ABG-VBG Analysis

Brian Locke, Anila Mehta

Table of contents

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.

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

# 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()))
```

```r
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

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: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

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")
}
```

```r
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")
}
```

```r
# 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.

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

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

```
set.seed(123)
rows_to_keep <- round(nrow(stata_data) * 0.1)
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 
17285 34275 
```

```
dim(subset_data)
```

```
[1] 51560   546
```

Generating Codebook for the Full Dataset

```
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 
46079  5481 
```

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

```
        0         1 
0.8936967 0.1063033 
```

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1063  0.0000  1.0000 
```

Table 1A and 1B:

```r
# 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

```
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")
```

NEW Table 1

```
# Status factors (column labels are taken from factor levels)
subset_data <- subset_data %>%
  mutate(
```

Variable	**No ABG** N = 32,809[1]	**ABG_NoHypercapnia** N = 12,966[1]	**ABG_Hypercapnia** N =
Age (years)	58.2 ± 18.0; 0.0/32,809.0 missing (0.0%)	60.9 ± 17.1; 0.0/12,966.0 missing (0.0%)	62.0 ± 16.4; 0.0/5,785.0 missin
Current BMI kg/m2	32.3 ± 8.8; 18,425.0/32,809.0 missing (56.2%)	28.6 ± 6.9; 7,599.0/12,966.0 missing (58.6%)	29.8 ± 7.7; 3,380.0/5,785.0 missi
sex_label			
Female	16,938 (52%)	5,868 (45%)	2,684 (46%)
Male	15,871 (48%)	7,098 (55%)	3,101 (54%)
race_ethnicity_label			
White	19,890 (61%)	8,189 (63%)	4,006 (69%)
Black or African American	6,257 (19%)	1,933 (15%)	785 (14%)
Hispanic	2,363 (7.2%)	755 (5.8%)	285 (4.9%)
Asian	496 (1.5%)	281 (2.2%)	67 (1.2%)
American Indian	190 (0.6%)	177 (1.4%)	30 (0.5%)
Pacific Islander	43 (0.1%)	8 (<0.1%)	6 (0.1%)
Unknown	3,570 (11%)	1,623 (13%)	606 (10%)
location_label			
South	13,925 (42%)	7,181 (55%)	3,235 (56%)
Northeast	9,231 (28%)	2,308 (18%)	1,342 (23%)
Midwest	2,331 (7.1%)	1,071 (8.3%)	493 (8.5%)
West	7,322 (22%)	2,406 (19%)	715 (12%)
osa_label	5,991 (18%)	1,695 (13%)	1,173 (20%)
asthma_label	4,802 (15%)	1,269 (9.8%)	814 (14%)
copd_label	5,937 (18%)	2,124 (16%)	1,880 (32%)
chf_label	5,990 (18%)	2,542 (20%)	1,652 (29%)
nmd_label	1,141 (3.5%)	591 (4.6%)	240 (4.1%)
phtn_label	2,379 (7.3%)	1,064 (8.2%)	745 (13%)
ckd_label	5,573 (17%)	2,531 (20%)	1,187 (21%)
diabetes_label	9,196 (28%)	3,704 (29%)	1,818 (31%)
encounter_type_label			
Emergency	14,304 (44%)	1,963 (15%)	1,018 (18%)
Inpatient	18,505 (56%)	11,003 (85%)	4,767 (82%)
VBG PCO2	45.3 ± 10.4; 23,389.0/32,809.0 missing (71.3%)	42.2 ± 11.3; 9,251.0/12,966.0 missing (71.3%)	57.9 ± 19.0; 4,050.0/5,785.0 miss
Arterial PCO2	NA ± NA; 32,809.0/32,809.0 missing (100.0%)	35.4 ± 6.1; 0.0/12,966.0 missing (0.0%)	58.4 ± 19.6; 0.0/5,785.0 missin

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

Variable	**No VBG** N = 36,690[1]	**VBG_NoHypercapnia** N = 10,498[1]	**VBG_Hypercapnia** N =
Age (years)	59.4 ± 17.8; 0.0/36,690.0 missing (0.0%)	58.2 ± 17.7; 0.0/10,498.0 missing (0.0%)	60.9 ± 16.8; 0.0/4,372.0 missin
Current BMI kg/m2	31.8 ± 8.6; 19,381.0/36,690.0 missing (52.8%)	28.5 ± 7.1; 6,916.0/10,498.0 missing (65.9%)	29.2 ± 7.9; 3,107.0/4,372.0 missi
sex_label			
Female	18,538 (51%)	4,894 (47%)	2,058 (47%)
Male	18,152 (49%)	5,604 (53%)	2,314 (53%)
race_ethnicity_label			
White	24,215 (66%)	5,414 (52%)	2,456 (56%)
Black or African American	6,196 (17%)	1,949 (19%)	830 (19%)
Hispanic	2,268 (6.2%)	870 (8.3%)	265 (6.1%)
Asian	535 (1.5%)	229 (2.2%)	80 (1.8%)
American Indian	194 (0.5%)	178 (1.7%)	25 (0.6%)
Pacific Islander	49 (0.1%)	7 (<0.1%)	1 (<0.1%)
Unknown	3,233 (8.8%)	1,851 (18%)	715 (16%)
location_label			
South	19,761 (54%)	3,096 (29%)	1,484 (34%)
Northeast	6,559 (18%)	4,398 (42%)	1,924 (44%)
Midwest	2,542 (6.9%)	899 (8.6%)	454 (10%)
West	7,828 (21%)	2,105 (20%)	510 (12%)
osa_label	6,470 (18%)	1,523 (15%)	866 (20%)
asthma_label	4,921 (13%)	1,320 (13%)	644 (15%)
copd_label	7,039 (19%)	1,580 (15%)	1,322 (30%)
chf_label	6,932 (19%)	2,009 (19%)	1,243 (28%)
nmd_label	1,420 (3.9%)	378 (3.6%)	174 (4.0%)
phtn_label	2,766 (7.5%)	845 (8.0%)	577 (13%)
ckd_label	6,224 (17%)	2,152 (20%)	915 (21%)
diabetes_label	9,983 (27%)	3,323 (32%)	1,412 (32%)
encounter_type_label			
Emergency	12,612 (34%)	3,397 (32%)	1,276 (29%)
Inpatient	24,078 (66%)	7,101 (68%)	3,096 (71%)
VBG PCO2	NA ± NA; 36,690.0/36,690.0 missing (100.0%)	40.1 ± 6.6; 0.0/10,498.0 missing (0.0%)	60.3 ± 12.8; 0.0/4,372.0 missin
Arterial PCO2	42.2 ± 15.2; 23,389.0/36,690.0 missing (63.7%)	38.5 ± 15.1; 6,856.0/10,498.0 missing (65.3%)	52.4 ± 19.1; 2,564.0/4,372.0 miss

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

```r
    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
```

NEW Table 2

```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}%)"
    ),
```

	Table 1		
Variable	Everyone[1]	**Did not get ABG** N = 32,809[1]	**Did get ABG** N = 18
Age (years)	59.3 ± 17.7; 0.0/51,560.0 missing (0.0%)	58.2 ± 18.0; 0.0/32,809.0 missing (0.0%)	61.3 ± 16.9; 0.0/18,751.0 mis
Current BMI kg/m2	31.1 ± 8.4; 29,404.0/51,560.0 missing (57.0%)	32.3 ± 8.8; 18,425.0/32,809.0 missing (56.2%)	29.0 ± 7.2; 10,979.0/18,751.0 m
sex_label			
Female	25,490 (49%)	16,938 (52%)	8,552 (46%)
Male	26,070 (51%)	15,871 (48%)	10,199 (54%)
race_ethnicity_label			
White	32,085 (62%)	19,890 (61%)	12,195 (65%)
Black or African American	8,975 (17%)	6,257 (19%)	2,718 (14%)
Hispanic	3,403 (6.6%)	2,363 (7.2%)	1,040 (5.5%)
Asian	844 (1.6%)	496 (1.5%)	348 (1.9%)
American Indian	397 (0.8%)	190 (0.6%)	207 (1.1%)
Pacific Islander	57 (0.1%)	43 (0.1%)	14 (<0.1%)
Unknown	5,799 (11%)	3,570 (11%)	2,229 (12%)
location_label			
South	24,341 (47%)	13,925 (42%)	10,416 (56%)
Northeast	12,881 (25%)	9,231 (28%)	3,650 (19%)
Midwest	3,895 (7.6%)	2,331 (7.1%)	1,564 (8.3%)
West	10,443 (20%)	7,322 (22%)	3,121 (17%)
osa_label	8,859 (17%)	5,991 (18%)	2,868 (15%)
asthma_label	6,885 (13%)	4,802 (15%)	2,083 (11%)
copd_label	9,941 (19%)	5,937 (18%)	4,004 (21%)
chf_label	10,184 (20%)	5,990 (18%)	4,194 (22%)
nmd_label	1,972 (3.8%)	1,141 (3.5%)	831 (4.4%)
phtn_label	4,188 (8.1%)	2,379 (7.3%)	1,809 (9.6%)
ckd_label	9,291 (18%)	5,573 (17%)	3,718 (20%)
diabetes_label	14,718 (29%)	9,196 (28%)	5,522 (29%)
encounter_type_label			
Emergency	17,285 (34%)	14,304 (44%)	2,981 (16%)
Inpatient	34,275 (66%)	18,505 (56%)	15,770 (84%)
VBG PCO2	46.0 ± 12.8; 36,690.0/51,560.0 missing (71.2%)	45.3 ± 10.4; 23,389.0/32,809.0 missing (71.3%)	47.2 ± 16.0; 13,301.0/18,751.0 m
Arterial PCO2	42.5 ± 16.0; 32,809.0/51,560.0 missing (63.6%)	NA ± NA; 32,809.0/32,809.0 missing (100.0%)	42.5 ± 16.0; 0.0/18,751.0 mis

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

```r
    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") %>%
  body_add_flextable(ft2)

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

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

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

```
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)
```

Table 1

Variable	**Got ABG & Hypercapnia**[1]	**Got ABG & No hypercapnia**[1]	**Got VBG & Hypercapni**
Age (years)	62.0 ± 16.4; 0.0/5,785.0 missing (0.0%)	60.9 ± 17.1; 0.0/12,966.0 missing (0.0%)	60.9 ± 16.8; 0.0/4,372.0 missing
Current BMI kg/m2	29.8 ± 7.7; 3,380.0/5,785.0 missing (58.4%)	28.6 ± 6.9; 7,599.0/12,966.0 missing (58.6%)	29.2 ± 7.9; 3,107.0/4,372.0 missing
sex_label			
Female	2,684 (46%)	5,868 (45%)	2,058 (47%)
Male	3,101 (54%)	7,098 (55%)	2,314 (53%)
race_ethnicity_label			
White	4,006 (69%)	8,189 (63%)	2,456 (56%)
Black or African American	785 (14%)	1,933 (15%)	830 (19%)
Hispanic	285 (4.9%)	755 (5.8%)	265 (6.1%)
Asian	67 (1.2%)	281 (2.2%)	80 (1.8%)
American Indian	30 (0.5%)	177 (1.4%)	25 (0.6%)
Pacific Islander	6 (0.1%)	8 (<0.1%)	1 (<0.1%)
Unknown	606 (10%)	1,623 (13%)	715 (16%)
location_label			
South	3,235 (56%)	7,181 (55%)	1,484 (34%)
Northeast	1,342 (23%)	2,308 (18%)	1,924 (44%)
Midwest	493 (8.5%)	1,071 (8.3%)	454 (10%)
West	715 (12%)	2,406 (19%)	510 (12%)
osa_label	1,173 (20%)	1,695 (13%)	866 (20%)
asthma_label	814 (14%)	1,269 (9.8%)	644 (15%)
copd_label	1,880 (32%)	2,124 (16%)	1,322 (30%)
chf_label	1,652 (29%)	2,542 (20%)	1,243 (28%)
nmd_label	240 (4.1%)	591 (4.6%)	174 (4.0%)
phtn_label	745 (13%)	1,064 (8.2%)	577 (13%)
ckd_label	1,187 (21%)	2,531 (20%)	915 (21%)
diabetes_label	1,818 (31%)	3,704 (29%)	1,412 (32%)
encounter_type_label			
Emergency	1,018 (18%)	1,963 (15%)	1,276 (29%)
Inpatient	4,767 (82%)	11,003 (85%)	3,096 (71%)
VBG PCO2	57.9 ± 19.0; 4,050.0/5,785.0 missing (70.0%)	42.2 ± 11.3; 9,251.0/12,966.0 missing (71.3%)	60.3 ± 12.8; 0.0/4,372.0 missing
Arterial PCO2	58.4 ± 19.6; 0.0/5,785.0 missing (0.0%)	35.4 ± 6.1; 0.0/12,966.0 missing (0.0%)	52.4 ± 19.1; 2,564.0/4,372.0 missin

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

```
Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -2.27756    0.01609 -141.52   <2e-16 ***
hypercap_on_abg  1.24405    0.03394   36.65   <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: 36163  on 51559  degrees of freedom
Residual deviance: 34987  on 51558  degrees of freedom
AIC: 34991

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)	0.1025338	0.0160933	-141.52272	0	0.0993367	0.1058053
hypercap_on_abg	3.4696199	0.0339428	36.65126	0	3.2457090	3.7076382

```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)     -2.86448    0.02069 -138.44   <2e-16 ***
hypercap_on_abg  1.19903    0.04148   28.91   <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: 25001  on 51559  degrees of freedom
Residual deviance: 24289  on 51558  degrees of freedom
AIC: 24293

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)	0.0570129	0.0206904	-138.44511	0	0.0547336	0.059358
hypercap_on_abg	3.3168968	0.0414801	28.90616	0	3.0567651	3.596549

```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)     -2.23931    0.01585 -141.32   <2e-16 ***
hypercap_on_abg  0.76080    0.03734   20.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34928  on 51559  degrees of freedom
Residual deviance: 34557  on 51558  degrees of freedom
AIC: 34561

Number of Fisher Scoring iterations: 5
```

```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)	0.1065316	0.0158457	-141.32029	0	0.1032604	0.1098779
hypercap_on_abg	2.1399945	0.0373424	20.37373	0	1.9882450	2.3016930

```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)     -3.34239    0.02574 -129.87   <2e-16 ***
hypercap_on_abg  2.10203    0.04069   51.66   <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: 22106  on 51559  degrees of freedom
Residual deviance: 19789  on 51558  degrees of freedom
AIC: 19793

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)	0.0353524	0.0257374	-129.86515	0	0.0335999	0.0371669
hypercap_on_abg	8.1827599	0.0406902	51.65931	0	7.5550710	8.8616708

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

```r
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})",
```

	Intubated	NIV	Death	ICD Hyper
hypercap_on_abg	3.47 (3.25, 3.71)	3.32 (3.06, 3.60)	2.14 (1.99, 2.30)	8.18 (7.56, 8.86)

```
  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...")
```

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

Unweighted VBG Group

```
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)     -2.14597    0.01503 -142.74   <2e-16 ***
hypercap_on_vbg  0.70088    0.04132   16.96   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 36163  on 51559  degrees of freedom
Residual deviance: 35906  on 51558  degrees of freedom
AIC: 35910

Number of Fisher Scoring iterations: 4
```

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

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.1169551	0.0150345	-142.73616	0	0.1135458	0.1204391
hypercap_on_vbg	2.0155299	0.0413246	16.96041	0	1.8577805	2.1845114

```
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)     -2.73523    0.01925 -142.12   <2e-16 ***
```

```
hypercap_on_vbg  0.71954    0.05075   14.18   <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: 25001  on 51559  degrees of freedom
Residual deviance: 24826  on 51558  degrees of freedom
AIC: 24830

Number of Fisher Scoring iterations: 5
```

```r
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)	0.0648794	0.0192454	-142.12338	0	0.0624646	0.0673595
hypercap_on_vbg	2.0534976	0.0507456	14.17945	0	1.8573842	2.2662526

```r
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)     -2.17362    0.01520 -143.001   <2e-16 ***
hypercap_on_vbg  0.44832    0.04487    9.991   <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: 34928  on 51559  degrees of freedom
Residual deviance: 34837  on 51558  degrees of freedom
AIC: 34841

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)	0.1137651	0.0152000	-143.000903	0	0.1104125	0.1171918
hypercap_on_vbg	1.5656743	0.0448706	9.991324	0	1.4328498	1.7084439

```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)     -3.17064    0.02341 -135.43   <2e-16 ***
hypercap_on_vbg  1.90252    0.04339   43.84   <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: 22106  on 51559  degrees of freedom
Residual deviance: 20538  on 51558  degrees of freedom
AIC: 20542

Number of Fisher Scoring iterations: 6
```

```r
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)	0.0419767	0.0234120	-135.42817	0	0.040081	0.0439337
hypercap_on_vbg	6.7027596	0.0433921	43.84488	0	6.154710	7.2960164

Display model coefficients for binary hypercapnia on VBG logistic regression

```r
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
```

	Intubated	NIV	Death	ICD Hyper
hypercap_on_vbg	2.02 (1.86, 2.18)	2.05 (1.86, 2.27)	1.57 (1.43, 1.71)	6.70 (6.15, 7.30)

```
  output        = "gt"
) |>
  gt_pdf(title = "Odds Ratios for VBG Hypercapnia (>45 mmHg)'s association with...")
```

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

Calculated ABG from VBG Using Farkas equation - binary predictor

```
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 4730  249
               1 1439  254
```

```r
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)      -1.96138    0.04310 -45.506  < 2e-16 ***
hypercapnia_calc  0.53170    0.07515   7.075  1.5e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5428.9  on 6671  degrees of freedom
Residual deviance: 5380.9  on 6670  degrees of freedom
  (44888 observations deleted due to missingness)
AIC: 5384.9

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)	0.1406644	0.0431015	-45.506018	0	0.1291515	0.1529287
hypercapnia_calc	1.7018173	0.0751526	7.074897	0	1.4676332	1.9705834

```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)      -2.94423    0.06502  -45.28   <2e-16 ***
hypercapnia_calc  1.20986    0.09412   12.85   <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: 3567.7  on 6671  degrees of freedom
Residual deviance: 3408.6  on 6670  degrees of freedom
  (44888 observations deleted due to missingness)
AIC: 3412.6

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)	0.0526427	0.0650157	-45.28485	0	0.0462285	0.059655
hypercapnia_calc	3.3530090	0.0941222	12.85413	0	2.7880974	4.033003

```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)      -1.79059    0.04048 -44.231   <2e-16 ***
hypercapnia_calc  0.10629    0.07818   1.359    0.174
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5557.4  on 6671  degrees of freedom
Residual deviance: 5555.6  on 6670  degrees of freedom
  (44888 observations deleted due to missingness)
AIC: 5559.6

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)	0.166862	0.0404827	-44.230930	0.0000000	0.1540165	0.1805077
hypercapnia_calc	1.112142	0.0781839	1.359463	0.1739999	0.9528989	1.2947767

```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)      -3.20003    0.07306  -43.80   <2e-16 ***
hypercapnia_calc  2.22299    0.09116   24.39   <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: 4297.3  on 6671  degrees of freedom
Residual deviance: 3632.4  on 6670  degrees of freedom
  (44888 observations deleted due to missingness)
AIC: 3636.4

Number of Fisher Scoring iterations: 6
```

```r
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)	0.0407609	0.0730564	-43.80224	0	0.035209	0.0468914
hypercapnia_calc	9.2349051	0.0911596	24.38570	0	7.736516	11.0614527

	Intubated	NIV	Death	ICD Hyper
hypercapnia_calc	1.70 (1.47, 1.97)	3.35 (2.79, 4.03)	1.11 (0.95, 1.29)	9.23 (7.74, 11.06)

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

```r
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...")
```

