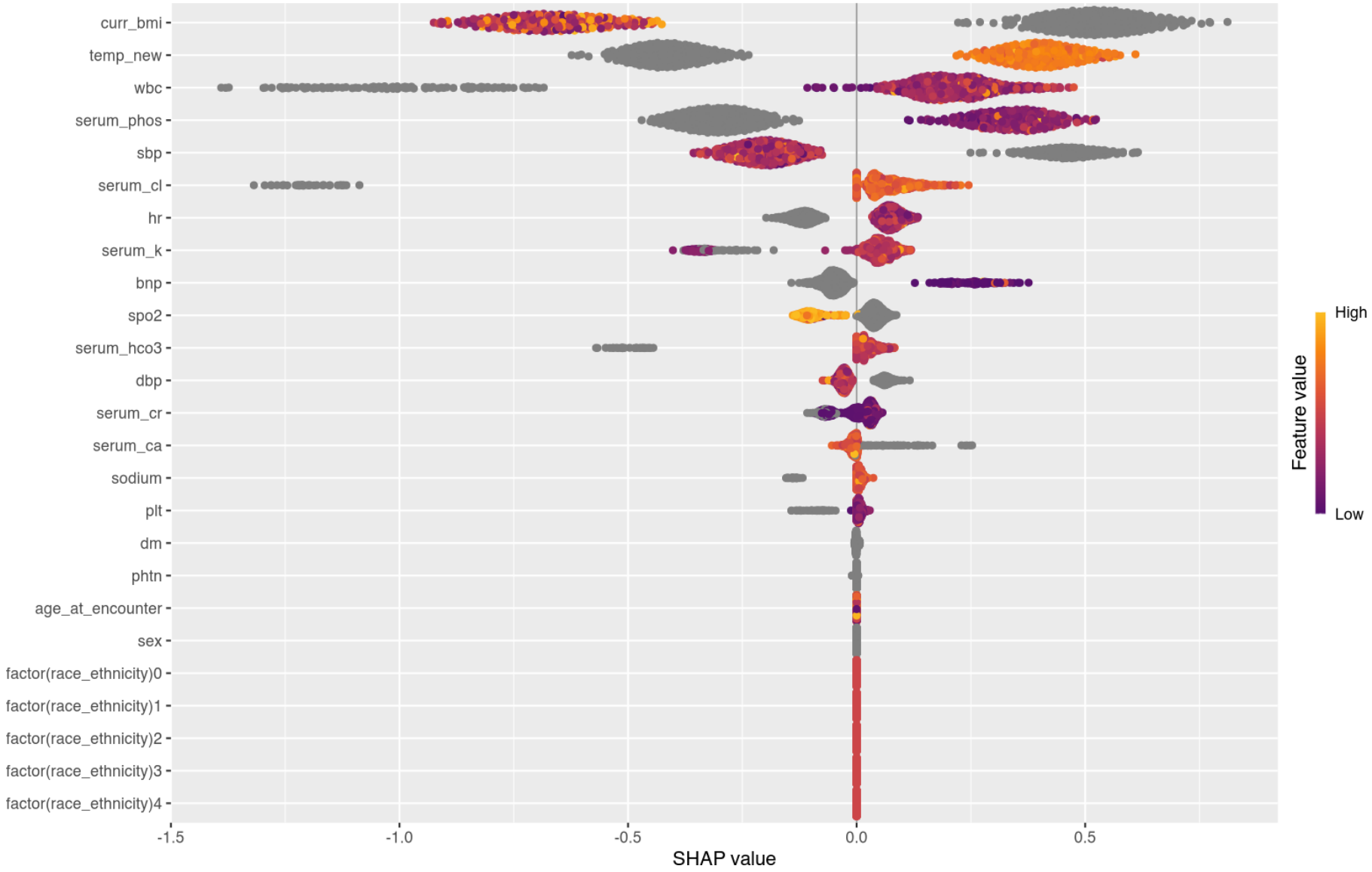



```
library(shapviz); library(fastshap)

t0 <- Sys.time()
sh_vbg_fast <- compute_shap_fast(w_vbg, top_k = 100, nsim = 32, frac_rows = 0.25, max_rows = 100000)
```

```r
t1 <- Sys.time(); message(sprintf("[compute_shap_fast VBG] %.2f s", as.numeric(difftime(t1, t0, units="secs"))))

S_vbg <- as.matrix(sh_vbg_fast$shap)
X_vbg <- as.data.frame(sh_vbg_fast$X)

for (nm in names(X_vbg)) {
  if (inherits(X_vbg[[nm]], "haven_labelled")) X_vbg[[nm]] <- labelled::to_factor(X_vbg[[nm]])
  if (is.factor(X_vbg[[nm]]))  X_vbg[[nm]] <- as.character(X_vbg[[nm]])
  if (is.character(X_vbg[[nm]])) suppressWarnings(X_vbg[[nm]] <- as.numeric(X_vbg[[nm]]))
}
if (is.null(colnames(S_vbg))) colnames(S_vbg) <- colnames(X_vbg)
S_vbg <- S_vbg[, intersect(colnames(S_vbg), colnames(X_vbg)), drop = FALSE]
X_vbg <- X_vbg[, colnames(S_vbg), drop = FALSE]

sv_vbg <- shapviz::shapviz(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(30, ncol(S_vbg))]]
sv_importance(sv_vbg, kind = "bar", v = topK_vbg)
```

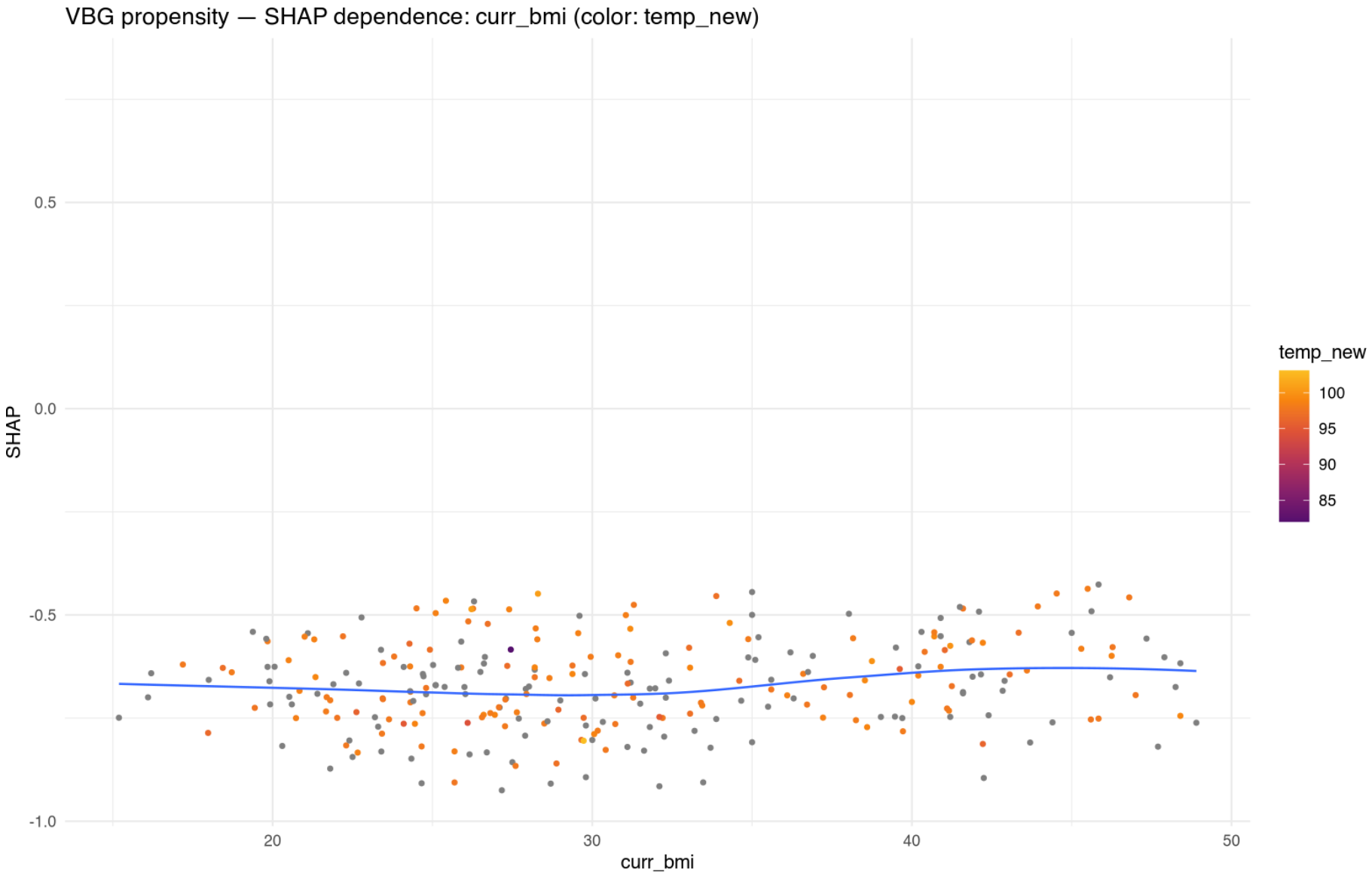



```
library(shapviz)
sv_importance(sv_vbg, kind = "beeswarm", max_display = 25)
```




```
library(ggplot2)

imp_order_vbg <- colnames(S_vbg)[order(colMeans(abs(S_vbg), na.rm = TRUE), decreasing = TRUE)]
pri_vbg <- imp_order_vbg[1]
```

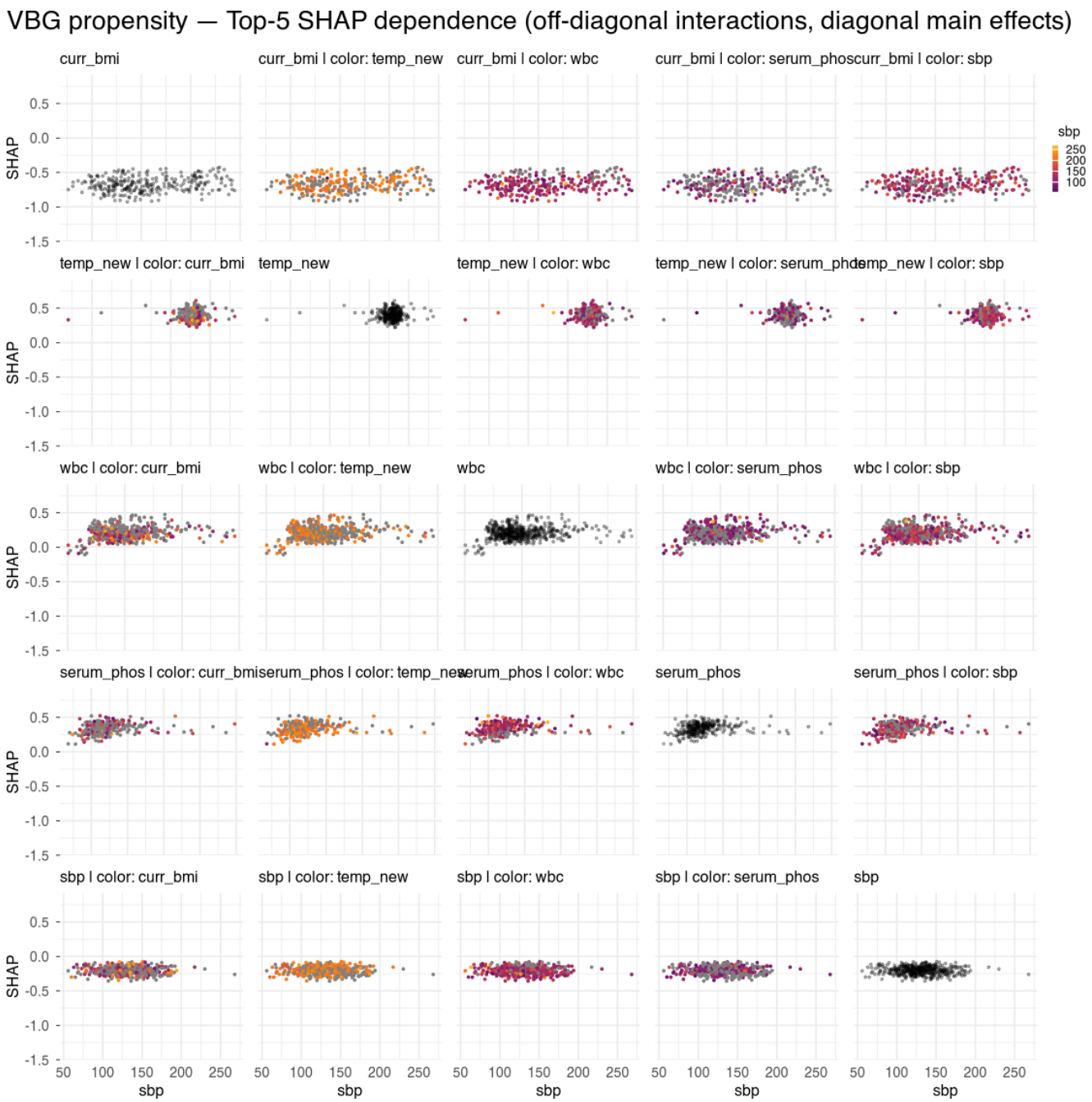
```r
aux_vbg <- imp_order_vbg[2]
if (identical(aux_vbg, pri_vbg) || !(aux_vbg %in% colnames(X_vbg))) aux_vbg <- imp_order_vbg[3]

shapviz::sv_dependence(sv_vbg, v = pri_vbg, color_var = aux_vbg, size = 1) +
  geom_smooth(se = FALSE, method = "loess", formula = y ~ x, linewidth = 0.6) +
  labs(title = sprintf("VBG propensity - SHAP dependence: %s (color: %s)", pri_vbg, aux_vbg),
       x = pri_vbg, y = "SHAP") +
  theme_minimal(base_size = 11)
```




```
library(shapviz); library(ggplot2); library(patchwork); library(grid)

stopifnot(is.matrix(S_vbg), is.data.frame(X_vbg))
```

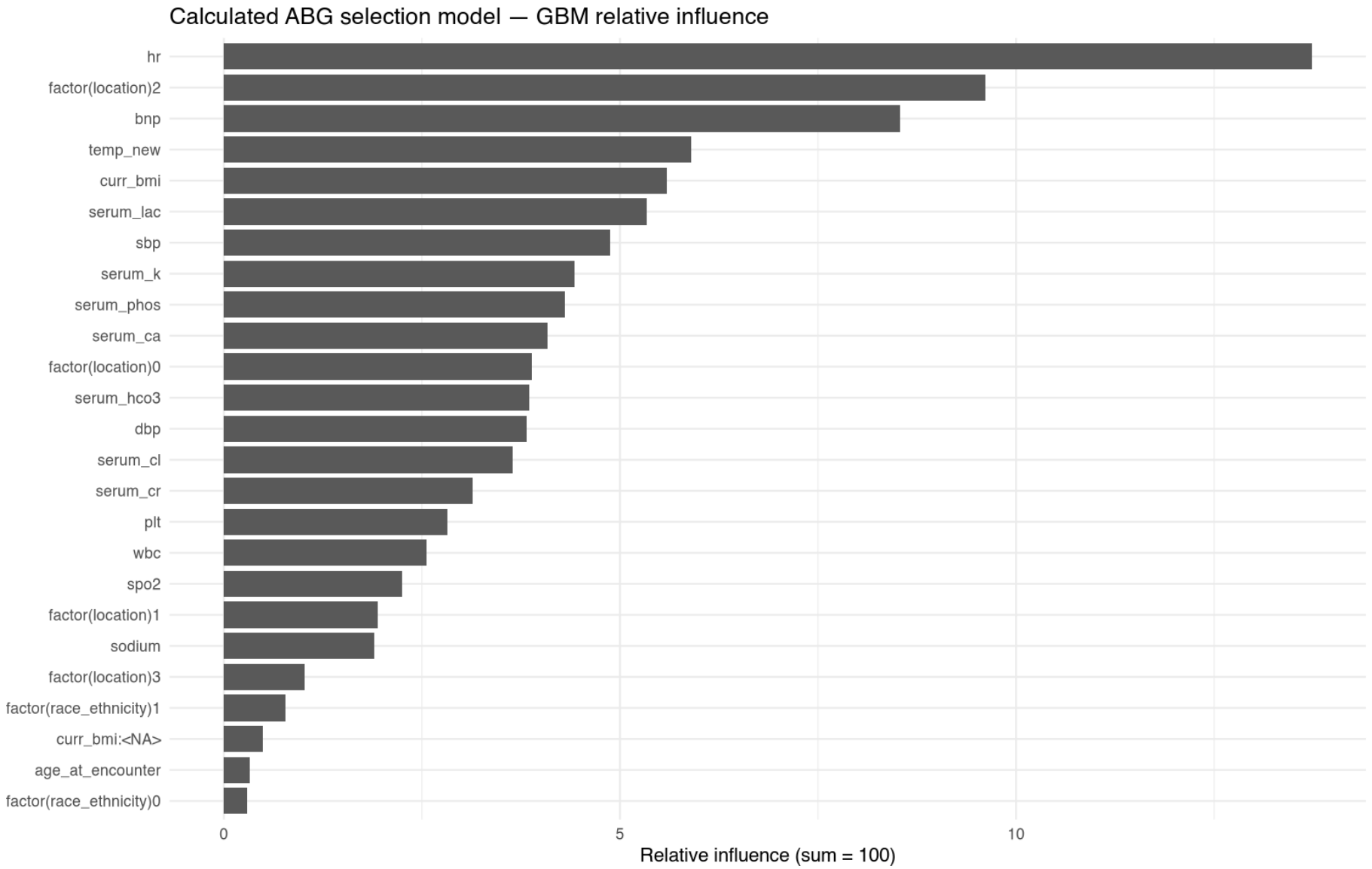
```r
ranked_vbg <- colnames(S_vbg)[order(colMeans(abs(S_vbg), na.rm = TRUE), decreasing = TRUE)]
top5_vbg   <- head(ranked_vbg, 5)
y_rng_vbg  <- range(unlist(lapply(top5_vbg, function(v) S_vbg[, v])), finite = TRUE)

theme_axes_compact <- function(show_y = FALSE, show_x = FALSE, base = 8) {
  theme_minimal(base_size = base) +
    theme(
      axis.title.y   = if (show_y) element_text(size = base) else element_blank(),
      axis.text.y    = if (show_y) element_text(size = base - 1) else element_blank(),
      axis.ticks.y   = if (show_y) element_line(linewidth = 0.2) else element_blank(),
      axis.title.x   = if (show_x) element_text(size = base) else element_blank(),
      axis.text.x    = if (show_x) element_text(size = base - 1) else element_blank(),
      plot.title     = element_text(size = base, hjust = 0),
      legend.title   = element_text(size = base - 1),
      legend.text    = element_text(size = base - 2),
      legend.key.height = unit(22, "pt"),
      legend.key.width  = unit(3,  "pt"),
      legend.margin     = margin(0, 0, 0, 0, "pt"),
      legend.box.margin = margin(0, 0, 0, 0, "pt")
    )
}

cell_plot_vbg <- function(v_row, v_col, i, j, n) {
  show_y <- (j == 1); show_x <- (i == n)
  if (identical(v_row, v_col)) {
    df <- data.frame(x = as.numeric(X_vbg[[v_row]]), shap = as.numeric(S_vbg[, v_row]))
    df <- df[is.finite(df$x) & is.finite(df$shap), , drop = FALSE]
    ggplot(df, aes(x = x, y = shap)) +
      geom_point(alpha = 0.30, size = 0.45, na.rm = TRUE) +
      scale_y_continuous(limits = y_rng_vbg) +
      labs(title = v_row, x = v_row, y = "SHAP") +
      theme_axes_compact(show_y, show_x, base = 8) +
      theme(legend.position = "none")
  } else {
    keep_legend <- (i == 1 && j == length(top5_vbg))
    shapviz::sv_dependence(sv_vbg, v = v_row, color_var = v_col, size = 0.4) +
```

```
      scale_y_continuous(limits = y_rng_vbg) +
      labs(title = paste0(v_row, " | color: ", v_col), x = v_row, y = "SHAP") +
      theme_axes_compact(show_y, show_x, base = 8) +
      guides(colour = guide_colorbar(barheight = unit(24, "pt"), barwidth = unit(3, "pt"),
                                     title.position = "top", title.hjust = 0.5, label.position = "right")) +
      theme(legend.position = if (keep_legend) "right" else "none")
  }
}

n <- length(top5_vbg); plots <- vector("list", n * n); k <- 1
for (i in seq_len(n)) for (j in seq_len(n)) {
  plots[[k]] <- cell_plot_vbg(top5_vbg[i], top5_vbg[j], i, j, n); k <- k + 1
}
patchwork::wrap_plots(plots, ncol = n, guides = "keep") +
  plot_annotation(title = "VBG propensity - Top-5 SHAP dependence (off-diagonal interactions, diagonal main effects)")
```


VBG propensity — Top-5 SHAP dependence (off-diagonal interactions, diagonal main effects)
curr_bmi
curr_bmi | color: temp_new
curr_bmi | color: wbc
curr_bmi | color: serum_phos
curr_bmi | color: sbp
temp_new | color: curr_bmi
temp_new
temp_new | color: wbc
temp_new | color: serum_phos
temp_new | color: sbp
wbc | color: curr_bmi
wbc | color: temp_new
wbc
wbc | color: serum_phos
wbc | color: sbp
serum_phos | color: curr_bmi
serum_phos | color: temp_new
serum_phos | color: wbc
serum_phos
serum_phos | color: sbp
sbp | color: curr_bmi
sbp | color: temp_new
sbp | color: wbc
sbp | color: serum_phos
sbp
SHAP
0.5
0.0
-0.5
-1.0
-1.5
50
100
150
200
250
sbp
sbp
250
200
150
100

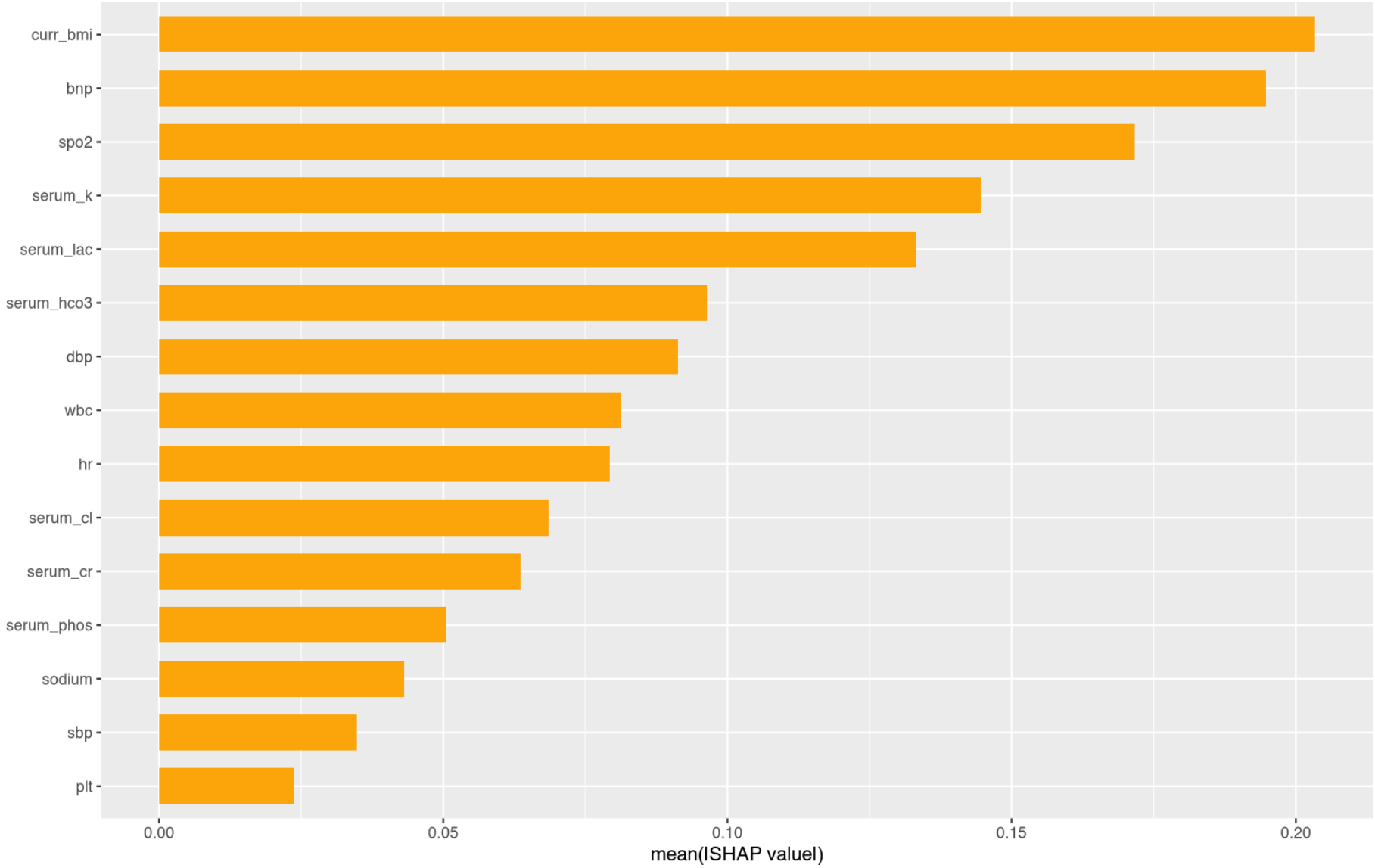
1.6.3 Calculated ABG explainability

```
library(WeightIt); library(gbm); library(dplyr); library(ggplot2)

stopifnot(exists("w_vbg_calc", inherits = TRUE))
w_vbg_calc <- ensure_gbm_obj(w_vbg_calc)

imp_calc   <- extract_gbm_importance(w_vbg_calc, top_n = 25)
p_imp_calc <- plot_gbm_importance(imp_calc, "Calculated ABG selection model - GBM relative influence")
p_imp_calc
```

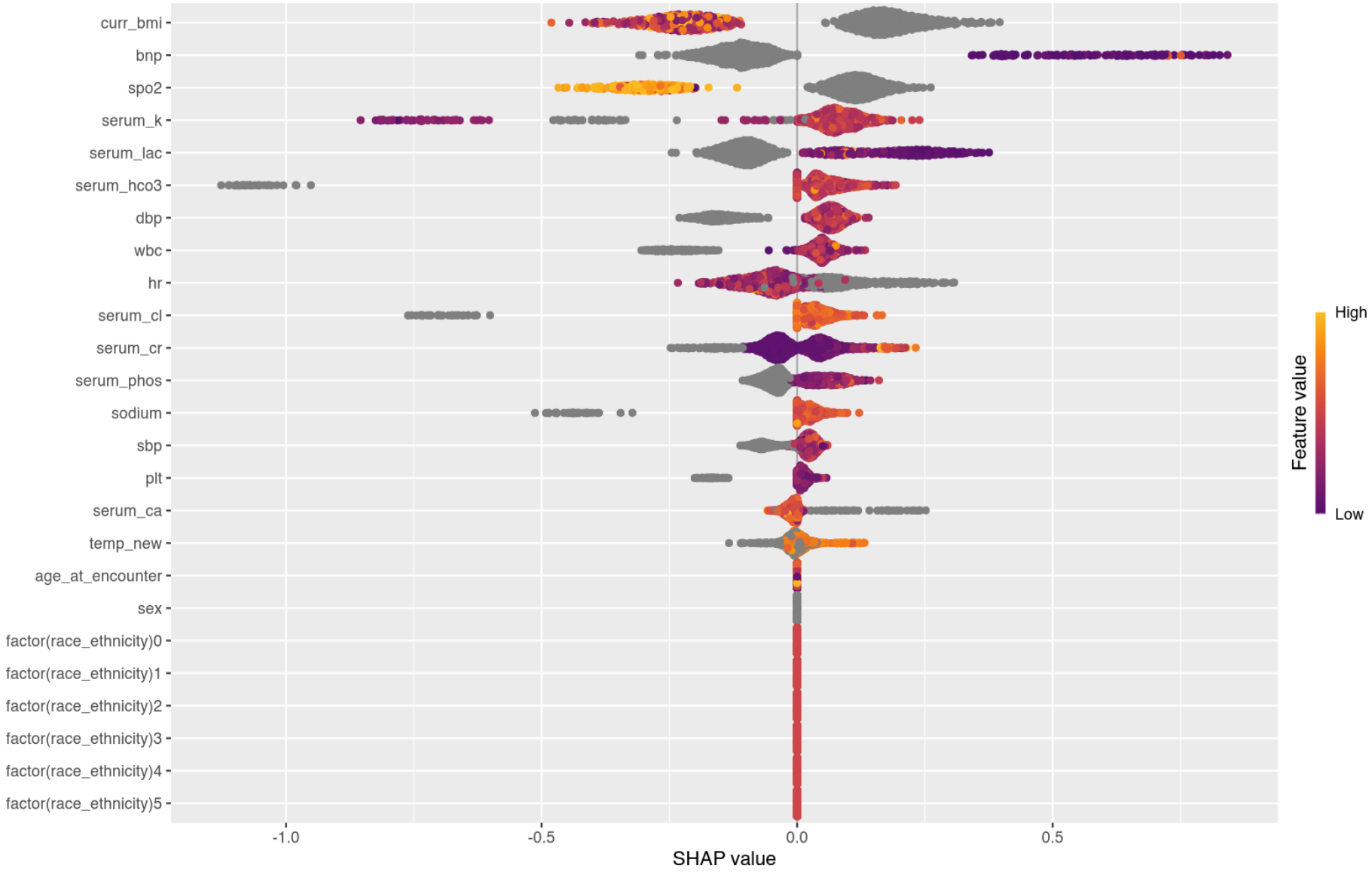



```
library(shapviz); library(fastshap)

t0 <- Sys.time()
sh_calc_fast <- compute_shap_fast(w_vbg_calc, top_k = 100, nsim = 32, frac_rows = 0.25, max_rows = 100000)
```

```r
t1 <- Sys.time(); message(sprintf("[compute_shap_fast Calc-ABG] %.2f s", as.numeric(difftime(t1, t0, units="secs"))))

S_calc <- as.matrix(sh_calc_fast$shap)
X_calc <- as.data.frame(sh_calc_fast$X)

for (nm in names(X_calc)) {
  if (inherits(X_calc[[nm]], "haven_labelled")) X_calc[[nm]] <- labelled::to_factor(X_calc[[nm]])
  if (is.factor(X_calc[[nm]]))  X_calc[[nm]] <- as.character(X_calc[[nm]])
  if (is.character(X_calc[[nm]])) suppressWarnings(X_calc[[nm]] <- as.numeric(X_calc[[nm]]))
}
if (is.null(colnames(S_calc))) colnames(S_calc) <- colnames(X_calc)
S_calc <- S_calc[, intersect(colnames(S_calc), colnames(X_calc)), drop = FALSE]
X_calc <- X_calc[, colnames(S_calc), drop = FALSE]

sv_calc <- shapviz::shapviz(S_calc, X = as.matrix(X_calc))

ord_calc  <- order(colMeans(abs(S_calc), na.rm = TRUE), decreasing = TRUE)
topK_calc <- colnames(S_calc)[ord_calc[1:min(30, ncol(S_calc))]]
sv_importance(sv_calc, kind = "bar", v = topK_calc)
```

