

ABG-VBG Analysis

Brian Locke, Anila Mehta

Table of contents

1 Data Pre-processing	1
1.0.1 Package Set Up	1
1.1 Helper functions for model diagnostics	1
1.1.1 Rendering and Diagram Setup	1
1.2 1) Seed escrow (reproducibility anchors)	1
1.2.1 Rendering support information	2
1.2.2 Input Data Formatting	2
1.2.3 Configuration for the IPW models	3
1.2.4 Creating subset_data	7
1.2.5 Generating Codebook for the Full Dataset	7
1.2.6 Outcome Variable Creation	8
1.3 2) Baseline tables	10
1.3.1 2.1 Table 1A and 1B:	10
1.3.2 2.2 Table 1 (Overall ABG/VBG status)	14
1.3.3 2.3 Table 2 (Hypercapnia within cohorts)	18
1.3.4 2.4 Generating Word Docs for Table 1 and Table 2 (separate files)	22
2 Unweighted Binary Logistic Regressions	22
2.0.1 3.1 ABG: Binary hypercapnia models	22
2.0.2 3.2 VBG: Binary hypercapnia models	31
2.0.3 3.3 Display model coefficients for binary hypercapnia on VBG logistic regression	39
2.1 3) Three-level PCO ₂ categories (unweighted)	40

2.2	4) Restricted cubic spline regressions (unweighted)	54
2.2.1	4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2	54
2.2.2	4.2 Unweighted, Restricted Cubic Spline - VBG	61
3	Inverse Propensity Weighting	69
3.0.1	5.1 ABG IPW weighting and diagnostics	69
3.0.2	5.2 ABG IPW spline models	71
3.0.3	5.3 ABG IPW spline models (2–98th percentile)	80
3.0.4	5.4 VBG IPW weighting and spline models	84
3.1	5) Weighted effect estimates	93
3.1.1	5.5 Three-level PCO ₂ categories (weighted; ABG, VBG)	106
3.1.2	5.6 Three-level PCO ₂ categories (weighted; ABG vs VBG only)	119
3.2	6) Propensity score diagnostics	123
4	Multiple Imputation Analysis	129
4.1	7) Packages and reproducibility	129
4.1.1	Missing data / imputation summary (pre-imputation)	131
4.1.2	7.1 Missingness audit (what, where, how much)	131
4.1.3	7.2 Missingness structure and drivers	136
4.1.4	7.3 Monte Carlo error check after MI	153
4.2	8) Pre-imputation data prep (consistent types & predictors)	154
4.3	9) Imputation model specification (MICE)	154
4.3.1	9.1 Predictor matrix & methods. Run MICE (moderate settings for scale)	154
4.3.2	9.2 Convergence & plausibility checks	205
4.3.3	9.3 Observed vs imputed distributions (by strata)	208
4.4	10) Refit propensity models within each imputation	251
4.4.1	FAIL-FAST CHECKS	251
4.4.2	10.1 ABG propensity (has_abg)	251
4.4.3	10.2 Balance diagnostics across imputations	255
4.4.4	10.3 VBG propensity (has_vbg)	260
4.4.5	10.4 VBG balance	264
4.5	11) Weighted outcome models within each imputation + pooling	268
4.5.1	11.1 Helper: fit + extract log-OR and SE from svyglm	268
4.5.2	11.2 ABG: MI pooled spline models (treated cohort only)	276
4.5.3	11.3 VBG: MI pooled spline models (treated cohort only)	283
4.6	12) Explainability on one representative imputation	291
4.7	13) MI three-level PCO ₂ helpers and checks	307

4.8	14) MI + IPW three-level PCO2 (ABG & VBG)	310
4.8.1	14.1 ABG: MI + IPW, three-level PCO2 outcomes	310
4.8.2	14.2 VBG: MI + IPW, three-level PCO2 outcomes	316
4.8.3	14.3 Table 3: MI-pooled IPW associations (3-level CO)	322
4.8.4	14.3 Visualization: pooled three-level ORs	323
4.9	15) Imputed, weighted spline PCO2 (ABG & VBG)	328
4.9.1	15.1 ABG, imputed, weighted, spline outcome	329
4.9.2	15.2 VBG, imputed, weighted, spline outcome	329
4.9.3	15.3 Visualization	330
4.10	Diagnostics	341
4.10.1	Diagnostics inputs and settings	341
4.10.2	Missingness diagnostics	341
4.10.3	MI convergence and mixing	341
4.10.4	MI stability across m	341
4.10.5	MI maxit sensitivity (sampled)	341
4.10.6	Propensity and weight diagnostics	342
4.10.7	Balance diagnostics	342
4.10.8	Outcome diagnostics	343
4.10.9	Diagnostics summary and audit	343
4.10.10	Performance / runtime log	343
4.11	16) Save, export, and session info	343

reminder: if rendering fails, consider deleting cached items: e.g. rm -rf Results/cache/abg-vbg-2025-12-11/

1 Data Pre-processing

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

1.0.1 Package Set Up

1.1 Helper functions for model diagnostics

1.1.1 Rendering and Diagram Setup

1.2 1) Seed escrow (reproducibility anchors)

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

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

Chunk seed-escrow runtime: 0.00 s

1.2.1 Rendering support information

Chunk gt-pdf-helper runtime: 0.00 s

1.2.2 Input Data Formatting

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

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

rdata_file <- file.path(data_dir_name, "full_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.

```
if (file.exists(rdata_file)) {  
  load(rdata_file)  
  message("Loaded existing dataset from 'full_trinetx.rdata'.")  
} else {  
  message("RData file not found. Reading Stata dataset...")  
  stata_data <- read_dta(stata_file)  
  
  message("Extracting variable labels...")  
  var_label(stata_data)  
  
  message("Extracting value labels...")  
  sapply(stata_data, function(x) if (is.labelled(x)) val_labels(x))  
  
  save(stata_data, file = rdata_file)  
  message("Dataset saved as 'full_trinetx.rdata'.")  
  
  load(rdata_file)  
  message("Loaded newly saved dataset from 'full_trinetx.rdata'.")  
}
```

Loaded existing dataset from 'full_trinetx.rdata'.