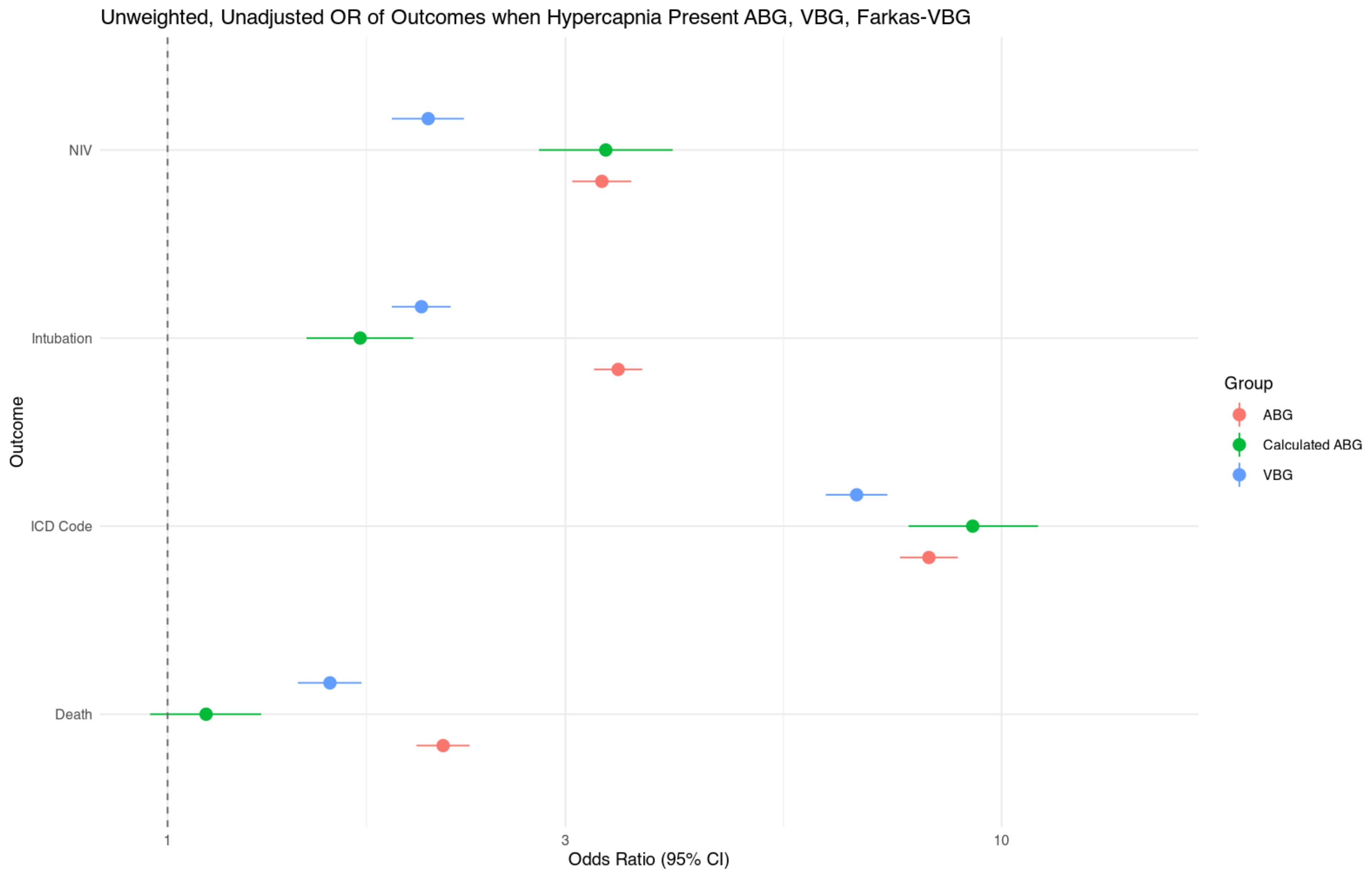
Odds Ratio Graph of all 3 simple, binary-predictor logistic regressions

```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 ---
```

```r
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

# --- 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

# --- Calculated ABG Models ---
calc_intub <- tidy_with_labels(glm(imv_proc ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculated ABG", "Int
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", "De
calc_icd   <- tidy_with_labels(glm(hypercap_resp_failure ~ hypercapnia_calc, data = subset_data, family = binomial), "Calculat

# --- 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
)
```

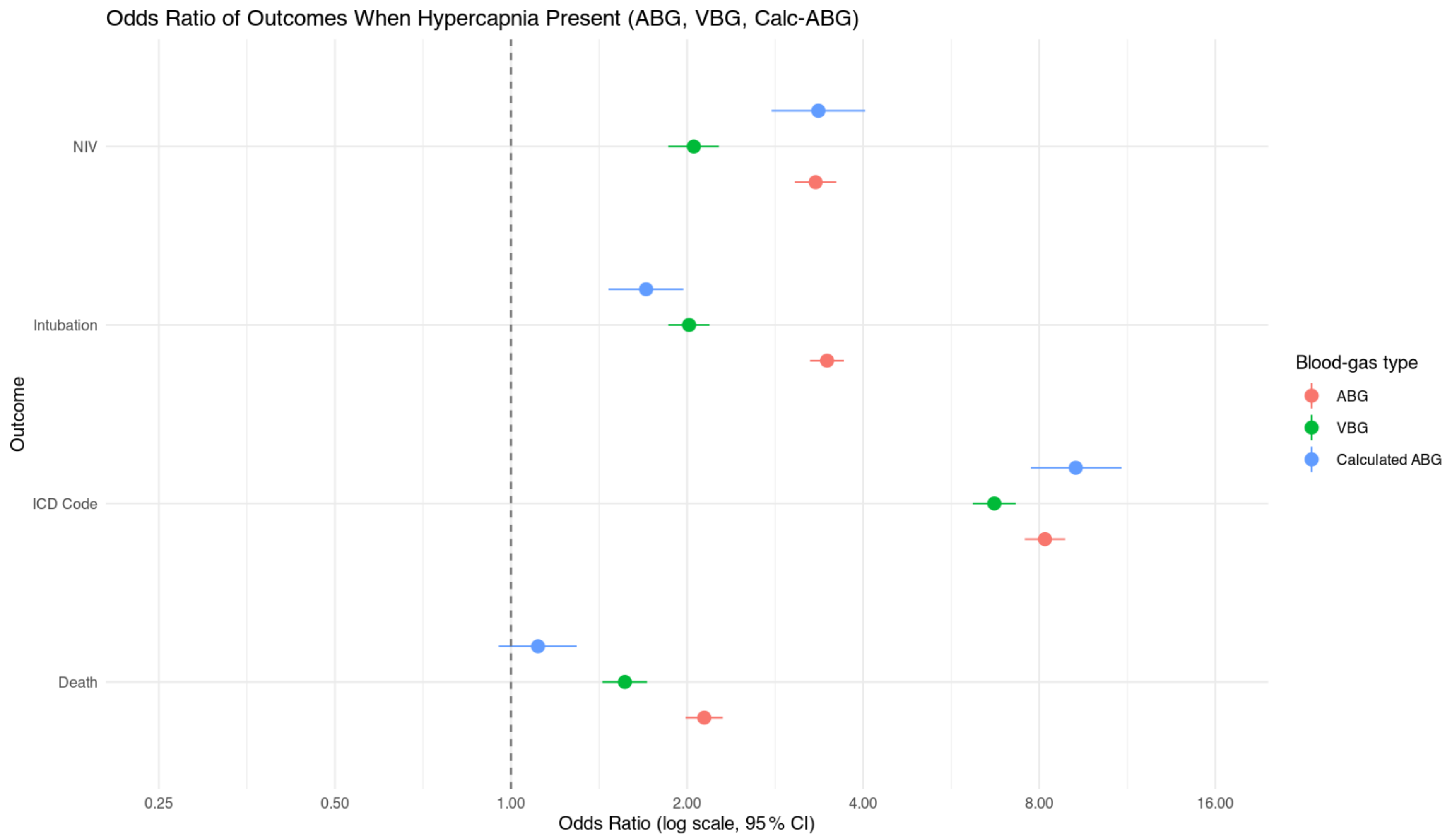
```r
ggplot(combined_or_df, aes(x = outcome, y = estimate, ymin = conf.low, ymax = conf.high, color = group)) +
  geom_pointrange(position = position_dodge(width = 0.5), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", color = "gray40") +
  coord_flip() +
  labs(
    title = "Unweighted, 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
```



```r
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))
```



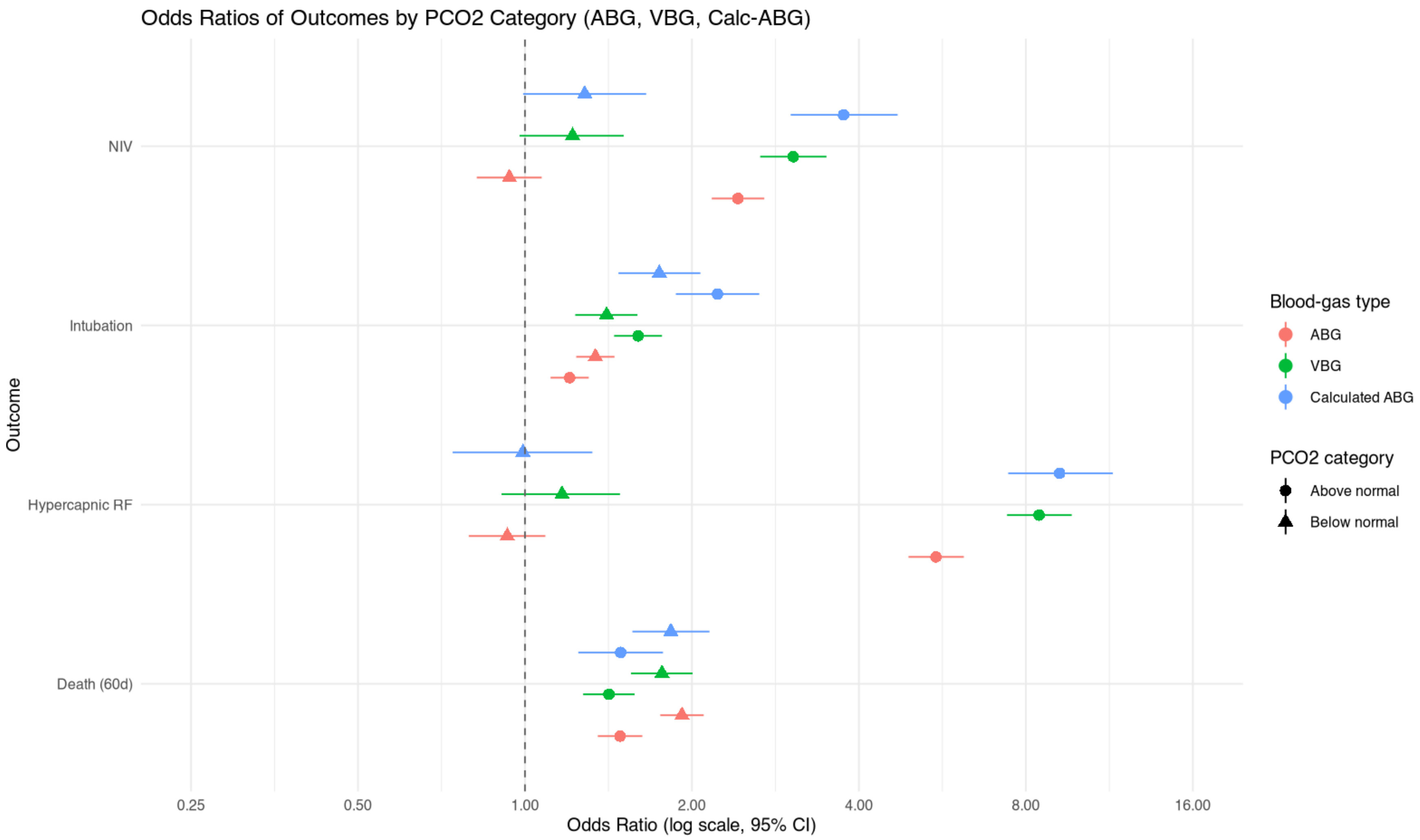
Odds ratios are computed *within* each blood-gas cohort.
Numerator = patients who received that blood-gas and **had** hypercapnia;
denominator = patients who received the same blood-gas and **did not** have hypercapnia.
Because the underlying cohorts differ (ABG, VBG, Calculated ABG),
denominators are not identical across groups.

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 < 35 ~ "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
    )
}

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")
)
```

```r
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))
```



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.

Restricted Cubic Spline Regressions

```r
# 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")
```

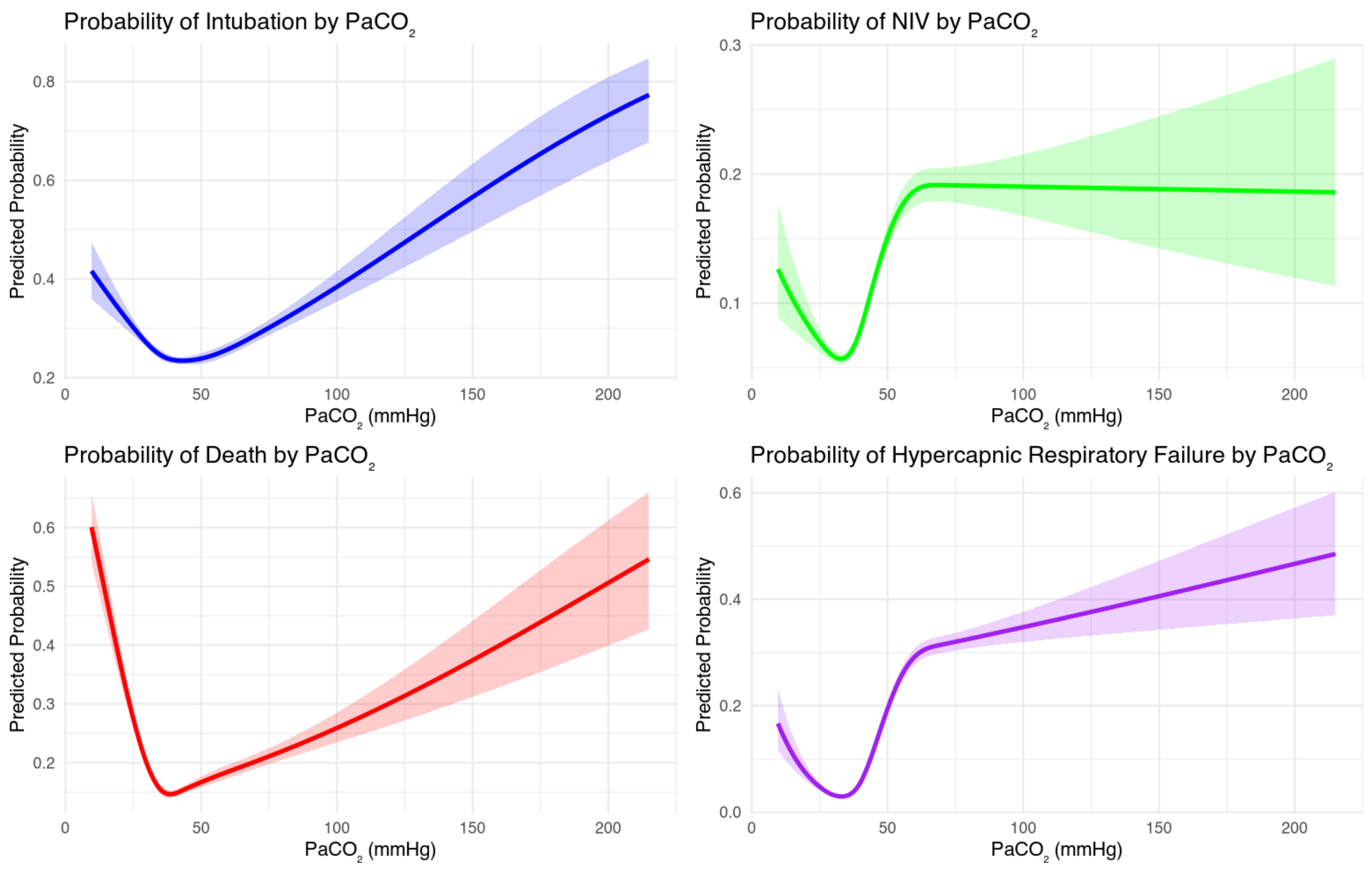
Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```r
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.
```

```r
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()
```