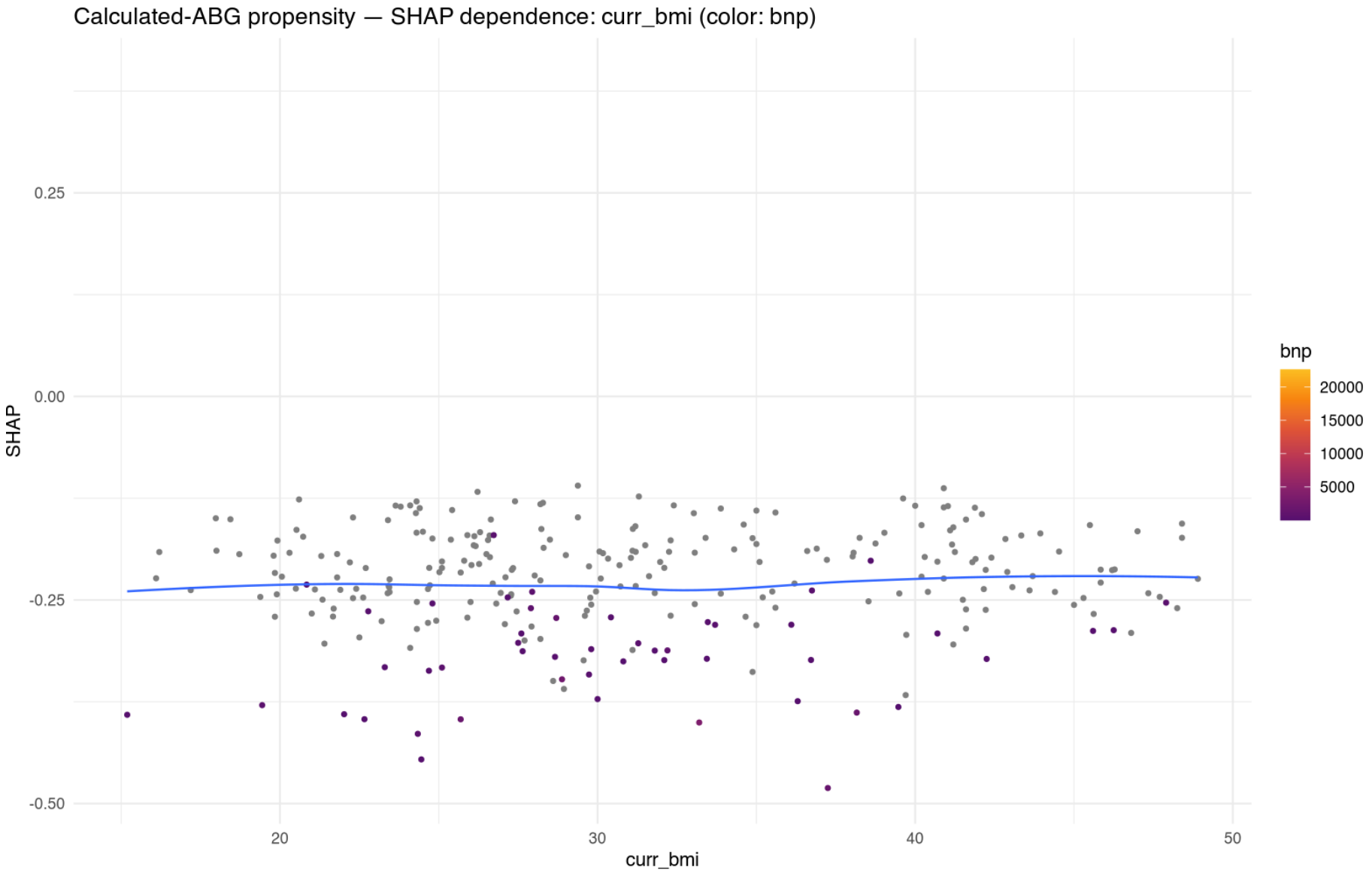



```
library(shapviz)
sv_importance(sv_calc, kind = "beeswarm", max_display = 25)
```




```
library(ggplot2)

imp_order_calc <- colnames(S_calc)[order(colMeans(abs(S_calc), na.rm = TRUE), decreasing = TRUE)]
pri_calc <- imp_order_calc[1]
```

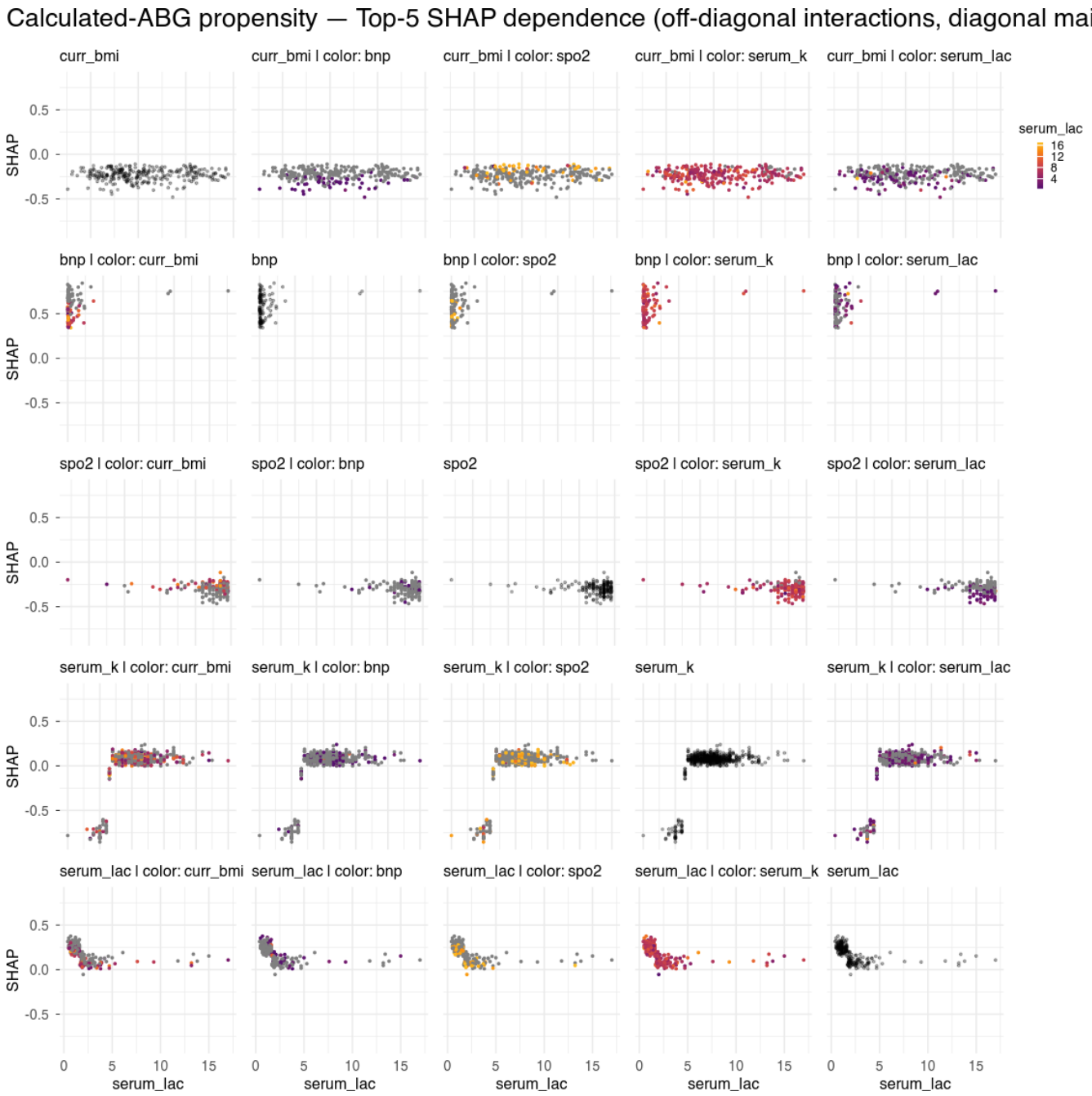
```r
aux_calc <- imp_order_calc[2]
if (identical(aux_calc, pri_calc) || !(aux_calc %in% colnames(X_calc))) aux_calc <- imp_order_calc[3]

shapviz::sv_dependence(sv_calc, v = pri_calc, color_var = aux_calc, size = 1) +
  geom_smooth(se = FALSE, method = "loess", formula = y ~ x, linewidth = 0.6) +
  labs(title = sprintf("Calculated-ABG propensity - SHAP dependence: %s (color: %s)", pri_calc, aux_calc),
       x = pri_calc, y = "SHAP") +
  theme_minimal(base_size = 11)
```




```
library(shapviz); library(ggplot2); library(patchwork); library(grid)

stopifnot(is.matrix(S_calc), is.data.frame(X_calc))
```

```
ranked_calc <- colnames(S_calc)[order(colMeans(abs(S_calc), na.rm = TRUE), decreasing = TRUE)]
top5_calc   <- head(ranked_calc, 5)
y_rng_calc  <- range(unlist(lapply(top5_calc, function(v) S_calc[, v])), finite = TRUE)

theme_axes_compact <- function(show_y = FALSE, show_x = FALSE, base = 8) {
  theme_minimal(base_size = base) +
    theme(
      axis.title.y   = if (show_y) element_text(size = base) else element_blank(),
      axis.text.y    = if (show_y) element_text(size = base - 1) else element_blank(),
      axis.ticks.y   = if (show_y) element_line(linewidth = 0.2) else element_blank(),
      axis.title.x   = if (show_x) element_text(size = base) else element_blank(),
      axis.text.x    = if (show_x) element_text(size = base - 1) else element_blank(),
      plot.title     = element_text(size = base, hjust = 0),
      legend.title   = element_text(size = base - 1),
      legend.text    = element_text(size = base - 2),
      legend.key.height = unit(22, "pt"),
      legend.key.width  = unit(3,  "pt"),
      legend.margin     = margin(0, 0, 0, 0, "pt"),
      legend.box.margin = margin(0, 0, 0, 0, "pt")
    )
}

cell_plot_calc <- function(v_row, v_col, i, j, n) {
  show_y <- (j == 1); show_x <- (i == n)
  if (identical(v_row, v_col)) {
    df <- data.frame(x = as.numeric(X_calc[[v_row]]), shap = as.numeric(S_calc[, v_row]))
    df <- df[is.finite(df$x) & is.finite(df$shap), , drop = FALSE]
    ggplot(df, aes(x = x, y = shap)) +
      geom_point(alpha = 0.30, size = 0.45, na.rm = TRUE) +
      scale_y_continuous(limits = y_rng_calc) +
      labs(title = v_row, x = v_row, y = "SHAP") +
      theme_axes_compact(show_y, show_x, base = 8) +
      theme(legend.position = "none")
  } else {
    keep_legend <- (i == 1 && j == length(top5_calc))
    shapviz::sv_dependence(sv_calc, v = v_row, color_var = v_col, size = 0.4) +
```

```
      scale_y_continuous(limits = y_rng_calc) +
      labs(title = paste0(v_row, " | color: ", v_col), x = v_row, y = "SHAP") +
      theme_axes_compact(show_y, show_x, base = 8) +
      guides(colour = guide_colorbar(barheight = unit(24, "pt"), barwidth = unit(3, "pt"),
                                     title.position = "top", title.hjust = 0.5, label.position = "right")) +
      theme(legend.position = if (keep_legend) "right" else "none")
  }
}

n <- length(top5_calc); plots <- vector("list", n * n); k <- 1
for (i in seq_len(n)) for (j in seq_len(n)) {
  plots[[k]] <- cell_plot_calc(top5_calc[i], top5_calc[j], i, j, n); k <- k + 1
}
patchwork::wrap_plots(plots, ncol = n, guides = "keep") +
  plot_annotation(title = "Calculated-ABG propensity - Top-5 SHAP dependence (off-diagonal interactions, diagonal main effects)"
```


Calculated-ABG propensity — Top-5 SHAP dependence (off-diagonal interactions, diagonal mair
curr_bmi
curr_bmi | color: bnp
curr_bmi | color: spo2
curr_bmi | color: serum_k
curr_bmi | color: serum_lac
bnp | color: curr_bmi
bnp
bnp | color: spo2
bnp | color: serum_k
bnp | color: serum_lac
spo2 | color: curr_bmi
spo2 | color: bnp
spo2
spo2 | color: serum_k
spo2 | color: serum_lac
serum_k | color: curr_bmi
serum_k | color: bnp
serum_k | color: spo2
serum_k
serum_k | color: serum_lac
serum_lac | color: curr_bmi
serum_lac | color: bnp
serum_lac | color: spo2
serum_lac | color: serum_k
serum_lac
SHAP
0.5
0.0
-0.5
0
5
10
15
serum_lac
serum_lac
16
12
8
4

1.7 Weighted effect estimates

New weighted binary regression figures.

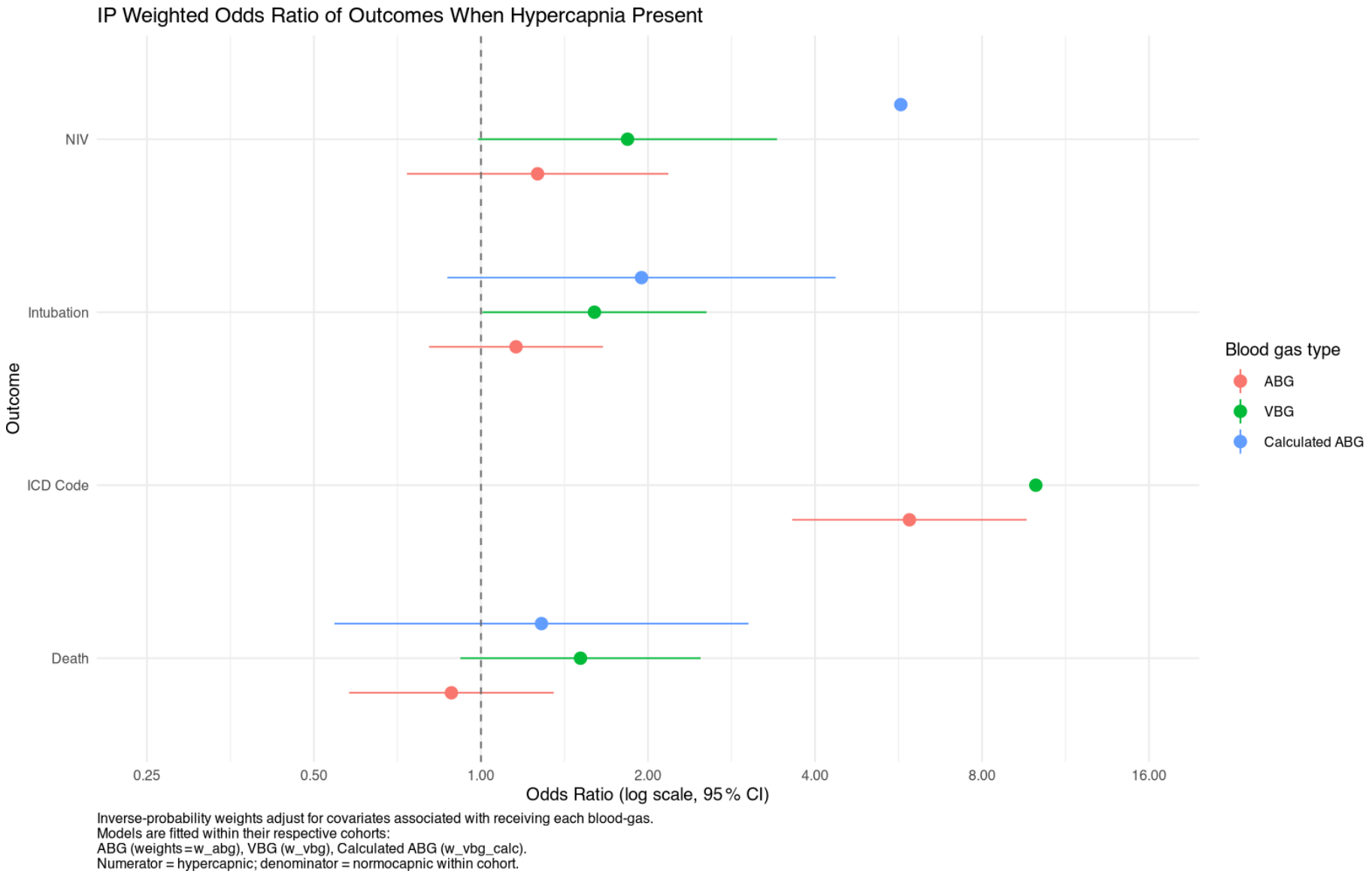
```
#  IP-weighted odds-ratio plot (ABG, VBG, Calculated-ABG)
#    - exact analogue of the un-weighted figure
#

# weights already attached earlier:
#   • w_abg         - propensity for *ABG*   (column in subset_data)
#   • w_vbg         - propensity for *VBG*   (column in subset_data)
#   • w_vbg_calc    - same weights, used for calculated ABG CO

# 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)
calc_df  <- subset_data %>% filter(!is.na(calc_abg))   # implies VBG present

# 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"),

  # Calculated ABG
  tidy_ipw(calc_df, "imv_proc",              "hypercapnia_calc","w_vbg_calc", "Calculated ABG","Intubation"),
  tidy_ipw(calc_df, "niv_proc",              "hypercapnia_calc","w_vbg_calc", "Calculated ABG","NIV"),
  tidy_ipw(calc_df, "death_60d",                 "hypercapnia_calc","w_vbg_calc", "Calculated ABG","Death"),
  tidy_ipw(calc_df, "hypercap_resp_failure", "hypercapnia_calc","w_vbg_calc", "Calculated ABG","ICD Code")
)

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

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) +
```

```r
  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), Calculated ABG (w_vbg_calc).",
      "Numerator = hypercapnic; denominator = normocapnic within cohort.",
      sep = "\n"
    )
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_pointrange()`).
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



1.7.1 Three-level PCO categories (weighted; ABG, VBG, Calc ABG)

Three Groups with Weights

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

#   1. Create PCO  categories
subset_data <- subset_data %>%
  mutate(
    pco2_cat_abg = case_when(
      !is.na(paco2) & paco2 < 35 ~ "Below normal",
      !is.na(paco2) & paco2 >= 35 & paco2 <= 45 ~ "Normal",
      !is.na(paco2) & paco2 > 45 ~ "Above normal",
      TRUE ~ NA_character_
    ),
    pco2_cat_vbg = case_when(
      !is.na(vbg_co2) & vbg_co2 < 40 ~ "Below normal",
      !is.na(vbg_co2) & vbg_co2 >= 40 & vbg_co2 <= 50 ~ "Normal",
      !is.na(vbg_co2) & vbg_co2 > 50 ~ "Above normal",
      TRUE ~ NA_character_
    ),
    pco2_cat_calc = case_when(
      !is.na(calc_abg) & calc_abg < 35 ~ "Below normal",
      !is.na(calc_abg) & calc_abg >= 35 & calc_abg <= 45 ~ "Normal",
      !is.na(calc_abg) & calc_abg > 45 ~ "Above normal",
      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]]),
```

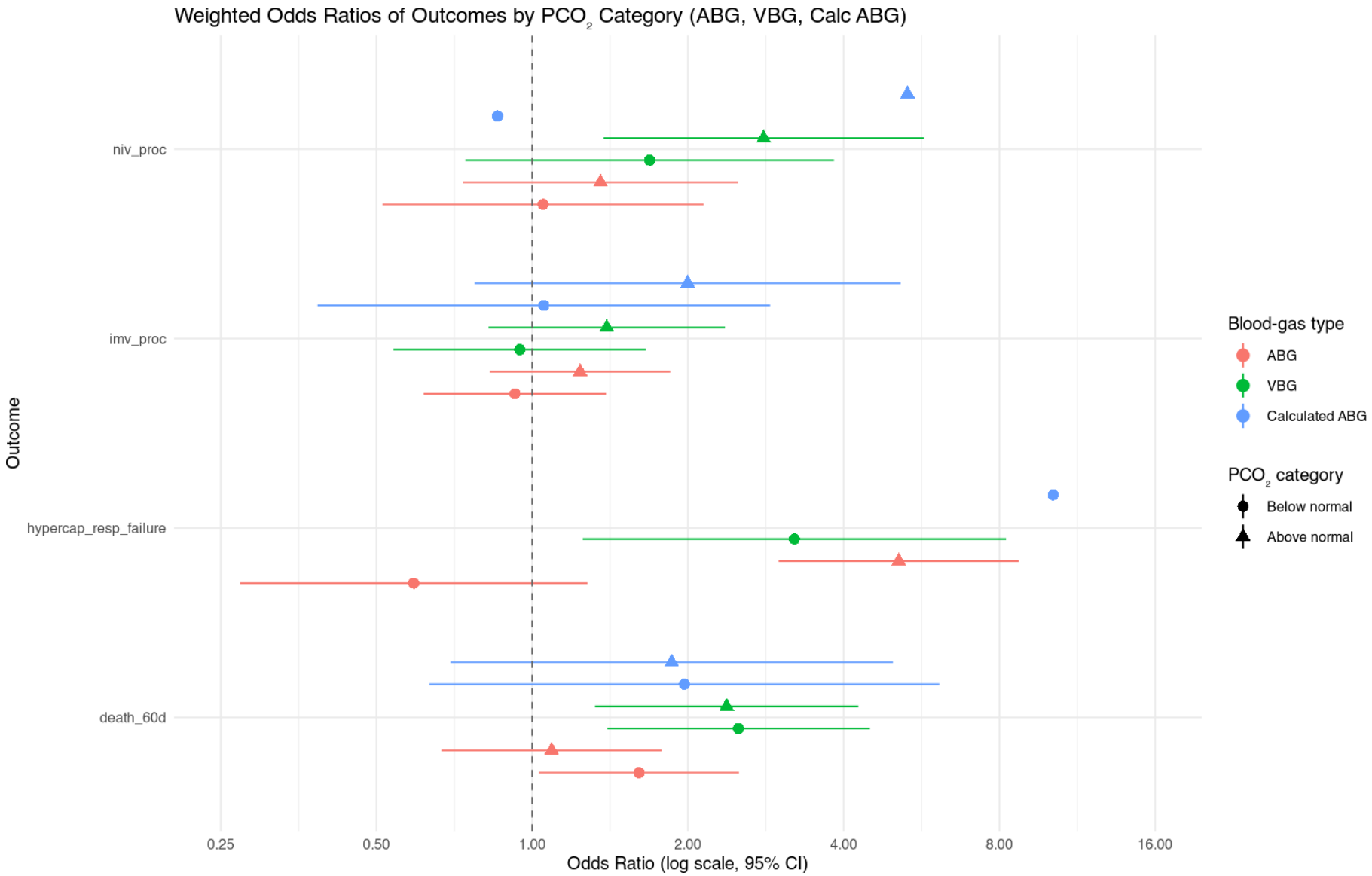
```r
      .data[[weight_var]] > 0
    ) %>%
    mutate(
      !!cat_var := factor(.data[[cat_var]],
                          levels = c("Normal", "Below normal", "Above normal"))
    ) %>%
    droplevels()

  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")),
  lapply(outcomes, function(out)
```

```
    run_weighted_or(subset_data, out, "pco2_cat_calc", "w_vbg_calc", "Calculated ABG"))
)

# Ensure nice ordering
combined_or_df$group    <- factor(combined_or_df$group,
                                  levels = c("ABG", "VBG", "Calculated ABG"))
combined_or_df$exposure <- factor(combined_or_df$exposure,
                                  levels = c("Below normal", "Above normal"))

#   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 PCO  Category (ABG, VBG, Calc ABG)",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95% CI)",
    color  = "Blood-gas type",
    shape  = "PCO  category"
  ) +
```

```
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_pointrange()`).
```

```
Warning: Removed 3 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



1.7.2 Three-level PCO categories (weighted; ABG vs VBG only)

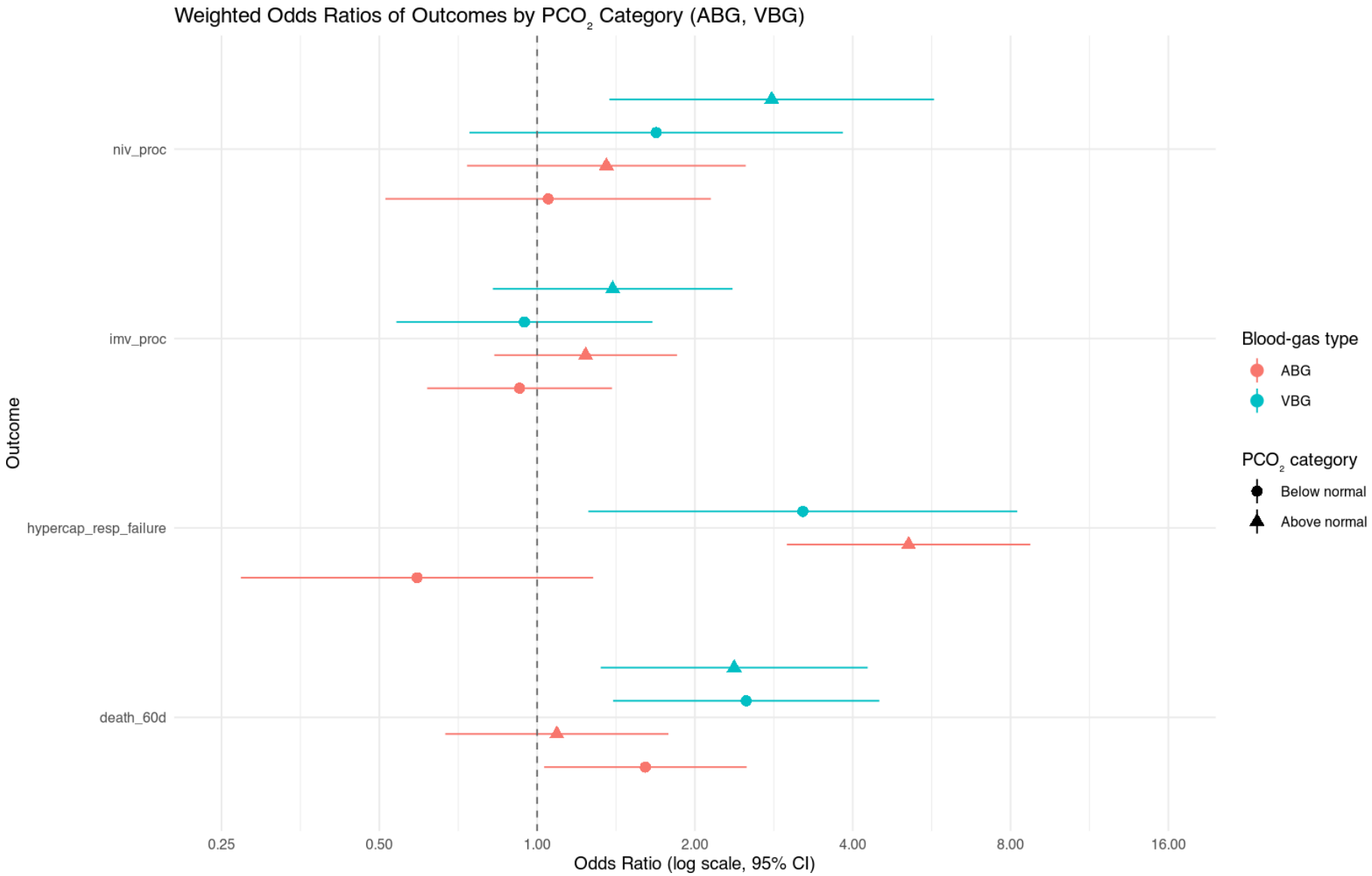
Three groups with weights: Just ABG and VBG