Chunk load-trinetx-data runtime: 7.28 s

1.2.3 Configuration for the IPW models

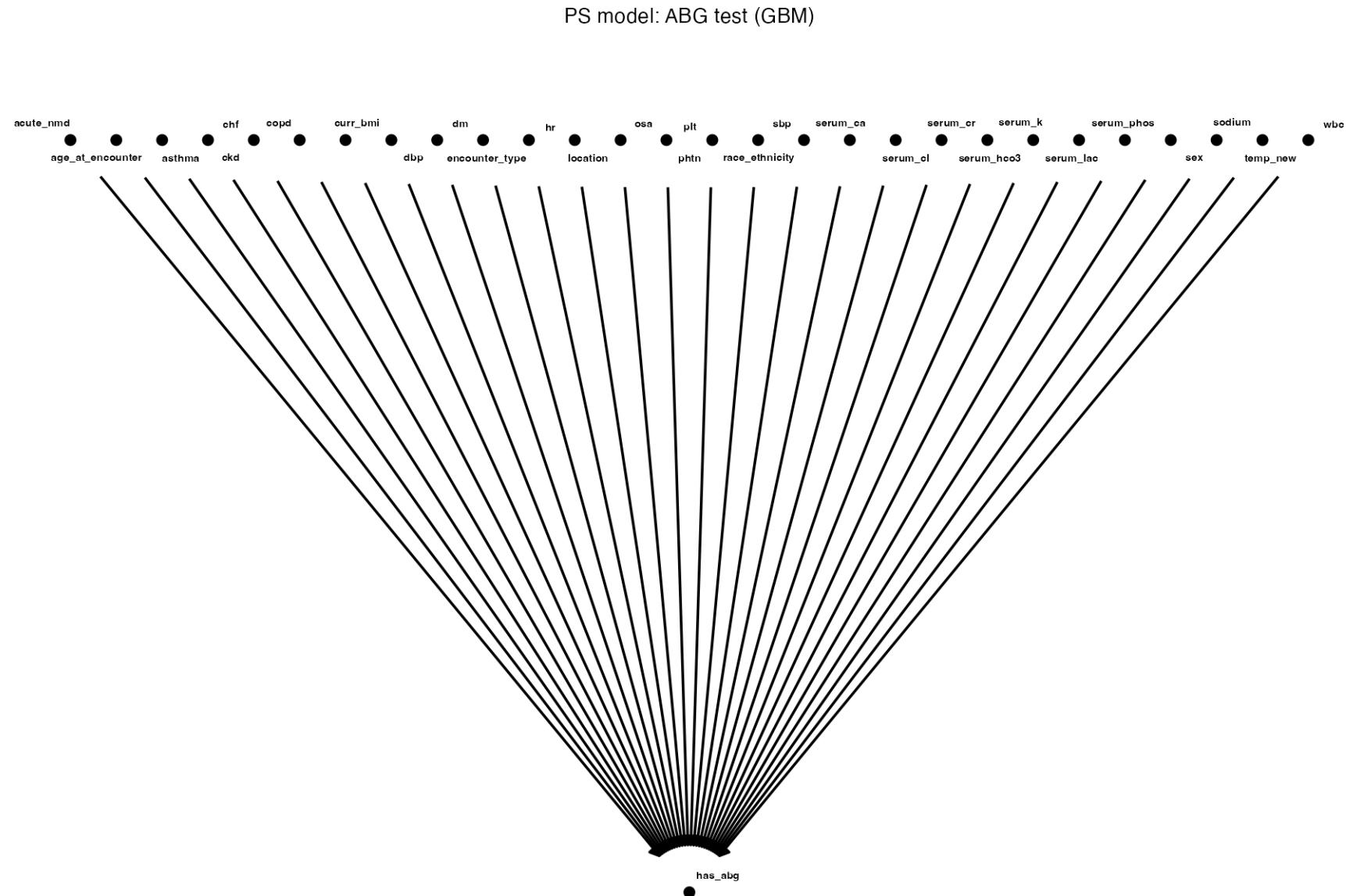
```
drop_vars_ultra_missing <- c("bpn", "spo2")
cat_vars <- c("sex", "race_ethnicity", "location", "encounter_type")
numeric_vars <- c(
  "age_at_encounter", "curr_bmi", "temp_new", "sbp", "dbp", "hr",
  "sodium", "serum_cr", "serum_hco3", "serum_cl", "serum_lac", "serum_k",
  "wbc", "plt", "serum_phos", "serum_ca"
)
co2_vars <- c("paco2", "vbg_co2", "vbg_o2sat")

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

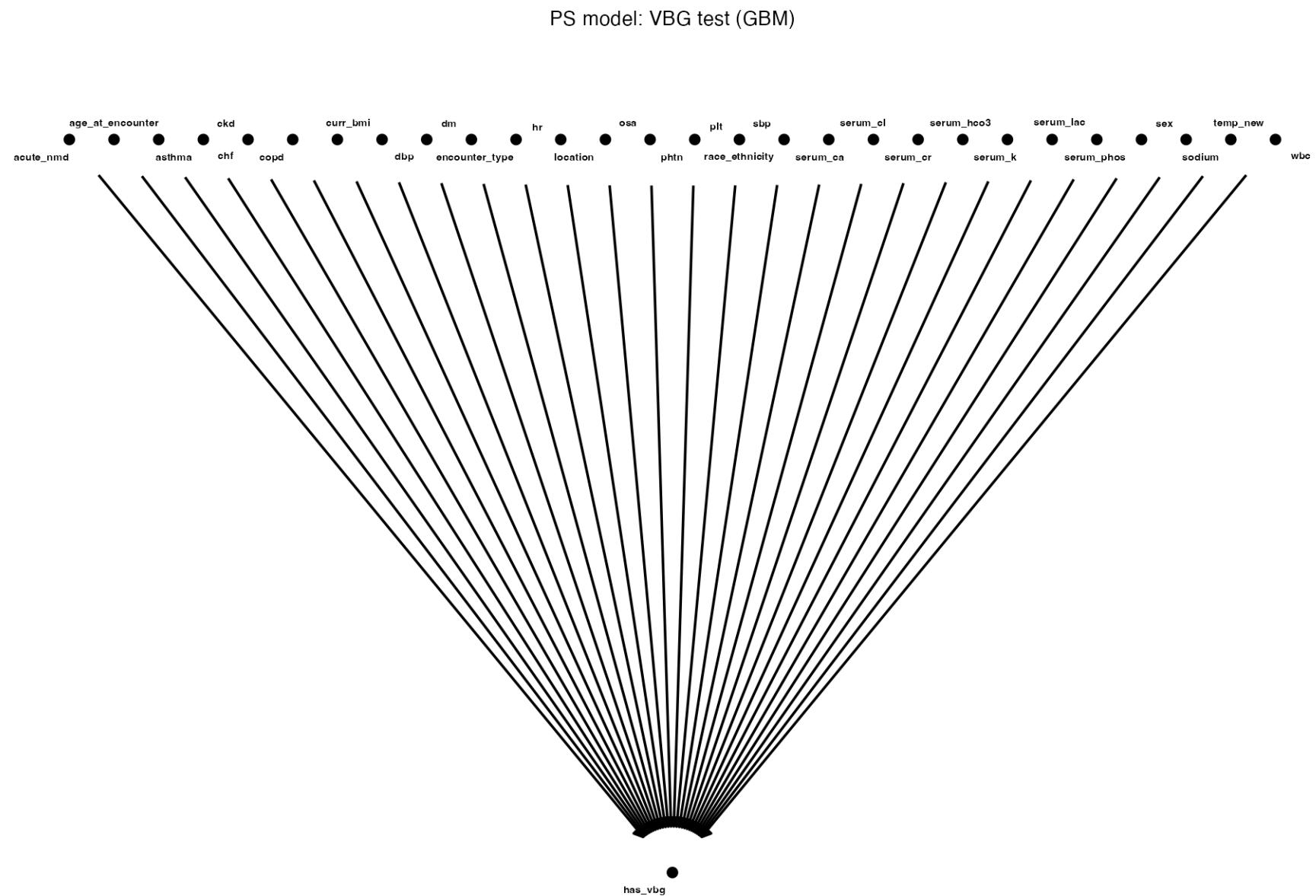
gbm_params <- list(
  n.trees = 1500,
  interaction.depth = 3,
  shrinkage = 0.01,
  bag.fraction = 0.8,
  cv.folds = 0,
  stop.method = "smd.max",
  n.cores = parallel::detectCores()
)
stopifnot(gbm_params$stop.method == "smd.max")
ps_trunc_quantile <- 0.01
stopifnot(ps_trunc_quantile > 0, ps_trunc_quantile < 0.5)

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

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