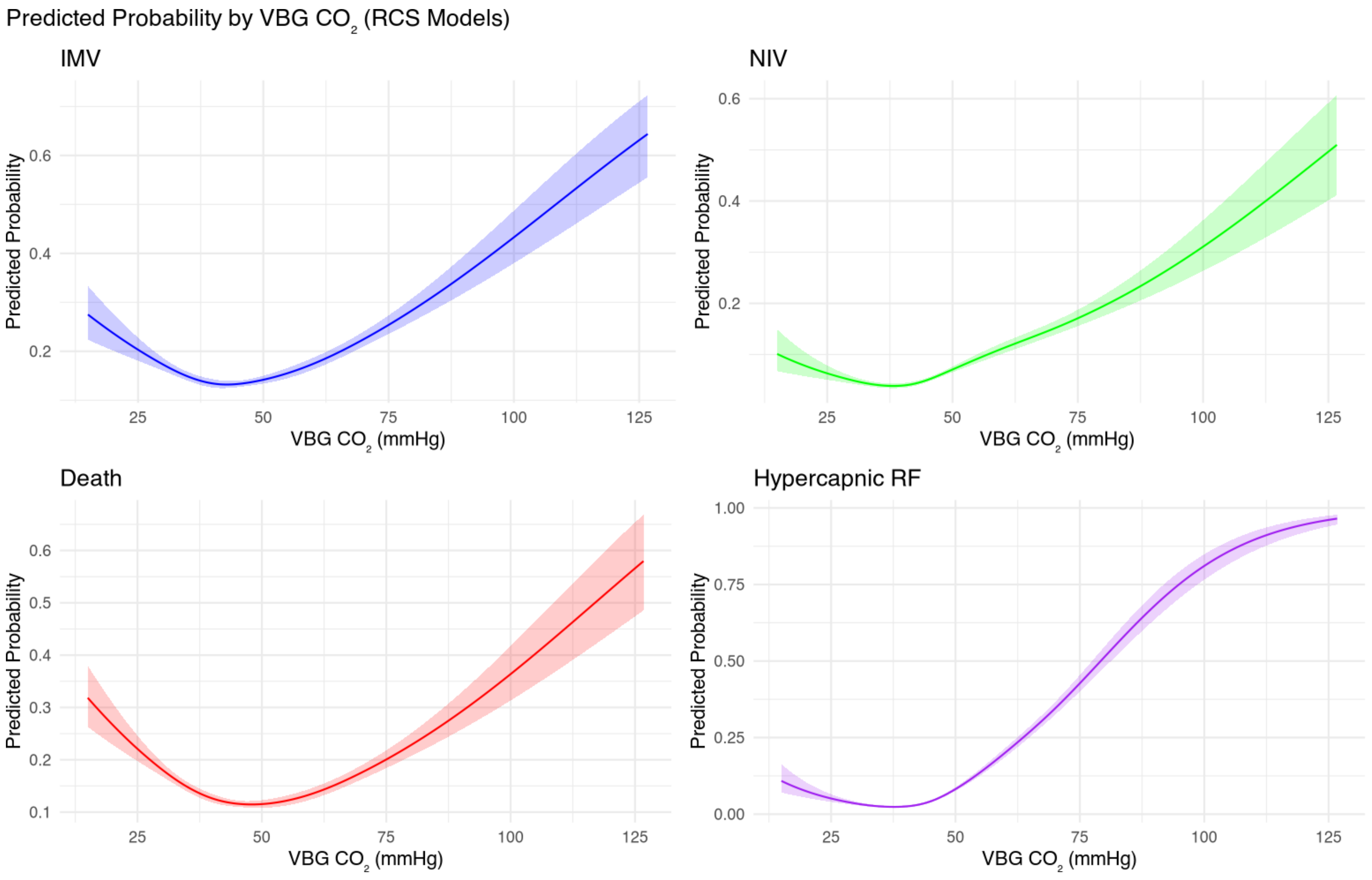
```r
fit_death <- lrm(death_60d ~ rcs(paco2, 4), data = subset_data_abg)
pred_death <- as.data.frame(Predict(fit_death, paco2, fun = plogis))

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

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

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

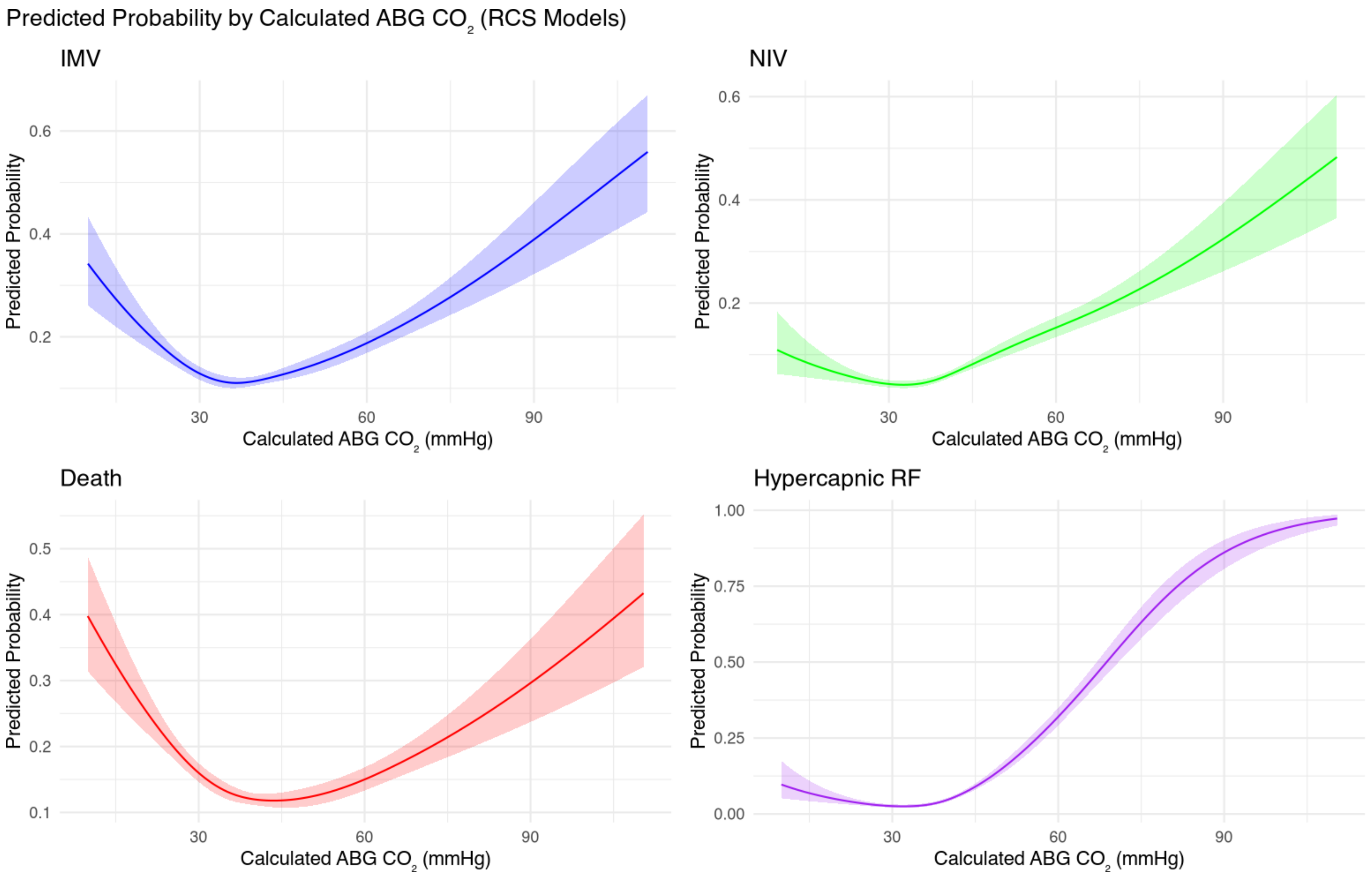
```r
subset_data_vbg <- subset_data %>%
  select(vbg_co2, imv_proc, niv_proc, death_60d, hypercap_resp_failure) %>%
  filter(!is.na(vbg_co2) & complete.cases(.))

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()

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)

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

```r
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)")
```



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.

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

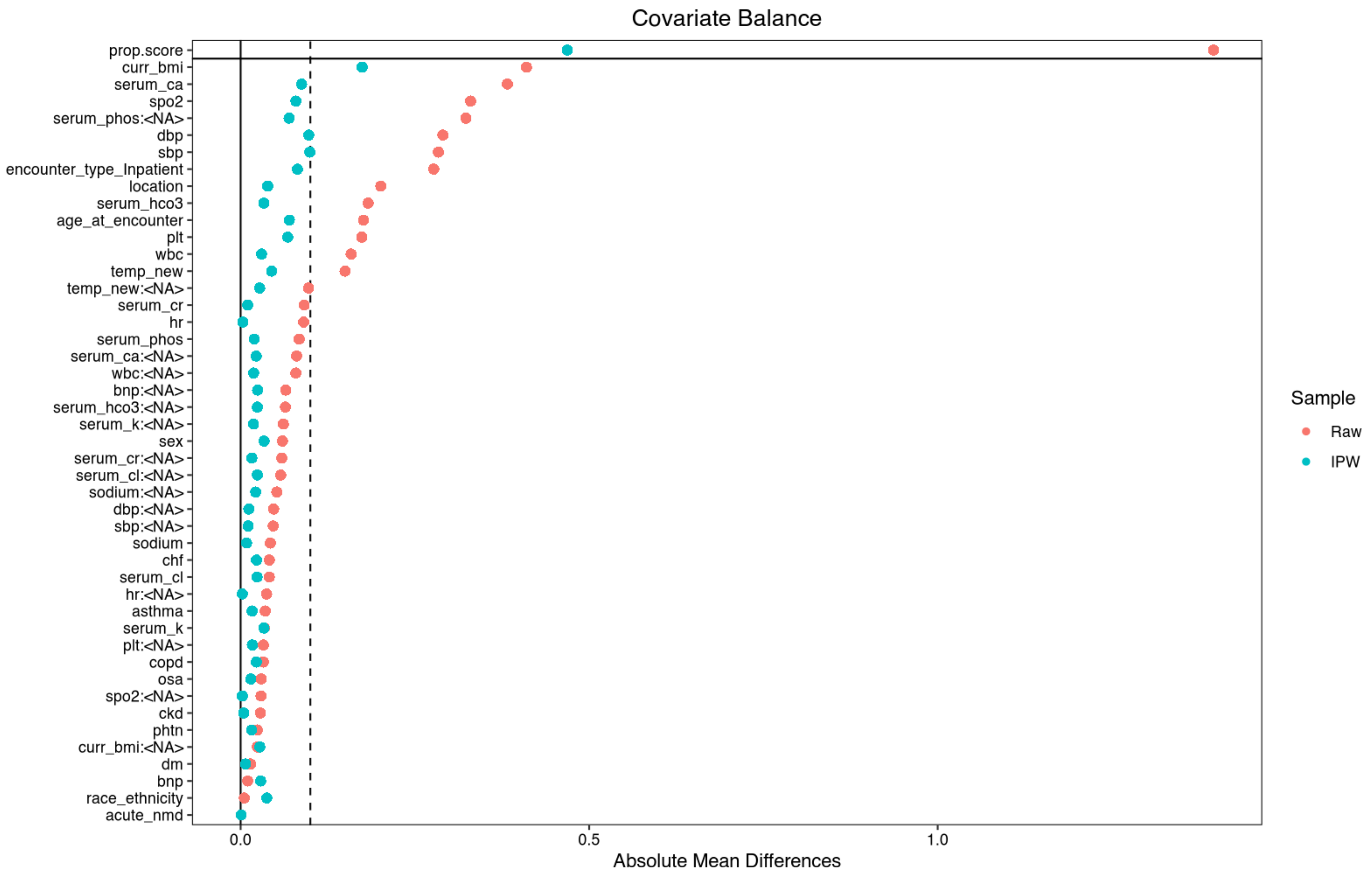
**Removed lactate from weights, decreased n.trees, increased bagging

```r
#   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 +
  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.
```

```r
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.
```



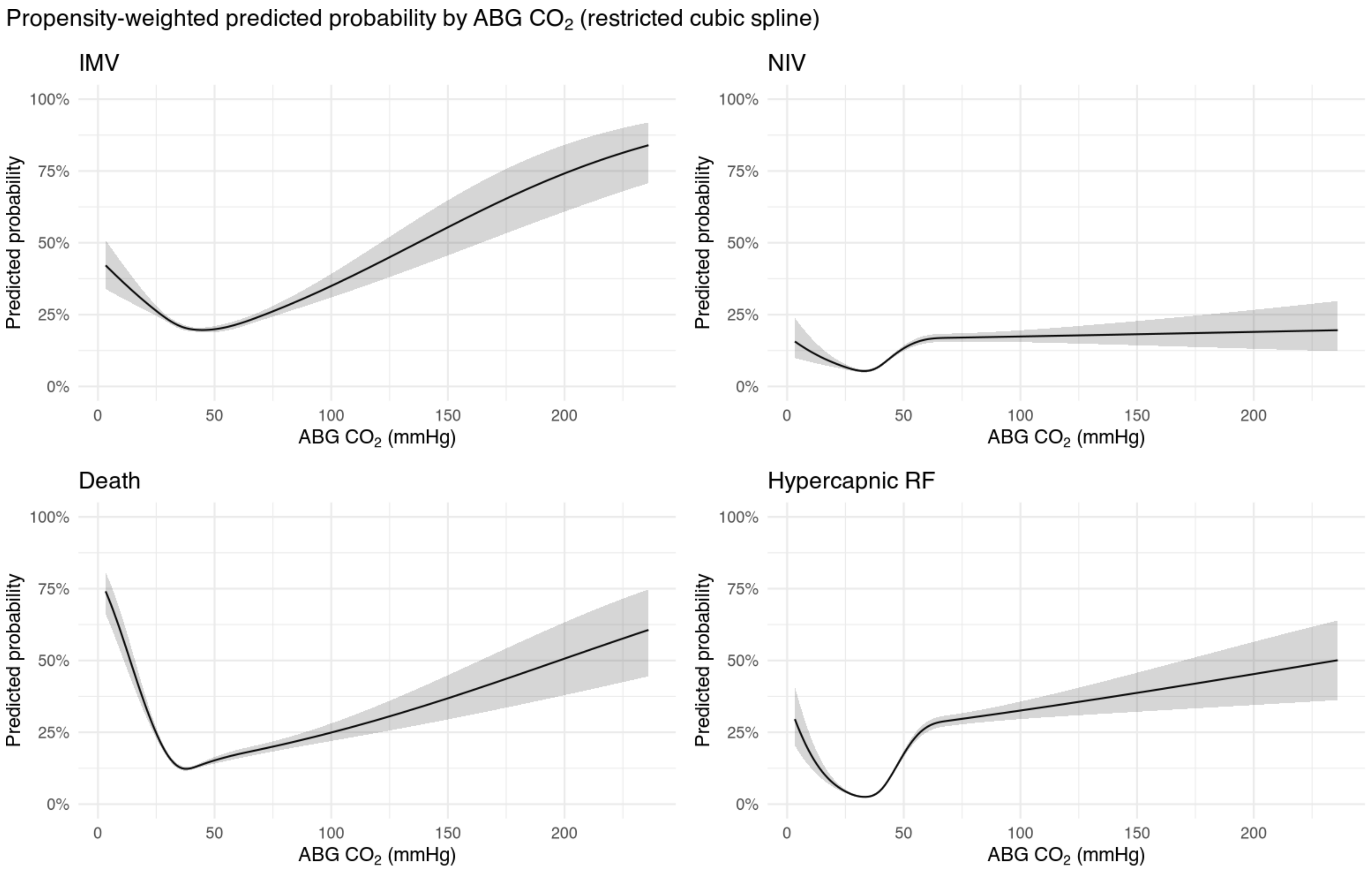
```r
#   4. survey design with the same weights
design <- svydesign(ids = ~1, weights = ~w_abg, data = subset_data)

#   5. outcome models (examples)
fit_niv   <- svyglm(niv_proc   ~ has_abg, design = design, family = quasibinomial())
fit_imv   <- svyglm(imv_proc   ~ has_abg, design = design, family = quasibinomial())
fit_death <- svyglm(death_60d       ~ has_abg, design = design, family = quasibinomial())
fit_icd   <- svyglm(hypercap_resp_failure ~ has_abg, design = design, family = quasibinomial())

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

```
$IMV
OR.has_abg        LCL        UCL 
  6.700189   6.180409   7.263684 

$NIV
OR.has_abg        LCL        UCL 
  1.831955   1.691254   1.984361 

$Death
OR.has_abg        LCL        UCL 
  2.135949   1.996133   2.285557 

$ICD
OR.has_abg        LCL        UCL 
  3.074365   2.807480   3.366622 
```

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

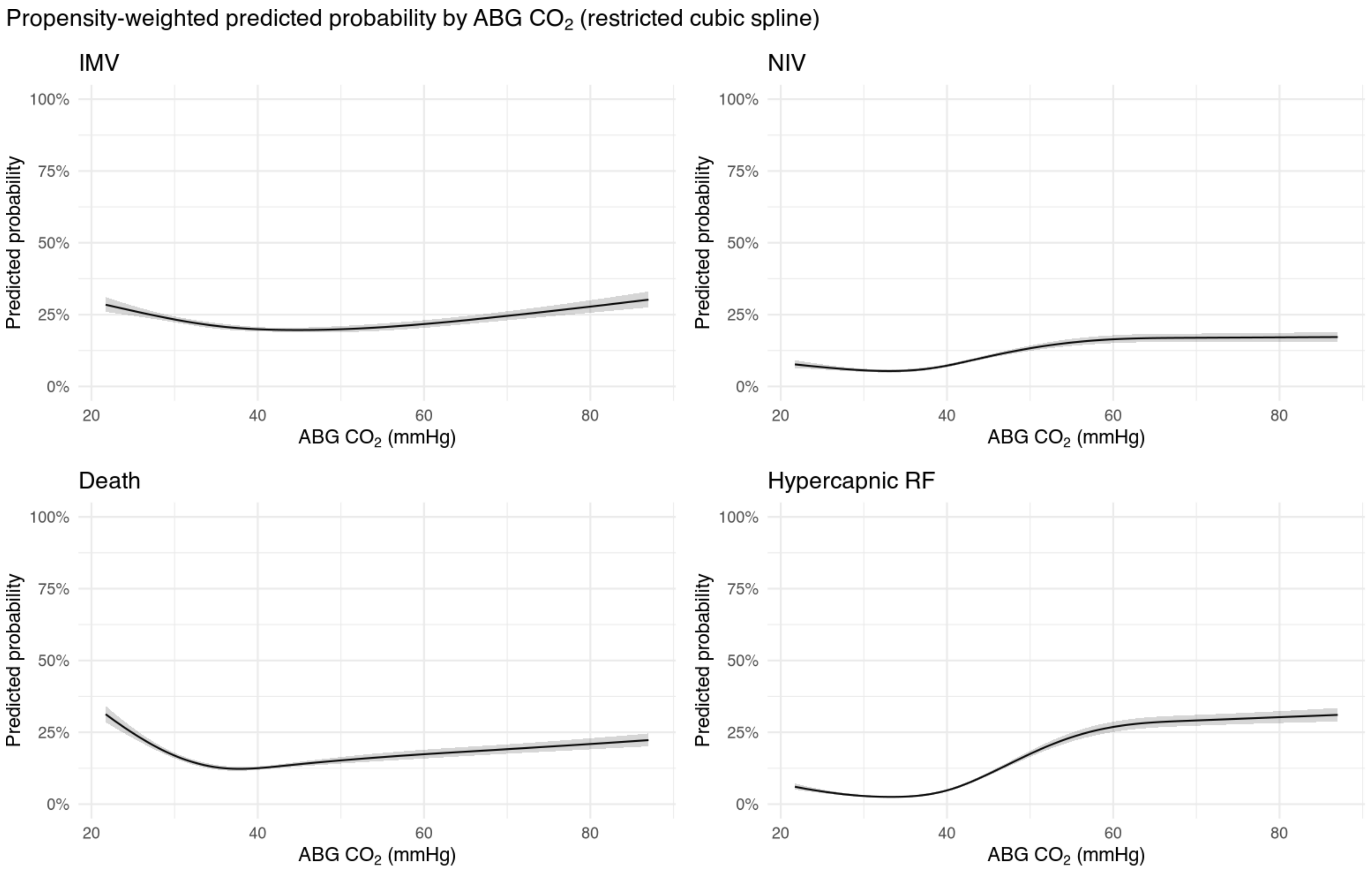
```
# set.seed(42)  # reproducible GBM fit
#
# #   1. inverse-probability weights for receiving an ABG
#
# # done in the last block, so not needed
#

#   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(
```

```r
    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

```r
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)

  # 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

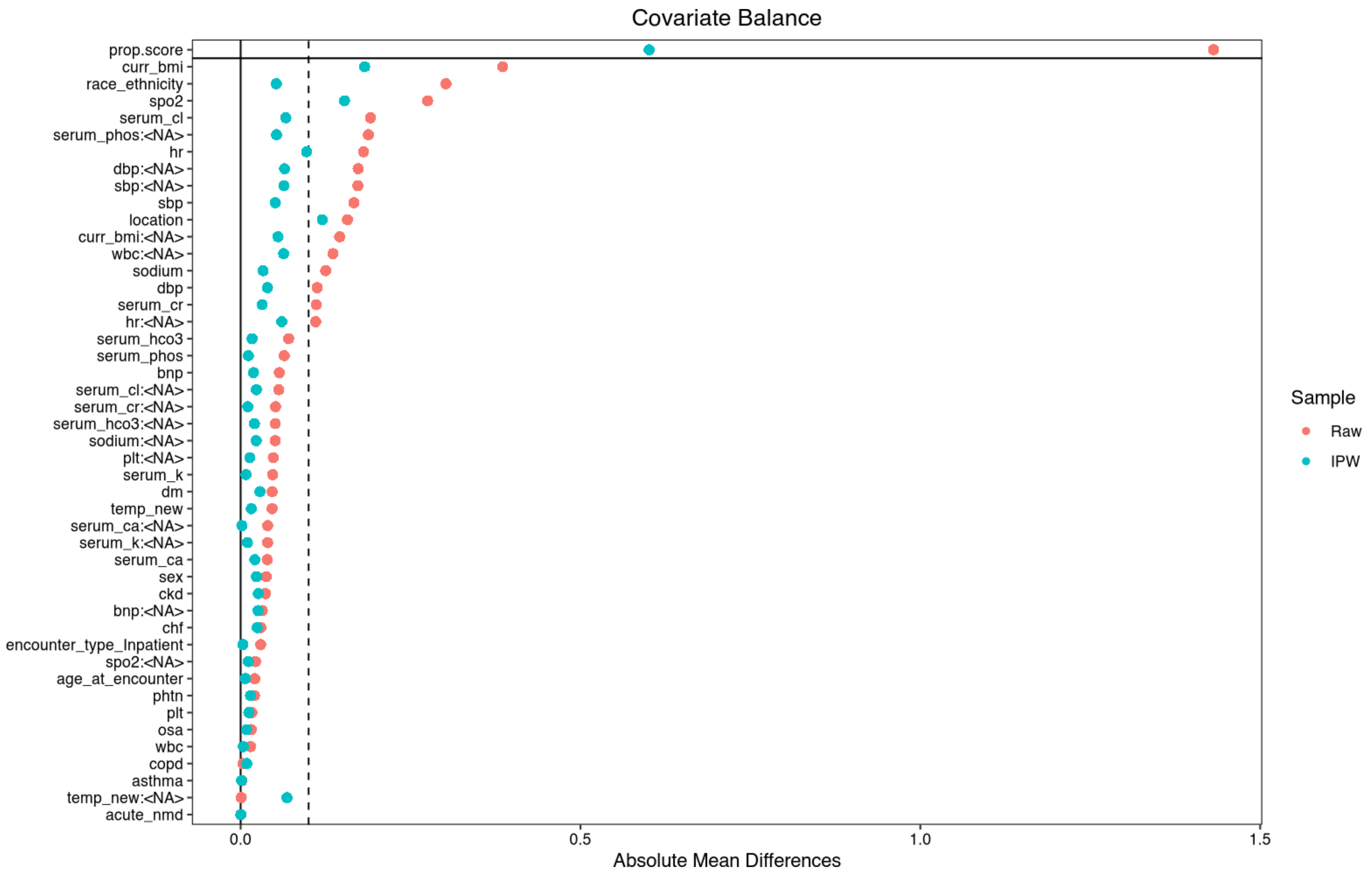
```r
#  Inverse-propensity weighting & outcome modelling for **VBG** cohort
#    - mirrored 1-to-1 to the validated ABG workflow