```r
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("Normal", "Below normal", "Above normal"))
    ) %>%
    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("Below normal", "Above normal"))

#   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 PCO  Category (ABG, VBG)",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95% CI)",
    color  = "Blood-gas type",
    shape  = "PCO  category"
  ) +
  theme_minimal(base_size = 10) +
  theme(plot.caption = element_text(hjust = 0))
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_pointrange()`).
```



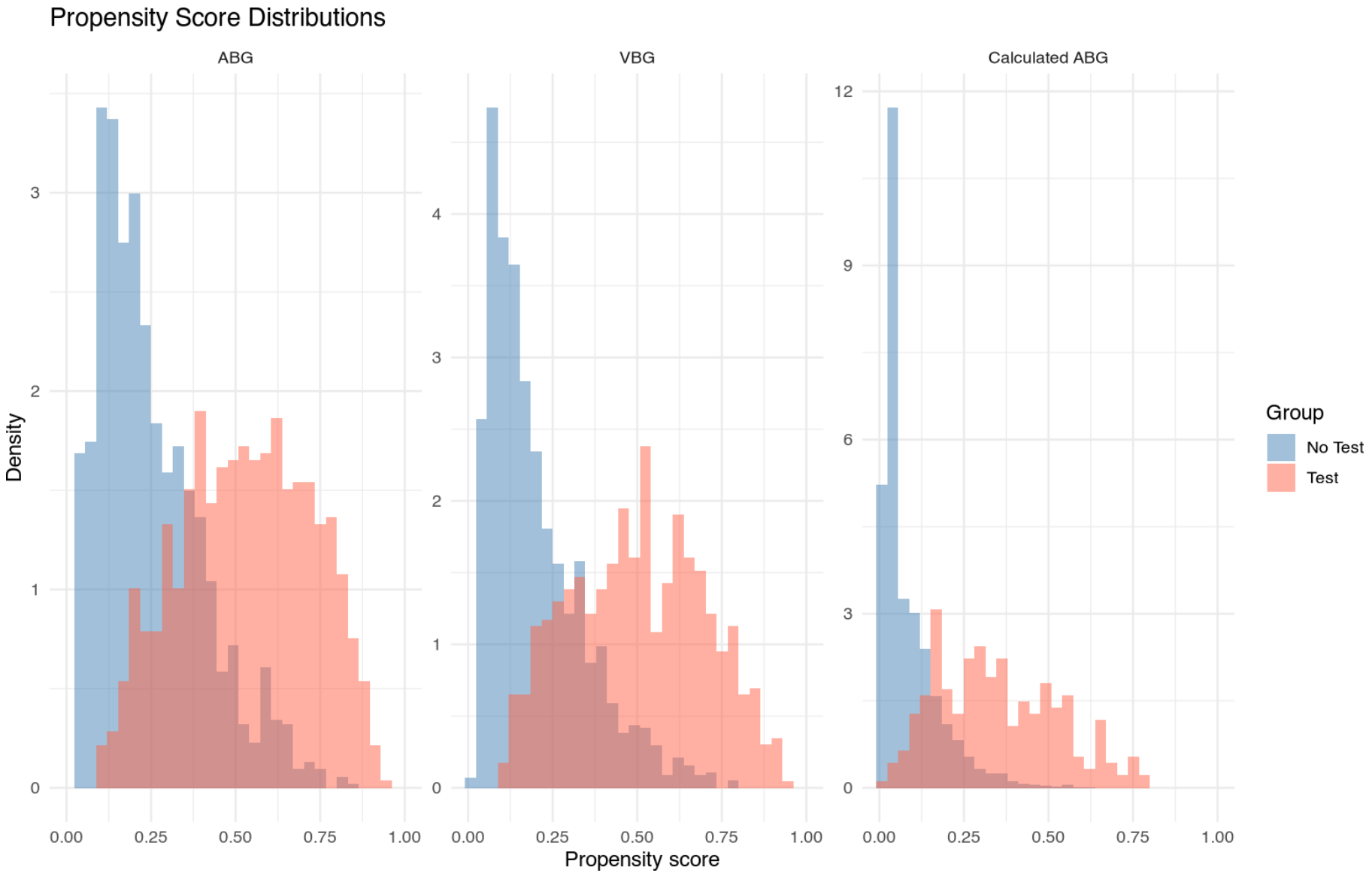
1.8 Propensity score diagnostics

Plotting propensity scores

```r
# --- 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
w_vbg_calc_obj <- if (exists("w_vbg_calc")) w_vbg_calc else 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.")
}

# Calculated ABG uses the VBG selection model; prefer a dedicated `w_vbg_calc` if present
if (!is.null(w_vbg_calc_obj) && "has_vbg_co2_o2_sat" %in% names(subset_data)) {
```

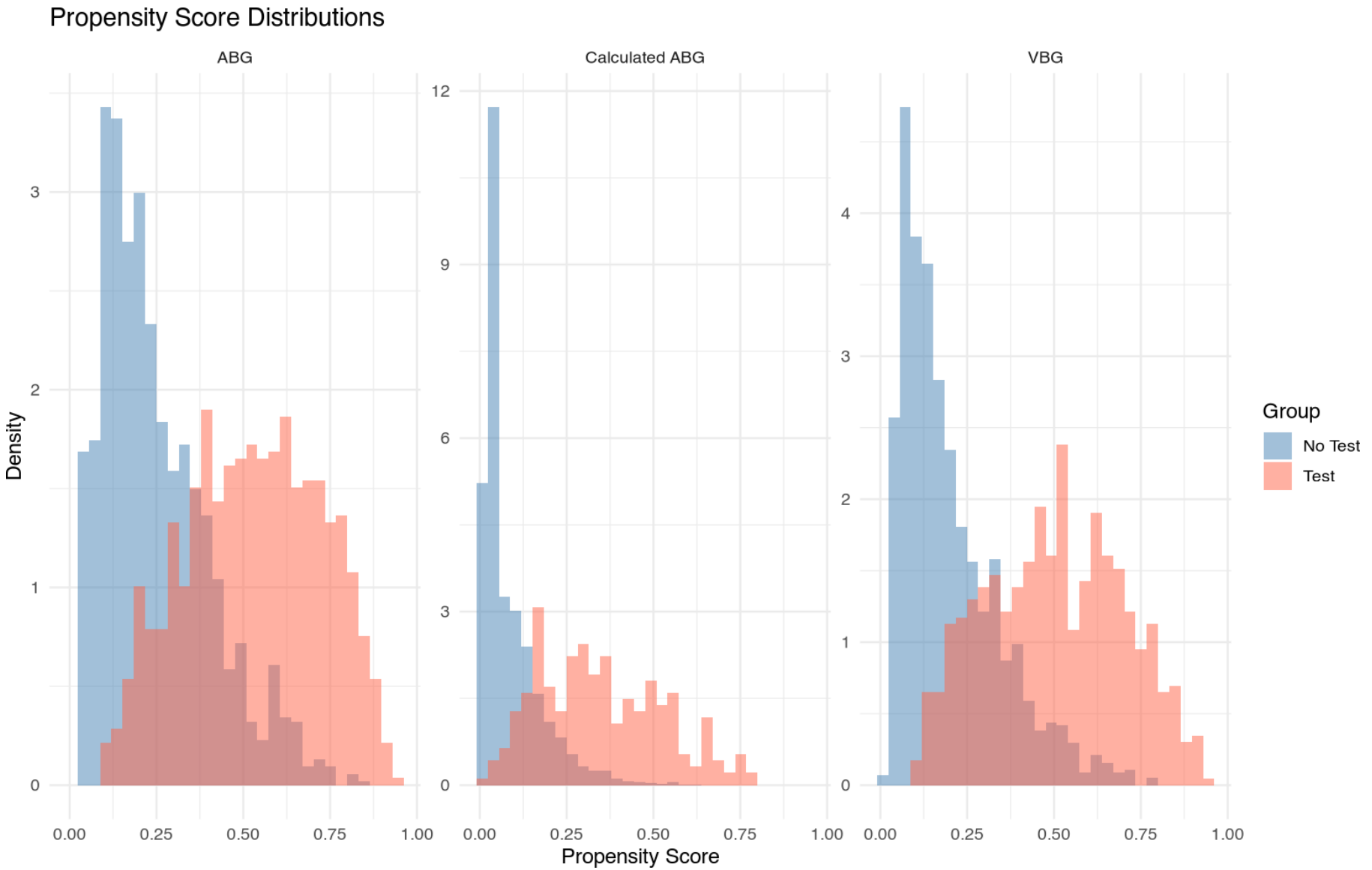
```r
  ps_dfs$CalcABG <- data.frame(
    ps        = w_vbg_calc_obj$ps,
    treat     = subset_data$has_vbg_co2_o2_sat,
    ScoreType = "Calculated ABG"
  )
} else if (is.null(w_vbg_calc_obj)) {
  message("Note: Calculated-ABG WeightIt object `w_vbg_calc` (or fallback `w_vbg`) not found; skipping Calc-ABG panel.")
}

# Bind, clean, and factorize for plotting
df_ps <- bind_rows(ps_dfs) %>%
  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", "Calculated ABG"))
  )

# 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.
```




```
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$has_vbg,
    ScoreType = "VBG"
  ),
  data.frame(
    ps        = w_vbg_calc$ps,
    treat     = subset_data$has_vbg_co2_o2_sat,
    ScoreType = "Calculated ABG"
  )
) %>%
  mutate(
    treat = factor(treat, levels = c(0,1), labels = c("No Test", "Test"))
  )

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
ABG
Calculated ABG
VBG
Density
Propensity Score
Group
No Test
Test
0.00
0.25
0.50
0.75
1.00

2 Multiple Imputation Analysis

added 12/6/2025

2.1 1.) 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(mitools)      # 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
```

```r
# 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
```

2.1.1 1.1) Missingness audit (what, where, how much)