```
register_model_diagram("PS model: VBG test (GBM)", formula_vbg, width = 10, height = 7)
```



Chunk propensity-config runtime: 1.49 s

1.2.4 Creating subset_data

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

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

table(subset_data$encounter_type)
```

```
2      3
171727 343559
```

```
dim(subset_data)
```

```
[1] 515286    546
```

Chunk sample-subset-data runtime: 4.84 s

```
Emergency Inpatient
171727    343559
```

Chunk schema-normalize runtime: 1.81 s

1.2.5 Generating Codebook for the Full Dataset

Chunk codebook-export-full runtime: 90.45 s

1.2.6 Outcome Variable Creation

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

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

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

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

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

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

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

```

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

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

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

```

Chunk derive-death-60d runtime: 1.59 s

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

0	1
461485	53801

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

0	1
0.89559	0.10441

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

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

Chunk death-60d-summary runtime: 0.03 s

1.3 2) Baseline tables

1.3.1 2.1 Table 1A and 1B:

```
# Robust derivation of analysis variables + helper for Table 1 production
# -----
# helper: label binary 0/1 → "No"/"Yes"
bin_lab <- function(x) {
  y <- to01(x)
  if (all(is.na(y))) {
    return(factor(y, levels = c(0, 1), labels = c("No", "Yes")))
  }
  factor(y, levels = c(0, 1), labels = c("No", "Yes"))
}

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

```

}

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),
           ~ to01(.)),

    ## 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 (robust to factor or numeric codes)
    sex_label = label_from_codes(sex, c(0, 1), c("Female", "Male")),
    race_ethnicity_label = label_from_codes(

```

```

race_ethnicity,
0:6,
c("White", "Black or African American", "Hispanic",
  "Asian", "American Indian", "Pacific Islander", "Unknown")
),
location_label = label_from_codes(
  location,
  0:3,
  c("South", "Northeast", "Midwest", "West")
),
encounter_type_label = label_from_codes(
  encounter_type,
  c(2, 3),
  c("Emergency", "Inpatient")
),
osa_label      = bin_lab(osa),
asthma_label   = bin_lab(asthma),
copd_label     = bin_lab(copd),
chf_label      = bin_lab(chf),
nmd_label      = bin_lab(nmd),
phtn_label     = bin_lab(phtn),
ckd_label      = bin_lab(ckd),
diabetes_label = bin_lab(dm)
)

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

# Table 1 constructor

```

```

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

  df <- data %>%
    filter(!is.na (!!group_sym),                                # drop explicit NA
           !!group_sym != "Missing") %>%                      # drop "Missing" cohort
    mutate (!!group_sym := droplevels (!!group_sym)) %>%  # only drop group levels
    select(all_of(c(group_var, vars_baseline)))

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

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

cat("sex_label: non-NA=", sum(!is.na(subset_data$sex_label)),
    " levels=", length(levels(subset_data$sex_label)), "\n")

```

sex_label: non-NA= 515286 levels= 2

```

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

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

table1A

```

```
table1B
```

Chunk derive-table1-cohorts runtime: 5.94 s

Generating Word Doc for Table 1A & 1B

Chunk export-table1a-table1b-word runtime: 0.63 s

1.3.2 2.2 Table 1 (Overall ABG/VBG status)

```

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

# ABG table with "Everyone" column first
tbl1_abg <- subset_data %>%

```

Variable	No ABG N = 328,044¹	ABG_NoHypercapnia N = 129,429¹	ABG_Hypercapnia N = 5
age_at_encounter	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	62.1 ± 16.4; 0.0/57,813.0 missing (0.0%)
curr_bmi	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.8 ± 7.9; 33,496.0/57,813.0 missing (58.6%)
sex_label			
Female	169,023 (52%)	57,767 (45%)	27,116 (47%)
Male	159,021 (48%)	71,662 (55%)	30,697 (53%)
race_ethnicity_label			
White	200,033 (61%)	81,357 (63%)	39,784 (69%)
Black or African American	62,418 (19%)	19,197 (15%)	8,082 (14%)
Hispanic	23,548 (7.2%)	7,464 (5.8%)	2,757 (4.8%)
Asian	4,880 (1.5%)	2,739 (2.1%)	789 (1.4%)
American Indian	1,971 (0.6%)	1,768 (1.4%)	316 (0.5%)
Pacific Islander	460 (0.1%)	162 (0.1%)	56 (<0.1%)
Unknown	34,734 (11%)	16,742 (13%)	6,029 (10%)
location_label			
South	138,843 (42%)	70,729 (55%)	32,694 (57%)
Northeast	93,209 (28%)	23,262 (18%)	12,975 (22%)
Midwest	22,924 (7.0%)	10,703 (8.3%)	4,844 (8.4%)
West	73,068 (22%)	24,735 (19%)	7,300 (13%)
osa_label	60,653 (18%)	17,709 (14%)	11,965 (21%)
asthma_label	48,456 (15%)	13,049 (10%)	8,268 (14%)
copd_label	60,214 (18%)	21,195 (16%)	18,846 (33%)
chf_label	59,770 (18%)	25,469 (20%)	16,219 (28%)
nmd_label	11,891 (3.6%)	5,861 (4.5%)	2,487 (4.3%)
phtn_label	23,854 (7.3%)	10,513 (8.1%)	7,347 (13%)
ckd_label	54,528 (17%)	24,849 (19%)	11,769 (20%)
diabetes_label	93,007 (28%)	37,426 (29%)	18,521 (32%)
encounter_type_label			
Emergency	142,713 (44%)	19,196 (15%)	9,818 (17%)
Inpatient	185,331 (56%)	110,233 (85%)	47,995 (83%)

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

Variable	No VBG N = 365,623¹	VBG_NoHypercapnia N = 105,646¹	VBG_Hypercapnia N = 49,977¹
age_at_encounter	59.4 ± 17.8; 0.0/365,623.0 missing (0.0%)	58.1 ± 17.8; 0.0/105,646.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 missing (0.0%)
curr_bmi	31.8 ± 8.5; 192,892.0/365,623.0 missing (52.8%)	28.7 ± 7.2; 69,615.0/105,646.0 missing (65.9%)	29.3 ± 7.9; 31,038.0/44,017.0 missing (52.8%)
sex_label			
Female	184,619 (50%)	48,931 (46%)	20,356 (46%)
Male	181,004 (50%)	56,715 (54%)	23,661 (54%)
race_ethnicity_label			
White	241,114 (66%)	55,100 (52%)	24,960 (57%)
Black or African American	61,814 (17%)	19,199 (18%)	8,684 (20%)
Hispanic	22,951 (6.3%)	8,354 (7.9%)	2,464 (5.6%)
Asian	5,439 (1.5%)	2,293 (2.2%)	676 (1.5%)
American Indian	2,128 (0.6%)	1,683 (1.6%)	244 (0.6%)
Pacific Islander	543 (0.1%)	110 (0.1%)	25 (<0.1%)
Unknown	31,634 (8.7%)	18,907 (18%)	6,964 (16%)
location_label			
South	196,774 (54%)	30,426 (29%)	15,066 (34%)
Northeast	65,537 (18%)	44,405 (42%)	19,504 (44%)
Midwest	24,891 (6.8%)	9,178 (8.7%)	4,402 (10%)
West	78,421 (21%)	21,637 (20%)	5,045 (11%)
osa_label	65,748 (18%)	15,634 (15%)	8,945 (20%)
asthma_label	49,810 (14%)	13,419 (13%)	6,544 (15%)
copd_label	70,950 (19%)	16,459 (16%)	12,846 (29%)
chf_label	68,964 (19%)	20,573 (19%)	11,921 (27%)
nmd_label	14,796 (4.0%)	3,754 (3.6%)	1,689 (3.8%)
phtn_label	27,731 (7.6%)	8,534 (8.1%)	5,449 (12%)
ckd_label	61,091 (17%)	21,290 (20%)	8,765 (20%)
diabetes_label	101,173 (28%)	33,665 (32%)	14,116 (32%)
encounter_type_label			
Emergency	124,405 (34%)	34,711 (33%)	12,611 (29%)
Inpatient	241,218 (66%)	70,935 (67%)	31,406 (71%)

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