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

```r
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")
)
```

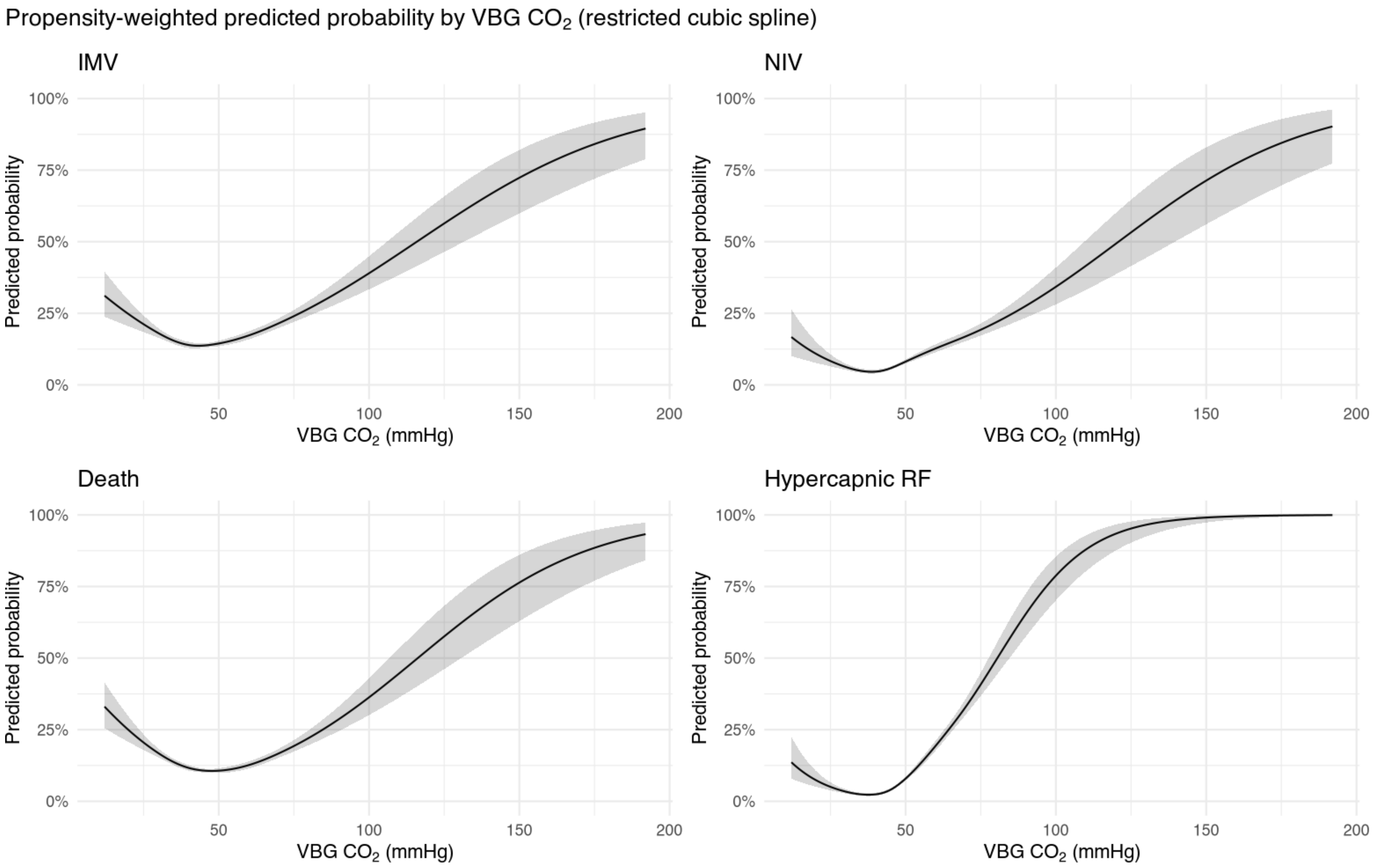
```
Warning: Standardized mean differences and raw mean differences are present in
the same plot. Use the `stars` argument to distinguish between them and
appropriately label the x-axis. See `?love.plot` for details.
```



```r
# 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),
```

```r
    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

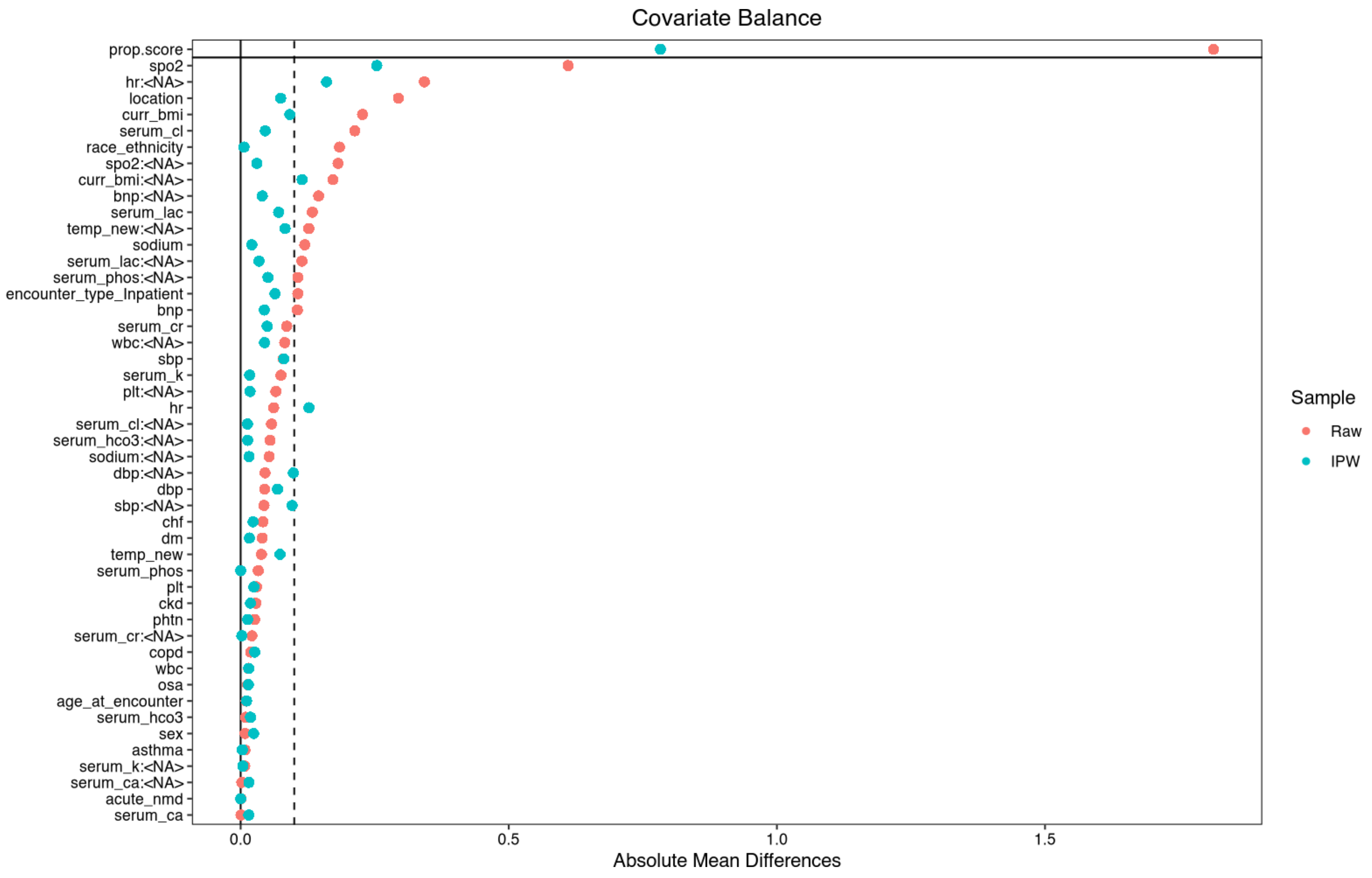
```r
#  Propensity-weighted spline models for **Calculated ABG CO **
#    (weights still derive from propensity to receive a VBG)

#   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 + ph
  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()
)
```

```r
#   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
```

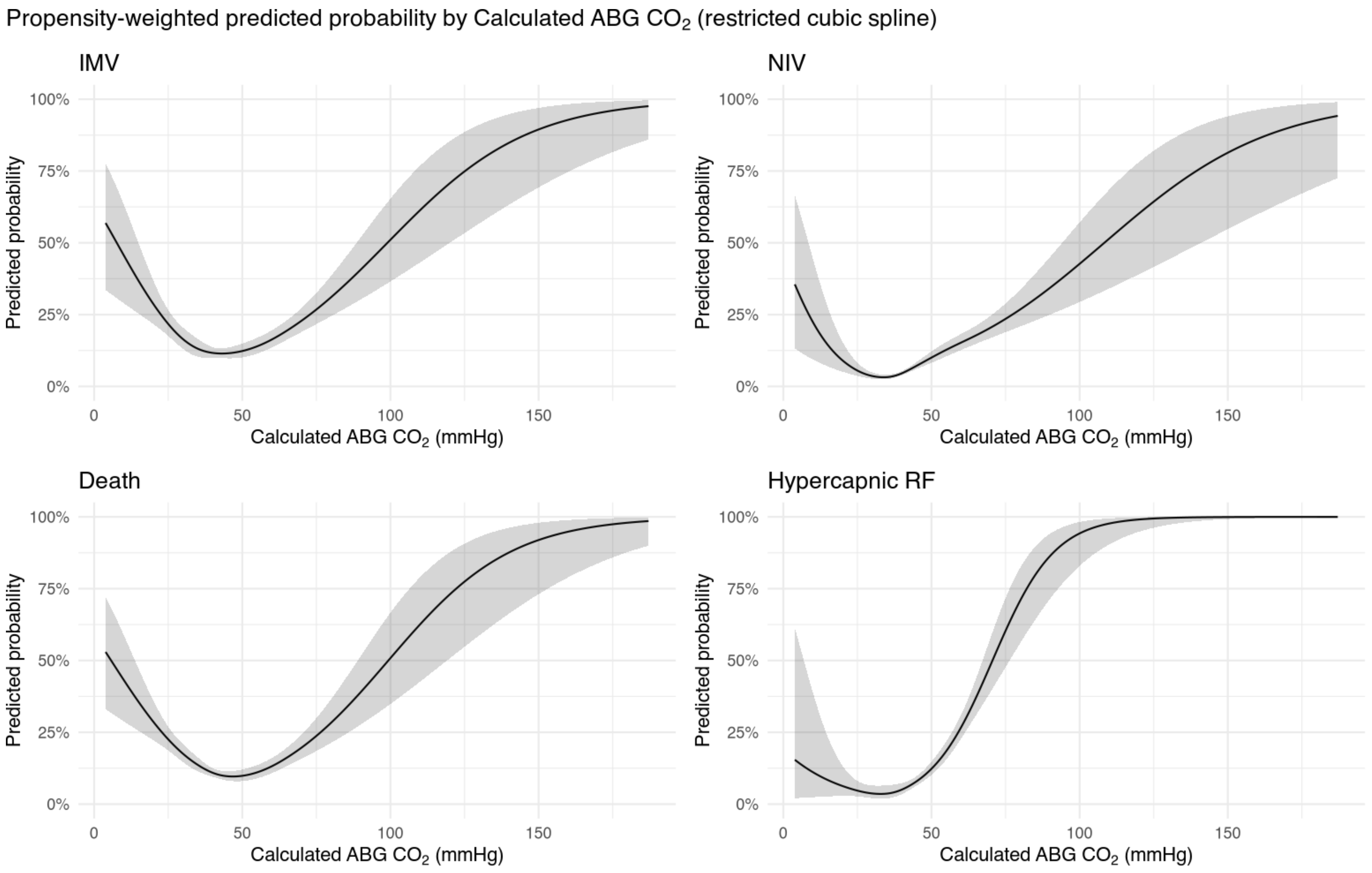
```
Warning: Missing values exist in the covariates. Displayed values omit these
observations.
```

```r
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.
```



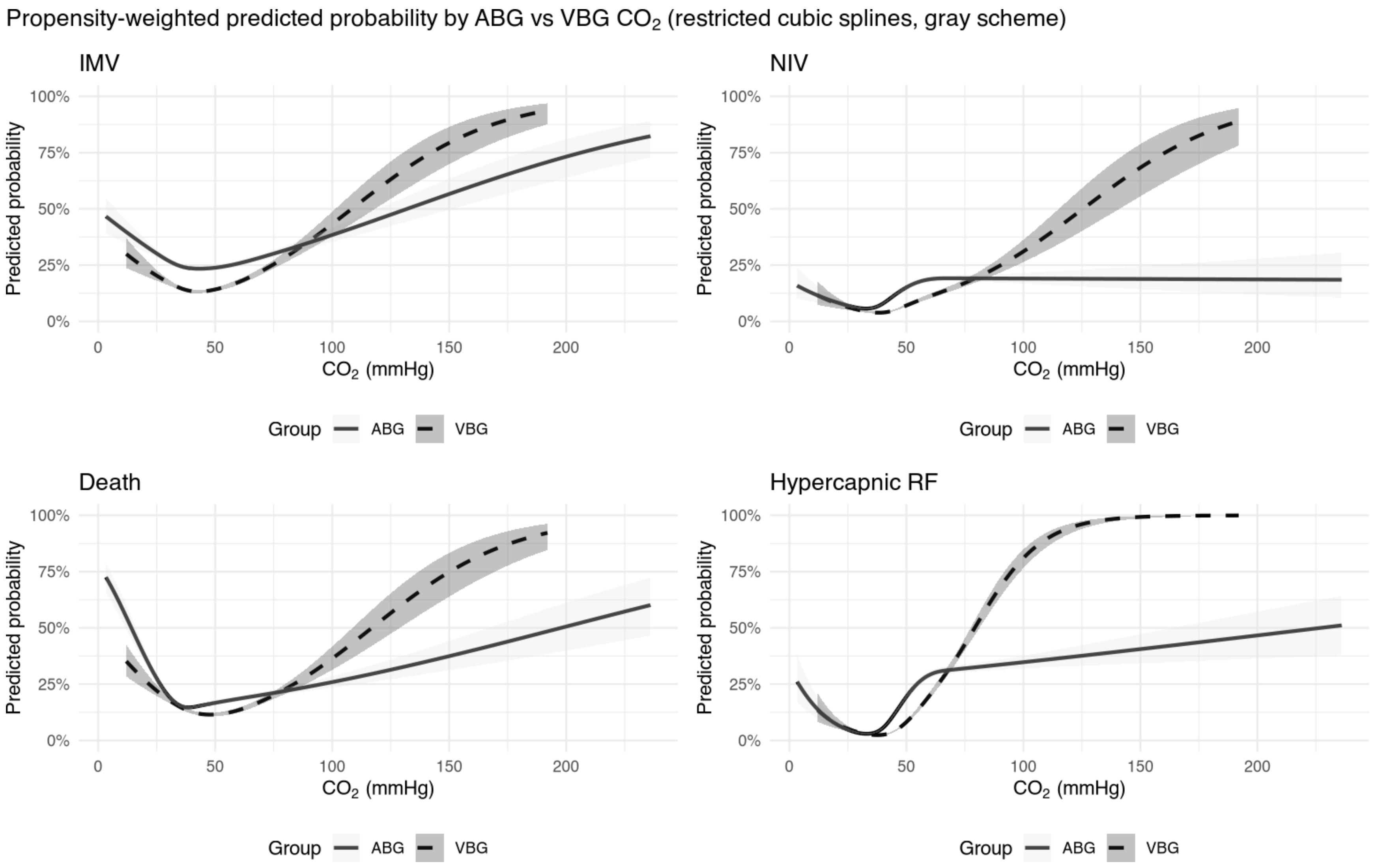
```r
# 2. Analysis sample: rows with a calculated ABG CO ---------------------------
subset_data_calc <- subset_data %>%
  filter(!is.na(calc_abg)) %>%                        # implies has_vbg == 1
  select(calc_abg, imv_proc, niv_proc, death_60d,
         hypercap_resp_failure, w_vbg_calc) %>%
  filter(complete.cases(.))