```r
# High-level profiles
skimr::skim(subset_data)
```

Table 13: Data summary

Name	subset_data
Number of rows	2491
Number of columns	579
_____	
Column type frequency:	
character	14
Date	3

factor	19
numeric	543
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
encounter_id	0	1.00	4	6	0	2491	0
rfs	0	1.00	3	14	0	7	0
bnp_date	0	1.00	0	10	2034	275	0
serum_phos_date	0	1.00	0	10	1310	355	0
serum_ca_date	0	1.00	0	10	198	366	0
serum_albumin_date	0	1.00	0	10	857	361	0
serum_tprot_date	0	1.00	0	10	945	362	0
principal_diagnosis_indicator	0	1.00	0	1	2402	4	0
admitting_diagnosis	0	1.00	0	1	2402	2	0
reason_for_visit	0	1.00	0	1	2402	2	0
pat_enc_hash	0	1.00	9	12	0	2491	0
pco2_cat_abg	1630	0.35	6	12	0	3	0
pco2_cat_vbg	1778	0.29	6	12	0	3	0
pco2_cat_calc	2200	0.12	6	12	0	3	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
encounter_date	0	1.00	2022-01-01	2022-12-31	2022-06-10	365
death_abs	2144	0.14	1995-06-01	2023-05-01	2022-07-01	29
ref_ym	0	1.00	2024-06-01	2024-06-01	2024-06-01	1

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
encounter_type	0	1	FALSE	2	Inp: 1629, Eme: 862
abg_group	0	1	FALSE	3	No : 1630, ABG: 605, ABG: 256, Mis: 0
vbg_group	0	1	FALSE	3	No : 1778, VBG: 516, VBG: 197, Mis: 0
sex_label	0	1	FALSE	2	Mal: 1248, Fem: 1243
race_ethnicity_label	0	1	FALSE	7	Whi: 1487, Bla: 460, Unk: 319, His: 165
location_label	0	1	FALSE	4	Sou: 1156, Nor: 631, Wes: 529, Mid: 175
encounter_type_label	0	1	FALSE	2	Inp: 1629, Eme: 862
osa_label	0	1	FALSE	2	No: 2071, Yes: 420
asthma_label	0	1	FALSE	2	No: 2191, Yes: 300
copd_label	0	1	FALSE	2	No: 2028, Yes: 463
chf_label	0	1	FALSE	2	No: 2009, Yes: 482
nmd_label	0	1	FALSE	2	No: 2389, Yes: 102
phtn_label	0	1	FALSE	2	No: 2317, Yes: 174
ckd_label	0	1	FALSE	2	No: 2009, Yes: 482
diabetes_label	0	1	FALSE	2	No: 1755, Yes: 736
abg_status	0	1	FALSE	2	Did: 1630, Did: 861
vbg_status	0	1	FALSE	2	Did: 1778, Did: 713
hyper_abg	0	1	FALSE	2	Got: 2235, Got: 256
hyper_vbg	0	1	FALSE	2	Got: 2294, Got: 197

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sex	0	1.00	0.50	0.50	0.00	0.00	1.00	1.00	1.00	
race	0	1.00	0.60	0.89	0.00	0.00	0.00	1.00	5.00	
ethnicity	0	1.00	0.43	0.78	0.00	0.00	0.00	0.00	2.00	
location	0	1.00	1.03	1.18	0.00	0.00	1.00	2.00	3.00	
age_at_encounter	0	1.00	58.77	17.57	18.00	46.00	61.00	72.00	90.00	
los	0	1.00	9.95	28.71	1.00	2.00	4.00	9.00	365.00	
curr_bmi	1409	0.43	31.22	8.59	13.60	24.40	29.75	38.86	49.78	
spo2	1776	0.29	94.39	6.22	54.00	93.00	96.00	98.00	99.40	
hr	892	0.64	85.57	19.63	35.00	71.00	84.00	98.00	192.00	
curr_height	678	0.73	66.91	4.26	56.00	64.00	67.00	70.00	81.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
vbg_temp	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
abg_temp	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
vbg_o2sat	2186	0.12	64.29	21.36	12.50	49.80	65.70	82.00	99.10	
abg_o2sat	1994	0.20	88.83	16.92	14.00	90.00	96.00	98.00	99.90	
sao2_blood	2372	0.05	80.74	22.77	11.00	68.90	92.00	97.00	99.30	
value_prev_weight	2262	0.09	195.17	61.40	82.00	147.00	187.00	231.00	398.00	
value_prev_height	2169	0.13	67.00	4.04	56.00	64.00	67.00	70.00	76.00	
value_prev_bmi	2337	0.06	29.42	8.48	15.00	23.00	28.00	36.00	49.00	
value_highest_20198	1930	0.23	47.89	20.02	15.40	37.00	44.00	52.00	198.00	
value_highest_115576	2254	0.10	45.81	13.31	20.60	38.00	44.00	51.00	125.00	
value_highest_327718	2375	0.05	65.20	50.39	22.00	38.00	43.00	63.00	247.00	
vbg_co2	1778	0.29	45.98	13.13	13.00	38.00	44.00	50.00	136.00	
highest_vbg_co2	1777	0.29	48.96	15.41	19.30	40.12	46.00	54.00	145.00	
pco2_nos	2221	0.11	41.33	10.70	19.00	34.00	40.10	47.00	97.00	
highest_pco2_nos	2221	0.11	46.94	13.89	20.60	38.00	45.00	52.22	125.00	
abg_ph	1619	0.35	7.37	0.11	6.83	7.32	7.38	7.44	7.70	
vbg_ph	1735	0.30	7.35	0.10	6.76	7.31	7.36	7.41	7.62	
abg_hco3	1964	0.21	23.35	6.08	4.00	20.05	23.20	26.00	51.80	
vbg_hco3	1851	0.26	24.83	6.04	3.10	21.97	24.70	28.00	56.00	
sodium	137	0.95	137.17	4.49	107.00	135.00	138.00	140.00	160.00	
serum_cr	217	0.91	1.45	1.80	0.20	0.76	0.98	1.32	19.40	
hgb	272	0.89	12.38	2.65	4.00	10.70	12.60	14.20	23.60	
serum_hco3	147	0.94	23.77	4.80	3.00	21.00	24.00	26.40	50.30	
serum_cl	154	0.94	102.15	5.55	66.00	99.00	103.00	106.00	123.00	
serum_lac	1501	0.40	2.10	2.10	0.30	1.00	1.50	2.30	21.80	
serum_k	189	0.92	4.08	0.67	2.00	3.70	4.00	4.40	8.90	
temp_cor_oxygen	2435	0.02	50.09	14.27	31.00	41.75	46.50	56.25	84.00	
vbg_ph_temp_cor	2437	0.02	7.36	0.10	7.12	7.32	7.37	7.41	7.55	
vbg_po2	1865	0.25	43.74	23.43	14.00	30.00	38.00	51.00	247.00	
vbg_lactate	2397	0.04	2.07	2.02	0.11	1.10	1.60	2.30	15.89	
vbg_hco3_calc	2415	0.03	25.57	6.94	8.00	21.75	25.50	29.00	47.00	
abg_po2	1956	0.21	134.21	92.71	20.50	74.00	100.00	162.20	495.00	
abg_po2_temp_cor	2373	0.05	104.54	98.20	20.00	37.00	77.00	118.25	497.00	
abg_ph_temp_cor	2389	0.04	7.37	0.10	7.06	7.32	7.38	7.43	7.63	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
abg_lactate	2405	0.03	2.22	2.08	0.50	0.90	1.50	2.80	10.91	
ph_blood	2249	0.10	7.32	0.15	6.91	7.28	7.36	7.41	7.60	
po2_blood	2262	0.09	83.26	80.96	11.00	38.00	58.00	87.00	481.00	
wbc	421	0.83	10.28	5.48	0.10	6.78	9.02	12.54	79.20	
plt	204	0.92	249.39	105.10	2.80	183.00	234.00	296.00	1159.00	
bnp	2052	0.18	1008.89	4626.79	2.00	59.25	180.00	580.50	80400.00	
serum_phos	1343	0.46	3.65	1.46	0.90	2.80	3.40	4.10	13.80	
serum_ca	242	0.90	8.95	0.87	4.50	8.50	9.00	9.40	15.80	
serum_albumin	897	0.64	3.59	0.72	1.10	3.10	3.70	4.10	5.60	
serum_tprot	947	0.62	6.80	1.02	2.70	6.20	6.90	7.50	10.10	
has_j9612	0	1.00	0.01	0.08	0.00	0.00	0.00	0.00	1.00	
has_j9622	0	1.00	0.02	0.13	0.00	0.00	0.00	0.00	1.00	
has_j9602	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
has_j9692	0	1.00	0.01	0.08	0.00	0.00	0.00	0.00	1.00	
ohs_code	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
has_j9600	0	1.00	0.04	0.19	0.00	0.00	0.00	0.00	1.00	
has_j9601	0	1.00	0.19	0.39	0.00	0.00	0.00	0.00	1.00	
has_j961	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
has_j9610	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
has_j9611	0	1.00	0.02	0.14	0.00	0.00	0.00	0.00	1.00	
has_j962	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
has_j9620	0	1.00	0.00	0.06	0.00	0.00	0.00	0.00	1.00	
has_j9621	0	1.00	0.04	0.20	0.00	0.00	0.00	0.00	1.00	
has_j9690	0	1.00	0.03	0.16	0.00	0.00	0.00	0.00	1.00	
has_j9691	0	1.00	0.01	0.11	0.00	0.00	0.00	0.00	1.00	
other_abn_of_br	0	1.00	0.02	0.12	0.00	0.00	0.00	0.00	1.00	
cfdo	0	1.00	0.00	0.04	0.00	0.00	0.00	0.00	1.00	
has_i50_acute	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
acute_nmd	0	1.00	0.00	0.03	0.00	0.00	0.00	0.00	1.00	
sepsis_dx	0	1.00	0.11	0.32	0.00	0.00	0.00	0.00	1.00	
stupor_dx	0	1.00	0.03	0.16	0.00	0.00	0.00	0.00	1.00	
cog_signs_dx	0	1.00	0.11	0.31	0.00	0.00	0.00	0.00	1.00	
mal_fat_dx	0	1.00	0.10	0.31	0.00	0.00	0.00	0.00	1.00	
resp_acid_dx	0	1.00	0.01	0.07	0.00	0.00	0.00	0.00	1.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sleep_hypovent_dx	0	1.00	0.00	0.03	0.00	0.00	0.00	0.00	1.00	
cchs_dx	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
other_sleep_hypovent_dx	0	1.00	0.00	0.03	0.00	0.00	0.00	0.00	1.00	
acidosis_unspec	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
headache_dx	0	1.00	0.03	0.17	0.00	0.00	0.00	0.00	1.00	
cpap	0	1.00	0.07	0.25	0.00	0.00	0.00	0.00	1.00	
tte_proc	0	1.00	0.12	0.33	0.00	0.00	0.00	0.00	1.00	
aero	0	1.00	0.12	0.33	0.00	0.00	0.00	0.00	1.00	
inh_teaching	0	1.00	0.02	0.16	0.00	0.00	0.00	0.00	1.00	
cxr1v	0	1.00	0.26	0.44	0.00	0.00	0.00	1.00	1.00	
cxr2v	0	1.00	0.03	0.17	0.00	0.00	0.00	0.00	1.00	
ctcnoncon	0	1.00	0.03	0.17	0.00	0.00	0.00	0.00	1.00	
ctccon	0	1.00	0.05	0.21	0.00	0.00	0.00	0.00	1.00	
cc_time	0	1.00	0.22	0.41	0.00	0.00	0.00	0.00	1.00	
meas_venous_o2_proc	0	1.00	0.02	0.13	0.00	0.00	0.00	0.00	1.00	
meas_arterial_gas_proc	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
blood_cx_proc	0	1.00	0.13	0.34	0.00	0.00	0.00	0.00	1.00	
art_punct_proc	0	1.00	0.06	0.23	0.00	0.00	0.00	0.00	1.00	
ctabdpelv	0	1.00	0.05	0.23	0.00	0.00	0.00	0.00	1.00	
osa	0	1.00	0.17	0.37	0.00	0.00	0.00	0.00	1.00	
asthma	0	1.00	0.12	0.33	0.00	0.00	0.00	0.00	1.00	
copd	0	1.00	0.19	0.39	0.00	0.00	0.00	0.00	1.00	
chf	0	1.00	0.19	0.40	0.00	0.00	0.00	0.00	1.00	
stroke	0	1.00	0.07	0.26	0.00	0.00	0.00	0.00	1.00	
ckd	0	1.00	0.19	0.40	0.00	0.00	0.00	0.00	1.00	
pvd	0	1.00	0.09	0.28	0.00	0.00	0.00	0.00	1.00	
oud	0	1.00	0.06	0.23	0.00	0.00	0.00	0.00	1.00	
sedatives	0	1.00	0.01	0.11	0.00	0.00	0.00	0.00	1.00	
phtn	0	1.00	0.07	0.25	0.00	0.00	0.00	0.00	1.00	
polycy	0	1.00	0.01	0.10	0.00	0.00	0.00	0.00	1.00	
po_steroid	0	1.00	0.43	0.50	0.00	0.00	0.00	1.00	1.00	
narcan	0	1.00	0.19	0.39	0.00	0.00	0.00	0.00	1.00	
inpt_inh	0	1.00	0.41	0.49	0.00	0.00	0.00	1.00	1.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
vasodilators	0	1.00	0.01	0.08	0.00	0.00	0.00	0.00	1.00	
ip_diuretics	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
ip_abx	0	1.00	0.02	0.14	0.00	0.00	0.00	0.00	1.00	
paralytic	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
op_diuretics	0	1.00	0.37	0.48	0.00	0.00	0.00	1.00	1.00	
op_opiate	0	1.00	0.63	0.48	0.00	0.00	1.00	1.00	1.00	
op_mat	0	1.00	0.04	0.20	0.00	0.00	0.00	0.00	1.00	
op_nrt	0	1.00	0.12	0.33	0.00	0.00	0.00	0.00	1.00	
copd_med	0	1.00	0.49	0.50	0.00	0.00	0.00	1.00	1.00	
muscle_relax	0	1.00	0.24	0.43	0.00	0.00	0.00	0.00	1.00	
ABG_rfs	0	1.00	0.25	0.43	0.00	0.00	0.00	0.00	1.00	
is_amb	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
age_by_ten	0	1.00	5.88	1.76	1.80	4.60	6.10	7.20	9.00	
age_decade	0	1.00	4.34	1.75	1.00	3.00	5.00	6.00	7.00	
death_date	2144	0.14	-88.69	838.11	-10049.00	-12.00	4.00	54.50	348.00	
died	0	1.00	0.14	0.35	0.00	0.00	0.00	0.00	1.00	
months_death_or_cens	2157	0.13	1.74	2.44	0.00	0.00	1.00	2.00	12.00	
curr_weight	779	0.69	198.63	61.73	80.00	152.20	190.00	237.18	432.00	
curr_weight_date	779	0.69	-0.11	6.81	-70.00	0.00	0.00	0.00	59.00	
prev_weight_date	2262	0.09	-7.04	12.72	-70.00	-6.00	-2.00	-1.00	-1.00	
curr_height_date	678	0.73	-0.38	5.98	-70.00	0.00	0.00	0.00	49.00	
prev_height_date	2169	0.13	-5.18	9.57	-70.00	-4.00	-2.00	-1.00	-1.00	
curr_bmi_date	1409	0.43	-0.49	6.43	-67.00	0.00	0.00	0.00	59.00	
prev_bmi_date	2337	0.06	-6.38	11.30	-67.00	-5.00	-2.00	-1.00	-1.00	
height	677	0.73	66.92	4.26	56.00	64.00	67.00	70.00	81.00	
height_date	677	0.73	-0.41	6.08	-70.00	0.00	0.00	0.00	49.00	
weight	777	0.69	198.68	61.74	80.00	152.25	190.00	237.00	432.00	
weight_date	777	0.69	-0.14	6.90	-70.00	0.00	0.00	0.00	59.00	
calc_bmi	1027	0.59	31.61	9.24	13.60	24.70	30.00	37.90	75.40	
calc_bmi_date	1027	0.59	-0.50	6.50	-70.00	0.00	0.00	0.00	49.00	
working_bmi	1408	0.43	31.22	8.58	13.60	24.40	29.70	38.85	49.80	
working_bmi_date	1408	0.43	-0.53	6.58	-67.00	0.00	0.00	0.00	59.00	
bmi	852	0.66	31.56	9.21	13.60	24.70	29.90	37.90	75.40	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
bmi_date	852	0.66	-0.44	6.63	-70.00	0.00	0.00	0.00	59.00	
bmi_int	852	0.66	31.61	9.21	14.00	25.00	30.00	38.00	75.00	
bmi_by_five	852	0.66	6.31	1.84	2.72	4.94	5.98	7.58	15.08	
rr	1355	0.46	18.03	4.89	3.00	16.00	18.00	20.00	55.00	
rr_date	1350	0.46	-0.95	4.34	-70.00	0.00	0.00	0.00	20.00	
temp_new	1198	0.52	97.84	2.05	47.00	97.38	98.00	98.38	105.80	
new_temp_date	1198	0.52	-0.95	4.88	-70.00	0.00	0.00	0.00	27.00	
sbp	767	0.69	128.03	26.18	34.00	112.00	127.00	144.00	268.00	
sbp_date	765	0.69	0.20	7.12	-70.00	0.00	0.00	0.00	64.00	
dbp	778	0.69	72.28	15.95	21.00	62.00	72.00	83.00	150.00	
dbp_date	773	0.69	0.19	7.11	-70.00	0.00	0.00	0.00	64.00	
spo2_date	1770	0.29	-0.46	2.66	-25.00	0.00	0.00	0.00	20.00	
hr_date	892	0.64	-0.71	4.85	-70.00	0.00	0.00	0.00	40.00	
abg_ph_date	1975	0.21	0.09	1.78	-7.00	0.00	0.00	0.00	29.00	
vbg_ph_date	1813	0.27	0.09	2.34	-16.00	0.00	0.00	0.00	19.00	
abg_hco3_date	1952	0.22	0.07	1.56	-7.00	0.00	0.00	0.00	29.00	
abg_hco3_int	1964	0.21	23.38	6.09	4.00	20.00	23.00	26.00	52.00	
vbg_hco3_date	1837	0.26	0.08	2.56	-17.00	0.00	0.00	0.00	22.00	
sodium_date	137	0.95	-0.56	2.54	-31.00	0.00	0.00	0.00	40.00	
serum_k_date	189	0.92	-0.56	2.56	-31.00	0.00	0.00	0.00	40.00	
hgb_date	221	0.91	-0.51	2.54	-31.00	0.00	0.00	0.00	40.00	
wbc_date	376	0.85	-0.53	2.48	-31.00	0.00	0.00	0.00	40.00	
plt_date	176	0.93	-0.54	2.49	-31.00	0.00	0.00	0.00	40.00	
serum_hco3_date	144	0.94	-0.55	2.53	-31.00	0.00	0.00	0.00	40.00	
any_bicarb	123	0.95	23.95	4.82	3.05	21.05	24.00	26.50	56.00	
int_bicarb	123	0.95	23.99	4.82	3.00	21.00	24.00	27.00	56.00	
hco3_cat	147	0.94	0.88	0.98	0.00	0.00	1.00	1.00	3.00	
serum_cl_date	154	0.94	-0.55	2.54	-31.00	0.00	0.00	0.00	40.00	
serum_cr_date	200	0.92	-0.53	2.55	-27.00	0.00	0.00	0.00	40.00	
serum_lac_date	1484	0.40	-0.28	2.63	-31.00	0.00	0.00	0.00	20.00	
vbg_co2_date	1776	0.29	0.00	2.41	-17.00	0.00	0.00	0.00	19.00	
pco2_nos_date	2221	0.11	0.37	2.97	-16.00	0.00	0.00	0.00	27.00	
highest_vbg_co2_date	1776	0.29	0.47	2.77	-13.00	0.00	0.00	0.00	24.00	
highest_pco2_nos_date	2221	0.11	1.43	5.40	-16.00	0.00	0.00	0.00	44.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
paco2	1630	0.35	42.74	18.69	13.40	34.00	39.00	46.50	235.00	
paco2_date_1	1930	0.23	0.06	1.44	-7.00	0.00	0.00	0.00	29.00	
paco2_date_2	2254	0.10	0.26	2.57	-16.00	0.00	0.00	0.00	26.00	
paco2_date_3	2375	0.05	-0.05	0.56	-6.00	0.00	0.00	0.00	0.00	
paco2_date	1629	0.35	0.10	1.74	-16.00	0.00	0.00	0.00	29.00	
paco2_int	1630	0.35	42.75	18.69	13.00	34.00	39.00	47.00	235.00	
highest_paco2	1629	0.35	49.89	25.97	15.40	37.05	44.00	52.25	247.00	
paco2_date_highest_1	1930	0.23	1.37	4.42	-5.00	0.00	0.00	0.00	54.00	
paco2_date_highest_2	2254	0.10	0.87	4.40	-16.00	0.00	0.00	0.00	44.00	
paco2_date_highest_3	2375	0.05	0.83	2.70	-6.00	0.00	0.00	0.00	17.00	
paco2_date_highest	1629	0.35	1.14	4.28	-16.00	0.00	0.00	0.00	54.00	
temp_cor_oxygen_date	2424	0.03	0.10	3.10	-17.00	0.00	0.00	0.00	15.00	
temp_cor_vbg_ph_date	2437	0.02	-0.39	2.25	-11.00	0.00	0.00	0.00	4.00	
vbg_po2_date	1858	0.25	0.13	2.45	-17.00	0.00	0.00	0.00	19.00	
vbg_lactate_date	2396	0.04	0.86	6.26	-17.00	0.00	0.00	0.50	47.00	
vbg_hco3_calc_date	2414	0.03	-0.06	2.09	-12.00	0.00	0.00	0.00	9.00	
abg_po2_date	1954	0.22	0.09	1.65	-7.00	0.00	0.00	0.00	29.00	
abg_po2_temp_cor_date	2373	0.05	-0.03	0.58	-6.00	0.00	0.00	0.00	1.00	
abg_ph_temp_cor_date	2389	0.04	-0.03	0.62	-6.00	0.00	0.00	0.00	1.00	
abg_lactate_date	2405	0.03	0.02	1.32	-5.00	0.00	0.00	0.00	8.00	
ph_blood_date	2249	0.10	0.51	5.40	-16.00	0.00	0.00	0.00	74.00	
po2_blood_date	2260	0.09	0.26	2.60	-16.00	0.00	0.00	0.00	26.00	
vbg_temp_date	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
abg_temp_date	2483	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
vbg_o2sat_date	2185	0.12	0.22	2.50	-13.00	0.00	0.00	0.00	19.00	
abg_o2sat_date	1931	0.22	0.12	1.58	-7.00	0.00	0.00	0.00	29.00	
sao2_blood_date	2344	0.06	0.88	6.78	-31.00	0.00	0.00	1.00	46.00	
paco2_flag	1630	0.35	0.30	0.46	0.00	0.00	0.00	1.00	1.00	
highest_paco2_flag	1629	0.35	0.48	0.50	0.00	0.00	0.00	1.00	1.00	
paco2_52_flag	1630	0.35	0.15	0.36	0.00	0.00	0.00	0.00	1.00	
vbg_co2_flag	1778	0.29	0.28	0.45	0.00	0.00	0.00	1.00	1.00	
highest_vbg_co2_flag	1777	0.29	0.36	0.48	0.00	0.00	0.00	1.00	1.00	
miss_paco2_flag	0	1.00	0.10	0.30	0.00	0.00	0.00	0.00	1.00	
miss_vbg_co2_flag	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
miss_vbg_or_abg_co2_flag	0	1.00	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
hco3_flag	147	0.94	0.25	0.43	0.00	0.00	0.00	0.00	1.00	
not_paco2_flag	1630	0.35	0.70	0.46	0.00	0.00	1.00	1.00	1.00	
not_hco3_flag	147	0.94	0.75	0.43	0.00	1.00	1.00	1.00	1.00	
k_cat	189	0.92	1.20	0.75	0.00	1.00	1.00	2.00	3.00	
acidemia	1154	0.54	0.32	0.47	0.00	0.00	0.00	1.00	1.00	
abg_sbe	1990	0.20	-2.10	6.81	-28.03	-5.63	-1.90	1.36	26.35	
vbg_sbe	1857	0.25	-0.93	6.89	-28.18	-4.32	-0.70	2.55	29.45	
cw_simple_acute_resp_acid	2109	0.15	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
paco2_52_comp_flag	1630	0.35	0.04	0.19	0.00	0.00	0.00	0.00	1.00	
po_steroid_date	1410	0.43	1.20	6.94	-82.00	0.00	0.00	1.00	76.00	
narcan_date	2012	0.19	1.87	8.31	-109.00	0.00	0.00	3.00	70.00	
inpt_inh_date	1476	0.41	0.29	10.44	-233.00	0.00	0.00	1.00	26.00	
vasodilators_date	2476	0.01	2.13	5.85	-4.00	0.00	0.00	1.50	22.00	
ip_diuretics_date	2469	0.01	2.36	2.80	0.00	0.00	1.00	4.75	9.00	
ip_abx_date	2444	0.02	1.28	4.00	-12.00	0.00	0.00	1.00	17.00	
paralytic_date	2471	0.01	1.25	4.49	-7.00	0.00	0.00	0.25	13.00	
inpt_inh_0	0	1.00	0.21	0.41	0.00	0.00	0.00	0.00	1.00	
ip_abx_0	0	1.00	0.01	0.10	0.00	0.00	0.00	0.00	1.00	
ip_diuretics_0	0	1.00	0.00	0.06	0.00	0.00	0.00	0.00	1.00	
narcan_0	0	1.00	0.09	0.29	0.00	0.00	0.00	0.00	1.00	
paralytic_0	0	1.00	0.01	0.07	0.00	0.00	0.00	0.00	1.00	
po_steroid_0	0	1.00	0.24	0.43	0.00	0.00	0.00	0.00	1.00	
vasodilators_0	0	1.00	0.00	0.05	0.00	0.00	0.00	0.00	1.00	
op_diuretics_first_date	1568	0.37	-1601.18	1421.49	-8424.00	-2371.00	-1270.00	-461.50	0.00	
op_diuretics_last_date	1569	0.37	-1592.15	1418.32	-8424.00	-2341.50	-1259.50	-460.25	0.00	
op_diuretics_365d	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
op_opiate_first_date	919	0.63	-1671.39	1507.19	-8673.00	-2573.50	-1356.00	-368.50	0.00	
op_opiate_last_date	928	0.63	-1678.57	1505.63	-8673.00	-2572.50	-1373.00	-375.50	0.00	
op_opiate_365d	0	1.00	0.16	0.36	0.00	0.00	0.00	0.00	1.00	
op_mat_first_date	2384	0.04	-1310.29	1292.01	-6507.00	-1803.00	-930.00	-352.00	0.00	
op_mat_last_date	2384	0.04	-1307.50	1293.30	-6507.00	-1799.00	-901.00	-349.50	0.00	
op_mat_365d	0	1.00	0.01	0.11	0.00	0.00	0.00	0.00	1.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
op_nrt_first_date	2192	0.12	-1616.59	1443.99	-8385.00	-2462.00	-1298.00	-472.50	0.00	
op_nrt_last_date	2193	0.12	-1617.68	1442.21	-8385.00	-2467.50	-1303.50	-478.00	0.00	
op_nrt_365d	0	1.00	0.03	0.16	0.00	0.00	0.00	0.00	1.00	
copd_med_first_date	1260	0.49	-1604.86	1403.48	-9520.00	-2412.50	-1285.00	-480.50	0.00	
copd_med_last_date	1263	0.49	-1602.27	1400.42	-9520.00	-2409.00	-1287.50	-475.25	0.00	
copd_med_365d	0	1.00	0.11	0.31	0.00	0.00	0.00	0.00	1.00	
muscle_relax_first_date	1897	0.24	-1552.14	1341.47	-8325.00	-2338.25	-1263.00	-479.25	0.00	
muscle_relax_last_date	1898	0.24	-1550.32	1342.05	-8325.00	-2340.00	-1254.00	-477.00	0.00	
muscle_relax_365d	0	1.00	0.05	0.22	0.00	0.00	0.00	0.00	1.00	
tte_proc_first_date	2180	0.12	0.47	3.60	-19.00	0.00	1.00	2.00	18.00	
tte_proc_last_date	2180	0.12	2.04	6.74	-19.00	0.00	1.00	2.00	61.00	
vent_proc	0	1.00	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
niv_proc	0	1.00	0.06	0.24	0.00	0.00	0.00	0.00	1.00	
imv_proc	0	1.00	0.11	0.32	0.00	0.00	0.00	0.00	1.00	
cpap_first_date	2327	0.07	1.02	3.01	-7.00	0.00	0.00	1.00	16.00	
cpap_last_date	2327	0.07	4.16	6.06	-3.00	0.00	2.00	5.00	40.00	
niv_proc_first_date	2343	0.06	1.22	4.22	-11.00	0.00	0.00	1.00	28.00	
niv_proc_last_date	2343	0.06	1.36	4.31	-11.00	0.00	0.00	1.00	28.00	
imv_proc_first_date	2205	0.11	0.63	3.64	-22.00	0.00	0.00	0.00	27.00	
imv_proc_last_date	2205	0.11	3.22	7.08	-22.00	0.00	0.50	4.00	58.00	
vent_proc_first_date	2089	0.16	0.47	3.07	-22.00	0.00	0.00	0.00	27.00	
vent_proc_last_date	2089	0.16	2.61	6.42	-22.00	0.00	0.00	3.00	58.00	
aero_first_date	2189	0.12	0.36	3.95	-30.00	0.00	0.00	1.00	41.00	
aero_last_date	2189	0.12	4.99	8.71	-18.00	0.00	2.50	6.00	74.00	
inh_teaching_first_date	2429	0.02	1.95	3.30	-2.00	0.00	1.00	2.00	17.00	
inh_teaching_last_date	2429	0.02	4.27	5.54	-2.00	1.00	2.00	5.00	24.00	
cxr1v_first_date	1836	0.26	-0.18	3.11	-31.00	0.00	0.00	0.00	26.00	
cxr1v_last_date	1836	0.26	3.19	8.07	-11.00	0.00	0.00	3.00	86.00	
cxr2v_first_date	2416	0.03	2.32	9.46	-17.00	0.00	0.00	2.50	59.00	
cxr2v_last_date	2416	0.03	3.45	9.36	-13.00	0.00	1.00	5.00	59.00	
ctcnoncon_first_date	2416	0.03	2.32	9.46	-17.00	0.00	0.00	2.50	59.00	
ctcnoncon_last_date	2416	0.03	3.45	9.36	-13.00	0.00	1.00	5.00	59.00	
ctccon_first_date	2377	0.05	0.38	3.83	-14.00	0.00	0.00	0.00	20.00	
ctccon_last_date	2377	0.05	1.09	4.25	-9.00	0.00	0.00	0.00	20.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ctabdpelv_first_date	2356	0.05	2.17	9.45	-12.00	0.00	0.00	1.00	87.00	
ctabdpelv_last_date	2356	0.05	2.79	10.02	-12.00	0.00	0.00	1.00	87.00	
meas_venous_o2_proc_first_date	2447	0.02	-0.09	1.63	-9.00	0.00	0.00	0.00	4.00	
meas_venous_o2_proc_last_date	2447	0.02	1.09	3.46	-2.00	0.00	0.00	1.00	22.00	
meas_art_gas_proc_first_date	2469	0.01	0.14	0.35	0.00	0.00	0.00	0.00	1.00	
meas_art_gas_proc_last_date	2469	0.01	1.86	3.18	0.00	0.00	1.00	2.00	13.00	
blood_cx_proc_first_date	2160	0.13	-0.27	3.94	-31.00	0.00	0.00	0.00	23.00	
blood_cx_proc_last_date	2160	0.13	1.61	6.57	-19.00	0.00	0.00	1.00	74.00	
art_punct_proc_first_date	2350	0.06	0.72	3.33	-5.00	0.00	0.00	0.00	29.00	
art_punct_proc_last_date	2350	0.06	2.76	6.92	-5.00	0.00	0.00	2.00	44.00	
cc_time_first_date	1944	0.22	0.00	3.12	-31.00	0.00	0.00	0.00	25.00	
cc_time_last_date	1944	0.22	3.72	8.09	-13.00	0.00	1.00	4.00	63.00	
aero_0	0	1.00	0.06	0.25	0.00	0.00	0.00	0.00	1.00	
blood_cx_proc_0	0	1.00	0.09	0.29	0.00	0.00	0.00	0.00	1.00	
cc_time_0	0	1.00	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
cpap_0	0	1.00	0.04	0.19	0.00	0.00	0.00	0.00	1.00	
ctabdpelv_0	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
ctccon_0	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
ctcnoncon_0	0	1.00	0.01	0.11	0.00	0.00	0.00	0.00	1.00	
cxr1v_0	0	1.00	0.20	0.40	0.00	0.00	0.00	0.00	1.00	
cxr2v_0	0	1.00	0.01	0.11	0.00	0.00	0.00	0.00	1.00	
imv_proc_0	0	1.00	0.09	0.29	0.00	0.00	0.00	0.00	1.00	
inh_teaching_0	0	1.00	0.01	0.08	0.00	0.00	0.00	0.00	1.00	
niv_proc_0	0	1.00	0.04	0.19	0.00	0.00	0.00	0.00	1.00	
tte_proc_0	0	1.00	0.04	0.20	0.00	0.00	0.00	0.00	1.00	
vent_proc_0	0	1.00	0.12	0.33	0.00	0.00	0.00	0.00	1.00	
aero_dur	2189	0.12	4.63	8.09	0.00	0.00	2.00	6.00	74.00	
blood_cx_proc_dur	2160	0.13	1.87	6.02	0.00	0.00	0.00	0.50	74.00	
cpap_dur	2327	0.07	3.13	5.10	0.00	0.00	1.00	4.00	29.00	
ctabdpelv_dur	2356	0.05	0.61	2.98	0.00	0.00	0.00	0.00	21.00	
ctccon_dur	2377	0.05	0.71	3.69	0.00	0.00	0.00	0.00	32.00	
ctcnoncon_dur	2416	0.03	1.13	3.48	0.00	0.00	0.00	0.00	18.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cxr1v_dur	1836	0.26	3.37	8.21	0.00	0.00	0.00	3.00	81.00	
cxr2v_dur	2416	0.03	1.13	3.48	0.00	0.00	0.00	0.00	18.00	
imv_proc_dur	2205	0.11	2.59	6.29	0.00	0.00	0.00	2.00	58.00	
inh_teaching_dur	2429	0.02	2.32	4.73	0.00	0.00	0.00	2.00	24.00	
niv_proc_dur	2343	0.06	0.14	1.08	0.00	0.00	0.00	0.00	11.00	
tte_proc_dur	2180	0.12	1.57	5.69	0.00	0.00	0.00	0.00	55.00	
hypercap_resp_failure	0	1.00	0.06	0.23	0.00	0.00	0.00	0.00	1.00	
j9612_date	2475	0.01	0.75	4.77	-7.00	-0.50	0.00	0.00	12.00	
j9612_pcpl	2475	0.01	0.12	0.34	0.00	0.00	0.00	0.00	1.00	
j9612_adm	2475	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9612_vr	2475	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9622_date	2448	0.02	0.21	3.70	-7.00	0.00	0.00	0.00	21.00	
j9622_pcpl	2448	0.02	0.12	0.32	0.00	0.00	0.00	0.00	1.00	
j9622_adm	2448	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9622_vr	2448	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9602_date	2412	0.03	1.13	7.25	-24.00	0.00	0.00	0.00	43.00	
j9602_pcpl	2414	0.03	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
j9602_adm	2412	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9602_vr	2412	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9692_date	2474	0.01	1.94	7.25	-7.00	-1.00	0.00	3.00	24.00	
j9692_pcpl	2474	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9692_adm	2474	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
j9692_vr	2474	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ohs_code_date	2470	0.01	-0.38	1.99	-6.00	0.00	0.00	0.00	5.00	
e662_pcpl	2470	0.01	0.10	0.30	0.00	0.00	0.00	0.00	1.00	
e662_adm	2470	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
e662_vr	2470	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
hypercap_resp_failure_date	2352	0.06	0.78	6.23	-24.00	0.00	0.00	0.00	43.00	
other_resp_failure	0	1.00	0.26	0.44	0.00	0.00	0.00	1.00	1.00	
j9600_date	2402	0.04	1.74	6.57	-22.00	0.00	0.00	1.00	30.00	
j9601_date	2026	0.19	0.12	5.28	-24.00	0.00	0.00	0.00	59.00	
j961_date	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
j9610_date	2469	0.01	3.05	9.19	-4.00	0.00	0.00	0.00	39.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
j9611_date	2444	0.02	0.13	2.57	-7.00	0.00	0.00	0.00	14.00	
j962_date	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
j9620_date	2481	0.00	1.00	1.76	-2.00	0.00	0.50	2.00	4.00	
j9621_date	2389	0.04	0.29	5.62	-9.00	0.00	0.00	0.00	46.00	
j9690_date	2428	0.03	3.71	11.54	-22.00	0.00	0.00	1.00	51.00	
j9691_date	2458	0.01	4.00	10.01	-7.00	0.00	0.00	1.00	43.00	
other_resp_failure_date	1854	0.26	0.17	4.78	-24.00	0.00	0.00	0.00	59.00	
sepsis_dx_date	2211	0.11	-0.51	3.97	-31.00	0.00	0.00	0.00	18.00	
stupor_dx_date	2426	0.03	1.09	6.00	-10.00	0.00	0.00	0.00	36.00	
cog_signs_dx_date	2225	0.11	0.88	6.34	-22.00	0.00	0.00	0.00	46.00	
mal_fat_dx_date	2232	0.10	0.40	4.99	-22.00	0.00	0.00	0.00	28.00	
resp_acid_dx_date	2478	0.01	-0.15	1.28	-3.00	0.00	0.00	0.00	3.00	
sleep_hypovent_dx_date	2488	0.00	3.00	3.61	0.00	1.00	2.00	4.50	7.00	
cchs_dx_date	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
other_sleep_hypovent_dx_date	2488	0.00	4.33	7.51	0.00	0.00	0.00	6.50	13.00	
acidosis_unspec_date	2410	0.03	0.20	5.36	-11.00	0.00	0.00	0.00	43.00	
headache_dx_date	2419	0.03	-0.44	2.71	-18.00	0.00	0.00	0.00	5.00	
dysp_dx	0	1.00	0.14	0.35	0.00	0.00	0.00	0.00	1.00	
dysp_dx_date	2145	0.14	0.39	4.25	-13.00	0.00	0.00	0.00	51.00	
symp_obs	0	1.00	0.01	0.08	0.00	0.00	0.00	0.00	1.00	
symp_obs_date	2476	0.01	0.80	2.96	-1.00	0.00	0.00	0.00	11.00	
abn_br_dx	0	1.00	0.00	0.03	0.00	0.00	0.00	0.00	1.00	
abn_br_dx_date	2489	0.00	2.00	2.83	0.00	1.00	2.00	3.00	4.00	
resp_abnormality	0	1.00	0.02	0.13	0.00	0.00	0.00	0.00	1.00	
r0689_date	2453	0.02	0.00	6.47	-22.00	-1.75	0.00	0.00	26.00	
resp_abnormality_date	2447	0.02	0.11	6.10	-22.00	-1.25	0.00	0.00	26.00	
other_abn_of_br_date	2453	0.02	0.00	6.47	-22.00	-1.75	0.00	0.00	26.00	
fast_br	0	1.00	0.00	0.04	0.00	0.00	0.00	0.00	1.00	
fast_br_date	2486	0.00	-1.80	5.22	-11.00	0.00	0.00	0.00	2.00	
pulm_edema_dx	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
pulm_edema_dx_date	2404	0.03	-1.14	21.20	-181.00	0.00	0.00	0.00	51.00	
pna_dx	0	1.00	0.14	0.35	0.00	0.00	0.00	0.00	1.00	
pna_dx_date	2132	0.14	-0.12	3.65	-16.00	0.00	0.00	0.00	43.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
acute_chf	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
acute_old	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
acute_old_date	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
resp_dep_compl	0	1.00	0.03	0.18	0.00	0.00	0.00	0.00	1.00	
resp_dep_compl_date	2404	0.03	-0.64	3.15	-22.00	0.00	0.00	0.00	4.00	
acute_nmd_date	2488	0.00	-0.67	1.15	-2.00	-1.00	0.00	0.00	0.00	
osa_first_date	2070	0.17	-1388.39	1048.44	-6106.00	-2013.00	-1229.00	-536.00	0.00	
osa_last_date	2071	0.17	-1387.46	1048.76	-6106.00	-2013.50	-1228.00	-547.25	0.00	
asthma_first_date	2191	0.12	-1475.97	1279.04	-12433.00	-2137.00	-1326.50	-560.50	0.00	
asthma_last_date	2192	0.12	-1461.98	1226.16	-12433.00	-2137.00	-1333.00	-558.00	0.00	
copd_first_date	2026	0.19	-1323.12	1164.76	-9568.00	-1936.00	-1193.00	-494.00	0.00	
copd_last_date	2026	0.19	-1098.76	1025.13	-9568.00	-1611.00	-934.00	-303.00	0.00	
chf_first_date	2005	0.20	-939.30	871.73	-5954.00	-1475.50	-717.50	-225.75	0.00	
chf_last_date	2005	0.20	-934.13	871.14	-5954.00	-1464.75	-717.50	-220.50	0.00	
stroke_first_date	2316	0.07	-966.97	1143.58	-7453.00	-1494.00	-595.00	-147.50	0.00	
stroke_last_date	2316	0.07	-963.98	1142.30	-7453.00	-1485.00	-595.00	-147.50	0.00	
ckd_first_date	2009	0.19	-1191.63	977.79	-5617.00	-1878.00	-1041.00	-339.25	0.00	
ckd_last_date	2012	0.19	-1189.95	975.59	-5617.00	-1871.00	-1039.00	-337.50	0.00	
ctd	0	1.00	0.05	0.22	0.00	0.00	0.00	0.00	1.00	
ctd_first_date	2367	0.05	-1294.99	1046.70	-6297.00	-1864.50	-1174.50	-482.25	0.00	
ctd_last_date	2367	0.05	-1179.76	955.48	-6297.00	-1752.00	-1102.00	-378.25	0.00	
dem	0	1.00	0.06	0.24	0.00	0.00	0.00	0.00	1.00	
dem_first_date	2339	0.06	-980.14	1172.06	-8434.00	-1352.50	-647.00	-204.75	0.00	
dem_last_date	2340	0.06	-823.91	929.02	-5826.00	-1140.50	-539.00	-161.50	0.00	
dm	0	1.00	0.30	0.46	0.00	0.00	0.00	1.00	1.00	
dm_first_date	1755	0.30	-1529.07	1257.93	-10111.00	-2241.50	-1483.00	-505.50	0.00	
dm_last_date	1756	0.30	-1259.91	985.75	-5881.00	-1951.00	-1224.00	-362.50	0.00	
pvd_first_date	2268	0.09	-961.51	1017.73	-8011.00	-1538.50	-659.00	-100.00	0.00	
pvd_last_date	2268	0.09	-961.46	1017.76	-8011.00	-1538.50	-659.00	-100.00	0.00	
oud_first_date	2353	0.06	-1206.01	869.81	-4689.00	-1783.00	-1171.50	-443.25	0.00	
oud_last_date	2354	0.05	-1212.72	866.23	-4689.00	-1791.00	-1179.00	-453.00	0.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sedatives_first_date	2462	0.01	-987.69	818.61	-2468.00	-1724.00	-921.00	-288.00	0.00	
sedatives_last_date	2462	0.01	-987.21	819.09	-2468.00	-1724.00	-921.00	-288.00	0.00	
cfdo_first_date	2486	0.00	-1991.80	1721.90	-4600.00	-2261.00	-2109.00	-988.00	-1.00	
cfdo_last_date	2486	0.00	-1991.80	1721.90	-4600.00	-2261.00	-2109.00	-988.00	-1.00	
phtn_first_date	2316	0.07	-938.86	728.69	-2720.00	-1561.00	-777.00	-299.50	0.00	
phtn_last_date	2317	0.07	-937.51	730.63	-2720.00	-1569.00	-762.00	-299.25	0.00	
polycy_first_date	2467	0.01	-1105.83	810.15	-2720.00	-1628.25	-1131.00	-343.25	0.00	
polycy_last_date	2467	0.01	-1105.83	810.15	-2720.00	-1628.25	-1131.00	-343.25	0.00	
nmd	0	1.00	0.04	0.20	0.00	0.00	0.00	0.00	1.00	
nmd_first_date	2388	0.04	-1186.89	1687.98	-13004.00	-1625.50	-717.00	-146.00	0.00	
nmd_last_date	2388	0.04	-1069.35	1576.17	-13004.00	-1519.00	-643.00	-114.50	0.00	
nic	0	1.00	0.25	0.43	0.00	0.00	0.00	1.00	1.00	
nic_first_date	1857	0.25	-1363.35	1091.37	-8591.00	-1956.25	-1298.00	-601.75	0.00	
nic_last_date	1857	0.25	-1361.15	1092.65	-8591.00	-1956.25	-1298.00	-593.50	0.00	
ovs	0	1.00	0.06	0.24	0.00	0.00	0.00	0.00	1.00	
ats_ohs_flag	2223	0.11	0.41	0.49	0.00	0.00	0.00	1.00	1.00	
pos_ohs_flag	0	1.00	0.41	0.49	0.00	0.00	0.00	1.00	1.00	
ats_copd_flag	2235	0.10	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
guidelines	2235	0.10	0.54	0.64	0.00	0.00	0.00	1.00	2.00	
OBESITY_rfs	0	1.00	0.14	0.35	0.00	0.00	0.00	0.00	1.00	
_merge_amb_obes	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
PREDISPOSITION_rfs	0	1.00	0.54	0.50	0.00	0.00	1.00	1.00	1.00	
_merge_amb_predisp	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
RESPFAIL_rfs	0	1.00	0.22	0.42	0.00	0.00	0.00	0.00	1.00	
_merge_amb_respfail	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
VBG_rfs	0	1.00	0.33	0.47	0.00	0.00	0.00	1.00	1.00	
_merge_amb_vbg	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
VENTSUPPORT_rfs	0	1.00	0.18	0.38	0.00	0.00	0.00	0.00	1.00	
_merge_amb_ventsupp	2491	0.00	NaN	NA	NA	NA	NA	NA	NA	
is_emer	0	1.00	0.35	0.48	0.00	0.00	0.00	1.00	1.00	
_merge_emer_obes	2257	0.09	1.75	0.67	1.00	1.00	2.00	2.00	5.00	
_merge_emer_predisp	1855	0.26	2.15	1.17	1.00	2.00	2.00	2.00	5.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
_merge_emer_respfail	1831	0.26	1.36	1.09	1.00	1.00	1.00	1.00	5.00	
_merge_emer_vbg	1648	0.34	1.64	1.23	1.00	1.00	1.00	2.00	5.00	
_merge_emer_ventsupp	1629	0.35	1.25	0.94	1.00	1.00	1.00	1.00	5.00	
_merge_emer	1629	0.35	2.00	0.00	2.00	2.00	2.00	2.00	2.00	
is_inp	862	0.65	1.00	0.00	1.00	1.00	1.00	1.00	1.00	
_merge_inpat_obes	1810	0.27	1.44	0.96	1.00	1.00	1.00	2.00	5.00	
_merge_inpat_predisp	1255	0.50	2.42	1.52	1.00	1.00	2.00	2.00	5.00	
_merge_inpat_respfail	1113	0.55	2.07	1.68	1.00	1.00	1.00	2.00	5.00	
_merge_inpat_vbg	928	0.63	2.06	1.66	1.00	1.00	1.00	2.00	5.00	
_merge_inpat_ventsupp	862	0.65	1.81	1.57	1.00	1.00	1.00	1.00	5.00	
_merge_inpat	0	1.00	1.65	0.48	1.00	1.00	2.00	2.00	2.00	
patient_id	0	1.00	361588.69	213546.39	776.00	170503.00	363238.00	535806.50	748967.00	
rfsgroup	0	1.00	3.38	1.65	1.00	2.00	3.00	4.00	7.00	
first_encounter	0	1.00	0.87	0.34	0.00	1.00	1.00	1.00	1.00	
has_abg	0	1.00	0.35	0.48	0.00	0.00	0.00	1.00	1.00	
has_vbg	0	1.00	0.29	0.45	0.00	0.00	0.00	1.00	1.00	
max_hco3_or_its_met_alk	1630	0.35	29.91	26.70	-12.00	17.43	24.57	35.29	304.57	
prim_met_alk	2235	0.10	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
max_hco3_or_its_comb_met_alk	1630	0.35	27.09	7.48	15.36	23.60	25.60	28.60	104.00	
combo_met_alk	2235	0.10	0.13	0.34	0.00	0.00	0.00	0.00	1.00	
paco2_50_flag	1630	0.35	0.19	0.39	0.00	0.00	0.00	0.00	1.00	
hypercap_dx_adm_flag	2352	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
hypercap_dx_pcpl_flag	2354	0.05	0.09	0.28	0.00	0.00	0.00	0.00	1.00	
hypercap_dx_vr_flag	2352	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
hypercap_on_abg	0	1.00	0.10	0.30	0.00	0.00	0.00	0.00	1.00	
hypercap_on_vbg	0	1.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00	
has_both_abg_vbg	0	1.00	0.11	0.31	0.00	0.00	0.00	0.00	1.00	
has_neither_abg_vbg	0	1.00	0.48	0.50	0.00	0.00	0.00	1.00	1.00	
vbg_or_abg_co2_flag	0	1.00	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
dx_hypercap_on_abg	2352	0.06	0.46	0.50	0.00	0.00	0.00	1.00	1.00	
sugg_hypercap_dx_on_vbg	2352	0.06	0.35	0.48	0.00	0.00	0.00	1.00	1.00	
dx_hypercap_on_vbg	2352	0.06	0.17	0.38	0.00	0.00	0.00	0.00	1.00	
vbg_o2sat_calc	0	1.00	65.18	10.46	12.50	65.08	65.08	65.08	99.10	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
abg_o2sat_calc	0	1.00	90.83	8.03	14.00	91.05	91.05	91.05	99.90	
corr_vbg_co2	1778	0.29	40.54	13.14	7.45	32.62	38.97	44.92	129.62	
corr_vbg_co2_flag	1778	0.29	0.25	0.43	0.00	0.00	0.00	0.00	1.00	
corr_hypercap_on_vbg	0	1.00	0.07	0.26	0.00	0.00	0.00	0.00	1.00	
race_ethnicity	0	1.00	1.17	2.00	0.00	0.00	0.00	1.00	6.00	
has_vbg_and_cat	0	1.00	0.48	0.88	0.00	0.00	0.00	1.00	3.00	
has_bmi_and_cat	0	1.00	2.44	2.16	0.00	0.00	2.00	4.00	6.00	
has_weight_and_cat	0	1.00	2.35	1.87	0.00	0.00	3.00	4.00	6.00	
has_height_and_cat	0	1.00	2.25	1.66	0.00	0.00	2.00	4.00	6.00	
has_hr_and_cat	0	1.00	1.30	1.03	0.00	0.00	2.00	2.00	3.00	
has_sbp_and_cat	0	1.00	1.54	1.13	0.00	0.00	2.00	2.00	3.00	
has_rr_and_cat	0	1.00	1.03	1.18	0.00	0.00	0.00	2.00	3.00	
has_temp_and_cat	0	1.00	0.95	0.97	0.00	0.00	1.00	2.00	3.00	
has_spo2_and_cat	0	1.00	0.53	0.86	0.00	0.00	0.00	1.00	2.00	
has_cl_and_cat	0	1.00	1.92	0.74	0.00	2.00	2.00	2.00	3.00	
has_k_and_cat	0	1.00	1.99	0.93	0.00	2.00	2.00	3.00	4.00	
has_hco3_and_cat	0	1.00	2.32	1.23	0.00	1.00	2.00	3.00	5.00	
has_lactate_and_cat	0	1.00	0.52	0.70	0.00	0.00	0.00	1.00	2.00	
has_na_and_cat	0	1.00	1.62	0.61	0.00	1.00	2.00	2.00	3.00	
has_cr_and_cat	0	1.00	1.25	0.69	0.00	1.00	1.00	2.00	3.00	
has_hgb_and_cat	0	1.00	2.99	1.36	0.00	2.00	3.00	4.00	5.00	
has_wbc_and_cat	0	1.00	1.99	1.10	0.00	2.00	2.00	3.00	4.00	
has_plt_and_cat	0	1.00	2.65	0.91	0.00	3.00	3.00	3.00	4.00	
has_bnp_and_cat	0	1.00	0.32	0.77	0.00	0.00	0.00	0.00	3.00	
has_phos_and_cat	0	1.00	0.75	0.95	0.00	0.00	0.00	1.00	3.00	
has_ca_and_cat	0	1.00	1.59	0.71	0.00	1.00	2.00	2.00	3.00	
has_alb_and_cat	0	1.00	1.55	1.30	0.00	0.00	2.00	3.00	3.00	
has_tprot_and_cat	0	1.00	1.14	0.97	0.00	0.00	1.00	2.00	3.00	
encounter_type_dummy1	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
encounter_type_dummy2	0	1.00	0.35	0.48	0.00	0.00	0.00	1.00	1.00	
encounter_type_dummy3	0	1.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00	
female	0	1.00	0.50	0.50	0.00	0.00	0.00	1.00	1.00	
male	0	1.00	0.50	0.50	0.00	0.00	1.00	1.00	1.00	
white_race	0	1.00	0.62	0.48	0.00	0.00	1.00	1.00	1.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
black_race	0	1.00	0.18	0.39	0.00	0.00	0.00	0.00	1.00	
unknown_race	0	1.00	0.17	0.37	0.00	0.00	0.00	0.00	1.00	
asian_race	0	1.00	0.01	0.12	0.00	0.00	0.00	0.00	1.00	
nat_am_race	0	1.00	0.01	0.09	0.00	0.00	0.00	0.00	1.00	
nhpi_race	0	1.00	0.00	0.04	0.00	0.00	0.00	0.00	1.00	
not_hisp_eth	0	1.00	0.75	0.43	0.00	1.00	1.00	1.00	1.00	
hisp_eth	0	1.00	0.07	0.25	0.00	0.00	0.00	0.00	1.00	
unknown_eth	0	1.00	0.18	0.38	0.00	0.00	0.00	0.00	1.00	
location_dummy1	0	1.00	0.46	0.50	0.00	0.00	0.00	1.00	1.00	
location_dummy2	0	1.00	0.25	0.43	0.00	0.00	0.00	1.00	1.00	
location_dummy3	0	1.00	0.07	0.26	0.00	0.00	0.00	0.00	1.00	
location_dummy4	0	1.00	0.21	0.41	0.00	0.00	0.00	0.00	1.00	
ps	0	1.00	0.36	0.25	0.01	0.14	0.31	0.56	0.97	
tx_pr_decile	0	1.00	7.22	1.99	1.00	6.00	7.00	9.00	10.00	
sumofweights	0	1.00	1593981.00	0.00	1593981.00	1593981.00	1593981.00	1593981.00	1593981.00	
ipw	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
approx_ipw_fweight	0	1.00	12.61	16.22	6.00	7.00	9.00	12.00	410.00	
inp_ps	0	1.00	0.40	0.22	0.04	0.21	0.36	0.57	0.95	
inp_sumofweights	0	1.00	1419945.50	0.00	1419945.50	1419945.50	1419945.50	1419945.50	1419945.50	
inp_ipw	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
approx_inp_ipw_fweight	0	1.00	13.40	9.80	7.00	9.00	10.00	14.00	184.00	
vbg_ps	0	1.00	0.29	0.23	0.00	0.10	0.22	0.44	0.96	
vbg_tx_pr_decile	0	1.00	7.25	1.98	1.00	6.00	7.00	9.00	10.00	
vbg_sumofweights	0	1.00	1542183.62	0.00	1542183.62	1542183.62	1542183.62	1542183.62	1542183.62	
vbg_ipw	0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
vbg_approx_ipw_fweight	0	1.00	12.75	15.79	7.00	7.00	8.00	12.00	316.00	
died_1mo	13	0.99	0.06	0.23	0.00	0.00	0.00	0.00	1.00	
died_2mo	13	0.99	0.09	0.28	0.00	0.00	0.00	0.00	1.00	
death_time	2157	0.13	1.74	2.44	0.00	0.00	1.00	2.00	12.00	
death_60d	0	1.00	0.11	0.31	0.00	0.00	0.00	0.00	1.00	
calc_abg	2200	0.12	41.35	13.39	7.50	33.01	40.06	46.27	119.19	
hypercapnia_calc	2200	0.12	0.29	0.46	0.00	0.00	0.00	1.00	1.00	
w_abg	0	1.00	1.00	0.55	0.61	0.69	0.81	1.07	6.12	
w_vbg	0	1.00	1.00	0.59	0.63	0.69	0.78	1.03	6.68	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
has_vbg_co2_o2_sat	0	1.00	0.12	0.32	0.00	0.00	0.00	0.00	1.00	
w_vbg_calc	0	1.00	1.00	1.30	0.68	0.69	0.72	0.81	36.22	