```

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

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

library(gtsummary)

tbl1 <- tbl_merge(
  tbls = list(tbl1_abg, tbl1_vbg)
) %>%
  modify_caption("**Table 1. Baseline summary: Everyone, ABG status, and VBG status**")

```

tbl1

Chunk table1-everyone-abg-vbg runtime: 2.58 s

1.3.3 2.3 Table 2 (Hypercapnia within cohorts)

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

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

Table 1

Variable	Everyone¹	Did not get ABG N = 328,044¹	Did get ABG N = 18
age_at_encounter	59.2 ± 17.7; 0.0/515,286.0 missing (0.0%)	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	61.2 ± 16.9; 0.0/187,242.0 mi
curr_bmi	31.1 ± 8.4; 293,545.0/515,286.0 missing (57.0%)	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	29.0 ± 7.2; 109,322.0/187,242.0 r
sex_label			
Female	253,906 (49%)	169,023 (52%)	84,883 (45%)
Male	261,380 (51%)	159,021 (48%)	102,359 (55%)
race_ethnicity_label			
White	321,174 (62%)	200,033 (61%)	121,141 (65%)
Black or African American	89,697 (17%)	62,418 (19%)	27,279 (15%)
Hispanic	33,769 (6.6%)	23,548 (7.2%)	10,221 (5.5%)
Asian	8,408 (1.6%)	4,880 (1.5%)	3,528 (1.9%)
American Indian	4,055 (0.8%)	1,971 (0.6%)	2,084 (1.1%)
Pacific Islander	678 (0.1%)	460 (0.1%)	218 (0.1%)
Unknown	57,505 (11%)	34,734 (11%)	22,771 (12%)
location_label			
South	242,266 (47%)	138,843 (42%)	103,423 (55%)
Northeast	129,446 (25%)	93,209 (28%)	36,237 (19%)
Midwest	38,471 (7.5%)	22,924 (7.0%)	15,547 (8.3%)
West	105,103 (20%)	73,068 (22%)	32,035 (17%)
osa_label	90,327 (18%)	60,653 (18%)	29,674 (16%)
asthma_label	69,773 (14%)	48,456 (15%)	21,317 (11%)
copd_label	100,255 (19%)	60,214 (18%)	40,041 (21%)
chf_label	101,458 (20%)	59,770 (18%)	41,688 (22%)
nmd_label	20,239 (3.9%)	11,891 (3.6%)	8,348 (4.5%)
phtn_label	41,714 (8.1%)	23,854 (7.3%)	17,860 (9.5%)
ckd_label	91,146 (18%)	54,528 (17%)	36,618 (20%)
diabetes_label	148,954 (29%)	93,007 (28%)	55,947 (30%)
encounter_type_label			
Emergency	171,727 (33%)	142,713 (44%)	29,014 (15%)
Inpatient	343,559 (67%)	185,331 (56%)	158,228 (85%)

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

```

# VBG cohort (has_vbg == 1)
tbl2_vbg <- subset_data %>%
  filter(has_vbg == 1) %>%
  select(all_of(vars_vbg), 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)
tbl2 <- gtsummary::tbl_merge(
  tbls = list(tbl2_abg, tbl2_vbg),
  tab_spinner = c(NULL, NULL)
) %>%
  gtsummary::modify_caption("**Table 2. Baseline summary by hypercapnia within ABG and VBG cohorts**")

tbl2

```

Chunk tbl2-hypercapnia-cohorts runtime: 2.53 s

Chunk tbl2-outcome-panel runtime: 0.60 s

Table 1

Variable	Got ABG & Hypercapnia ¹	Got ABG & No hypercapnia ¹	Got VBG & Hypercapnia ¹
age_at_encounter	62.1 ± 16.4; 0.0/57,813.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 missing
curr_bmi	29.8 ± 7.9; 33,496.0/57,813.0 missing (57.9%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.3 ± 7.9; 31,038.0/44,017.0 missing
sex_label			
Female	27,116 (47%)	57,767 (45%)	20,356 (46%)
Male	30,697 (53%)	71,662 (55%)	23,661 (54%)
race_ethnicity_label			
White	39,784 (69%)	81,357 (63%)	24,960 (57%)
Black or African American	8,082 (14%)	19,197 (15%)	8,684 (20%)
Hispanic	2,757 (4.8%)	7,464 (5.8%)	2,464 (5.6%)
Asian	789 (1.4%)	2,739 (2.1%)	676 (1.5%)
American Indian	316 (0.5%)	1,768 (1.4%)	244 (0.6%)
Pacific Islander	56 (<0.1%)	162 (0.1%)	25 (<0.1%)
Unknown	6,029 (10%)	16,742 (13%)	6,964 (16%)
location_label			
South	32,694 (57%)	70,729 (55%)	15,066 (34%)
Northeast	12,975 (22%)	23,262 (18%)	19,504 (44%)
Midwest	4,844 (8.4%)	10,703 (8.3%)	4,402 (10%)
West	7,300 (13%)	24,735 (19%)	5,045 (11%)
osa_label	11,965 (21%)	17,709 (14%)	8,945 (20%)
asthma_label	8,268 (14%)	13,049 (10%)	6,544 (15%)
copd_label	18,846 (33%)	21,195 (16%)	12,846 (29%)
chf_label	16,219 (28%)	25,469 (20%)	11,921 (27%)
nmd_label	2,487 (4.3%)	5,861 (4.5%)	1,689 (3.8%)
phtn_label	7,347 (13%)	10,513 (8.1%)	5,449 (12%)
ckd_label	11,769 (20%)	24,849 (19%)	8,765 (20%)
diabetes_label	18,521 (32%)	37,426 (29%)	14,116 (32%)
encounter_type_label			
Emergency	9,818 (17%)	19,196 (15%)	12,611 (29%)
Inpatient	47,995 (83%)	110,233 (85%)	31,406 (71%)
paco2	58.5 ± 20.4; 0.0/57,813.0 missing (0.0%)	35.5 ± 6.1; 0.0/129,429.0 missing (0.0%)	60.2 ± 12.6; 0.0/44,017.0 missing
vbg_co2			

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

Table 2a. Crude outcomes by hypercapnia group

Cohort	Outcome	Hypercapnia	No hypercapnia
ABG	IMV	15373/57813 (26.6%)	32800/129429 (25.3%)
ABG	NIV	9246/57813 (16.0%)	9794/129429 (7.6%)
ABG	Death (60d)	10525/57813 (18.2%)	22193/129429 (17.1%)
ABG	Hypercapnic RF	13588/57813 (23.5%)	6688/129429 (5.2%)
VBG	IMV	8122/44017 (18.5%)	15095/105646 (14.3%)
VBG	NIV	5316/44017 (12.1%)	5383/105646 (5.1%)
VBG	Death (60d)	6697/44017 (15.2%)	13786/105646 (13.0%)
VBG	Hypercapnic RF	9391/44017 (21.3%)	3837/105646 (3.6%)

Counts and column percentages; denominator is non-missing records within each cohort.

1.3.4 2.4 Generating Word Docs for Table 1 and Table 2 (separate files)

Chunk export-table1-tbl2-word runtime: 0.84 s

2 Unweighted Binary Logistic Regressions

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

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

2.0.1 3.1 ABG: Binary hypercapnia models

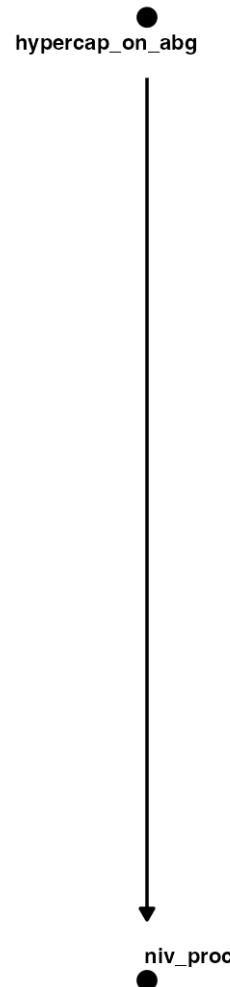
```
abg_bin_forms <- list(
  "ABG binary: IMV ~ hypercapnia"      = imv_proc ~ hypercap_on_abg,
  "ABG binary: NIV ~ hypercapnia"      = niv_proc ~ hypercap_on_abg,
  "ABG binary: Death60d ~ hypercapnia" = death_60d ~ hypercap_on_abg,
  "ABG binary: HCRF ~ hypercapnia"     = hypercap_resp_failure ~ hypercap_on_abg
)
register_model_diagrams(abg_bin_forms)
```

ABG binary: IMV ~ hypercapnia

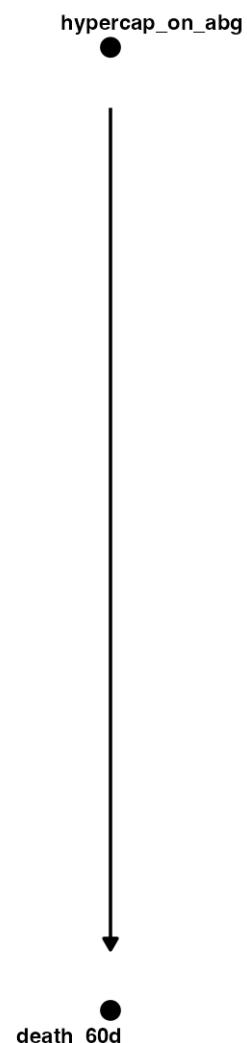
hypercap_on_abg



ABG binary: NIV ~ hypercapnia



ABG binary: Death60d ~ hypercapnia



ABG binary: HCRF ~ hypercapnia

hypercap_on_abg



hypercap_resp_failure

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

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

Coefficients:

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 5

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

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

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

Call:
glm(formula = niv_proc ~ hypercap_on_abg, family = binomial,