# 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)
  )
}

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

```r
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")
```

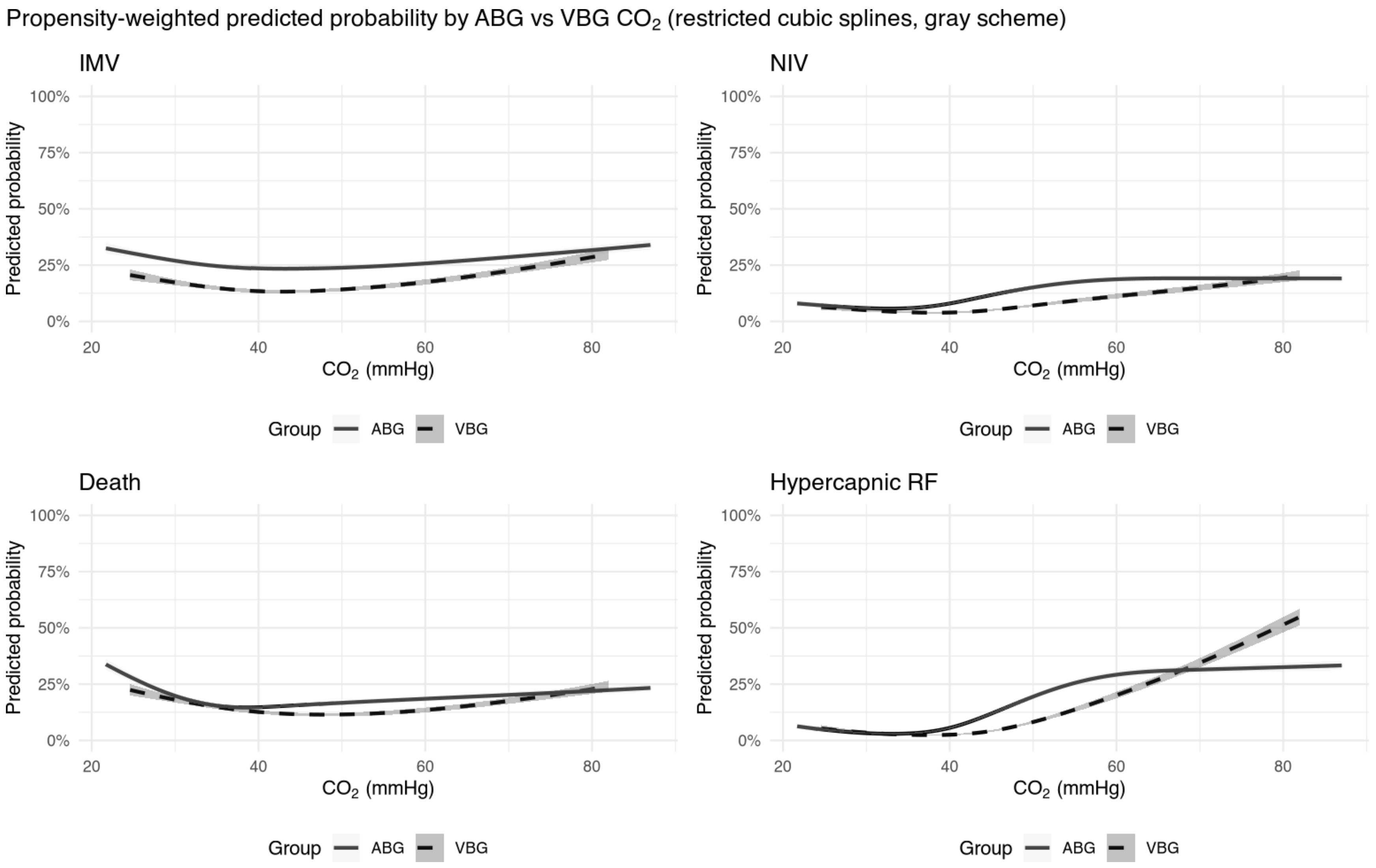
```r
# ABG
pred_imv_abg   <- mkpred(fit_imv_abg,   subset_data_abg, "paco2", "ABG")
pred_niv_abg   <- mkpred(fit_niv_abg,   subset_data_abg, "paco2", "ABG")
pred_death_abg <- mkpred(fit_death_abg, subset_data_abg, "paco2", "ABG")
pred_hcrf_abg  <- mkpred(fit_hcrf_abg,  subset_data_abg, "paco2", "ABG")

# 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)

```r
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")
```

```r
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)")
    )
  )
```



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).

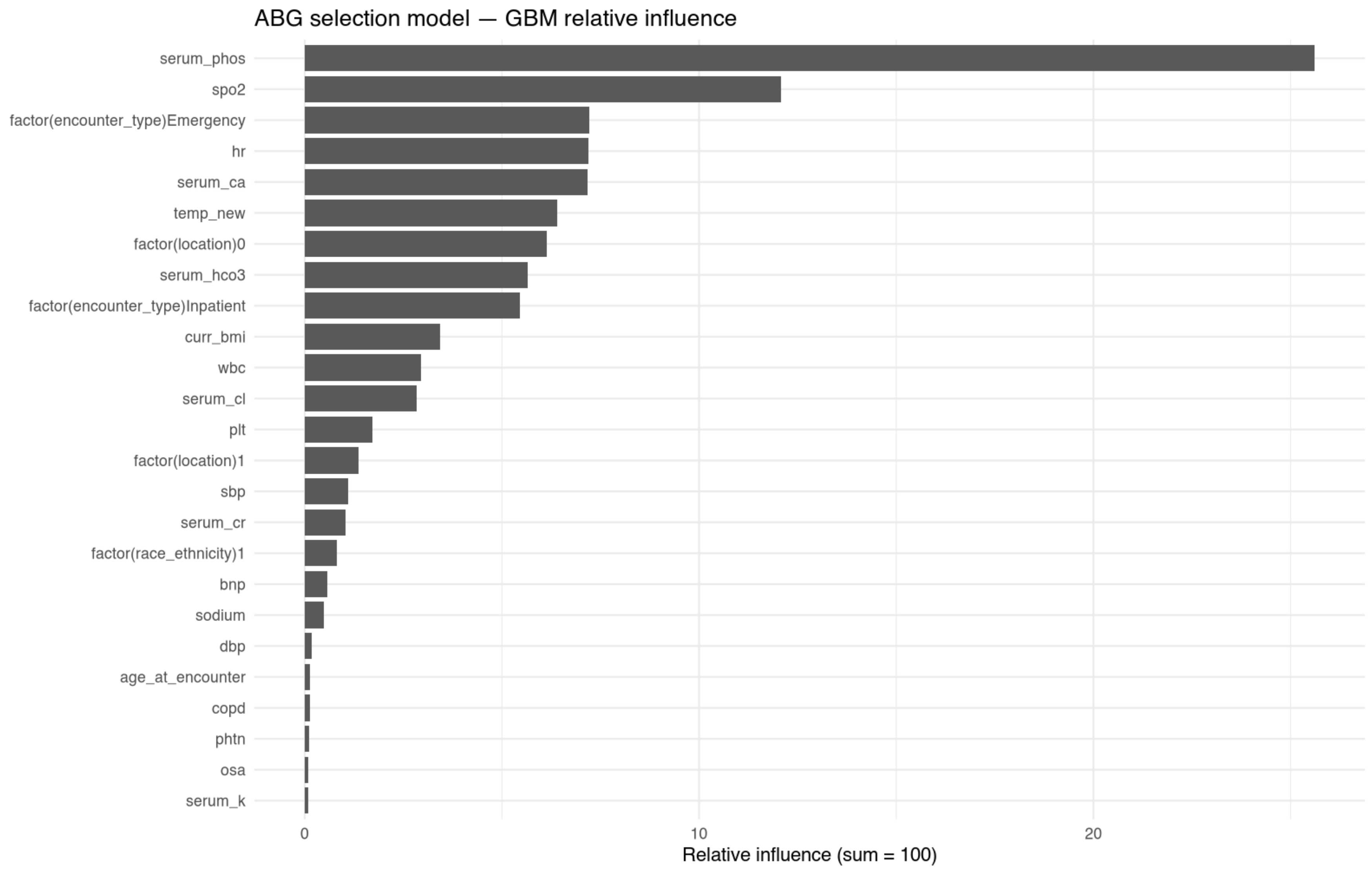
SHAP: a *local*, signed attribution for that subject: "by how much did feature j push this person's log-odds of receiving the test up or down vs baseline?, then global shap is mean absolute SHAP across subjects—i.e., the **typical magnitude** of a feature's contribution to predictions in your population. 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.**

ABG

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.

```r
# --- 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$model.obj)
  has_cov <- !is.null(W$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
  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
```



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

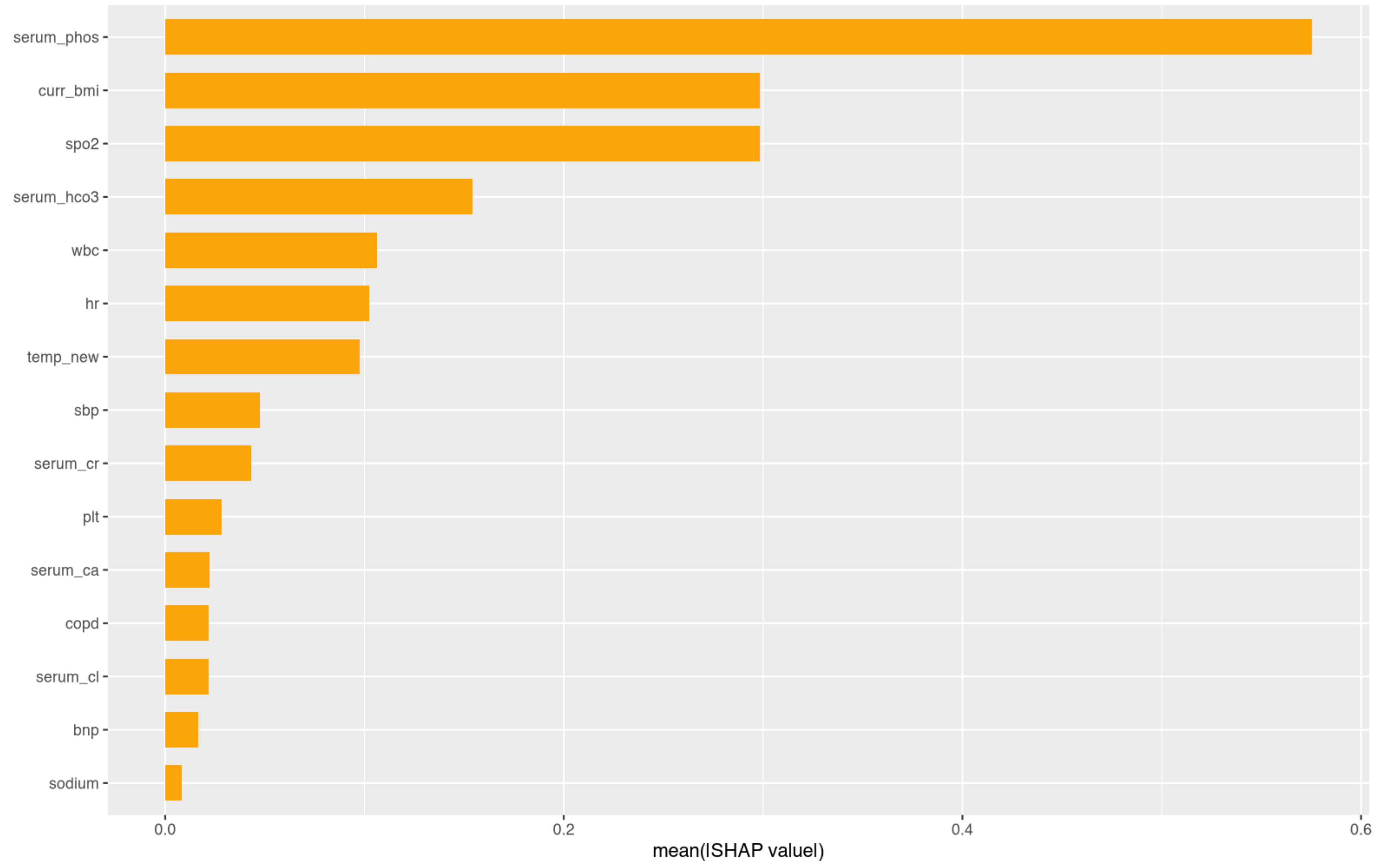
```r
# fast SHAP on LOGIT scale; prunes zero-variance features; subsamples rows
compute_shap_fast <- function(W, top_k = 30, nsim = 64, frac_rows = 0.50,
                              max_rows = 100000, seed = 123) {
  mats <- prep_design(W); X <- mats$X; gbm_fit <- mats$gbm_fit; best_tree <- mats$best_tree

  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)
    predict(object, newdata = newdata, 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)
```

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

```
[compute_shap_fast] 967.17 s
```

```r
# 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))

# --- 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)
```

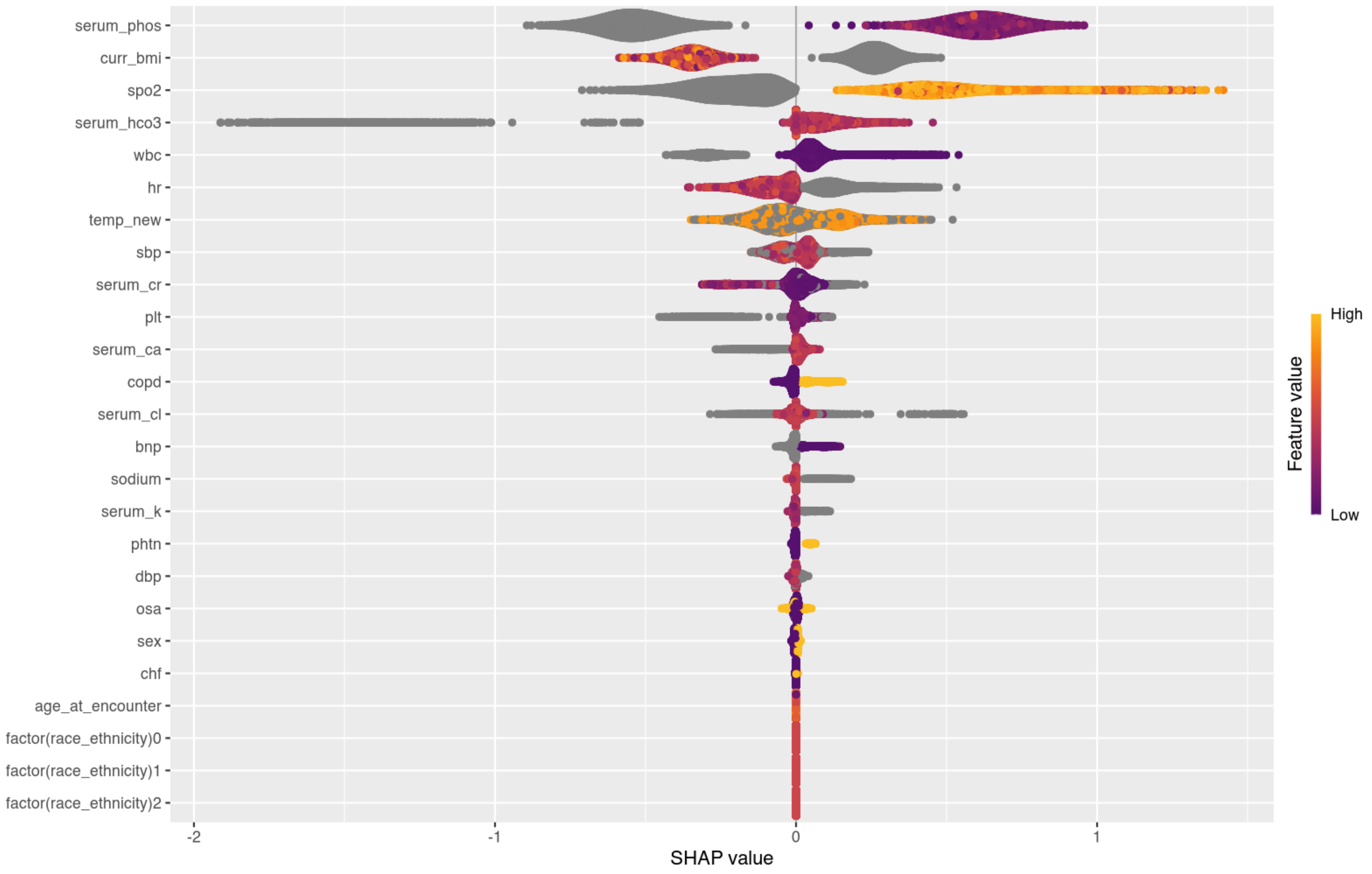
```
Warning: `label` cannot be a <ggplot2::element_blank> object.
```



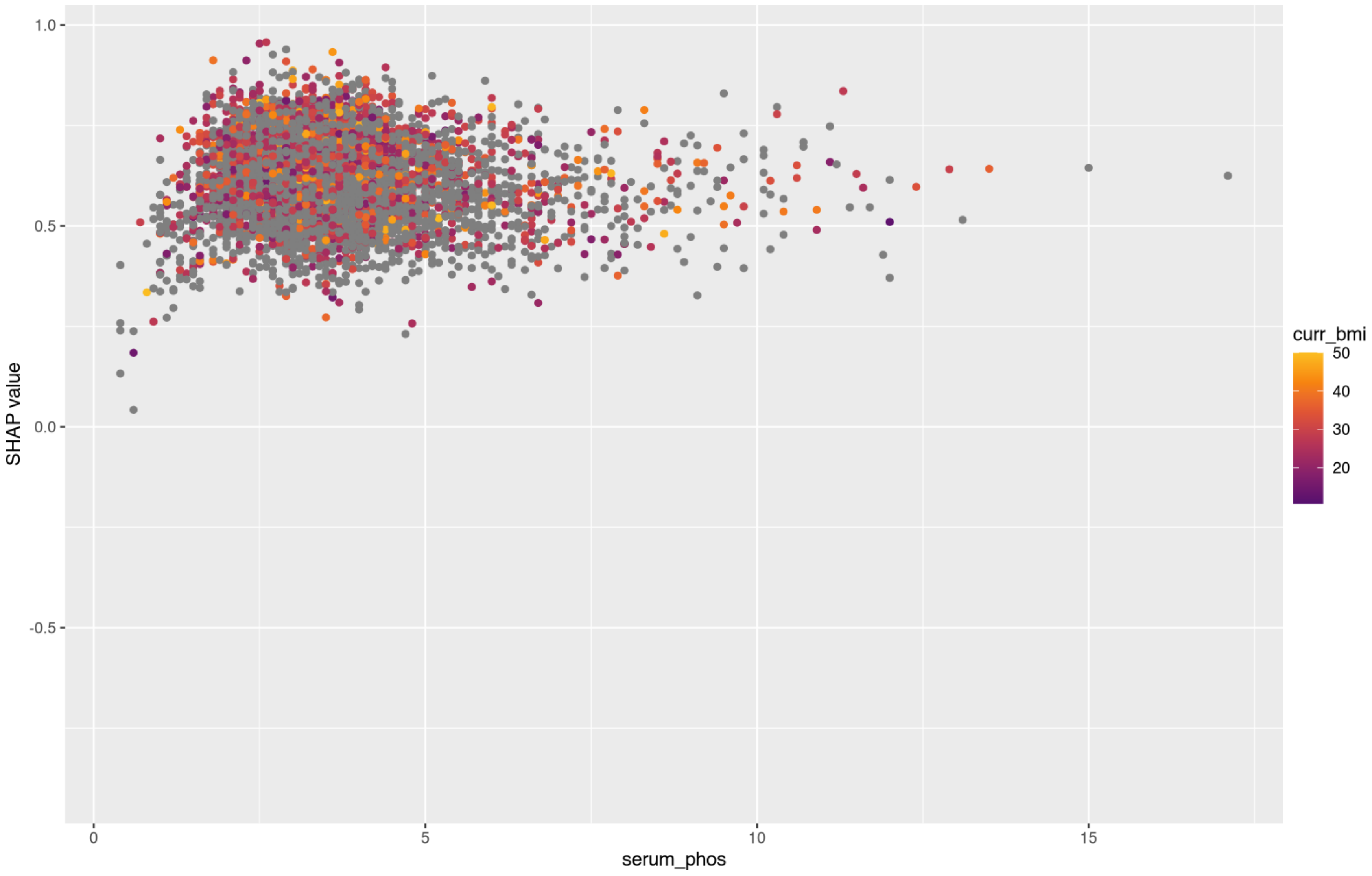
```r
library(shapviz)

S <- as.matrix(sh_abg_fast$shap)    # n x p SHAP values
```

```r
X <- as.data.frame(sh_abg_fast$X)  # same rows, p columns (features)