```
# Variable-wise % missing
naniar::miss_var_summary(subset_data)
```

variable	n_miss	pct_miss
vbg_temp	2491	100
abg_temp	2491	100
vbg_temp_date	2491	100
j961_date	2491	100
j962_date	2491	100
cchs_dx_date	2491	100
_merge_amb_obes	2491	100
_merge_amb_predisp	2491	100
_merge_amb_respfail	2491	100
_merge_amb_vbg	2491	100
_merge_amb_ventsupp	2491	100
abn_br_dx_date	2489	99.9
sleep_hypovent_dx_date	2488	99.9
other_sleep_hypovent_dx_date	2488	99.9
acute_nmd_date	2488	99.9
fast_br_date	2486	99.8
cfdo_first_date	2486	99.8
cfdo_last_date	2486	99.8
abg_temp_date	2483	99.7
j9620_date	2481	99.6
resp_acid_dx_date	2478	99.5
vasodilators_date	2476	99.4
symp_obs_date	2476	99.4
j9612_date	2475	99.4
j9612_pcpl	2475	99.4

variable	n_miss	pct_miss
j9612_adm	2475	99.4
j9612_vr	2475	99.4
j9692_date	2474	99.3
j9692_pcpl	2474	99.3
j9692_adm	2474	99.3
j9692_vr	2474	99.3
paralytic_date	2471	99.2
ohs_code_date	2470	99.2
e662_pcpl	2470	99.2
e662_adm	2470	99.2
e662_vr	2470	99.2
ip_diuretics_date	2469	99.1
meas_art_gas_proc_first_date	2469	99.1
meas_art_gas_proc_last_date	2469	99.1
j9610_date	2469	99.1
polycy_first_date	2467	99.0
polycy_last_date	2467	99.0
sedatives_first_date	2462	98.8
sedatives_last_date	2462	98.8
j9691_date	2458	98.7
r0689_date	2453	98.5
other_abn_of_br_date	2453	98.5
j9622_date	2448	98.3
j9622_pcpl	2448	98.3
j9622_adm	2448	98.3
j9622_vr	2448	98.3
meas_venous_o2_proc_first_date	2447	98.2
meas_venous_o2_proc_last_date	2447	98.2
resp_abnormality_date	2447	98.2
ip_abx_date	2444	98.1
j9611_date	2444	98.1
vbg_ph_temp_cor	2437	97.8
temp_cor_vbg_ph_date	2437	97.8
temp_cor_oxygen	2435	97.8
inh_teaching_first_date	2429	97.5

variable	n_miss	pct_miss
inh_teaching_last_date	2429	97.5
inh_teaching_dur	2429	97.5
j9690_date	2428	97.5
stupor_dx_date	2426	97.4
temp_cor_oxygen_date	2424	97.3
headache_dx_date	2419	97.1
cxr2v_first_date	2416	97.0
cxr2v_last_date	2416	97.0
ctcnoncon_first_date	2416	97.0
ctcnoncon_last_date	2416	97.0
ctcnoncon_dur	2416	97.0
cxr2v_dur	2416	97.0
vbg_hco3_calc	2415	96.9
vbg_hco3_calc_date	2414	96.9
j9602_pcpl	2414	96.9
j9602_date	2412	96.8
j9602_adm	2412	96.8
j9602_vr	2412	96.8
acidosis_unspec_date	2410	96.7
abg_lactate	2405	96.5
abg_lactate_date	2405	96.5
pulm_edema_dx_date	2404	96.5
resp_dep_compl_date	2404	96.5
j9600_date	2402	96.4
vbg_lactate	2397	96.2
vbg_lactate_date	2396	96.2
abg_ph_temp_cor	2389	95.9
abg_ph_temp_cor_date	2389	95.9
j9621_date	2389	95.9
nmd_first_date	2388	95.9
nmd_last_date	2388	95.9
op_mat_first_date	2384	95.7
op_mat_last_date	2384	95.7
ctccon_first_date	2377	95.4
ctccon_last_date	2377	95.4

variable	n_miss	pct_miss
ctccon_dur	2377	95.4
value_highest_327718	2375	95.3
paco2_date_3	2375	95.3
paco2_date_highest_3	2375	95.3
abg_po2_temp_cor	2373	95.3
abg_po2_temp_cor_date	2373	95.3
sao2_blood	2372	95.2
ctd_first_date	2367	95.0
ctd_last_date	2367	95.0
ctabdpelv_first_date	2356	94.6
ctabdpelv_last_date	2356	94.6
ctabdpelv_dur	2356	94.6
oud_last_date	2354	94.5
hypercap_dx_pcpl_flag	2354	94.5
oud_first_date	2353	94.5
hypercap_resp_failure_date	2352	94.4
hypercap_dx_adm_flag	2352	94.4
hypercap_dx_vr_flag	2352	94.4
dx_hypercap_on_abg	2352	94.4
sugg_hypercap_dx_on_vbg	2352	94.4
dx_hypercap_on_vbg	2352	94.4
art_punct_proc_first_date	2350	94.3
art_punct_proc_last_date	2350	94.3
sao2_blood_date	2344	94.1
niv_proc_first_date	2343	94.1
niv_proc_last_date	2343	94.1
niv_proc_dur	2343	94.1
dem_last_date	2340	93.9
dem_first_date	2339	93.9
value_prev_bmi	2337	93.8
prev_bmi_date	2337	93.8
cpap_first_date	2327	93.4
cpap_last_date	2327	93.4
cpap_dur	2327	93.4
phtn_last_date	2317	93.0

variable	n_miss	pct_miss
stroke_first_date	2316	93.0
stroke_last_date	2316	93.0
phtn_first_date	2316	93.0
pvd_first_date	2268	91.0
pvd_last_date	2268	91.0
value_prev_weight	2262	90.8
po2_blood	2262	90.8
prev_weight_date	2262	90.8
po2_blood_date	2260	90.7
_merge_emer_obes	2257	90.6
value_highest_115576	2254	90.5
paco2_date_2	2254	90.5
paco2_date_highest_2	2254	90.5
ph_blood	2249	90.3
ph_blood_date	2249	90.3
ats_copd_flag	2235	89.7
guidelines	2235	89.7
prim_met_alk	2235	89.7
combo_met_alk	2235	89.7
mal_fat_dx_date	2232	89.6
cog_signs_dx_date	2225	89.3
ats_ohs_flag	2223	89.2
pco2_nos	2221	89.2
highest_pco2_nos	2221	89.2
pco2_nos_date	2221	89.2
highest_pco2_nos_date	2221	89.2
sepsis_dx_date	2211	88.8
imv_proc_first_date	2205	88.5
imv_proc_last_date	2205	88.5
imv_proc_dur	2205	88.5
calc_abg	2200	88.3
hypercapnia_calc	2200	88.3
pco2_cat_calc	2200	88.3
op_nrt_last_date	2193	88.0
op_nrt_first_date	2192	88.0

variable	n_miss	pct_miss
asthma_last_date	2192	88.0
asthma_first_date	2191	88.0
aero_first_date	2189	87.9
aero_last_date	2189	87.9
aero_dur	2189	87.9
vbg_o2sat	2186	87.8
vbg_o2sat_date	2185	87.7
tte_proc_first_date	2180	87.5
tte_proc_last_date	2180	87.5
tte_proc_dur	2180	87.5
value_prev_height	2169	87.1
prev_height_date	2169	87.1
blood_cx_proc_first_date	2160	86.7
blood_cx_proc_last_date	2160	86.7
blood_cx_proc_dur	2160	86.7
months_death_or_cens	2157	86.6
death_time	2157	86.6
dysp_dx_date	2145	86.1
death_date	2144	86.1
death_abs	2144	86.1
pna_dx_date	2132	85.6
cw_simple_acute_resp_acid	2109	84.7
vent_proc_first_date	2089	83.9
vent_proc_last_date	2089	83.9
osa_last_date	2071	83.1
osa_first_date	2070	83.1
bnp	2052	82.4
j9601_date	2026	81.3
copd_first_date	2026	81.3
copd_last_date	2026	81.3
narcan_date	2012	80.8
ckd_last_date	2012	80.8
ckd_first_date	2009	80.7
chf_first_date	2005	80.5
chf_last_date	2005	80.5

variable	n_miss	pct_miss
abg_o2sat	1994	80.0
abg_sbe	1990	79.9
abg_ph_date	1975	79.3
abg_hco3	1964	78.8
abg_hco3_int	1964	78.8
abg_po2	1956	78.5
abg_po2_date	1954	78.4
abg_hco3_date	1952	78.4
cc_time_first_date	1944	78.0
cc_time_last_date	1944	78.0
abg_o2sat_date	1931	77.5
value_highest_20198	1930	77.5
paco2_date_1	1930	77.5
paco2_date_highest_1	1930	77.5
muscle_relax_last_date	1898	76.2
muscle_relax_first_date	1897	76.2
vbg_po2	1865	74.9
vbg_po2_date	1858	74.6
vbg_sbe	1857	74.5
nic_first_date	1857	74.5
nic_last_date	1857	74.5
_merge_emer_predisp	1855	74.5
other_resp_failure_date	1854	74.4
vbg_hco3	1851	74.3
vbg_hco3_date	1837	73.7
cxr1v_first_date	1836	73.7
cxr1v_last_date	1836	73.7
cxr1v_dur	1836	73.7
_merge_emer_respfail	1831	73.5
vbg_ph_date	1813	72.8
_merge_inpat_obes	1810	72.7
vbg_co2	1778	71.4
vbg_co2_flag	1778	71.4
corr_vbg_co2	1778	71.4
corr_vbg_co2_flag	1778	71.4

variable	n_miss	pct_miss
pco2_cat_vbg	1778	71.4
highest_vbg_co2	1777	71.3
highest_vbg_co2_flag	1777	71.3
spo2	1776	71.3
vbg_co2_date	1776	71.3
highest_vbg_co2_date	1776	71.3
spo2_date	1770	71.1
dm_last_date	1756	70.5
dm_first_date	1755	70.5
vbg_ph	1735	69.7
_merge_emer_vbg	1648	66.2
paco2	1630	65.4
paco2_int	1630	65.4
paco2_flag	1630	65.4
paco2_52_flag	1630	65.4
not_paco2_flag	1630	65.4
paco2_52_comp_flag	1630	65.4
max_hco3_or_its_met_alk	1630	65.4
max_hco3_or_its_comb_met_alk	1630	65.4
paco2_50_flag	1630	65.4
pco2_cat_abg	1630	65.4
paco2_date	1629	65.4
highest_paco2	1629	65.4
paco2_date_highest	1629	65.4
highest_paco2_flag	1629	65.4
_merge_emer_ventsupp	1629	65.4
_merge_emer	1629	65.4
abg_ph	1619	65.0
op_diuretics_last_date	1569	63.0
op_diuretics_first_date	1568	62.9
serum_lac	1501	60.3
serum_lac_date	1484	59.6
inpt_inh_date	1476	59.3
po_steroid_date	1410	56.6
curr_bmi	1409	56.6

variable	n_miss	pct_miss
curr_bmi_date	1409	56.6
working_bmi	1408	56.5
working_bmi_date	1408	56.5
rr	1355	54.4
rr_date	1350	54.2
serum_phos	1343	53.9
copd_med_last_date	1263	50.7
copd_med_first_date	1260	50.6
_merge_inpat_predisp	1255	50.4
temp_new	1198	48.1
new_temp_date	1198	48.1
acidemia	1154	46.3
_merge_inpat_respfail	1113	44.7
calc_bmi	1027	41.2
calc_bmi_date	1027	41.2
serum_tprot	947	38.0
op_opiate_last_date	928	37.3
_merge_inpat_vbg	928	37.3
op_opiate_first_date	919	36.9
serum_albumin	897	36.0
hr	892	35.8
hr_date	892	35.8
is_inp	862	34.6
_merge_inpat_ventsupp	862	34.6
bmi	852	34.2
bmi_date	852	34.2
bmi_int	852	34.2
bmi_by_five	852	34.2
curr_weight	779	31.3
curr_weight_date	779	31.3
dbp	778	31.2
weight	777	31.2
weight_date	777	31.2
dbp_date	773	31.0
sbp	767	30.8

variable	n_miss	pct_miss
sbp_date	765	30.7
curr_height	678	27.2
curr_height_date	678	27.2
height	677	27.2
height_date	677	27.2
wbc	421	16.9
wbc_date	376	15.1
hgb	272	10.9
serum_ca	242	9.71
hgb_date	221	8.87
serum_cr	217	8.71
plt	204	8.19
serum_cr_date	200	8.03
serum_k	189	7.59
serum_k_date	189	7.59
k_cat	189	7.59
plt_date	176	7.07
serum_cl	154	6.18
serum_cl_date	154	6.18
serum_hco3	147	5.90
hco3_cat	147	5.90
hco3_flag	147	5.90
not_hco3_flag	147	5.90
serum_hco3_date	144	5.78
sodium	137	5.50
sodium_date	137	5.50
any_bicarb	123	4.94
int_bicarb	123	4.94
died_1mo	13	0.522
died_2mo	13	0.522
encounter_id	0	0
rfs	0	0
sex	0	0
race	0	0
ethnicity	0	0

variable	n_miss	pct_miss
location	0	0
age_at_encounter	0	0
los	0	0
bnp_date	0	0
serum_phos_date	0	0
serum_ca_date	0	0
serum_albumin_date	0	0
serum_tprot_date	0	0
has_j9612	0	0
has_j9622	0	0
has_j9602	0	0
has_j9692	0	0
ohs_code	0	0
has_j9600	0	0
principal_diagnosis_indicator	0	0
admitting_diagnosis	0	0
reason_for_visit	0	0
has_j9601	0	0
has_j961	0	0
has_j9610	0	0
has_j9611	0	0
has_j962	0	0
has_j9620	0	0
has_j9621	0	0
has_j9690	0	0
has_j9691	0	0
other_abn_of_br	0	0
cfdo	0	0
has_i50_acute	0	0
acute_nmd	0	0
sepsis_dx	0	0
stupor_dx	0	0
cog_signs_dx	0	0
mal_fat_dx	0	0
resp_acid_dx	0	0

variable	n_miss	pct_miss
sleep_hypovent_dx	0	0
cchs_dx	0	0
other_sleep_hypovent_dx	0	0
acidosis_unspec	0	0
headache_dx	0	0
cpap	0	0
tte_proc	0	0
aero	0	0
inh_teaching	0	0
cxr1v	0	0
cxr2v	0	0
ctcnoncon	0	0
ctccon	0	0
cc_time	0	0
meas_venous_o2_proc	0	0
meas_arterial_gas_proc	0	0
blood_cx_proc	0	0
art_punct_proc	0	0
ctabdpelv	0	0
osa	0	0
asthma	0	0
copd	0	0
chf	0	0
stroke	0	0
ckd	0	0
pvd	0	0
oud	0	0
sedatives	0	0
phtn	0	0
polycy	0	0
po_steroid	0	0
narcan	0	0
inpt_inh	0	0
vasodilators	0	0
ip_diuretics	0	0

variable	n_miss	pct_miss
ip_abx	0	0
paralytic	0	0
op_diuretics	0	0
op_opiate	0	0
op_mat	0	0
op_nrt	0	0
copd_med	0	0
muscle_relax	0	0
pat_enc_hash	0	0
ABG_rfs	0	0
is_amb	0	0
encounter_date	0	0
age_by_ten	0	0
age_decade	0	0
died	0	0
miss_paco2_flag	0	0
miss_vbg_co2_flag	0	0
miss_vbg_or_abg_co2_flag	0	0
inpt_inh_0	0	0
ip_abx_0	0	0
ip_diuretics_0	0	0
narcan_0	0	0
paralytic_0	0	0
po_steroid_0	0	0
vasodilators_0	0	0
op_diuretics_365d	0	0
op_opiate_365d	0	0
op_mat_365d	0	0
op_nrt_365d	0	0
copd_med_365d	0	0
muscle_relax_365d	0	0
vent_proc	0	0
niv_proc	0	0
imv_proc	0	0
aero_0	0	0

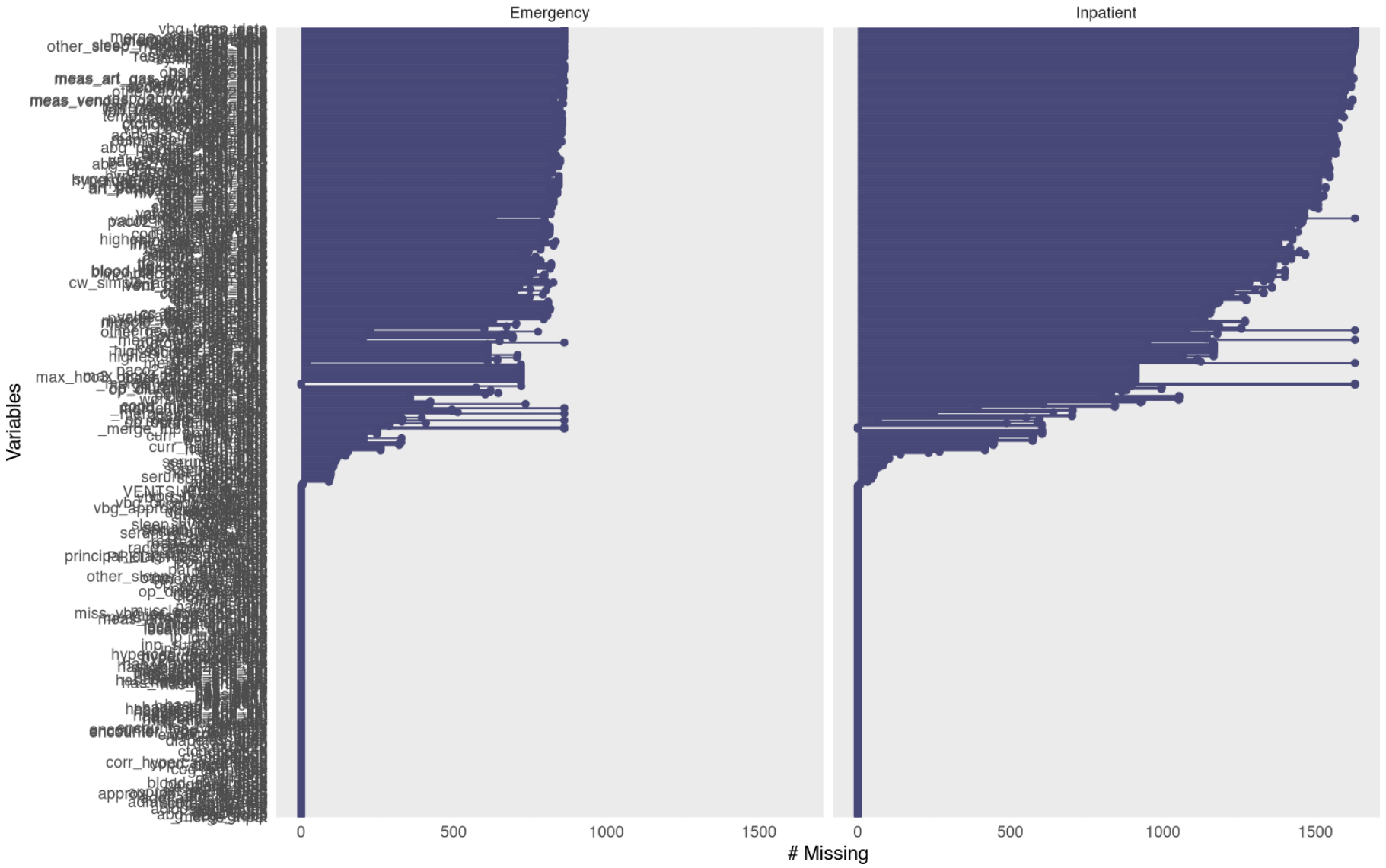
variable	n_miss	pct_miss
blood_cx_proc_0	0	0
cc_time_0	0	0
cpap_0	0	0
ctabdpelv_0	0	0
ctccon_0	0	0
ctcnoncon_0	0	0
cxr1v_0	0	0
cxr2v_0	0	0
imv_proc_0	0	0
inh_teaching_0	0	0
niv_proc_0	0	0
tte_proc_0	0	0
vent_proc_0	0	0
hypercap_resp_failure	0	0
other_resp_failure	0	0
dysp_dx	0	0
symp_obs	0	0
abn_br_dx	0	0
resp_abnormality	0	0
fast_br	0	0
pulm_edema_dx	0	0
pna_dx	0	0
acute_chf	0	0
acute_old	0	0
acute_old_date	0	0
resp_dep_compl	0	0
ctd	0	0
dem	0	0
dm	0	0
nmd	0	0
nic	0	0
ovs	0	0
pos_ohs_flag	0	0
OBESITY_rfs	0	0
PREDISPOSITION_rfs	0	0

variable	n_miss	pct_miss
RESPFAIL_rfs	0	0
VBG_rfs	0	0
VENTSUPPORT_rfs	0	0
is_emer	0	0
_merge_inpat	0	0
patient_id	0	0
rfsgroup	0	0
encounter_type	0	0
first_encounter	0	0
has_abg	0	0
has_vbg	0	0
hypercap_on_abg	0	0
hypercap_on_vbg	0	0
has_both_abg_vbg	0	0
has_neither_abg_vbg	0	0
vbg_or_abg_co2_flag	0	0
vbg_o2sat_calc	0	0
abg_o2sat_calc	0	0
corr_hypercap_on_vbg	0	0
race_ethnicity	0	0
has_vbg_and_cat	0	0
has_bmi_and_cat	0	0
has_weight_and_cat	0	0
has_height_and_cat	0	0
has_hr_and_cat	0	0
has_sbp_and_cat	0	0
has_rr_and_cat	0	0
has_temp_and_cat	0	0
has_spo2_and_cat	0	0
has_cl_and_cat	0	0
has_k_and_cat	0	0
has_hco3_and_cat	0	0
has_lactate_and_cat	0	0
has_na_and_cat	0	0
has_cr_and_cat	0	0

variable	n_miss	pct_miss
has_hgb_and_cat	0	0
has_wbc_and_cat	0	0
has_plt_and_cat	0	0
has_bnp_and_cat	0	0
has_phos_and_cat	0	0
has_ca_and_cat	0	0
has_alb_and_cat	0	0
has_tprot_and_cat	0	0
encounter_type_dummy1	0	0
encounter_type_dummy2	0	0
encounter_type_dummy3	0	0
female	0	0
male	0	0
white_race	0	0
black_race	0	0
unknown_race	0	0
asian_race	0	0
nat_am_race	0	0
nhpi_race	0	0
not_hisp_eth	0	0
hisp_eth	0	0
unknown_eth	0	0
location_dummy1	0	0
location_dummy2	0	0
location_dummy3	0	0
location_dummy4	0	0
ps	0	0
tx_pr_decile	0	0
sumofweights	0	0
ipw	0	0
approx_ipw_fweight	0	0
inp_ps	0	0
inp_sumofweights	0	0
inp_ipw	0	0
approx_inp_ipw_fweight	0	0

variable	n_miss	pct_miss
vbg_ps	0	0
vbg_tx_pr_decile	0	0
vbg_sumofweights	0	0
vbg_ipw	0	0
vbg_approx_ipw_fweight	0	0
ref_ym	0	0
death_60d	0	0
abg_group	0	0
vbg_group	0	0
sex_label	0	0
race_ethnicity_label	0	0
location_label	0	0
encounter_type_label	0	0
osa_label	0	0
asthma_label	0	0
copd_label	0	0
chf_label	0	0
nmd_label	0	0
phtn_label	0	0
ckd_label	0	0
diabetes_label	0	0
abg_status	0	0
vbg_status	0	0
hyper_abg	0	0
hyper_vbg	0	0
w_abg	0	0
w_vbg	0	0
has_vbg_co2_o2_sat	0	0
w_vbg_calc	0	0

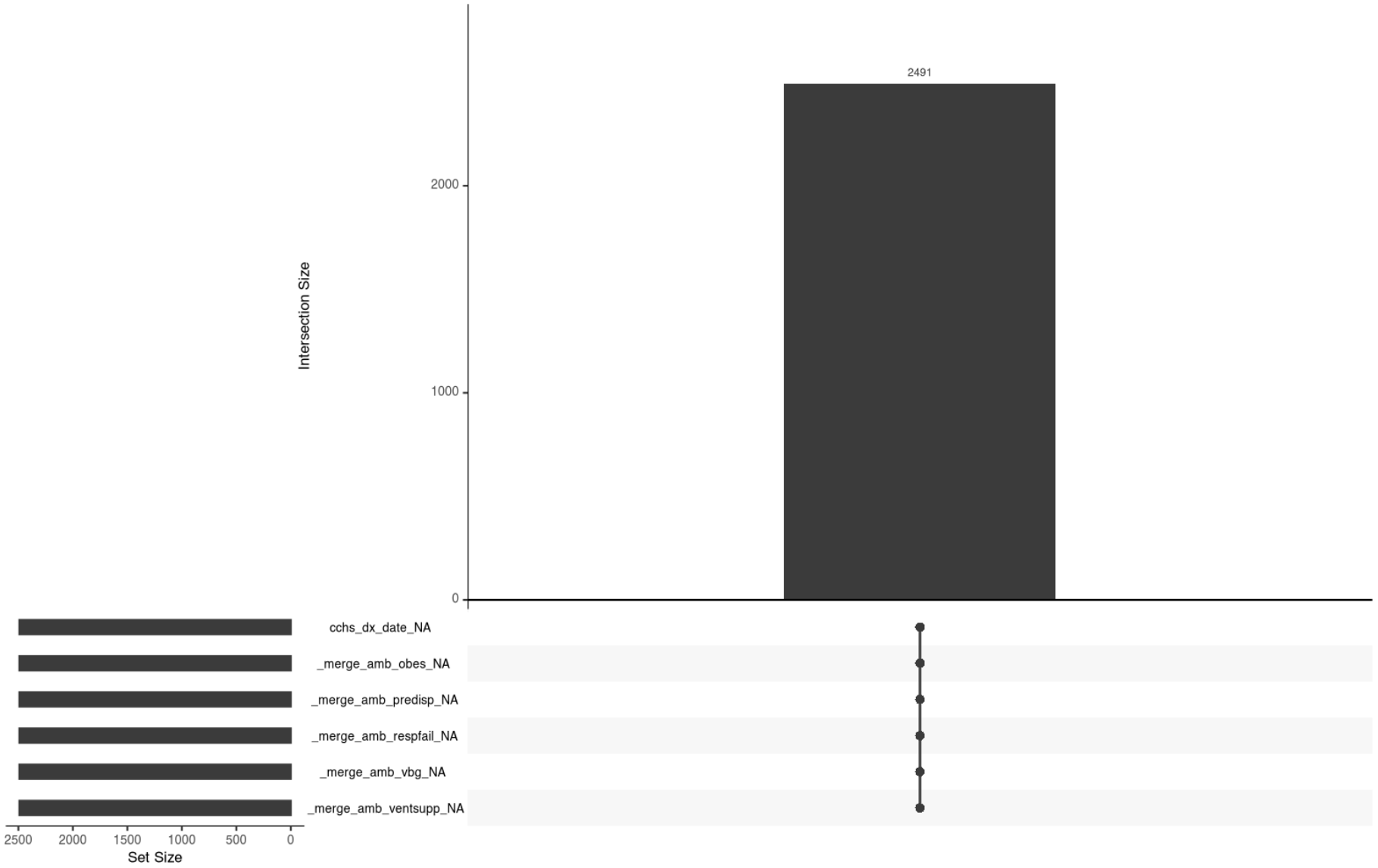
```
# Patterns
naniar::gg_miss_var(subset_data, facet = encounter_type)
```



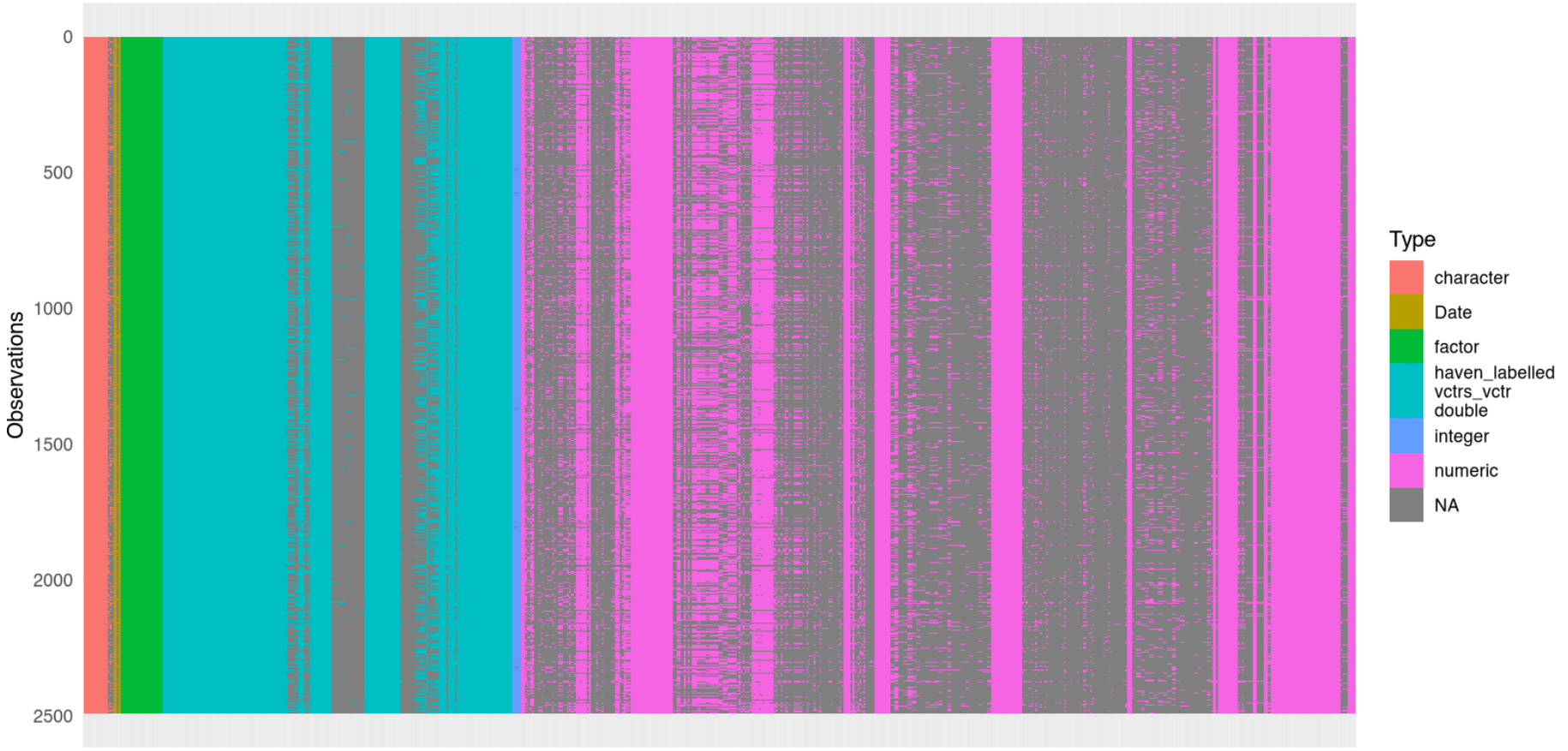