```

data = subset_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.860418  0.006533 -437.84 <2e-16 ***
hypercap_on_abg 1.201665  0.013093   91.78 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```

```

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

```

Number of Fisher Scoring iterations: 5

```

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

```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.0572448	0.0065330	-437.84069	0	0.0565151	0.0579811
hypercap_on_abg	3.3256498	0.0130929	91.77958	0	3.2412752	3.4119744

```

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

```

Call:

```

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

```

Coefficients:

```

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

(Dispersion parameter for binomial family taken to be 1)

```

```

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

```

Number of Fisher Scoring iterations: 5

```

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

```

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

```

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

```

Call:

```

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

```

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.355697   0.008192 -409.6 <2e-16 ***
hypercap_on_abg 2.175594   0.012779   170.2 <2e-16 ***

```

```

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 6

```

```

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

```

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

Chunk abg-binary-logit-models runtime: 18.35 s

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

```

modelsummary(
  list("Intubated" = logit_intubated_abg,
       "NIV"      = logit_niv_abg,
       "Death"    = logit_death_abg,
       "ICD Hyper" = logit_icd_abg),
  exponentiate = TRUE,
  conf_level   = 0.95,
  estimate     = "{estimate}",
  statistic    = "{conf.low}, {conf.high}",
  coef_omit    = "(Intercept)",
  gof_omit     = ".*",
  fmt          = 2,
  # drop all goodness-of-fit rows
  # 2 decimal places everywhere
)

```

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

```

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

```

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

Chunk abg-binary-or-table runtime: 21.51 s

Unweighted VBG Group

2.0.2 3.2 VBG: Binary hypercapnia models

```

vbg_bin_forms <- list(
  "VBG binary: IMV ~ hypercapnia"      = imv_proc ~ hypercap_on_vbg,
  "VBG binary: NIV ~ hypercapnia"      = niv_proc ~ hypercap_on_vbg,
  "VBG binary: Death60d ~ hypercapnia" = death_60d ~ hypercap_on_vbg,
  "VBG binary: HCRF ~ hypercapnia"     = hypercap_resp_failure ~ hypercap_on_vbg
)
register_model_diagrams(vbg_bin_forms)

```

VBG binary: IMV ~ hypercapnia

hypercap_on_vbg



imv_proc

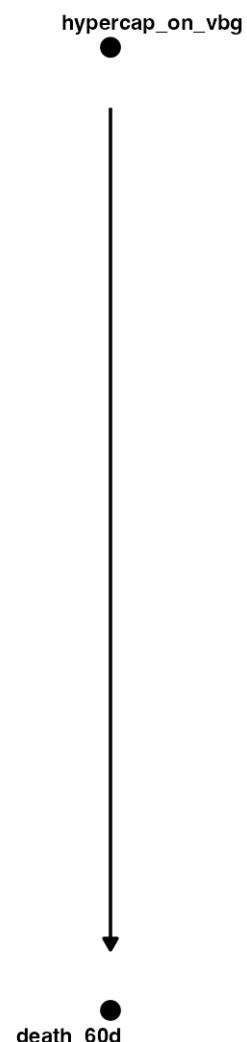
VBG binary: NIV ~ hypercapnia

hypercap_on_vbg



niv_proc

VBG binary: Death60d ~ hypercapnia



VBG binary: HCRF ~ hypercapnia

hypercap_on_vbg



hypercap_resp_failure

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

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

Coefficients:

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 4

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

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

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

Call:
glm(formula = niv_proc ~ hypercap_on_vbg, family = binomial,

```

data = subset_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.735699   0.006091 -449.13 <2e-16 ***
hypercap_on_vbg 0.750555   0.015845   47.37 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```

```

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

```

Number of Fisher Scoring iterations: 5

```

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

```

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

```

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

```

Call:

```

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

```

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept)     -2.197765   0.004856 -452.56 <2e-16 ***
hypercap_on_vbg  0.479895   0.014131   33.96 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 4

```

```

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

```

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

```

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

```

Call:

```

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

```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-3.136462	0.007293	-430.1	<2e-16 ***
hypercap_on_vbg	1.831609	0.013731	133.4	<2e-16 ***

```

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 6

```

```

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

```

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

Chunk vbg-binary-logit-models runtime: 16.95 s

2.0.3 3.3 Display model coefficients for binary hypercapnia on VBG logistic regression