# 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.
```



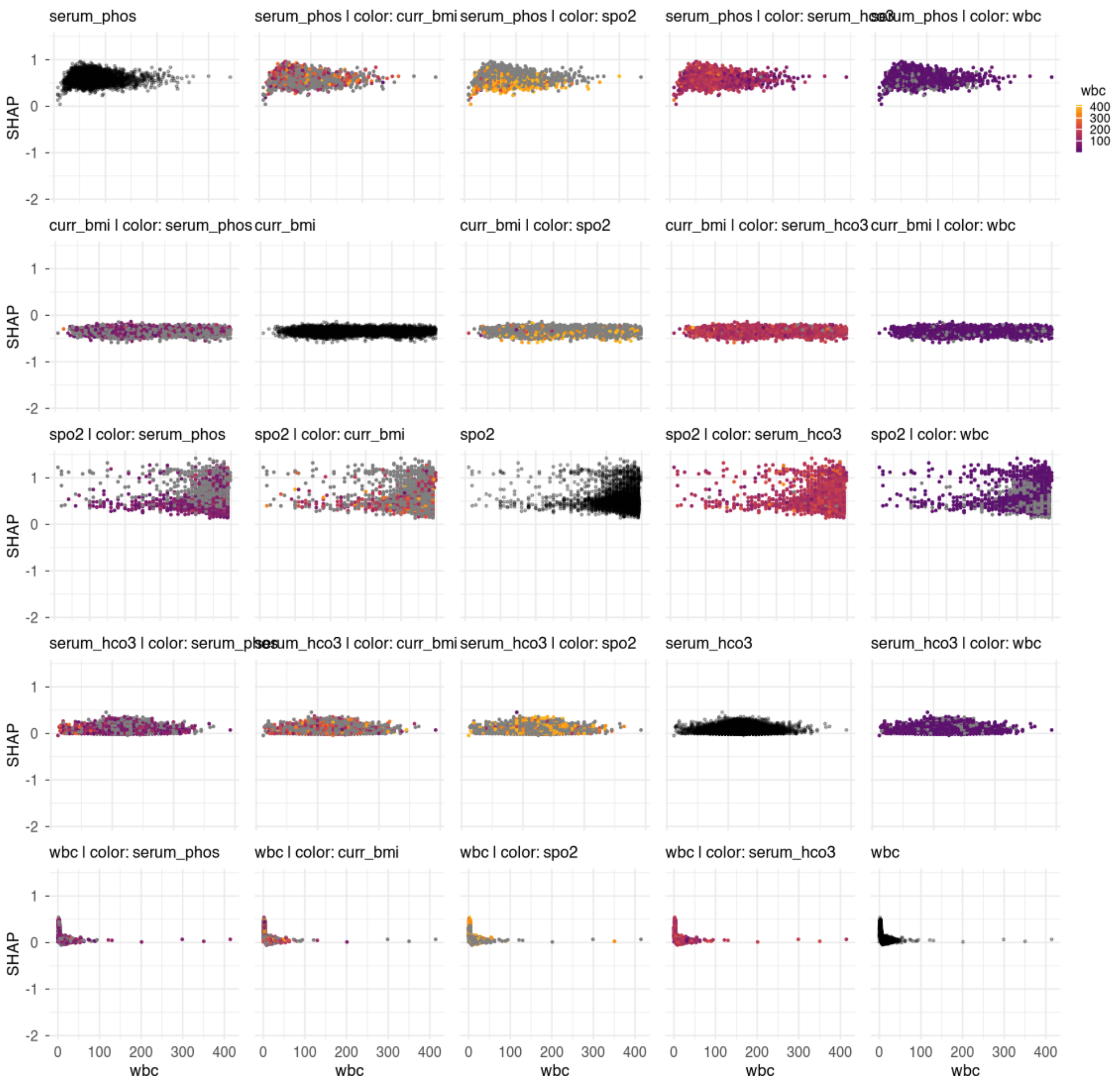
```
# Primary = top feature; color by next feature
imp_order <- colnames(S)[ord]
sv_dependence(sv, v = imp_order[1], color_var = imp_order[2], smooth = TRUE)
```



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

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

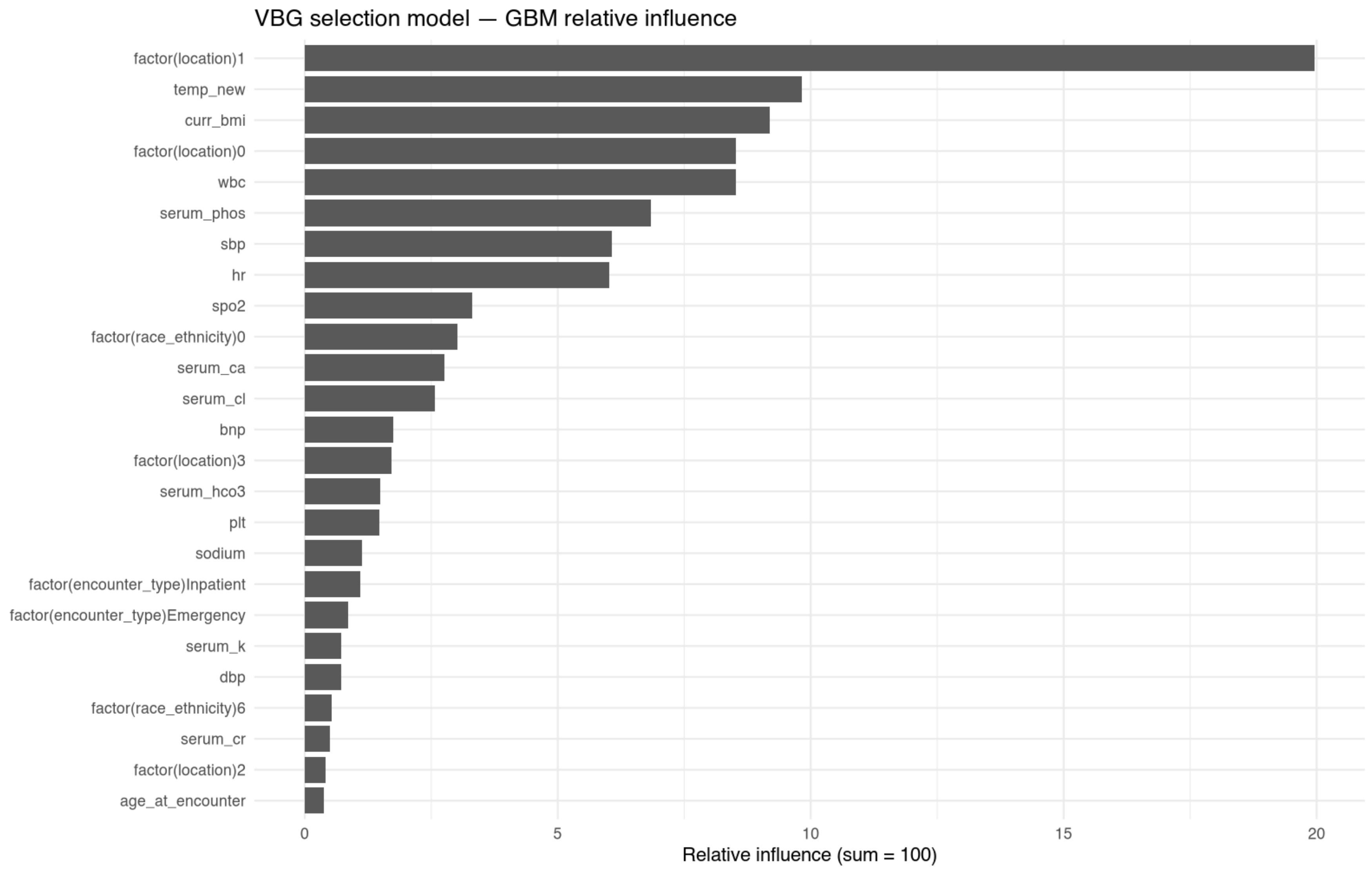
Top-5 SHAP dependence: interactions (off-diagonal) and main effects (diagonal)



VBG

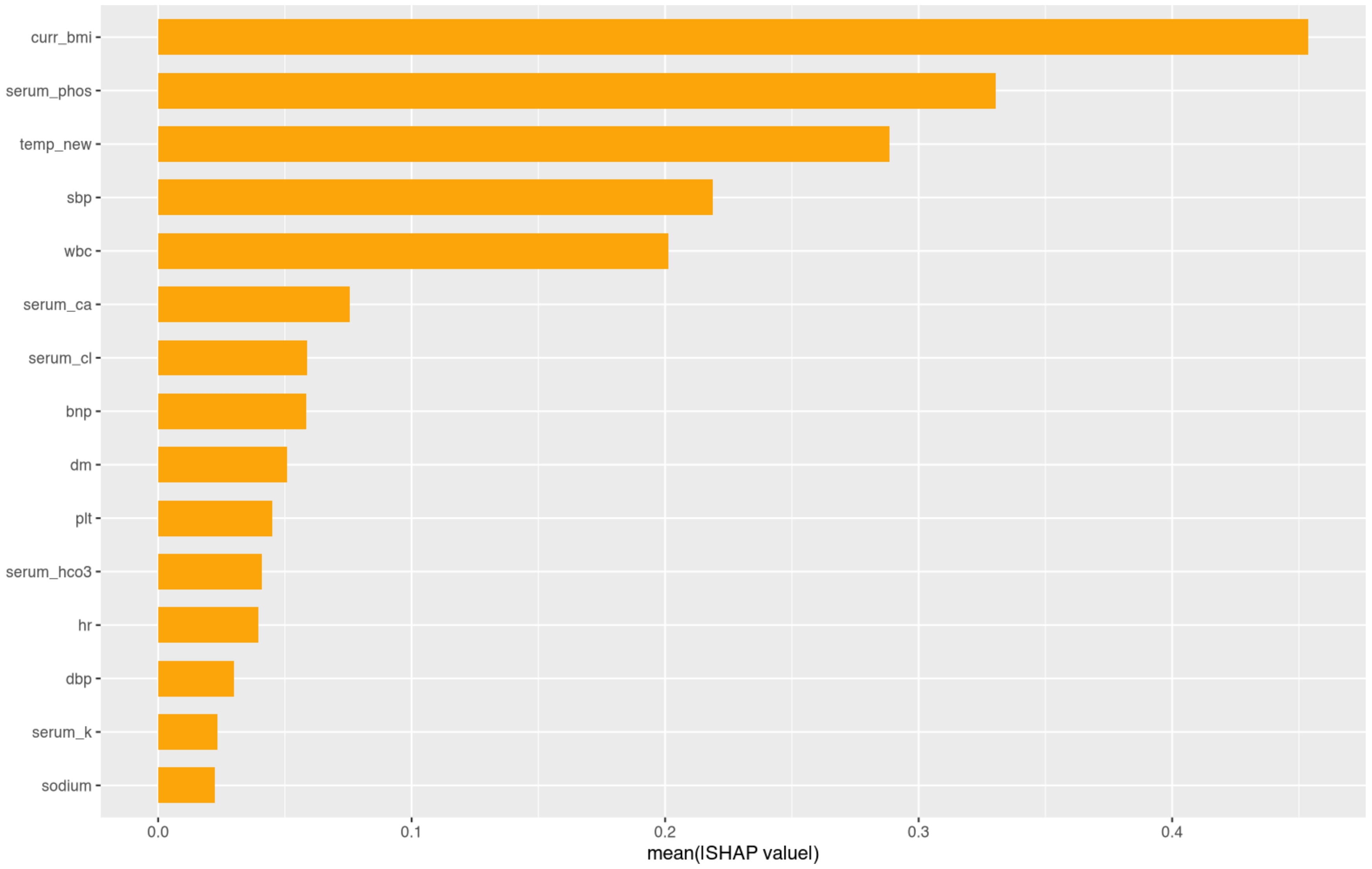
VBG Explainability

```r
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
```

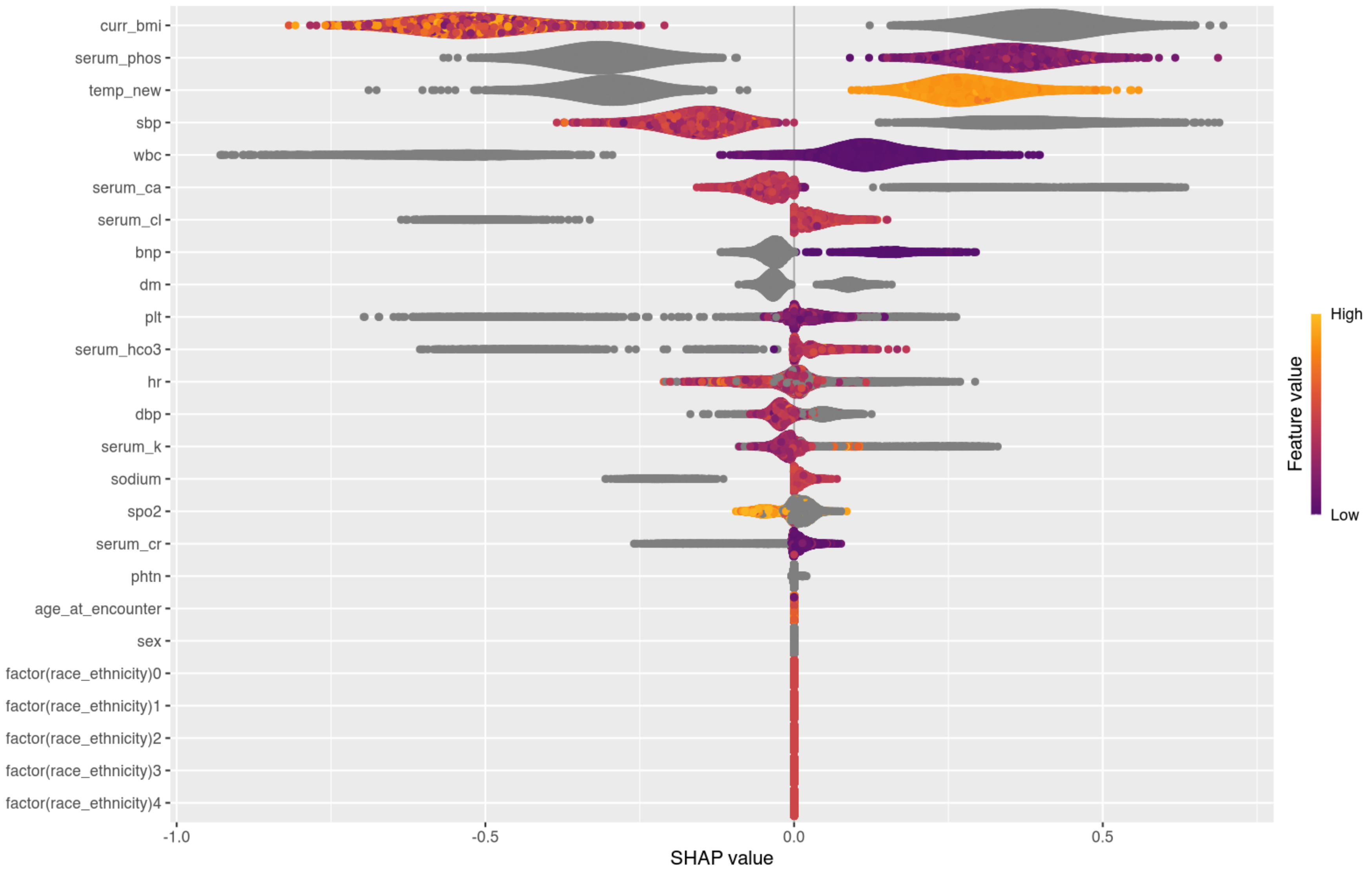


```r
library(shapviz); library(fastshap)

t0 <- Sys.time()
```

```r
sh_vbg_fast <- compute_shap_fast(w_vbg, top_k = 100, nsim = 32, frac_rows = 0.25, max_rows = 100000)
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)
```

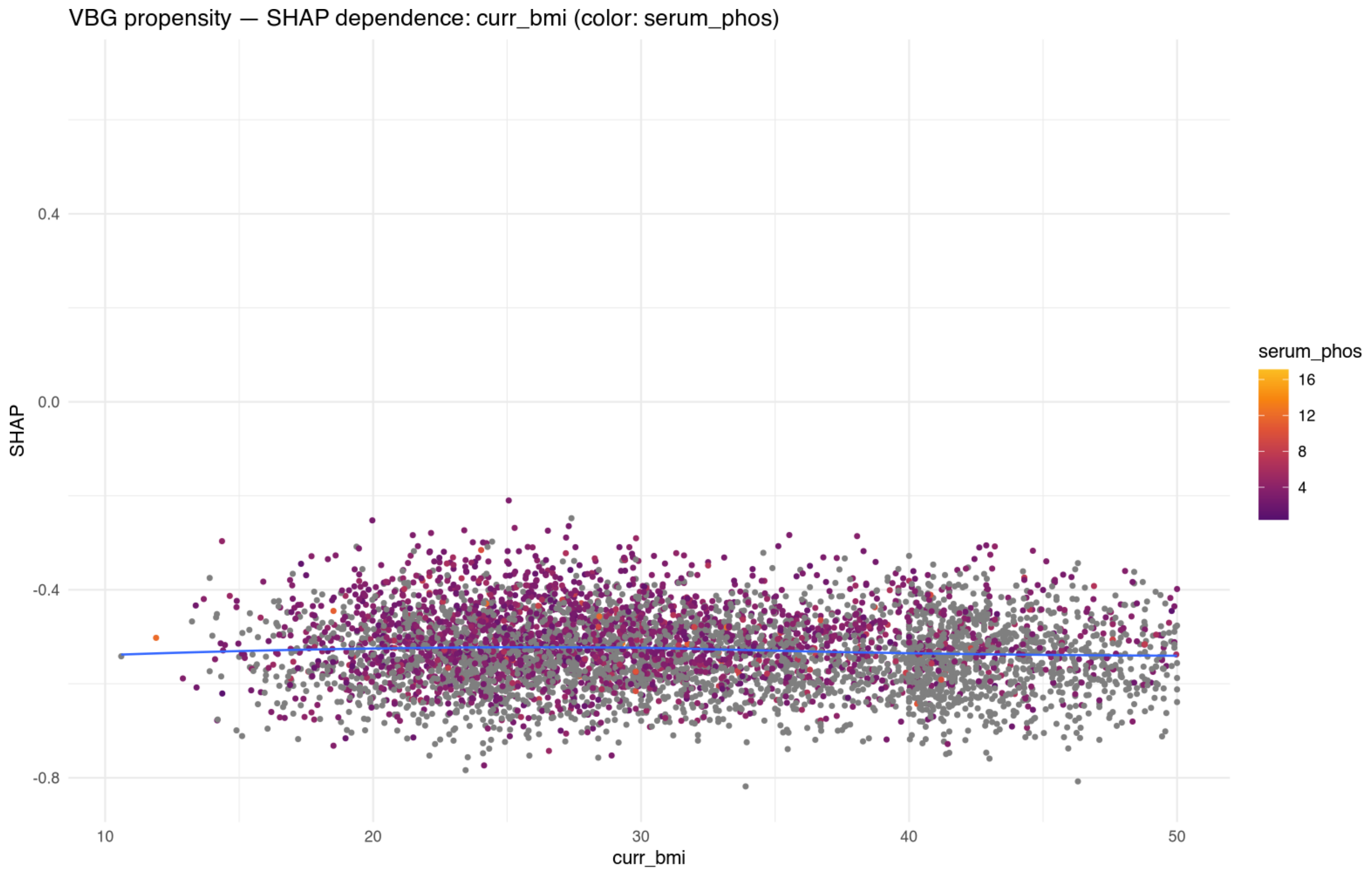