```
naniar::gg_miss_upset(subset_data, nsets = 6)
```

```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with `aes()`.
```

```
i See also `vignette("ggplot2-in-packages")` for more information.
i The deprecated feature was likely used in the UpSetR package.
  Please report the issue to the authors.

Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
i The deprecated feature was likely used in the UpSetR package.
  Please report the issue to the authors.
```




```
# Types & potential issues
visdat::vis_dat(subset_data, warn_large_data = FALSE)
```


Observations
0
500
1000
1500
2000
2500
Type
character
Date
factor
haven_labelled
vctrs_vctr
double
integer
numeric
NA

2.2 2) 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
      862      1629
```

```
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)

# --- Covariate set for GBM propensity models ---------------------------------
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"
)

# Verify presence
missing_covars <- setdiff(covars_gbm, names(subset_data))
if (length(missing_covars)) {
  warning("These covariates are missing from subset_data: ",
          paste(missing_covars, collapse = ", "))
} else {
  message("All GBM covariates present.")
}
```

```
All GBM covariates present.
```

2.3 3) Imputation model specification (MICE)

2.3.1 3.1) 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)
```

```r
library(dplyr)

# --- variables for GBM propensity (kept identical to main analysis) ---
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"
)

# --- 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)
  "imv_proc","niv_proc","death_60d","hypercap_resp_failure",  # outcomes (NOT imputed)
  co2_vars
))

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

# Make binary comorbids 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)
```

```r
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

# 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
  1   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  1   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  1   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  1   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  1   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  2   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  2   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  2   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  2   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  2   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  3   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  3   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  3   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  3   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  3   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  4   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  4   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  4   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  4   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  4   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  5   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  5   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  5   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  5   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  5   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  6   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
```

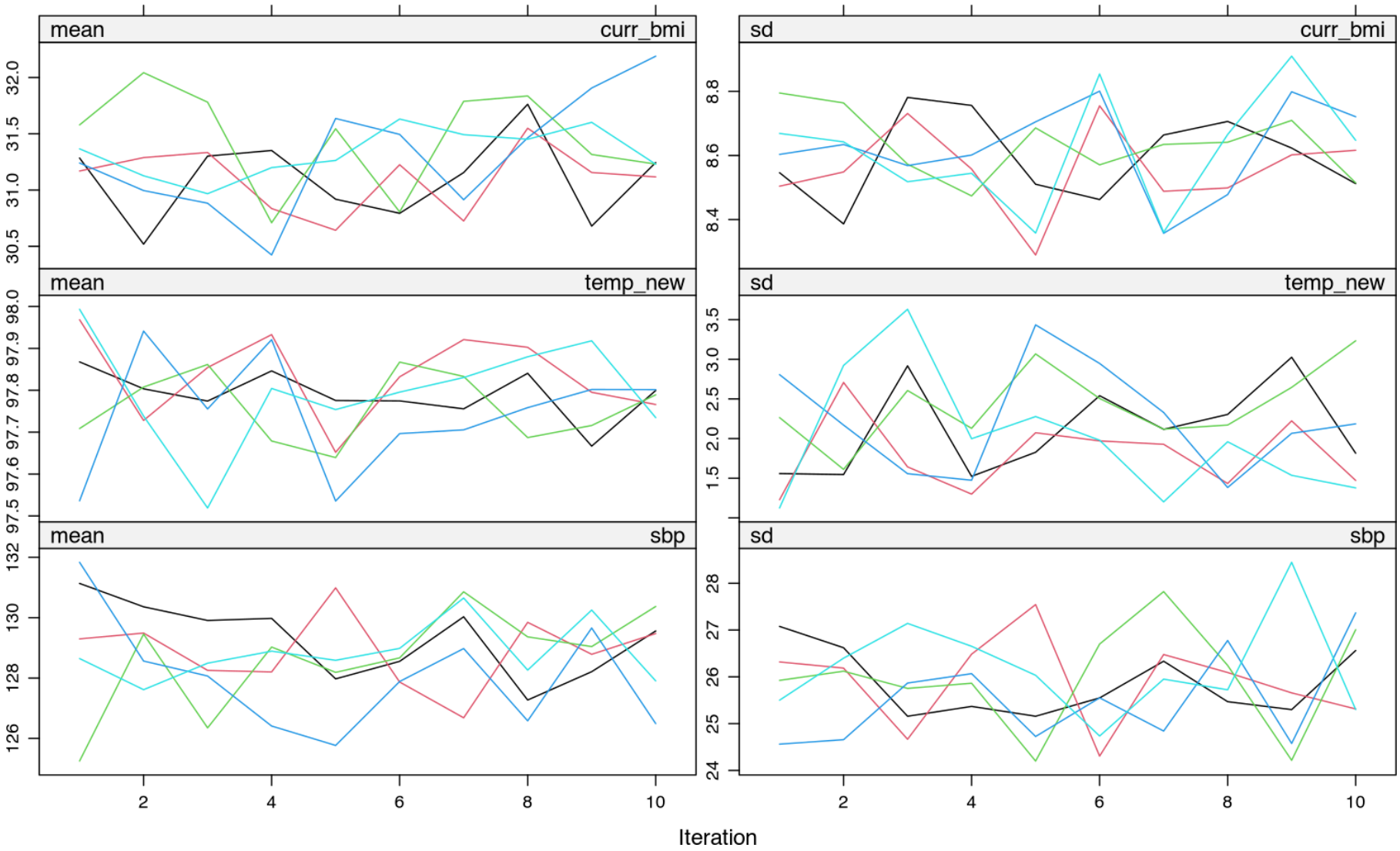
```
  6   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  6   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  6   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  6   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  7   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  7   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  7   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  7   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  7   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  8   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  8   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  8   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  8   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  8   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  9   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  9   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  9   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  9   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
  9   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
 10   1  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
 10   2  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
 10   3  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
 10   4  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
 10   5  curr_bmi  temp_new  sbp  dbp  hr  spo2  sodium  serum_cr  serum_hco3  serum_cl  serum_lac  serum_k  wbc  plt  bnp  seru
```

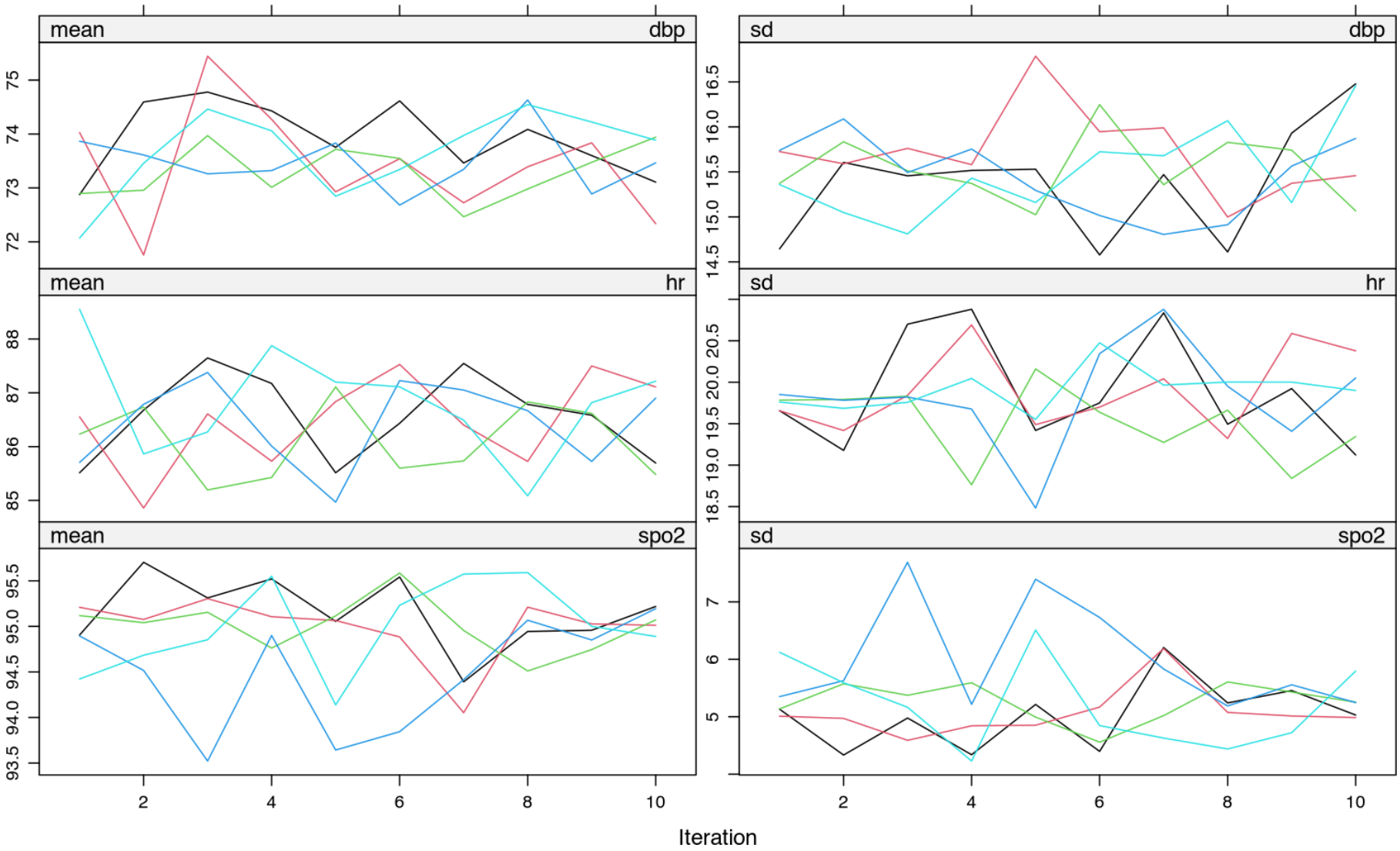
```
Warning: Number of logged events: 50
```

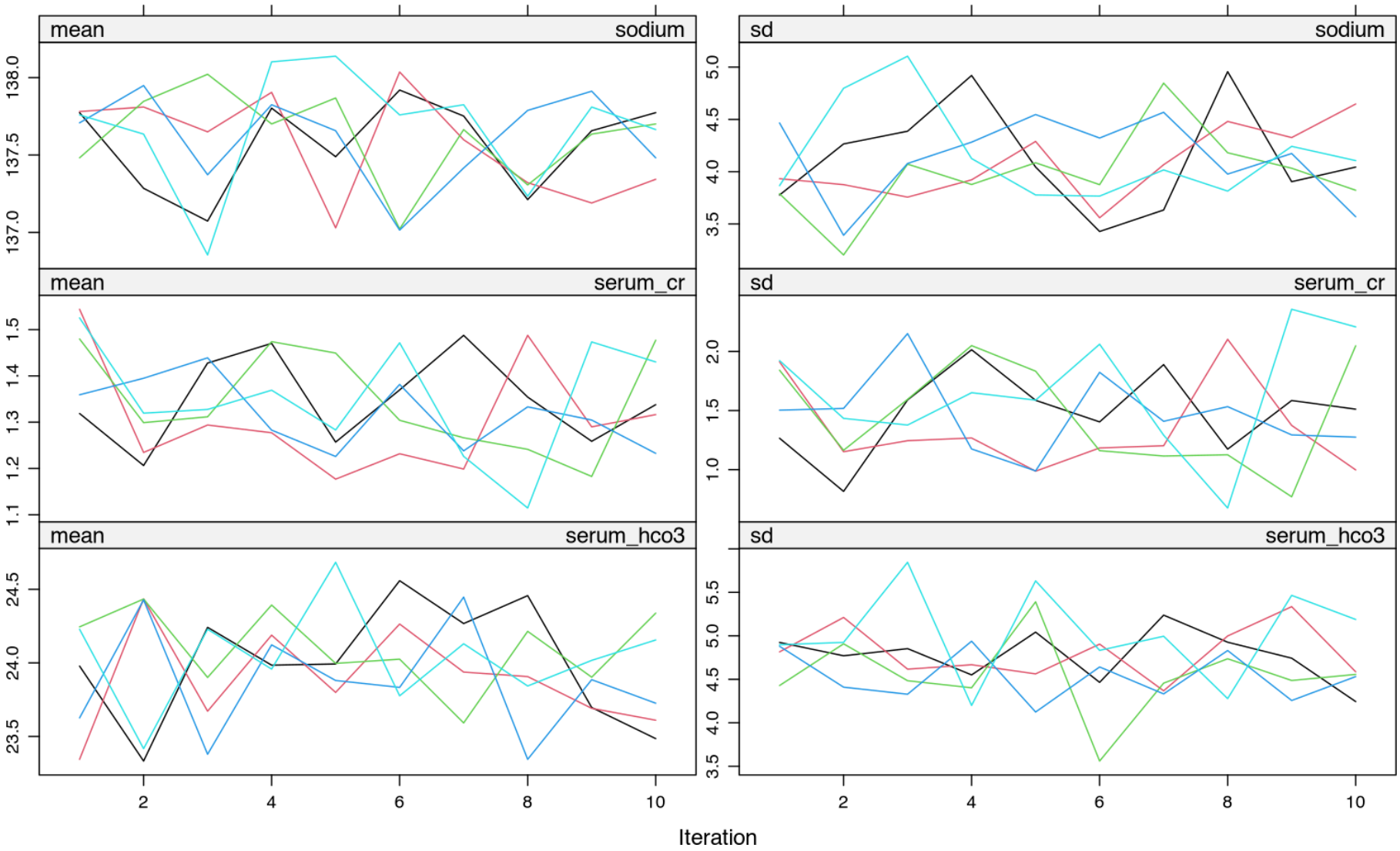
```r
saveRDS(imp, file = "mi_abg_vbg_mids.rds")