```

modelsummary(
  list("Intubated" = logit_intubated_vbg,
       "NIV"      = logit_niv_vbg,
       "Death"    = logit_death_vbg,
       "ICD Hyper" = logit_icd_vbg),
  exponentiate = TRUE,
  conf_level   = 0.95,
  estimate     = "{estimate}",
  statistic    = "({conf.low}, {conf.high})",
  coef_omit    = "(Intercept)",

```

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

```

gof OMIT      = ".*",
fmt           = 2,
output        = "gt"
) |>
gt_pdf(title = "Odds ratios for VBG hypercapnia (>45 mmHg) on outcomes")

```

Profiled confidence intervals may take longer time to compute.

Use `ci_method="wald"`` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci_method="wald"`` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci_method="wald"`` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci_method="wald"`` for faster computation of CIs.

Profiled confidence intervals may take longer time to compute.

Use `ci_method="wald"`` for faster computation of CIs.

Chunk vbg-binary-or-table runtime: 21.24 s

2.1 3) Three-level PCO2 categories (unweighted)

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

```

subset_data <- subset_data %>%
  mutate(
    pco2_cat_abg = case_when(
      !is.na(paco2) & paco2 < 35 ~ "Hypocapnia",
      !is.na(paco2) & paco2 > 45 ~ "Hypercapnia",
      !is.na(paco2) ~ "Eucapnia"
    ),
    pco2_cat_vbg = case_when(
      !is.na(vbg_co2) & vbg_co2 < 40 ~ "Hypocapnia",

```

```

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

library(broom)
library(tidyr)

```

Attaching package: 'tidyverse'

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

expand, pack, unpack

```

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
    )
}

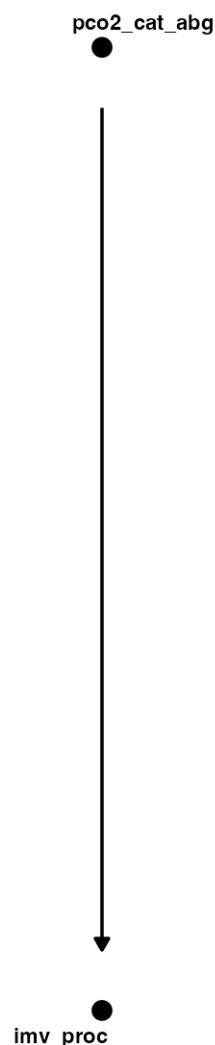
outcomes <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

three_level_forms <- list(

```

```
"ABG 3-level: IMV ~ CO2 category"      = imv_proc ~ pco2_cat_abg,
"ABG 3-level: NIV ~ CO2 category"      = niv_proc ~ pco2_cat_abg,
"ABG 3-level: Death60d ~ CO2 category" = death_60d ~ pco2_cat_abg,
"ABG 3-level: HCRF ~ CO2 category"     = hypercap_resp_failure ~ pco2_cat_abg,
"VBG 3-level: IMV ~ CO2 category"      = imv_proc ~ pco2_cat_vbg,
"VBG 3-level: NIV ~ CO2 category"      = niv_proc ~ pco2_cat_vbg,
"VBG 3-level: Death60d ~ CO2 category" = death_60d ~ pco2_cat_vbg,
"VBG 3-level: HCRF ~ CO2 category"     = hypercap_resp_failure ~ pco2_cat_vbg
)
register_model_diagrams(three_level_forms)
```

ABG 3-level: IMV ~ CO2 category



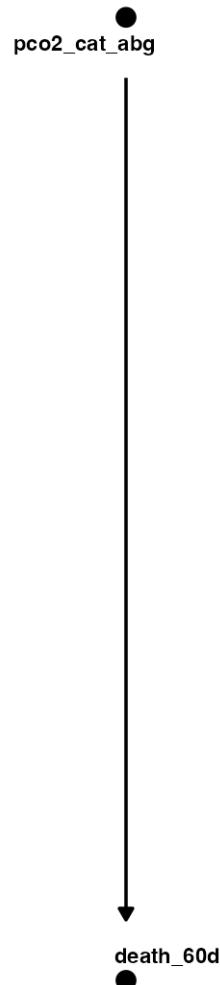
ABG 3-level: NIV ~ CO2 category

pco2_cat_abg

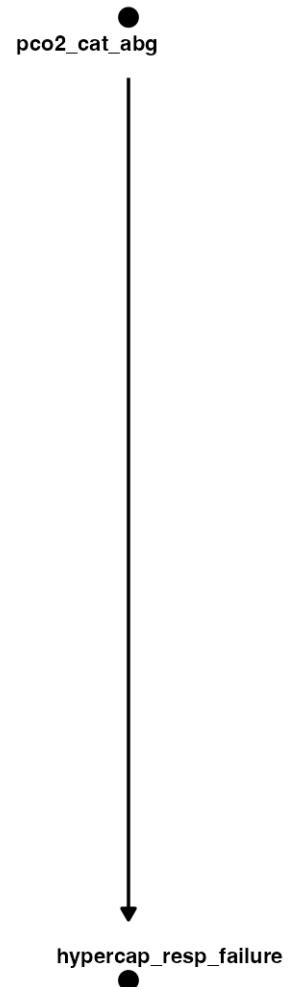


niv_proc

ABG 3-level: Death60d ~ CO2 category



ABG 3-level: HCRF ~ CO2 category



VBG 3-level: IMV ~ CO2 category

pco2_cat_vbg



imv_proc

VBG 3-level: NIV ~ CO2 category

pco2_cat_vbg



niv_proc

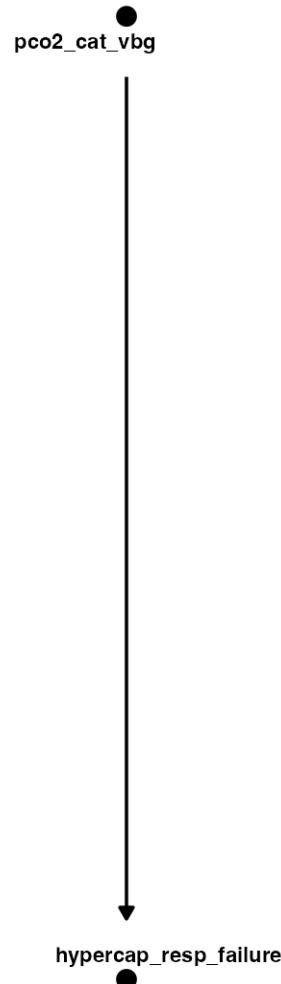
VBG 3-level: Death60d ~ CO2 category

pco2_cat_vbg



death_60d

VBG 3-level: HCRF ~ CO2 category



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

```

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

```

Chunk or-data-three-level-unweighted runtime: 20.74 s

```

library(scales)