```r
library(shapviz)
sv_importance(sv_vbg, kind = "beeswarm", max_display = 25)
```



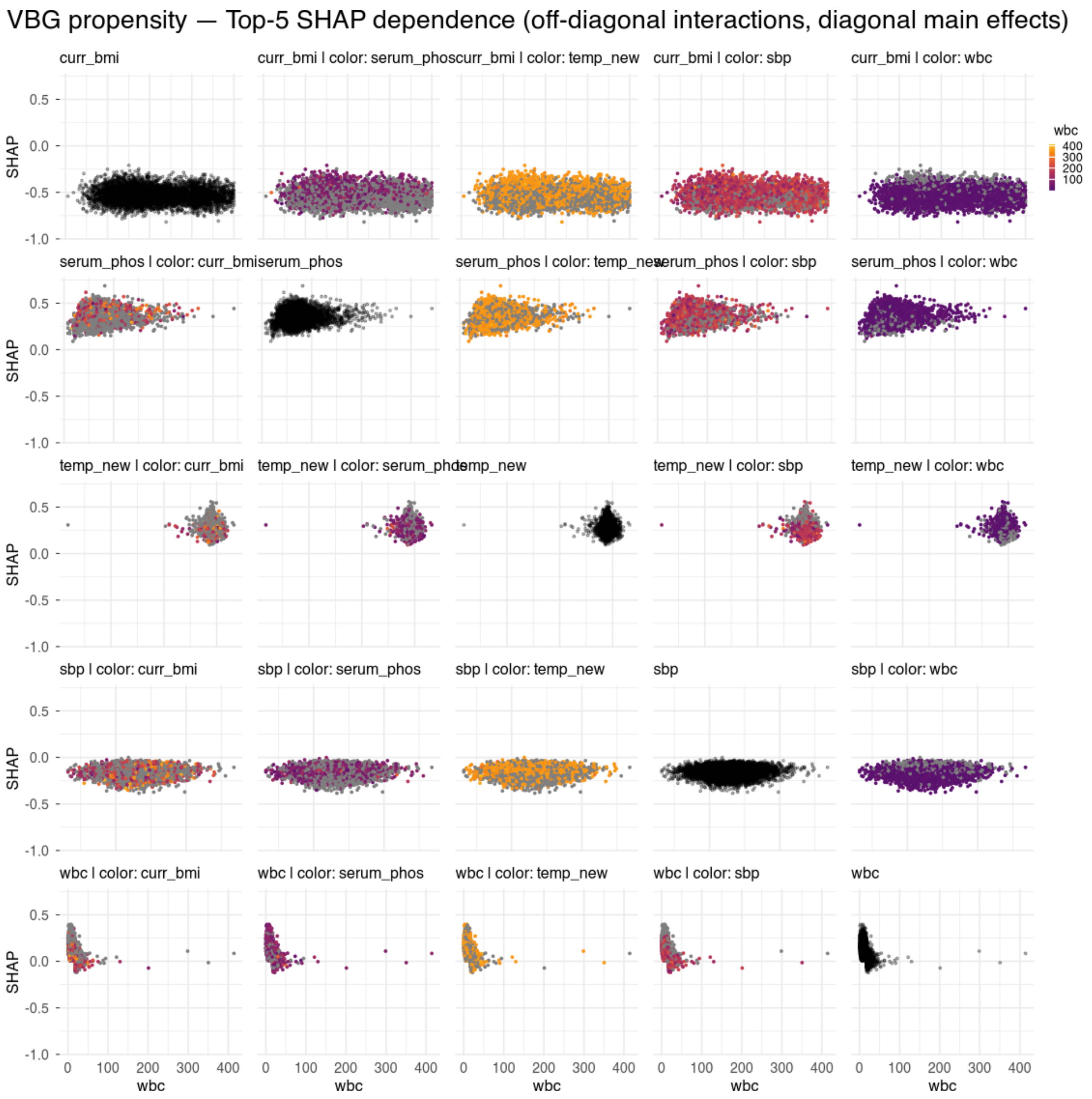
```r
library(ggplot2)

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

```r
pri_vbg <- imp_order_vbg[1]
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)
```



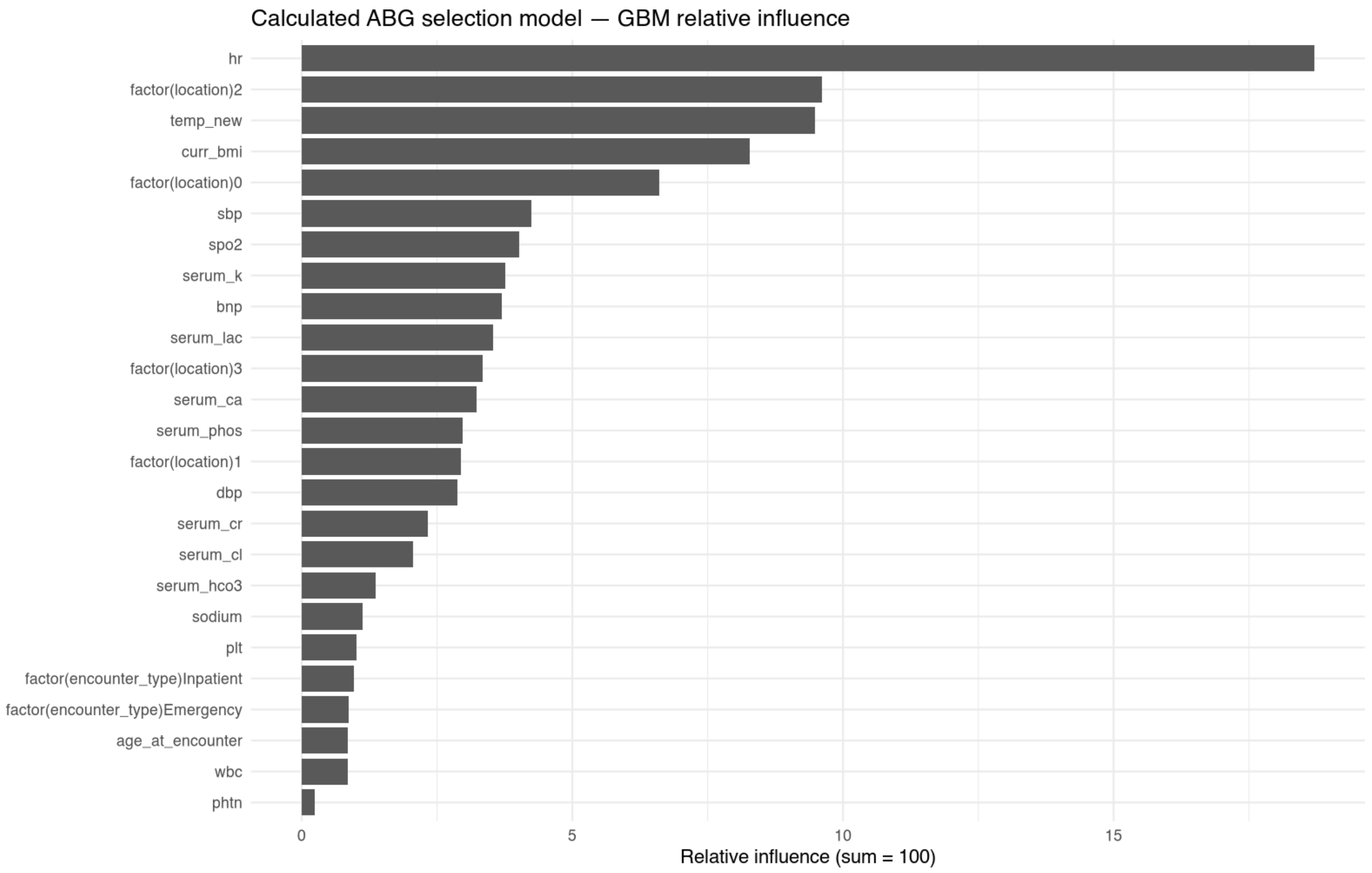
```
library(shapviz); library(ggplot2); library(patchwork); library(grid)

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: serum_phos
curr_bmi | color: temp_new
curr_bmi | color: sbp
curr_bmi | color: wbc
serum_phos | color: curr_bmi
serum_phos
serum_phos | color: temp_new
serum_phos | color: sbp
serum_phos | color: wbc
temp_new | color: curr_bmi
temp_new | color: serum_phos
temp_new
temp_new | color: sbp
temp_new | color: wbc
sbp | color: curr_bmi
sbp | color: serum_phos
sbp | color: temp_new
sbp
sbp | color: wbc
wbc | color: curr_bmi
wbc | color: serum_phos
wbc | color: temp_new
wbc | color: sbp
wbc
SHAP
0.5
0.0
-0.5
-1.0
0
100
200
300
400
wbc
wbc
400
300
200
100

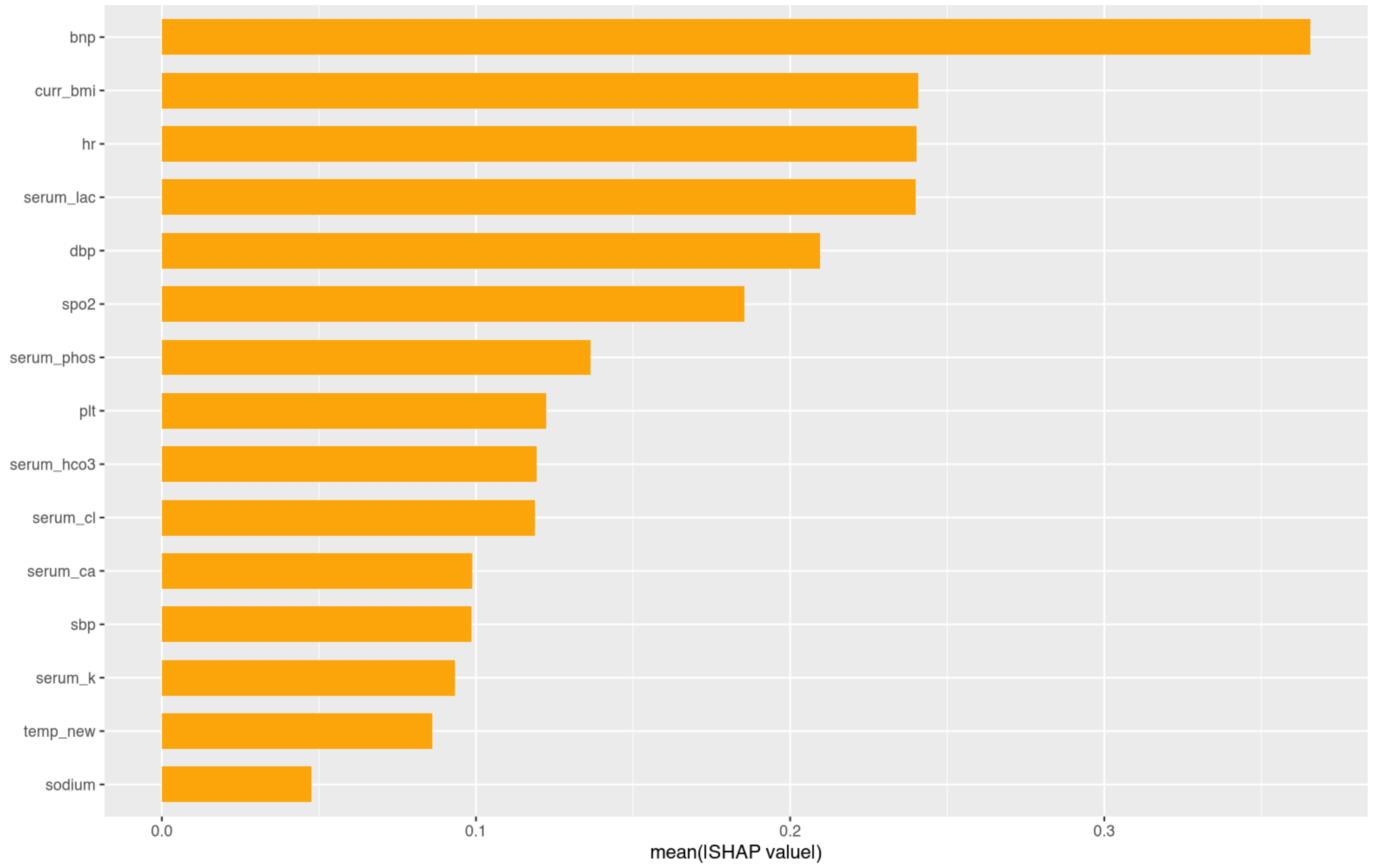
VBG-Calc

```r
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
```

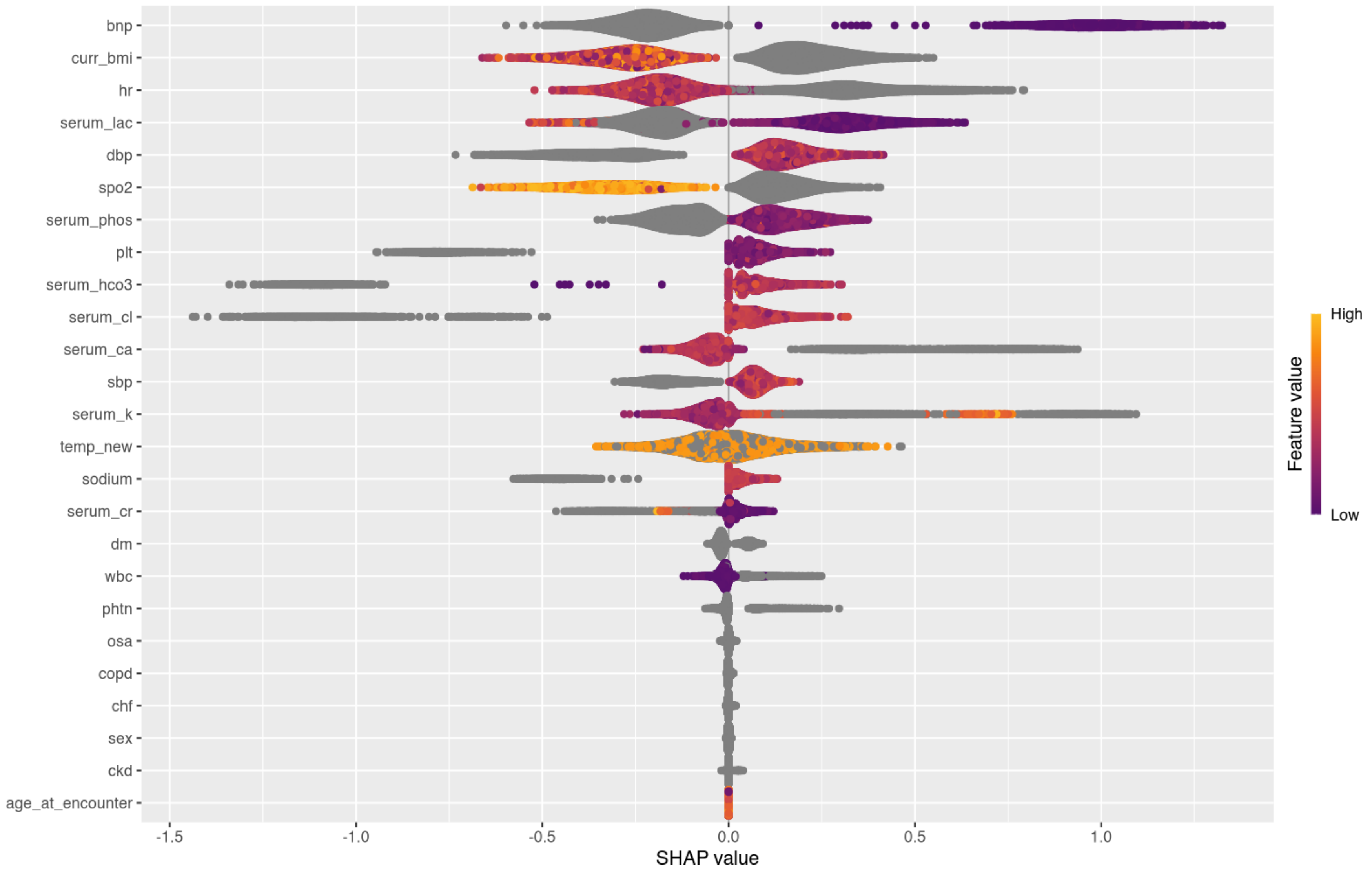


```r
library(shapviz); library(fastshap)

t0 <- Sys.time()
```

```r
sh_calc_fast <- compute_shap_fast(w_vbg_calc, top_k = 100, nsim = 32, frac_rows = 0.25, max_rows = 100000)
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)
```

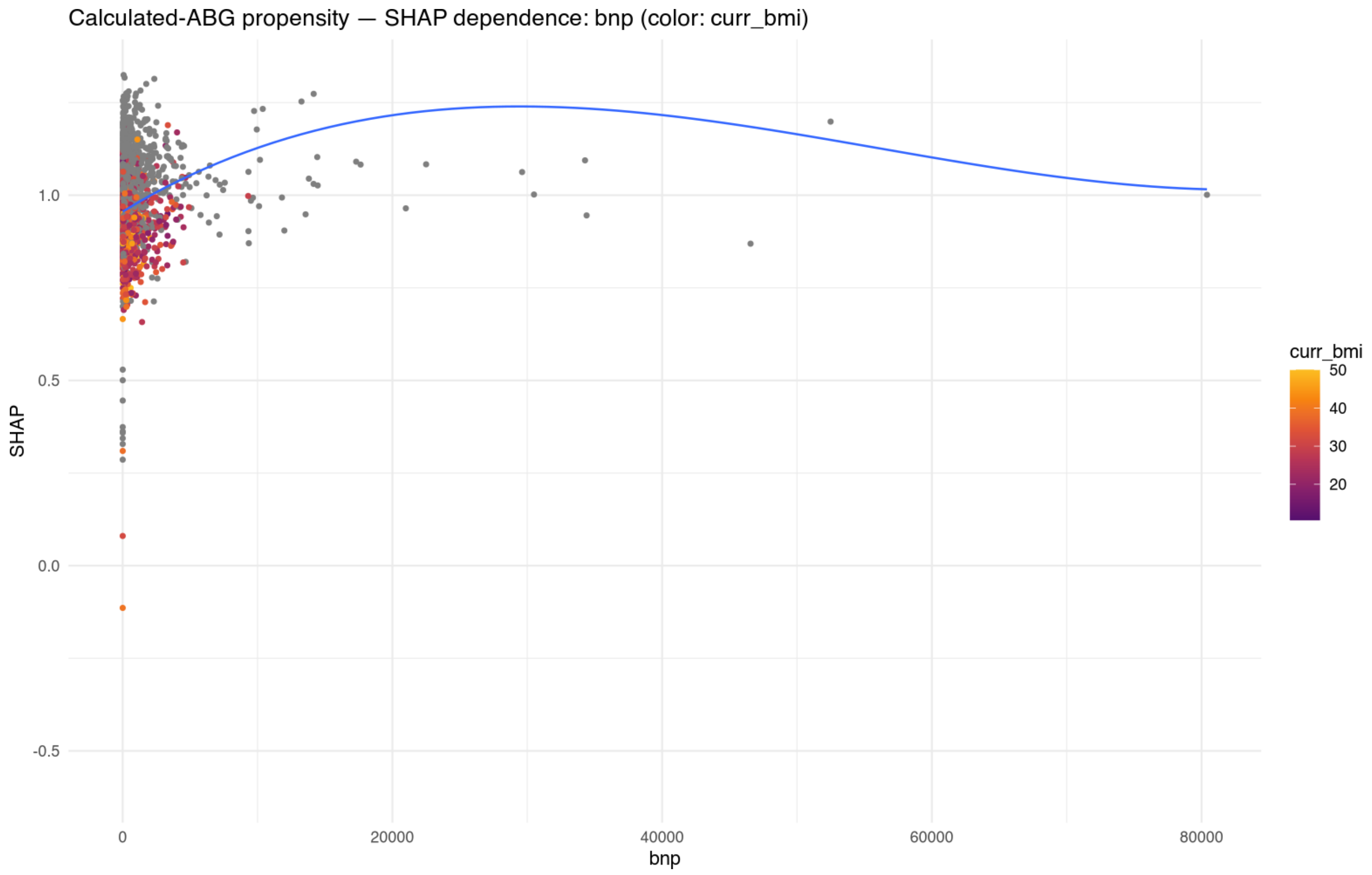


```r
library(shapviz)
sv_importance(sv_calc, kind = "beeswarm", max_display = 25)
```



```r
library(ggplot2)