# 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))
```

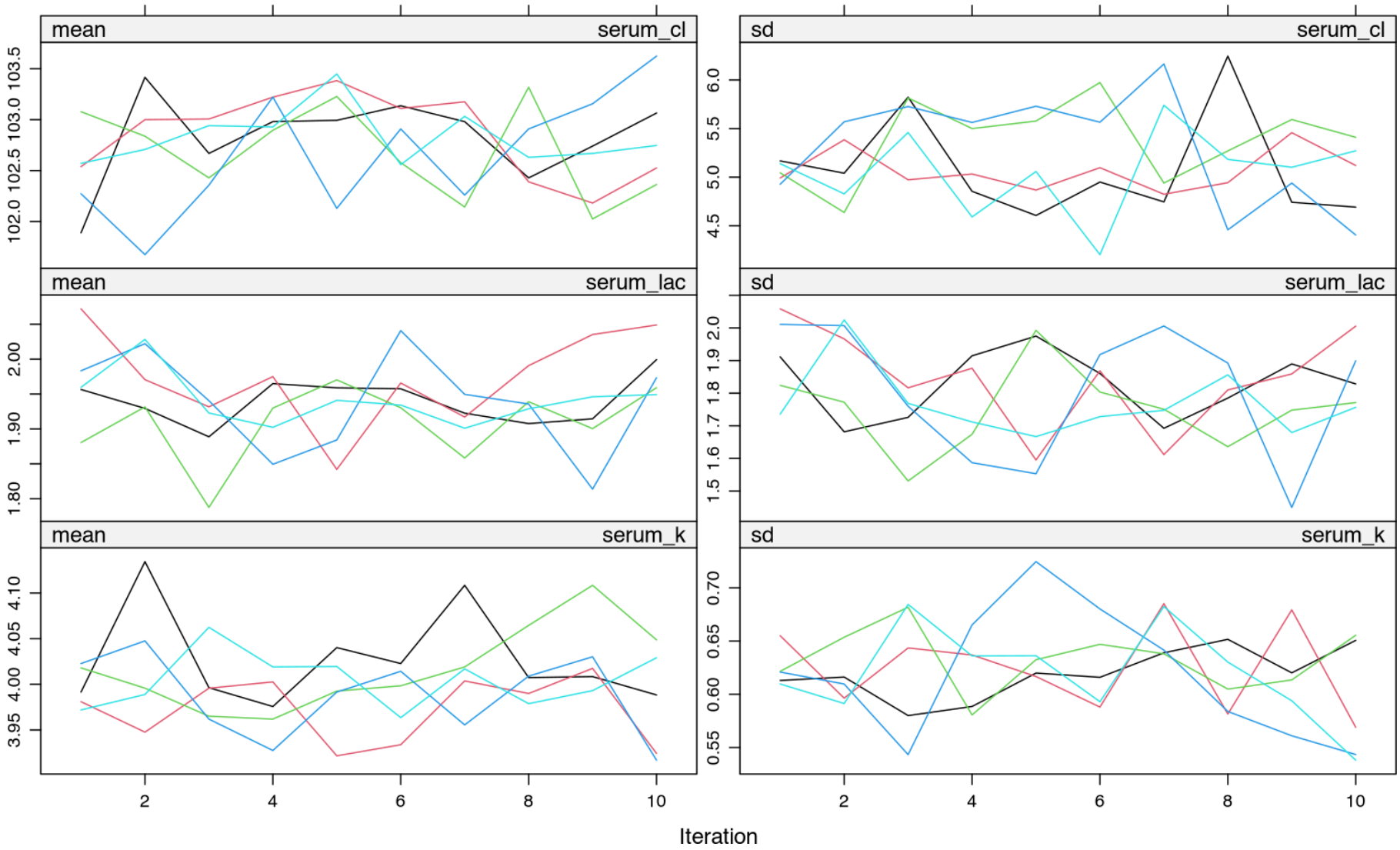
2.3.2 3.3) Convergence & plausibility checks

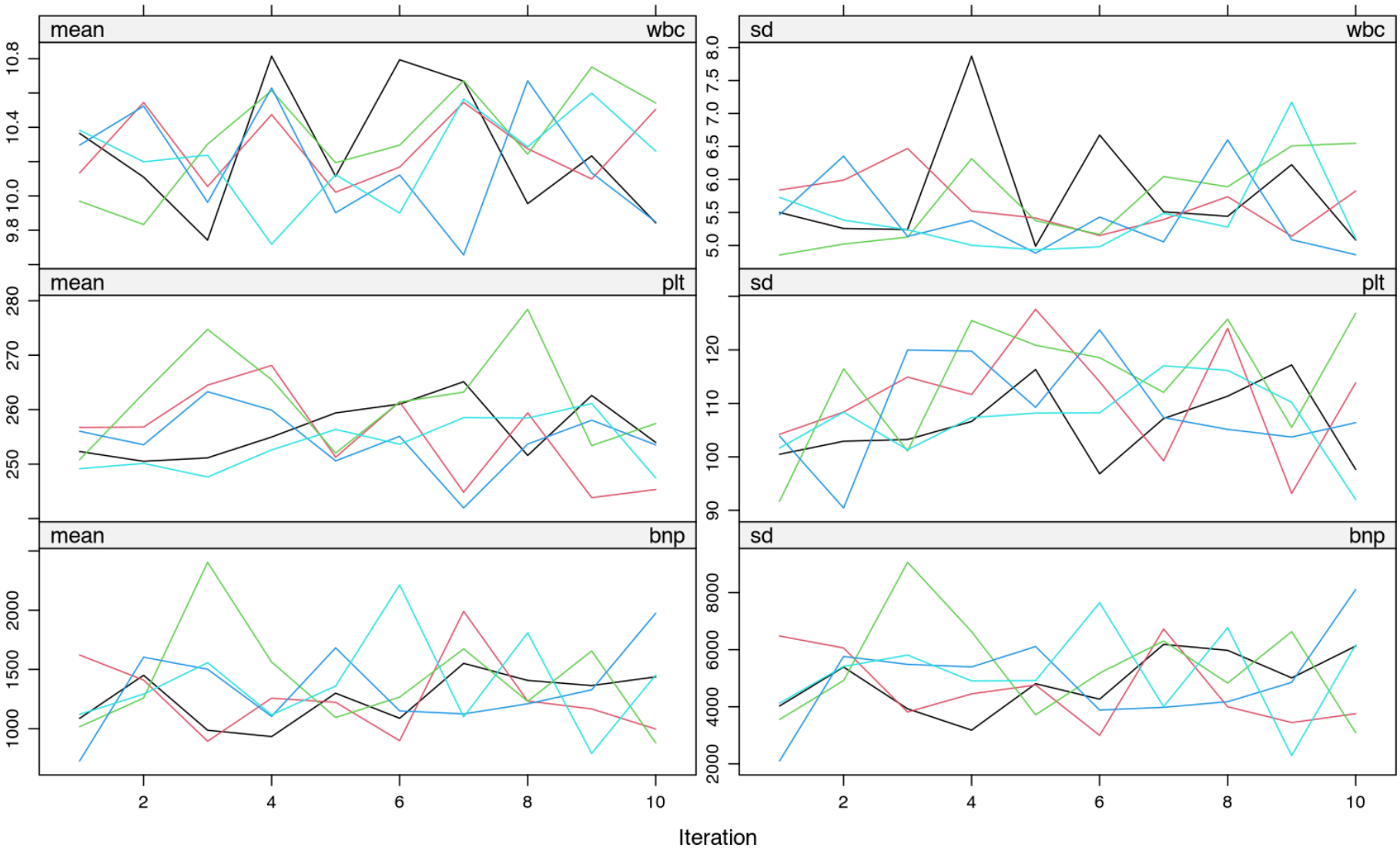
```
imp <- readRDS("mi_abg_vbg_mids.rds")

plot(imp)                                    # trace plots
```

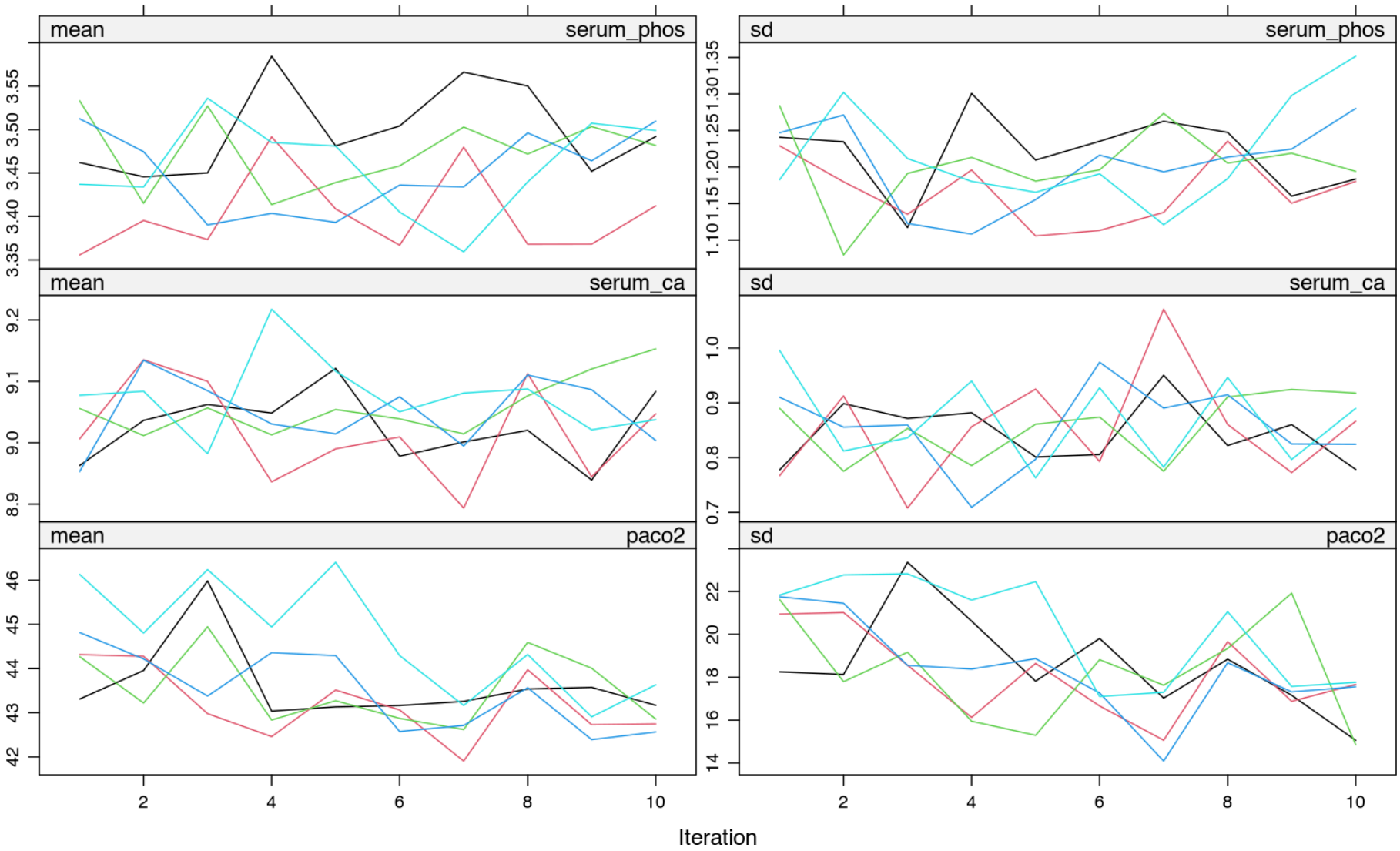

mean
curr_bmi
sd
curr_bmi
mean
temp_new
sd
temp_new
mean
sbp
sd
sbp
Iteration

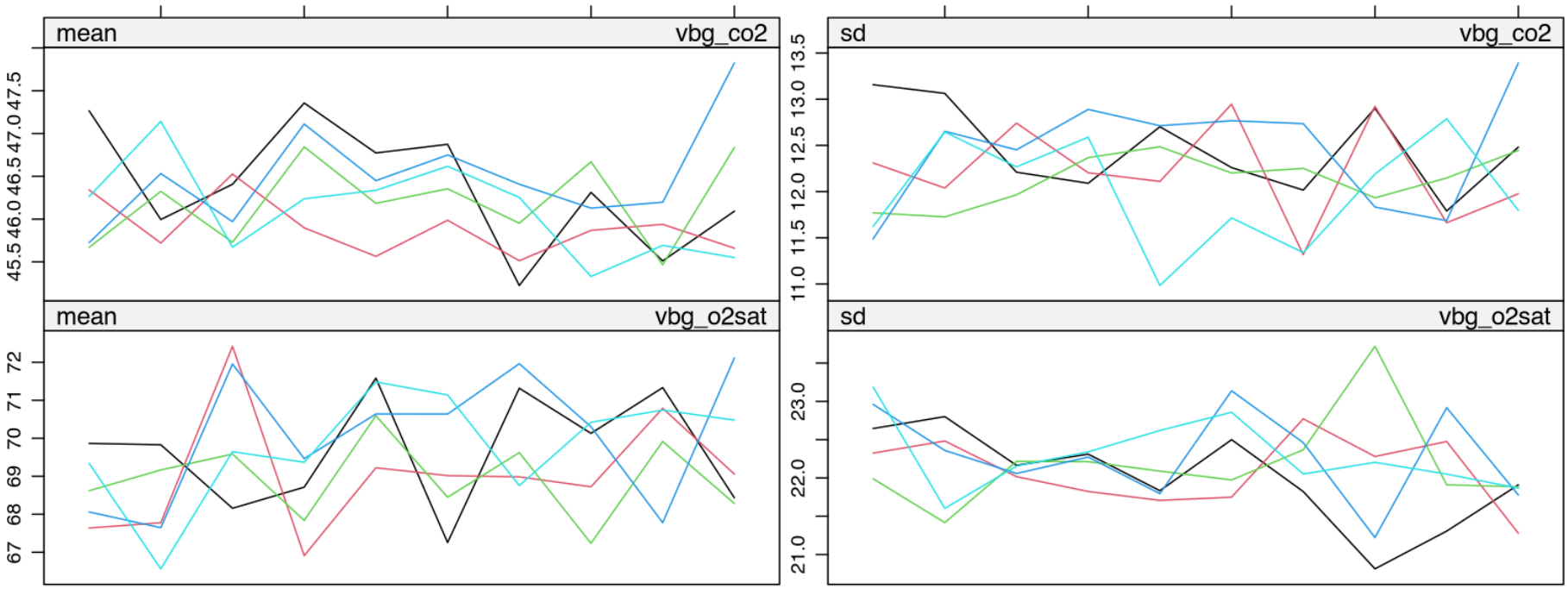

mean
dbp
sd
dbp
mean
hr
sd
hr
mean
spo2
sd
spo2
Iteration


mean
sodium
sd
sodium
mean
serum_cr
sd
serum_cr
mean
serum_hco3
sd
serum_hco3
137.0
137.5
138.0
3.5
4.0
4.5
5.0
1.1
1.2
1.3
1.4
1.5
1.0
1.5
2.0
23.5
24.0
24.5
3.5
4.0
4.5
5.0
5.5
2
4
6
8
10
Iteration

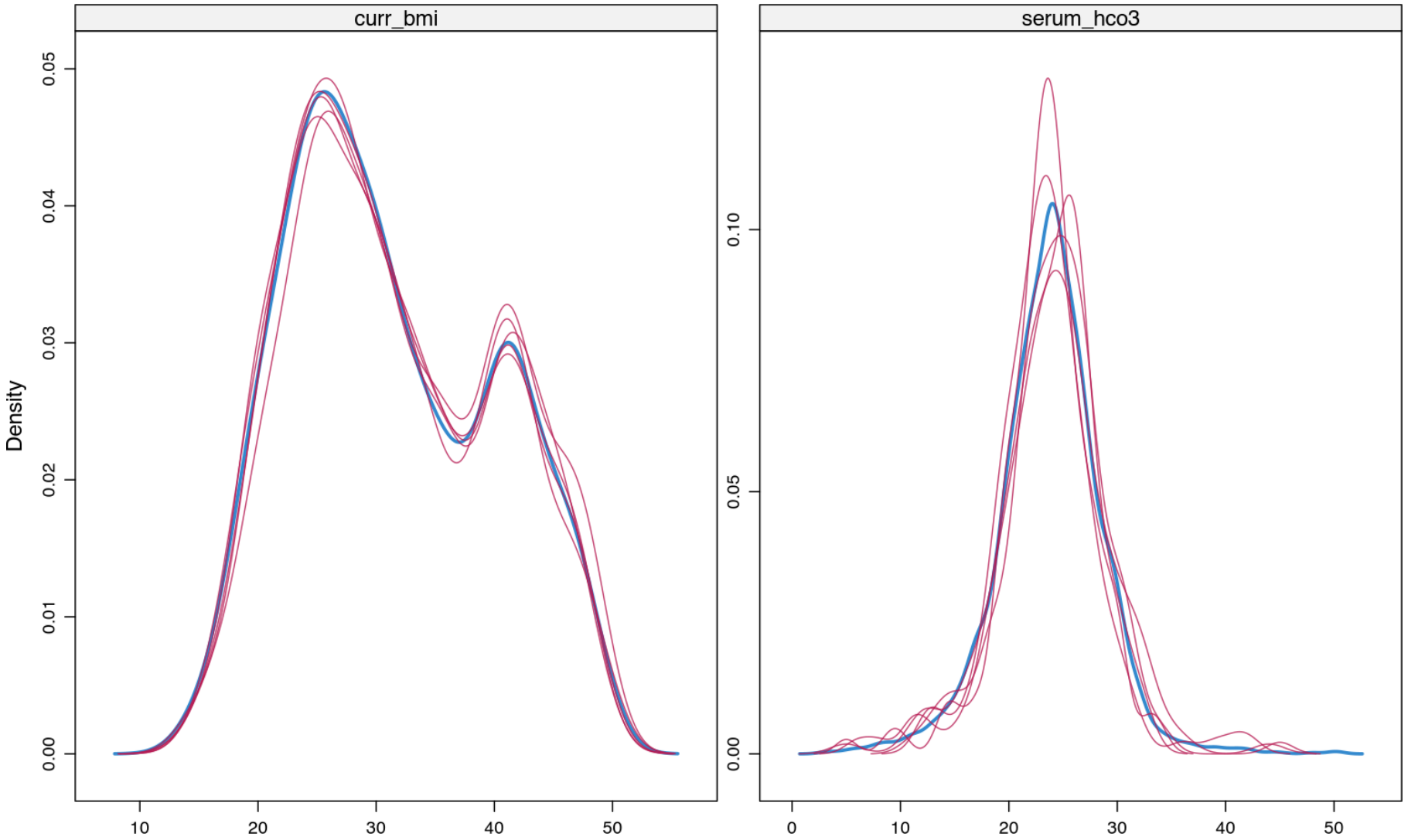

mean
serum_cl
sd
serum_cl
mean
serum_lac
sd
serum_lac
mean
serum_k
sd
serum_k
102.0 102.5 103.0 103.5
4.5 5.0 5.5 6.0
1.80 1.90 2.00
1.5 1.6 1.7 1.8 1.9 2.0
3.95 4.00 4.05 4.10
0.55 0.60 0.65 0.70
2 4 6 8 10
2 4 6 8 10
Iteration


mean
wbc
sd
wbc
mean
plt
sd
plt
mean
bnp
sd
bnp
Iteration

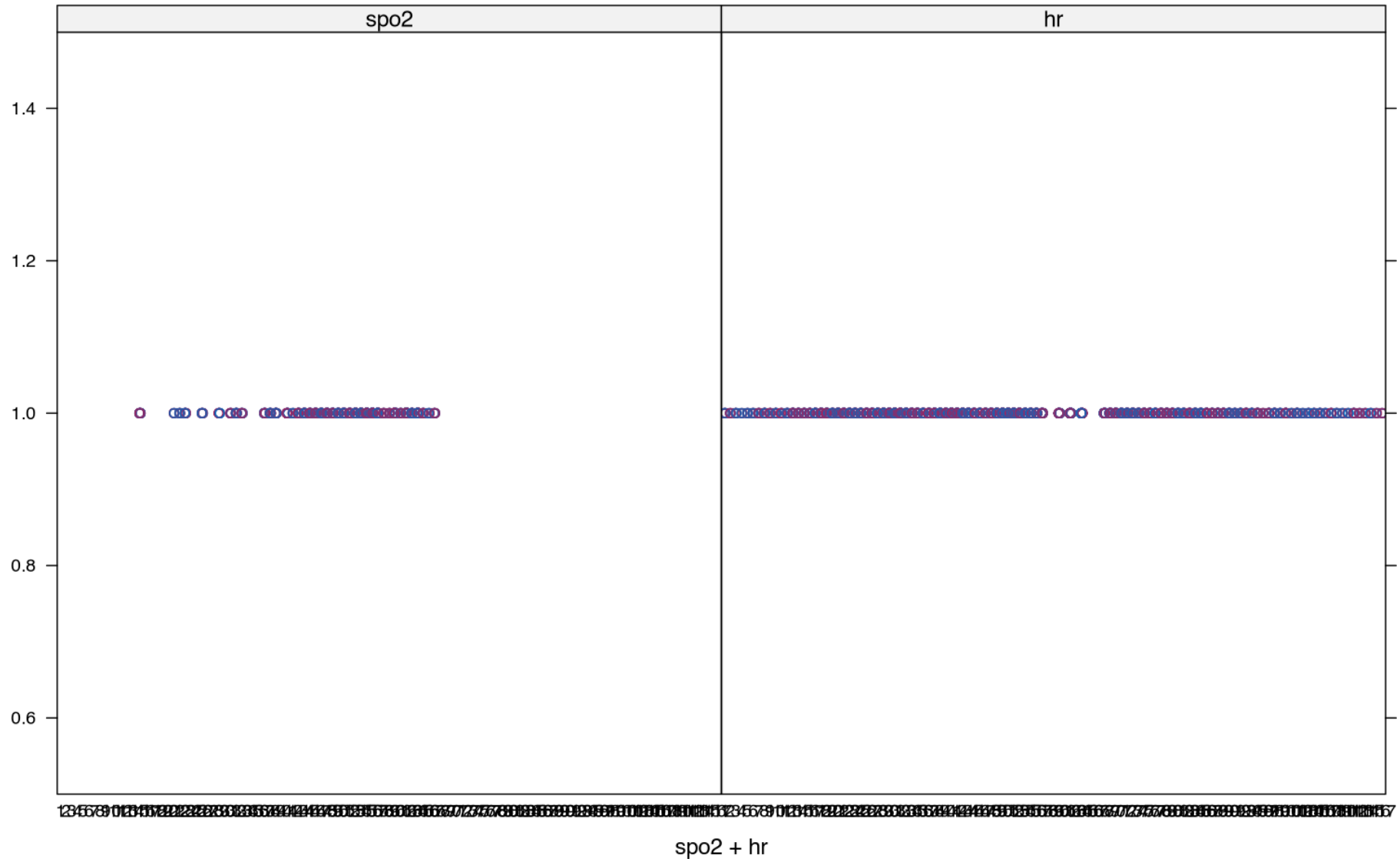

mean
serum_phos
sd
serum_phos
mean
serum_ca
sd
serum_ca
mean
paco2
sd
paco2
Iteration




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




```
stripplot(imp,  ~ spo2 + hr)                # distribution overlap
```




```
md.pattern(complete(imp, "long"))            # pattern after MI - should be complete
```

```
 /\     /\
{  `---'  }
```

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


```
      age_at_encounter sex race_ethnicity curr_bmi copd asthma osa chf
12455                1   1              1        1    1      1   1   1
                     0   0              0        0    0      0   0   0
      acute_nmd phtn ckd dm location encounter_type temp_new sbp dbp hr spo2
12455         1    1   1  1        1              1        1   1   1  1    1
              0    0   0  0        0              0        0   0   0  0    0
      sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt bnp
12455      1        1          1        1         1       1   1   1   1
           0        0          0        0         0       0   0   0   0
      serum_phos serum_ca has_abg has_vbg imv_proc niv_proc death_60d
12455          1        1       1       1        1        1         1
               0        0       0       0        0        0         0
      hypercap_resp_failure paco2 vbg_co2 vbg_o2sat .imp .id
12455                     1     1       1         1    1   1 0
                          0     0       0         0    0   0 0
```

2.4 4) 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.

2.4.1 4.1) ABG propensity (has_abg)

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

```r
# Fit ABG propensity weights in each imputation
fit_abg_one <- function(d) {
  w <- weightit(
    has_abg ~ .,
    data   = d[, c("has_abg", covars_gbm)],
    method = "gbm",
    estimand    = "ATE",
    include.obj = TRUE,
    n.trees     = 1000,
    interaction.depth = 3,
    shrinkage   = 0.02,
    bag.fraction= 0.6,
    cv.folds    = 2,
    stop.method = "es.mean"
  )
  # 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))
  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")
```

2.4.2 4.2) 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>")
```

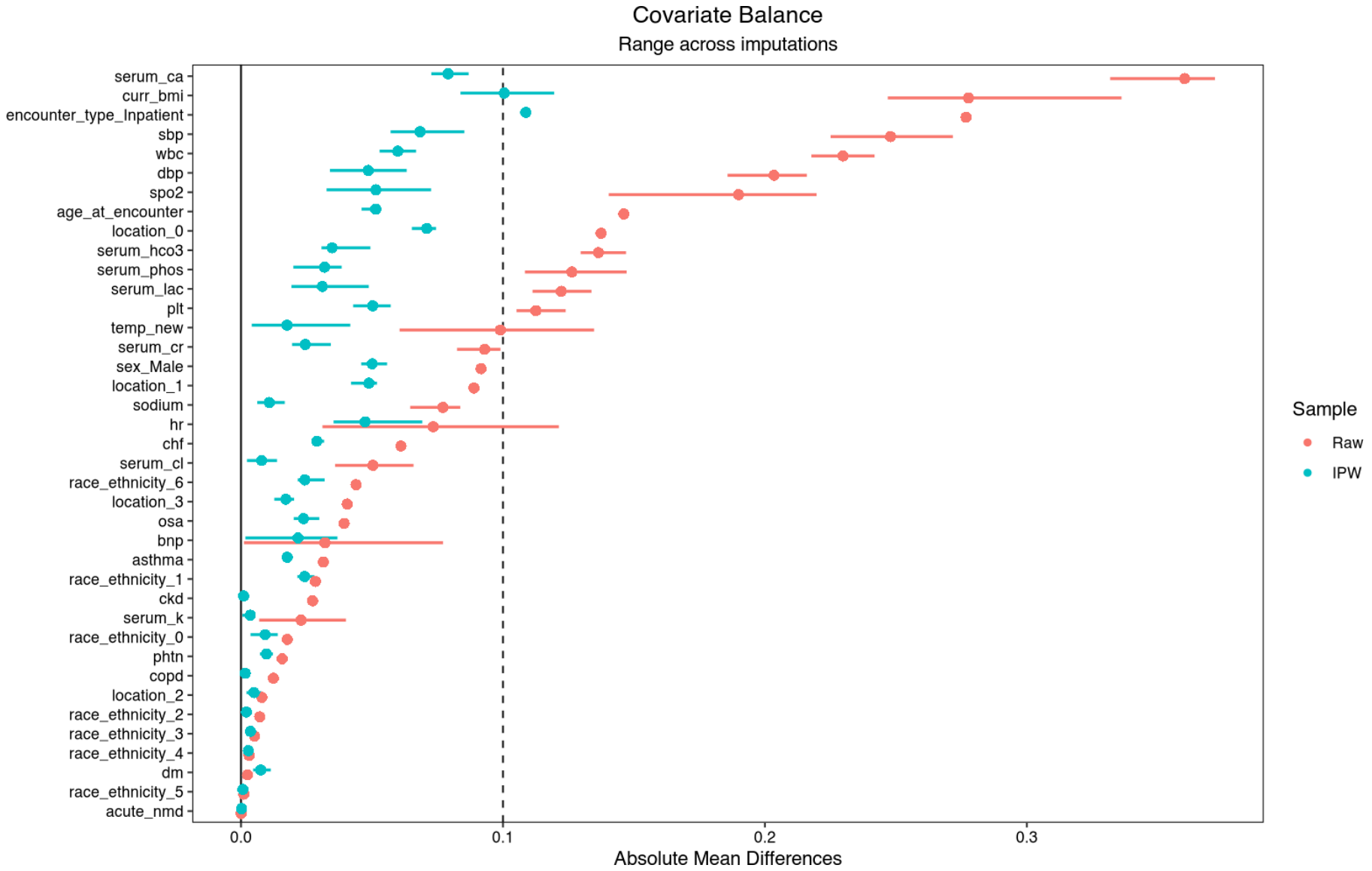
```
Offenders (post CC): <none>
```

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

```r
# 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(
```

```r
  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
)

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

```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```


Covariate Balance
Range across imputations
serum_ca
curr_bmi
encounter_type_Inpatient
sbp
wbc
dbp
spo2
age_at_encounter
location_0
serum_hco3
serum_phos
serum_lac
plt
temp_new
serum_cr
sex_Male
location_1
sodium
hr
chf
serum_cl
race_ethnicity_6
location_3
osa
bnp
asthma
race_ethnicity_1
ckd
serum_k
race_ethnicity_0
phtn
copd
location_2
race_ethnicity_2
race_ethnicity_3
race_ethnicity_4
dm
race_ethnicity_5
acute_nmd
0.0
0.1
0.2
0.3
Absolute Mean Differences
Sample
Raw
IPW

2.4.3 4.3) VBG propensity (has_vbg) — mirror of 4.1

```
fit_vbg_one <- function(d) {
  w <- weightit(
    has_vbg ~ .,
    data   = d[, c("has_vbg", covars_gbm)],
    method = "gbm",
    estimand    = "ATE",
    include.obj = TRUE,
    n.trees     = 2000,
    interaction.depth = 3,
    shrinkage   = 0.01,
    bag.fraction= 0.6,
    cv.folds    = 3,
    stop.method = "es.mean"
  )
  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))
  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
  )
})
```

```r
saveRDS(W_vbg_list, "mi_W_vbg_list.rds")
```

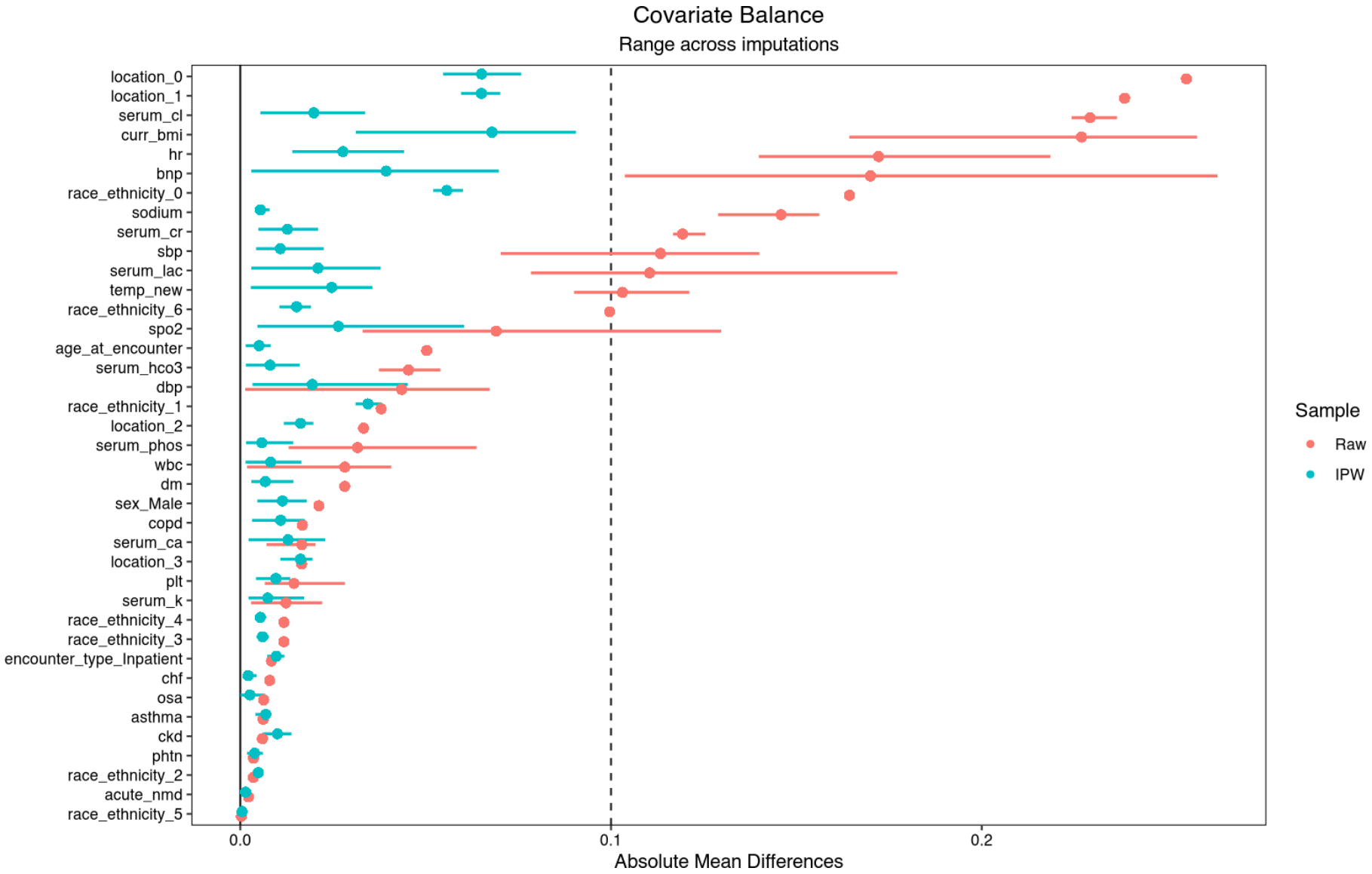
2.4.4 4.4) VBG Balance

```r
# --- VBG: balance across imputations (parallel to ABG) ------------------------

# Preconditions: dlist (m completed data sets), W_vbg_list (weights per imputation),
#                covars_gbm (covariate set). If W_vbg_list not in memory, load it.
if (!exists("W_vbg_list", inherits = TRUE)) {
  W_vbg_list <- readRDS("mi_W_vbg_list.rds")
}

# Helper(s) if not already defined above
if (!exists("vary_ok", inherits = TRUE)) {
  vary_ok <- function(z) {
    nz <- z[!is.na(z)]
    if (is.factor(nz)) nlevels(droplevels(nz)) > 1 else dplyr::n_distinct(nz) > 1
  }
}
if (!exists("make_long_for_cobalt", inherits = TRUE)) {
  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
    }))
  }
}

# 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) Long data (stack imputations), attach weights and imputation id
dlong_vbg <- make_long_for_cobalt(dlist, W_vbg_list, "has_vbg", vars_keep_vbg)
dlong_vbg <- droplevels(dlong_vbg)  # remove empty factor levels from stacking

# 3) Final guard: drop any factor that is 1-level in the stacked frame
one_level_factors_vbg <- names(Filter(function(x) is.factor(x) && nlevels(x) < 2,
                                      dlong_vbg[vars_keep_vbg]))
if (length(one_level_factors_vbg)) {
  message("Dropping 1-level factors in long VBG data: ",
          paste(one_level_factors_vbg, collapse = ", "))
  vars_keep_vbg <- setdiff(vars_keep_vbg, one_level_factors_vbg)
}

# 4) Balance with imputation identifiers
fml_vbg <- reformulate(termlabels = vars_keep_vbg, response = "has_vbg")

bal_vbg <- cobalt::bal.tab(
  fml_vbg,
  data        = dlong_vbg,
  weights     = dlong_vbg$.w,
  imp         = dlong_vbg$.imp,   # vector of imputation IDs
  estimand    = "ATE",
  un          = TRUE,
  m.threshold = 0.1
)

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

```
)
```

```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```



2.5 5) 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.