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

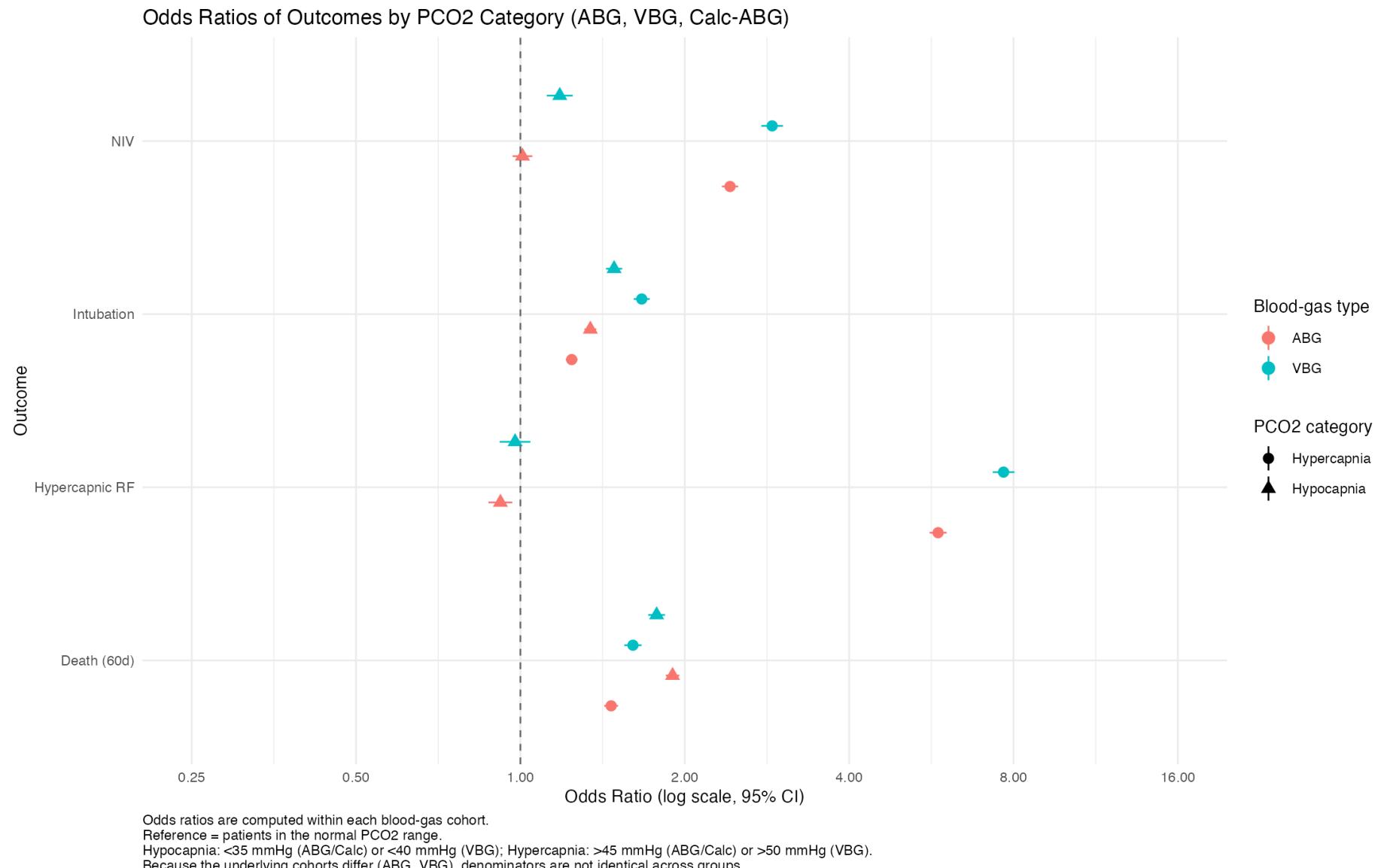
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +

```

```

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

```



Chunk or-plot-three-level-unweighted runtime: 0.30 s

2.2 4) Restricted cubic spline regressions (unweighted)

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

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

Chunk rcs-abg-data-prep runtime: 0.02 s

2.2.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```
abg_rcs_forms <- list(
  "ABG RCS: IMV ~ PaCO2"      = imv_proc ~ rcs(paco2, 4),
  "ABG RCS: NIV ~ PaCO2"      = niv_proc ~ rcs(paco2, 4),
  "ABG RCS: Death60d ~ PaCO2" = death_60d ~ rcs(paco2, 4),
  "ABG RCS: HCRF ~ PaCO2"     = hypercap_resp_failure ~ rcs(paco2, 4)
)
register_model_diagrams(abg_rcs_forms)
```

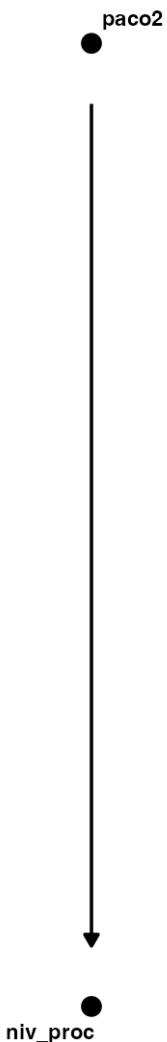
ABG RCS: IMV ~ PaCO₂

paCO₂

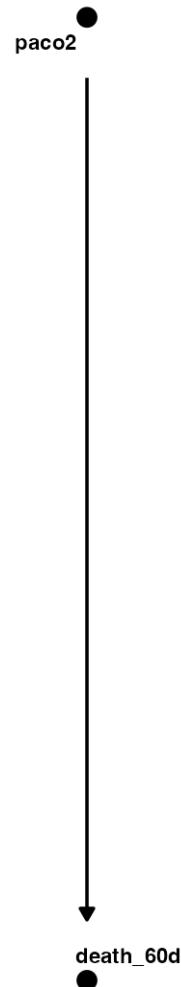


imv_proc

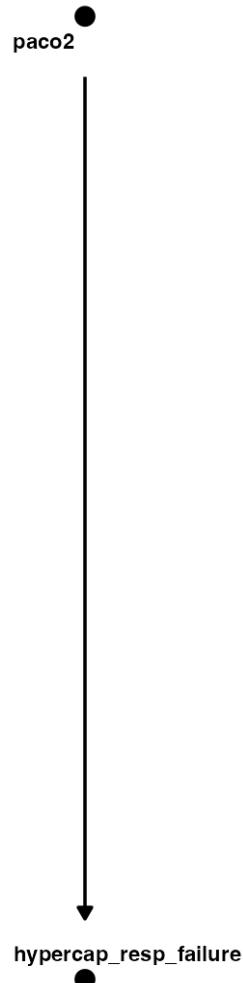
ABG RCS: NIV ~ PaCO₂



ABG RCS: Death60d ~ PaCO2



ABG RCS: HCRF ~ PaCO₂



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

plot_imv <- ggplot(pred_imv, aes(x = paco2, y = yhat)) +
```

```

geom_line(color = "blue", linewidth = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "blue", alpha = 0.2) +
  labs(title = "Probability of Intubation by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

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", linewidth = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "green", alpha = 0.2) +
  labs(title = "Probability of NIV by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