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

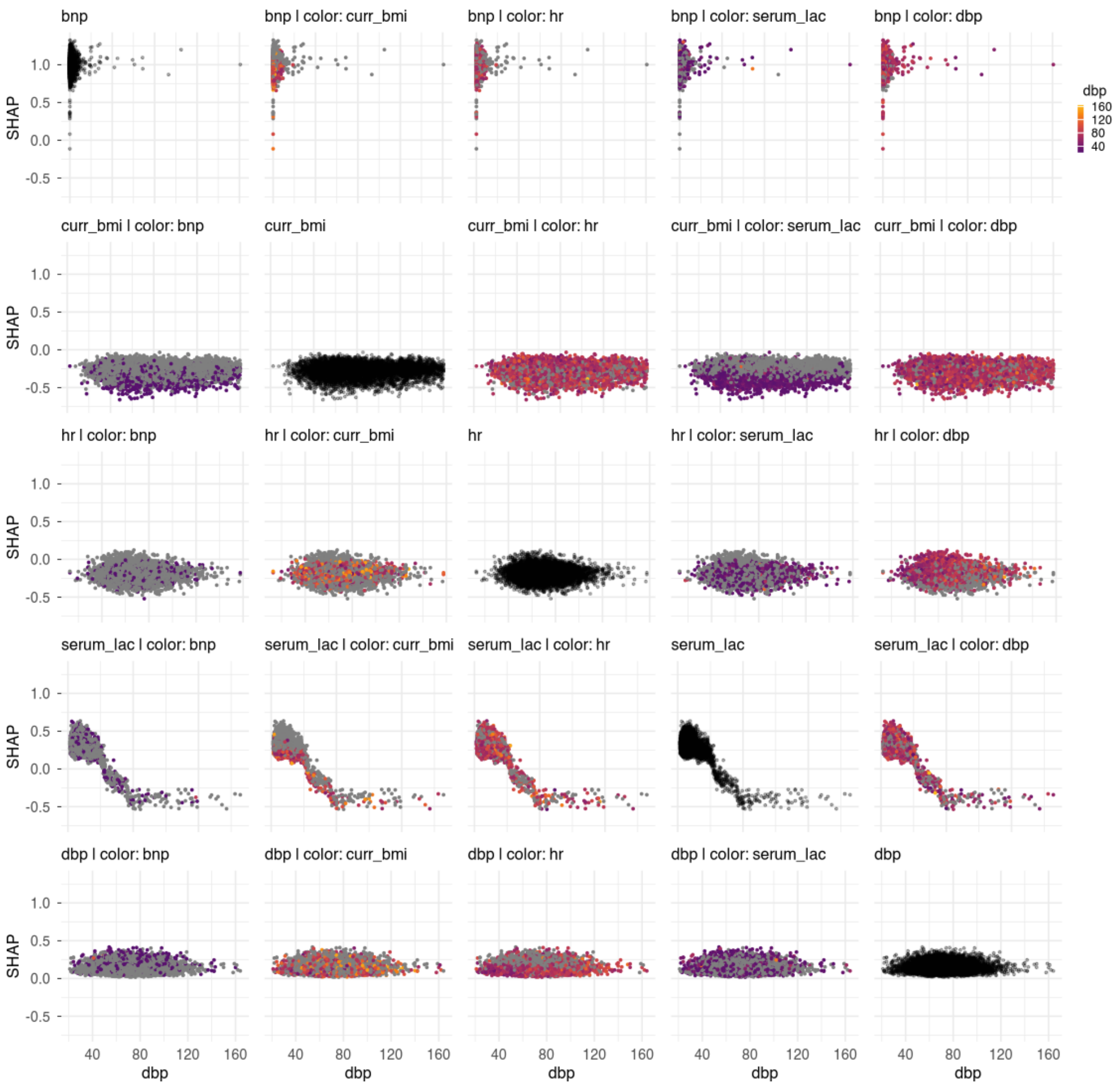
```r
pri_calc <- imp_order_calc[1]
aux_calc <- imp_order_calc[2]
if (identical(aux_calc, pri_calc) || !(aux_calc %in% colnames(X_calc))) aux_calc <- imp_order_calc[3]

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)
```



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

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

```r
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
```

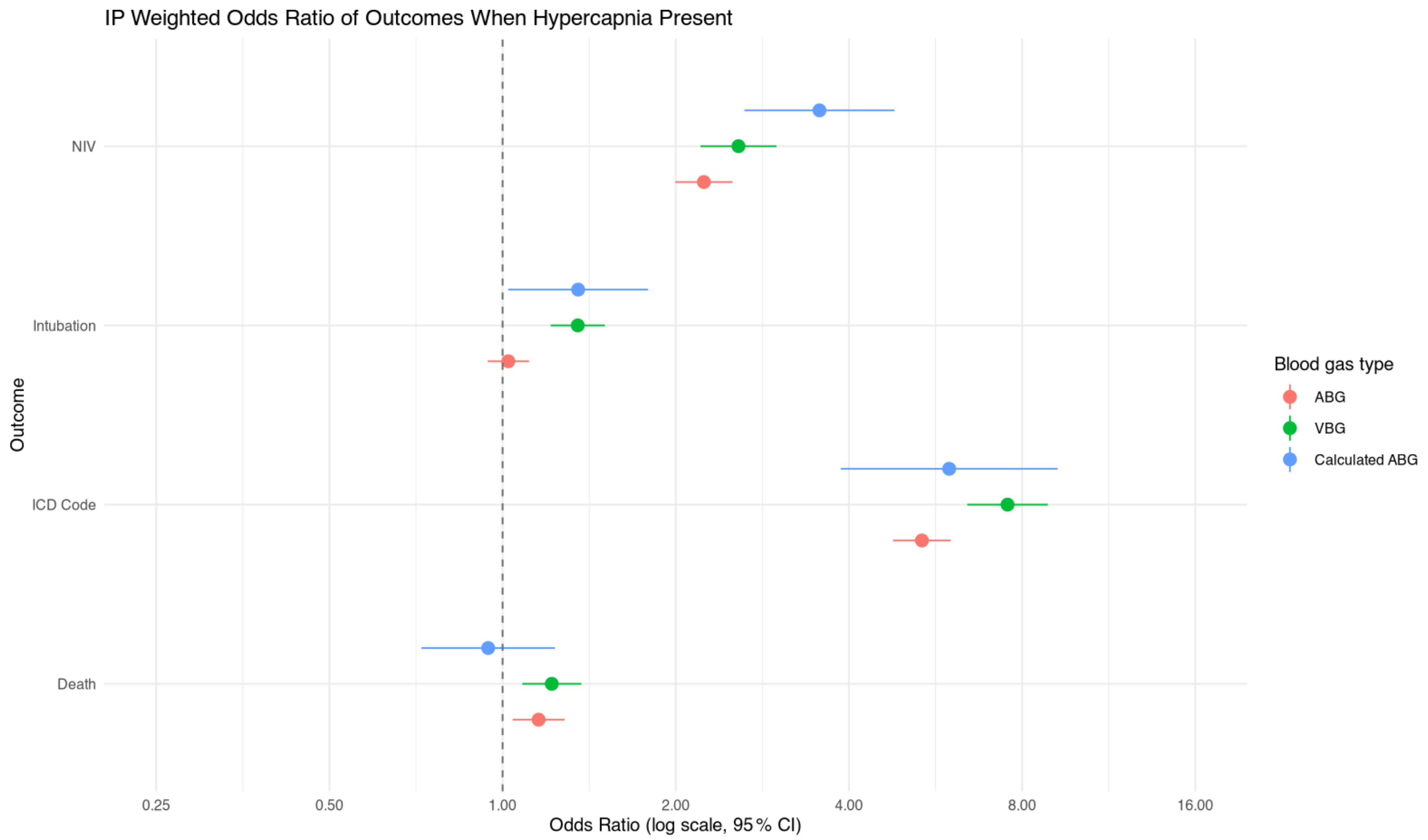
Calculated-ABG propensity — Top-5 SHAP dependence (off-diagonal interactions, diagonal mair



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
  )
```

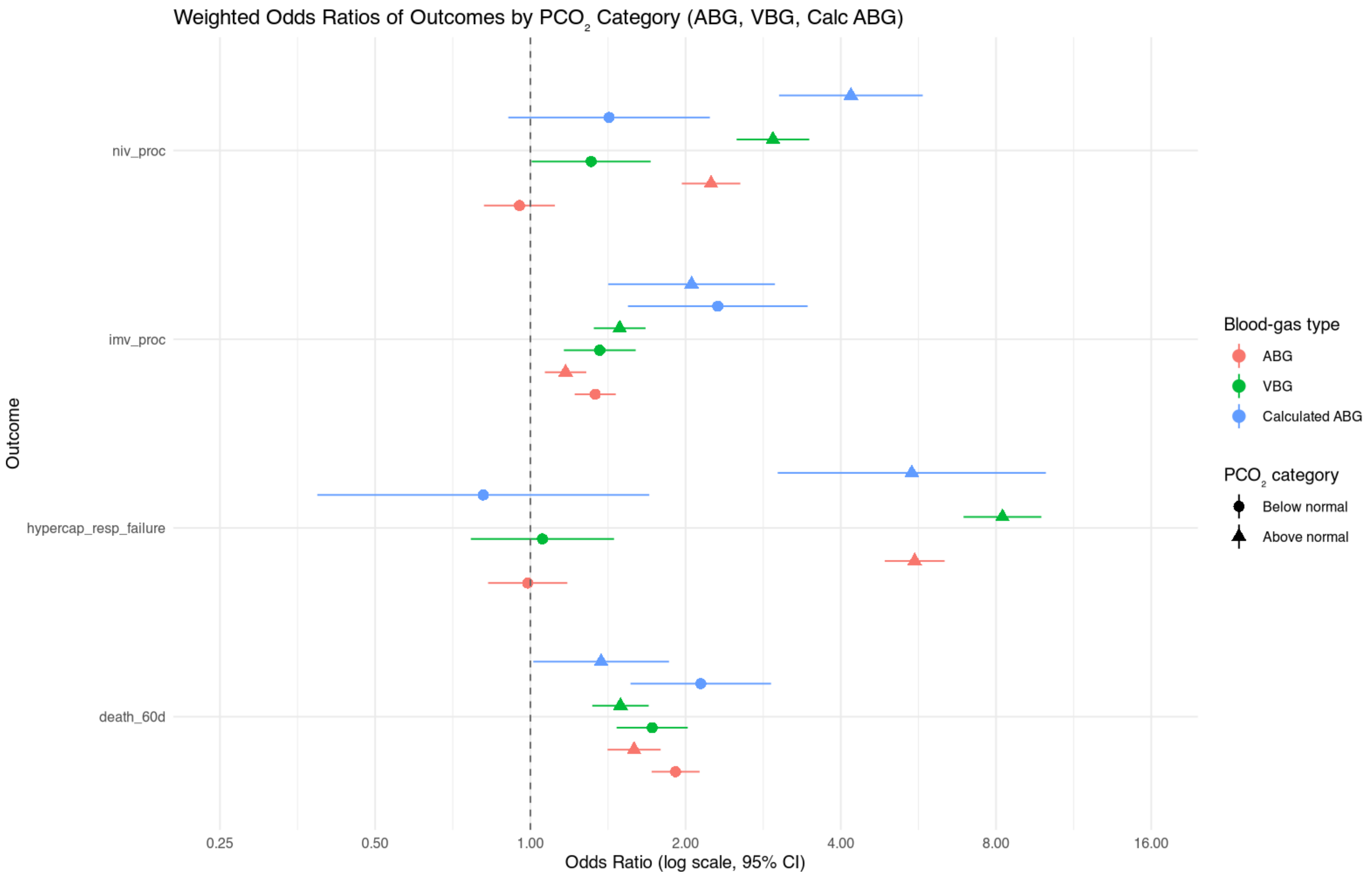
```r
) +
  geom_pointrange(position = position_dodge(width = 0.6), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = number_format(accuracy = 0.01)
  ) +
  coord_flip() +
  labs(
    title  = "IP Weighted Odds Ratio of Outcomes When Hypercapnia Present",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95 % CI)",
    color  = "Blood gas type",
    caption = paste(
      "Inverse-probability weights adjust for covariates associated with receiving each blood-gas.",
      "Models are fitted within their respective cohorts:",
      "ABG (weights = w_abg), VBG (w_vbg), 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))
```



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.

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 < 35 ~ "Below normal",
      !is.na(vbg_co2) & vbg_co2 >= 35 & 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]]),
      .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))
```

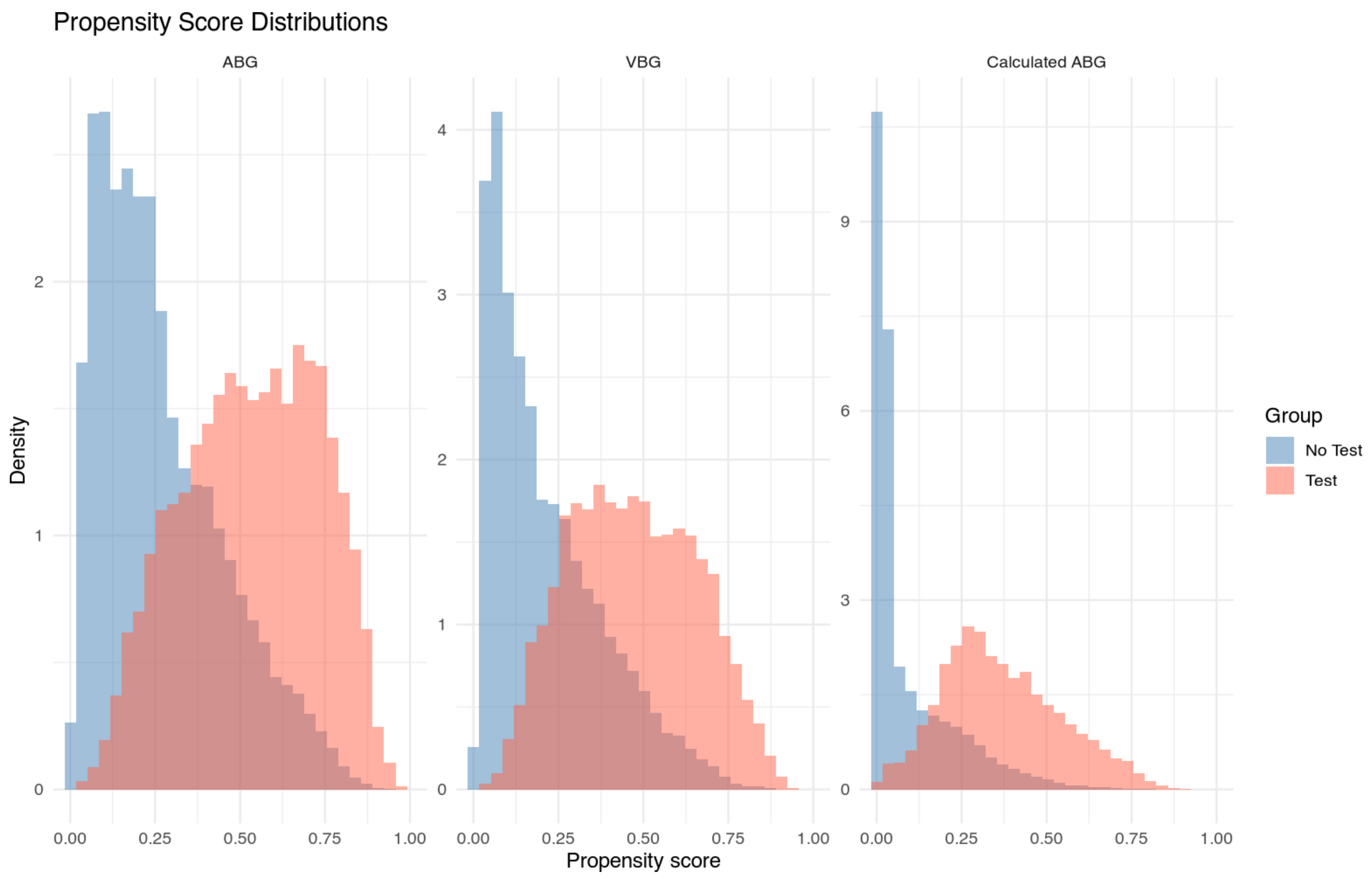


Plotting propensity scores

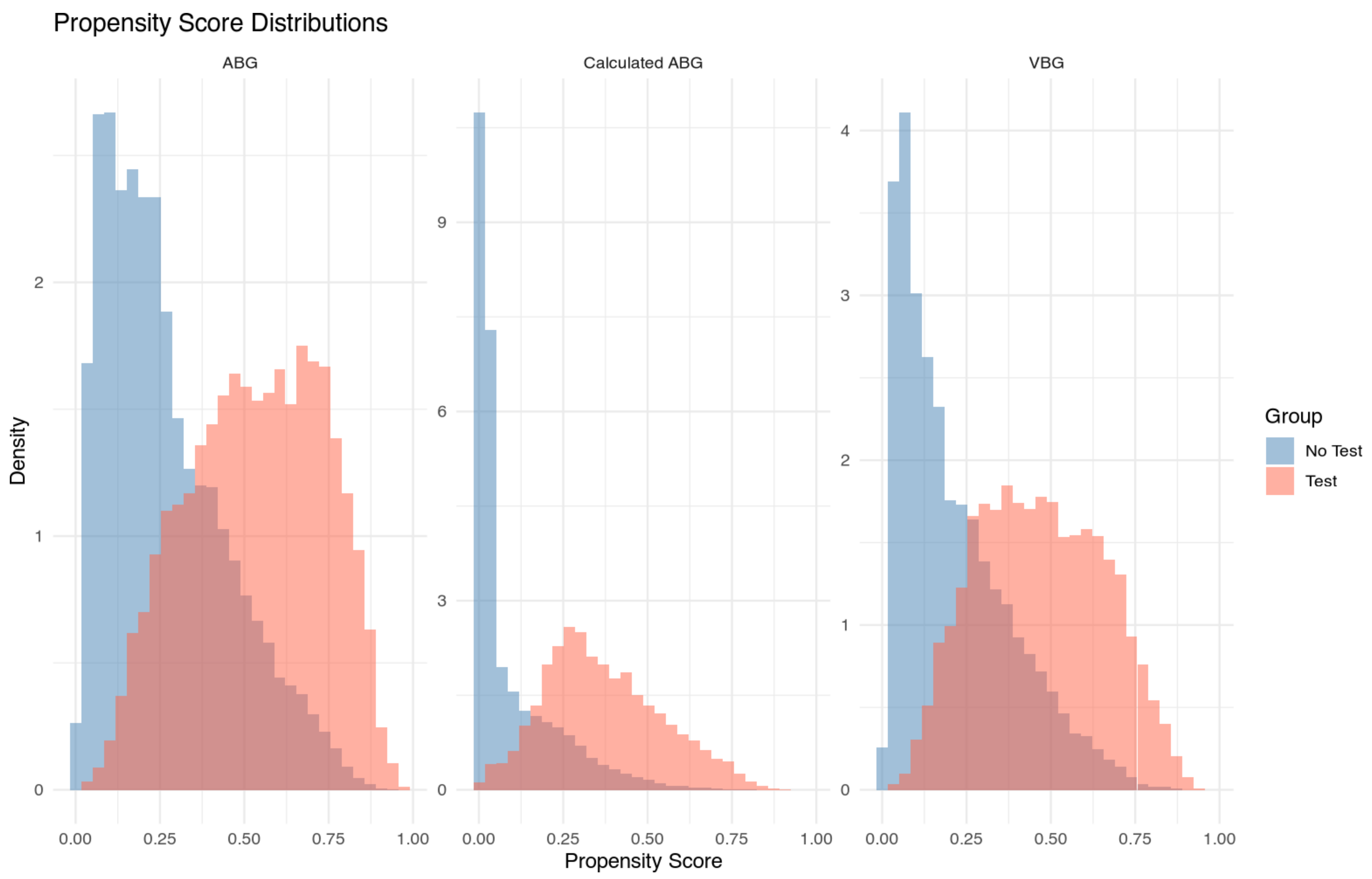
```r
# --- Propensity score histograms (ABG / VBG / Calculated-ABG) -----------------
# ABG = arterial blood gas; VBG = venous blood gas

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.")
}
```

```r
# 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)) {
  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.
```



```r
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
0
1
2
3
4
6
9
0.00
0.25
0.50
0.75
1.00
Group
No Test
Test