2.5.1 5.1) 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)
}
```

2.5.2 5.2) 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
)
```

```r
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	2.2698495	0.1798092	9.677944	6.803407	13.767014	5	5
niv_proc	has_abg	0.5107385	0.1849957	1.666521	1.159683	2.394873	5	5
death	has_abg	0.8606529	0.1423868	2.364704	1.788853	3.125928	5	5
hcrf	has_abg	1.3772955	0.2083079	3.964166	2.635340	5.963032	5	5

2.5.3 5.3) 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	0.5871228	0.1429037	1.798806	1.359384	2.380270	5	5
niv_proc	has_vbg	0.7636823	0.1928596	2.146164	1.470610	3.132048	5	5
death	has_vbg	0.3027193	0.1497856	1.353535	1.009181	1.815388	5	5
hcrf	has_vbg	0.9253467	0.1917419	2.522743	1.732443	3.673558	5	5

2.6 6) Explainability on one representative imputation

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

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

library(fastshap)
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]
```

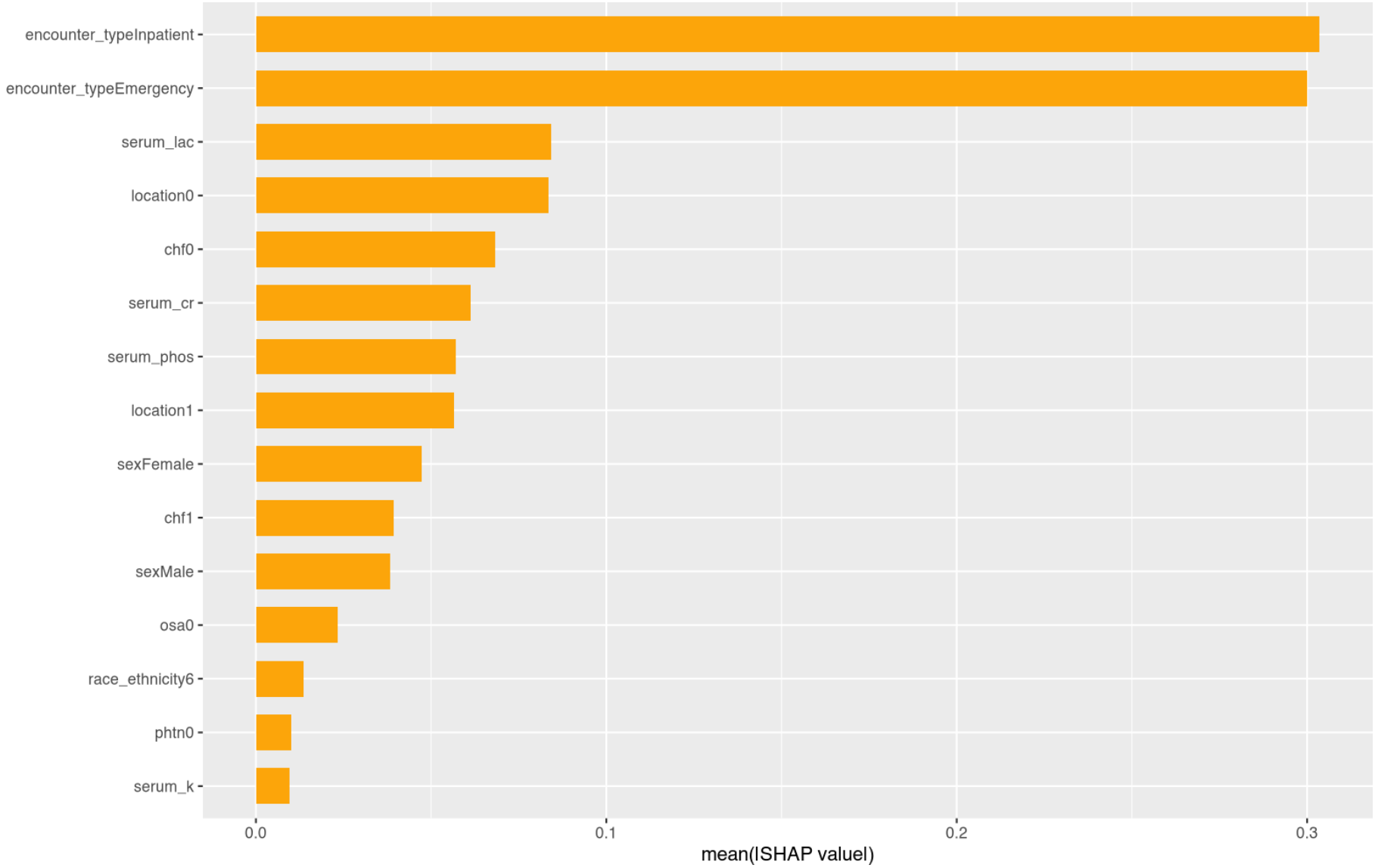
```
    # Case 1: exact column present
    if (v %in% cn) {
      z <- X_raw[[v]]
      out[, i] <- if (is.factor(z)) as.numeric(z) else as.numeric(z)
      next
    }

    # 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
```

```r
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)
}
```

```
# 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)

# --- 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)

# Fast SHAP on link (logit) scale
S_abg <- fastshap::explain(
  object       = bk_abg$gbm,
  X            = bk_abg$X,
  pred_wrapper = bk_abg$pred,
  nsim         = 32,
  adjust       = TRUE
)

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

# Bar importance (top 30)
ord_abg   <- order(colMeans(abs(S_abg), na.rm = TRUE), decreasing = TRUE)
topK_abg  <- colnames(S_abg)[ord_abg[1:min(30, 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]
p_dep_abg <- shapviz::sv_dependence(sv_abg, v = pri_abg, color_var = aux_abg) +
```

```
  ggplot2::geom_smooth(se = FALSE, linewidth = 0.5, method = "loess", formula = y ~ x)
p_dep_abg
```

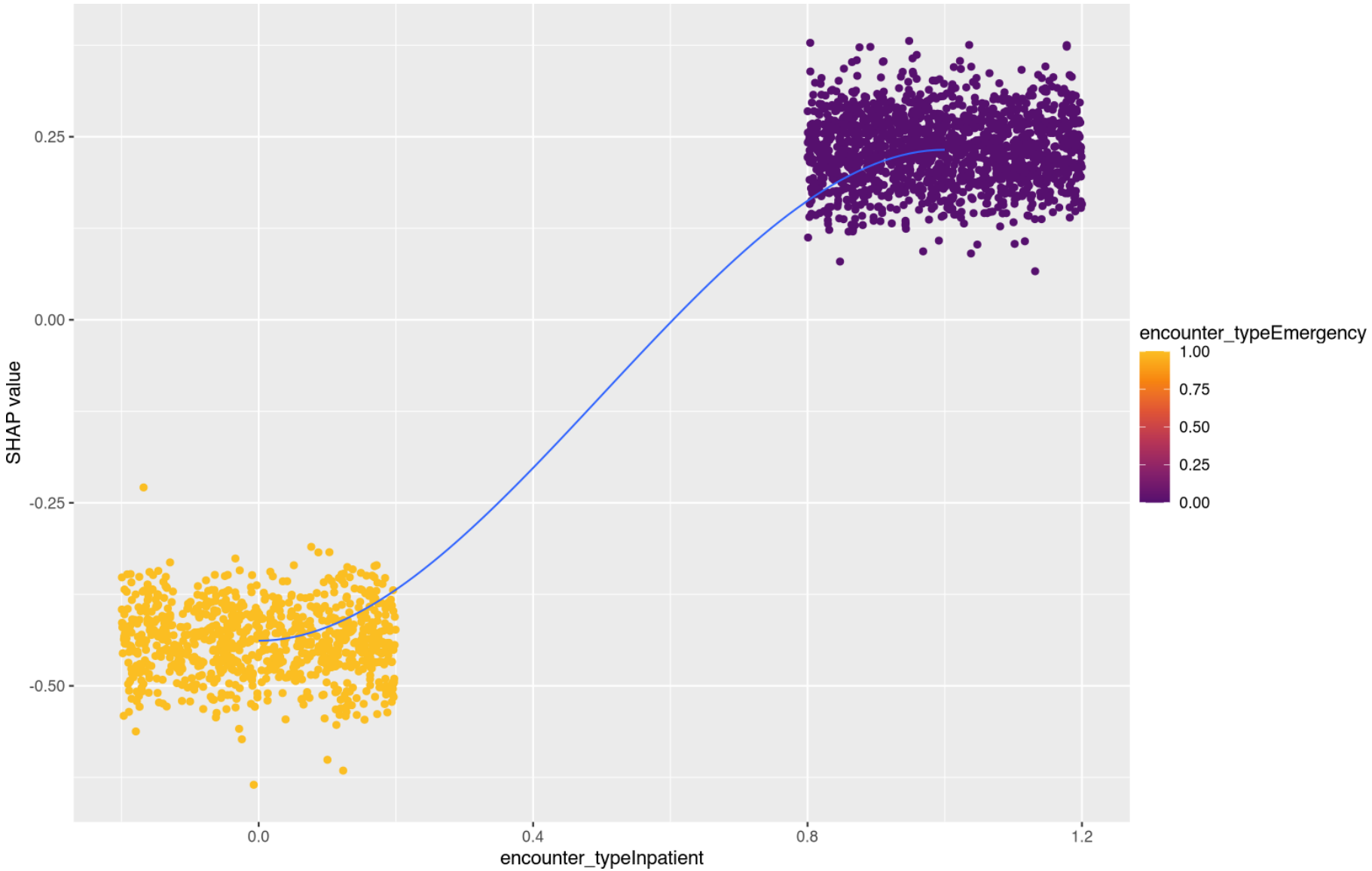
```
Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
: pseudoinverse used at -0.005
```

```
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 2.8434e-30
```

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

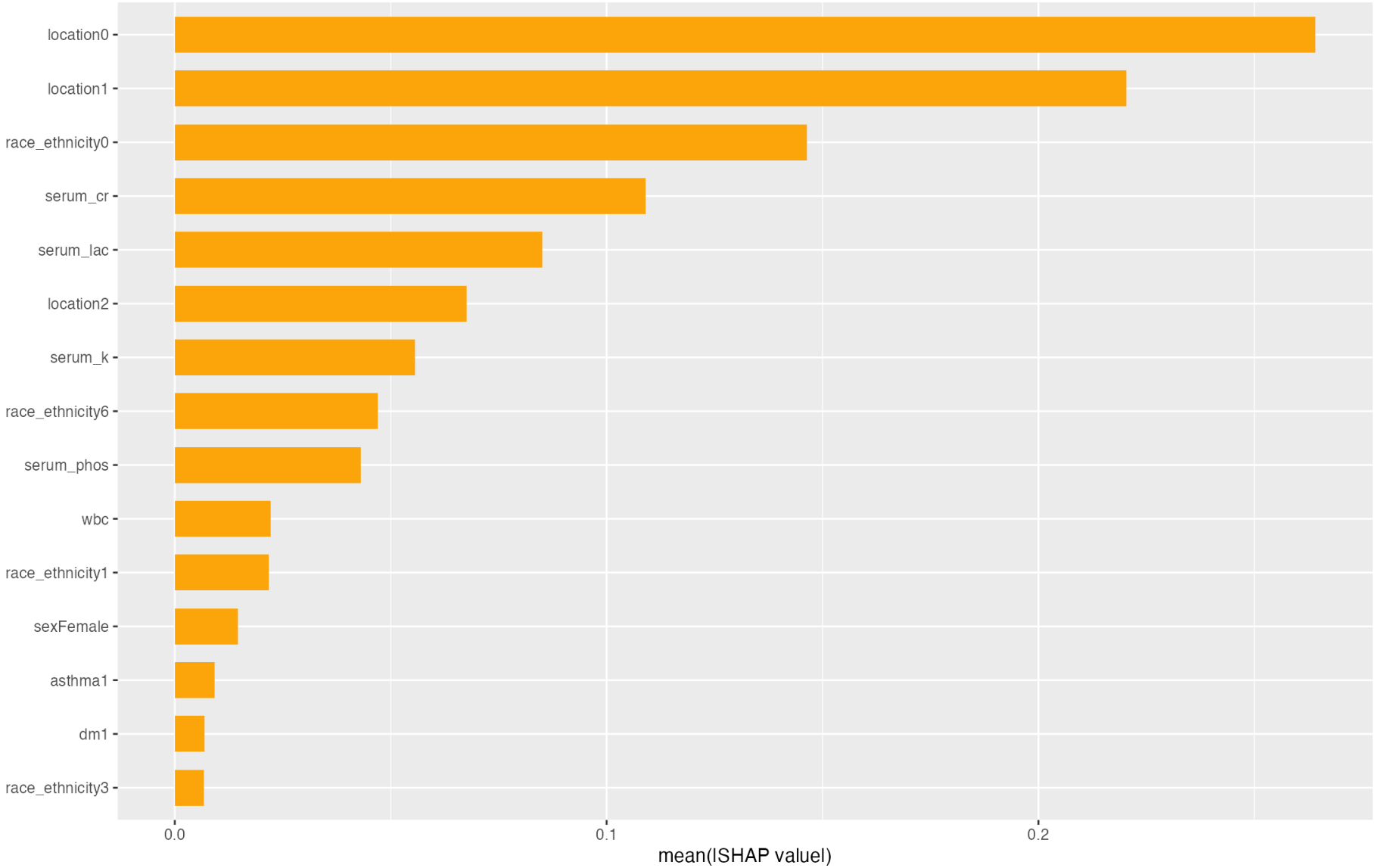
```
Warning: The following aesthetics were dropped during statistical transformation:
colour.
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```




```
# Repeat for VBG

# --- VBG explainability on imputation 1 ----------------------------------------
stopifnot(exists("W_vbg_list"), length(W_vbg_list) >= 1)
```

```r
W1_vbg <- W_vbg_list[[1]]

bk_vbg <- make_shap_backend_any(W1_vbg)

S_vbg <- fastshap::explain(
  object       = bk_vbg$gbm,
  X            = bk_vbg$X,
  pred_wrapper = bk_vbg$pred,
  nsim         = 32,
  adjust       = TRUE
)

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

ord_vbg   <- order(colMeans(abs(S_vbg), na.rm = TRUE), decreasing = TRUE)
topK_vbg  <- colnames(S_vbg)[ord_vbg[1:min(30, ncol(S_vbg))]]
p_bar_vbg <- shapviz::sv_importance(sv_vbg, kind = "bar", v = topK_vbg)
p_bar_vbg
```

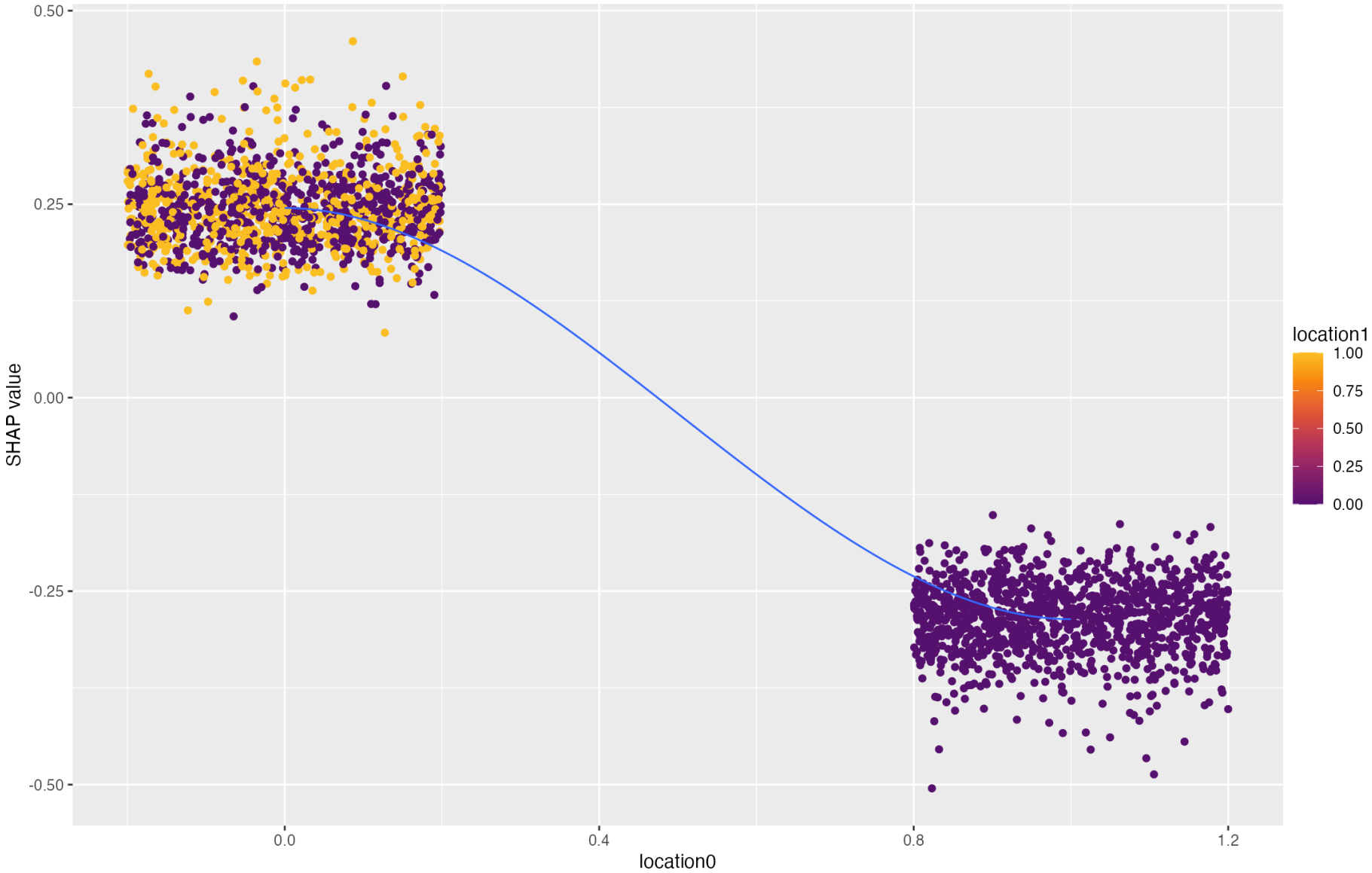
```
Warning: `label` cannot be a <ggplot2::element_blank> object.
```




```
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)
```

```
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 5.4603e-29

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?
```


0.50
0.25
0.00
-0.25
-0.50
SHAP value
0.0
0.4
0.8
1.2
location0
location1
1.00
0.75
0.50
0.25
0.00

2.7 7) Imputed, Weighted, Three-level PCO categories (ABG weighted; VBG weighted)

```
# --- helpers -----------------------------------------------------------
library(splines)
library(mitools)
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[grepl(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("Low","Mid","High")) {
  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 <- if (is.null(fixed_breaks)) stats::quantile(x_ok, probs = c(0, 1/3, 2/3, 1), na.rm = TRUE)
          else fixed_breaks
  brks <- unique(brks)
  if (length(brks) < 4) 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("Low","Mid","High")
ref_label_abg   <- if (exists("ref_label_abg")) ref_label_abg else "Mid"

# (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(" 45", "46-55", " 56")
ref_label_abg    <- "46-55"

use_fixed_vbg <- TRUE
co2_breaks_vbg   <- c(-Inf, 50, 60, Inf)
```

```r
co2_labels_vbg   <- c(" 50", "51-60", " 61")
ref_label_vbg    <- "51-60"

# 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))
}

# Pool a *spline* curve by pooling (b, V) then projecting onto a common grid
pool_spline_curve <- function(fits, terms_obj, grid_df) {
  fits <- Filter(function(m) inherits(m, "svyglm"), fits)
  stopifnot(length(fits) >= 2)

  Xg  <- model.matrix(stats::delete.response(terms_obj), grid_df)
  cn  <- colnames(Xg)

  results <- lapply(fits, function(m) setNames(coef(m)[cn], cn))
  vars    <- lapply(fits, function(m) {
    V <- vcov(m)[cn, cn, drop = FALSE]
    dimnames(V) <- list(cn, cn)
    V
  })
  ok <- which(sapply(results, function(x) all(names(x) == cn)) &
              sapply(vars,    function(V) identical(colnames(V), cn)))
  stopifnot(length(ok) >= 2)

  pooled <- MIcombine(results = results[ok], variances = vars[ok])
  b      <- coef(pooled)
  V      <- pooled$variance

  eta <- as.numeric(Xg %*% b)
  se  <- sqrt(rowSums((Xg %*% V) * Xg))
```

```
  tibble::tibble(
    x   = grid_df[[1L]],
    lp  = eta,
    se  = se,
    p   = stats::plogis(lp),
    lcl = stats::plogis(lp - 1.96 * se),
    ucl = stats::plogis(lp + 1.96 * se)
  )
}
```

2.7.1 ABG, imputed, weighted, 3-level outcome

```
# --- 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]
    d2$co2_cat <- make_co2_cat(
      d2$paco2,
      fixed_breaks = if (isTRUE(use_fixed_abg)) co2_breaks_abg else NULL,
      labels       = co2_labels_abg
    )
    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 }

    w <- W_abg_list[[i]]$weights[g]
    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)
```

2.7.2 VBG, imputed, weighted, 3-level

```
# 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]
```

```
    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
    )
    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 }

    w <- W_vbg_list[[i]]$weights[g]
    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)
```

```
# 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
```

```r
# 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_cat 45 0.1080410 0.2283396 1.114093 0.712123 1.742963
2 co2_cat 56 0.8128419 0.3108706 2.254305 1.225729 4.146018
```

2.8 8) Imputed, Weighted spline PCO (ABG weighted; VBG weighted)

2.8.1 ABG, imputed, weighted, spline outcome

```r
# --- ABG: outcome ~ CO2 category, IPW by W_abg_list -------------------------
fit_abg_cat <- function(outcome_var) {
  fits <- vector("list", length(dlist))
  for (i in seq_along(dlist)) {
    d <- dlist[[i]]
    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 (use_fixed_abg) co2_breaks_abg else NULL,
      labels       = co2_labels_abg
    )
    d2$co2_cat <- stats::relevel(droplevels(d2$co2_cat), ref = ref_label_abg)
    if (nlevels(d2$co2_cat) < 2) { fits[[i]] <- NULL; next }

    w <- W_abg_list[[i]]$weights[g]
    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_pattern = "^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)
abg_cat_results
```

outcome	term	logOR	SE	OR	LCL	UCL
IMV	co2_cat 45	0.1080410	0.2283396	1.1140934	0.7121230	1.7429630
IMV	co2_cat 56	0.8128419	0.3108706	2.2543055	1.2257287	4.1460181
NIV	co2_cat 45	-0.2129738	0.3492749	0.8081773	0.4075659	1.6025643
NIV	co2_cat 56	0.5868588	0.4580352	1.7983307	0.7327944	4.4132342
Death60d	co2_cat 45	0.2698914	0.2756205	1.3098222	0.7631317	2.2481497
Death60d	co2_cat 56	0.4896736	0.3985869	1.6317836	0.7470997	3.5640728
HCRF	co2_cat 45	-1.3467384	0.3009728	0.2600872	0.1441869	0.4691505
HCRF	co2_cat 56	1.0322254	0.3390583	2.8073064	1.4443670	5.4563481

2.8.2 VBG, imputed, weighted, spline outcome

```
# --- ABG: outcome ~ CO2 category, IPW by W_abg_list -------------------------
fit_abg_cat <- function(outcome_var) {
  fits <- vector("list", length(dlist))
  for (i in seq_along(dlist)) {
    d <- dlist[[i]]
    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 (use_fixed_abg) co2_breaks_abg else NULL,
      labels       = co2_labels_abg
    )
    d2$co2_cat <- stats::relevel(droplevels(d2$co2_cat), ref = ref_label_abg)
    if (nlevels(d2$co2_cat) < 2) { fits[[i]] <- NULL; next }

    w <- W_abg_list[[i]]$weights[g]
    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_pattern = "^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)
abg_cat_results
```

outcome	term	logOR	SE	OR	LCL	UCL
IMV	co2_cat 45	0.1080410	0.2283396	1.1140934	0.7121230	1.7429630
IMV	co2_cat 56	0.8128419	0.3108706	2.2543055	1.2257287	4.1460181
NIV	co2_cat 45	-0.2129738	0.3492749	0.8081773	0.4075659	1.6025643
NIV	co2_cat 56	0.5868588	0.4580352	1.7983307	0.7327944	4.4132342
Death60d	co2_cat 45	0.2698914	0.2756205	1.3098222	0.7631317	2.2481497
Death60d	co2_cat 56	0.4896736	0.3985869	1.6317836	0.7470997	3.5640728
HCRF	co2_cat 45	-1.3467384	0.3009728	0.2600872	0.1441869	0.4691505
HCRF	co2_cat 56	1.0322254	0.3390583	2.8073064	1.4443670	5.4563481

2.9 9) Save, export, and session info

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

```r
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] progressr_0.18.0    future.apply_1.20.0 future_1.67.0
 [4] mitools_2.4         skimr_2.2.1         visdat_0.6.0
 [7] naniar_1.1.0        miceadds_3.18-36    mice_3.18.0
[10] shapviz_0.10.2      fastshap_0.1.1      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] codebookr_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] xfun_0.53               chk_0.10.0              mgcv_1.9-3
 [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            UpSetR_1.4.0            tidyr_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         plyr_1.8.9
[109] forcats_1.0.1          fs_1.6.6               stringi_1.8.7
[112] viridisLite_0.4.2      tables_0.9.31          glmnet_4.1-10
[115] bayestestR_0.17.0      quantreg_6.1           fontquiver_0.2.1
[118] hms_1.1.4              rbibutils_2.4          xgboost_1.7.11.1
```