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

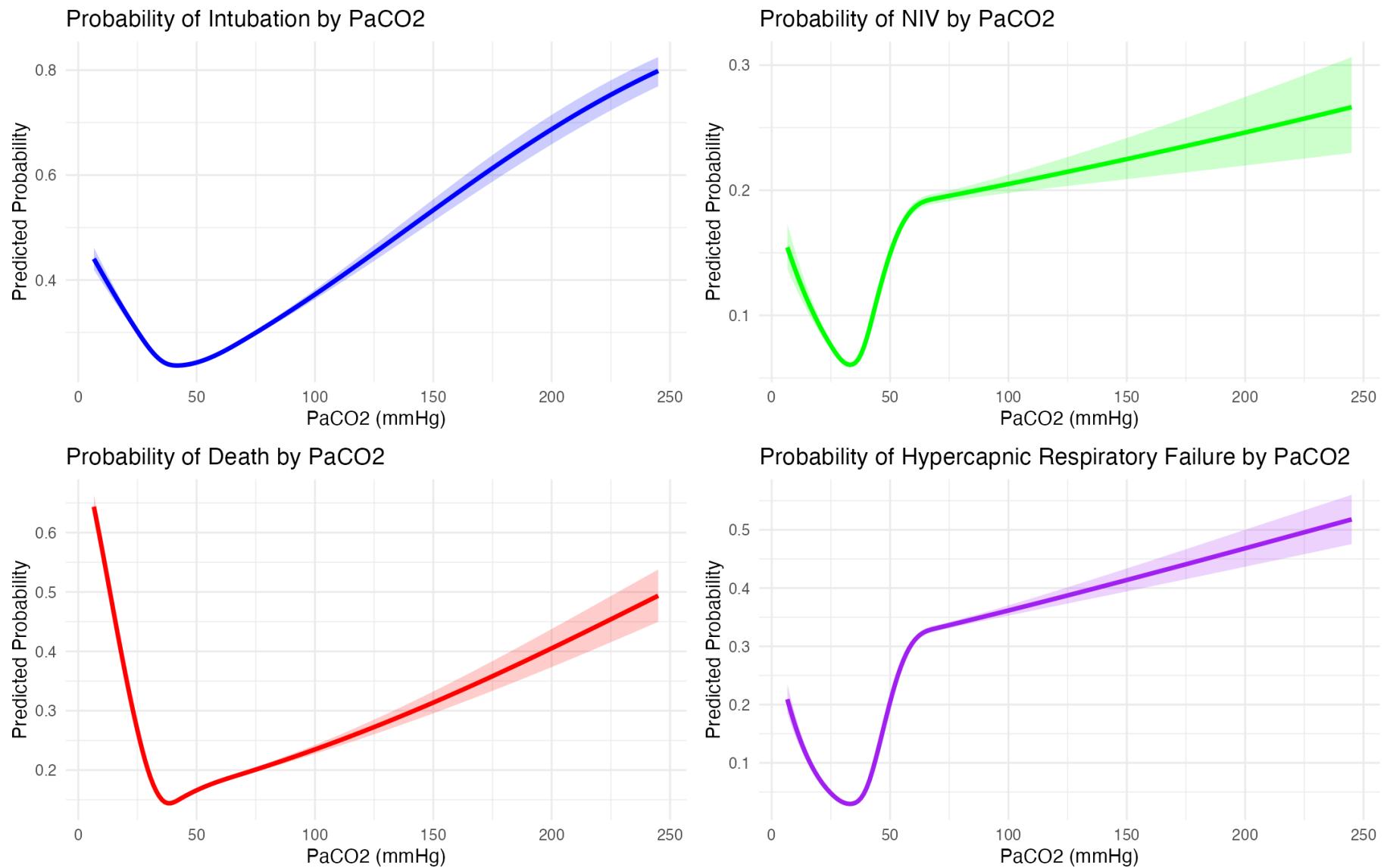
plot_death <- ggplot(pred_death, aes(x = paco2, y = yhat)) +
  geom_line(color = "red", linewidth = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.2) +
  labs(title = "Probability of Death by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

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

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

```

(plot_imv | plot_niv) / (plot_death | plot_hcrf)



Chunk rcs-abg-unweighted-models runtime: 2.36 s

2.2.2 4.2 Unweighted, Restricted Cubic Spline - VBG

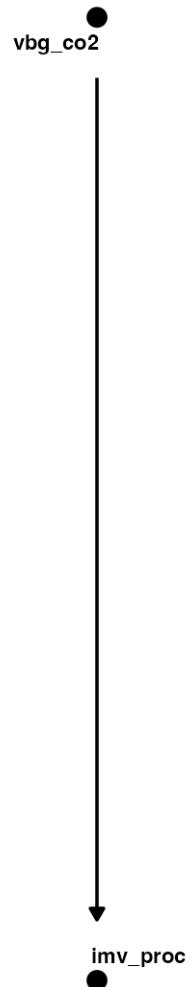
```
# --- VBG dataset ---
subset_data_vbg <- subset_data %>%
  dplyr::select(vbg_co2, imv_proc, niv_proc, death_60d, hypercap_resp_failure) %>%
  dplyr::filter(!is.na(vbg_co2) & complete.cases(.))

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

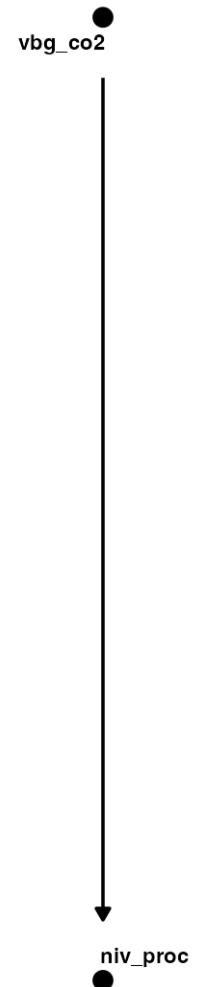
Chunk rcs-vbg-data-prep runtime: 0.03 s

```
vbg_rcs_forms <- list(
  "VBG RCS: IMV ~ C02"      = imv_proc ~ rcs(vbg_co2, 4),
  "VBG RCS: NIV ~ C02"      = niv_proc ~ rcs(vbg_co2, 4),
  "VBG RCS: Death60d ~ C02" = death_60d ~ rcs(vbg_co2, 4),
  "VBG RCS: HCRF ~ C02"     = hypercap_resp_failure ~ rcs(vbg_co2, 4)
)
register_model_diagrams(vbg_rcs_forms)
```

VBG RCS: IMV ~ CO2



VBG RCS: NIV ~ CO2



VBG RCS: Death60d ~ CO2

vbg_co2



death_60d

VBG RCS: HCRF ~ CO2



```

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

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

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

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

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

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

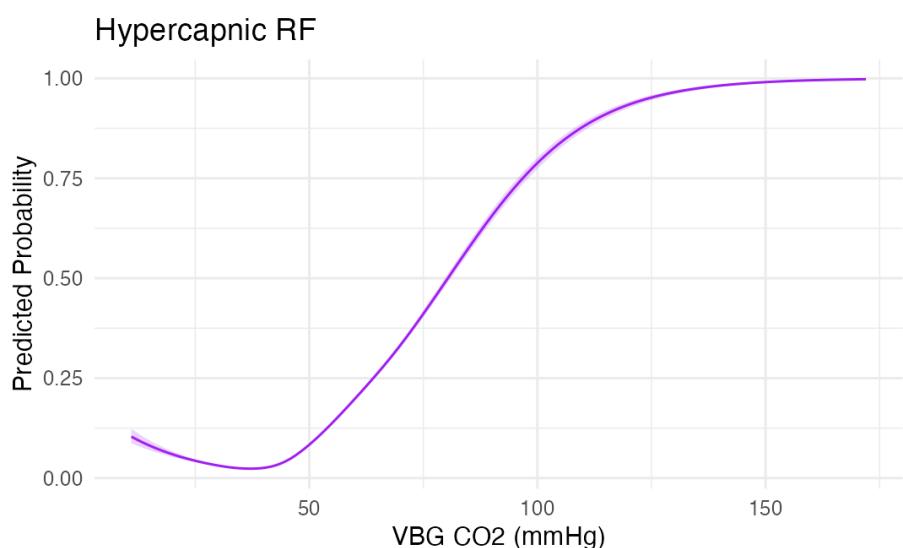
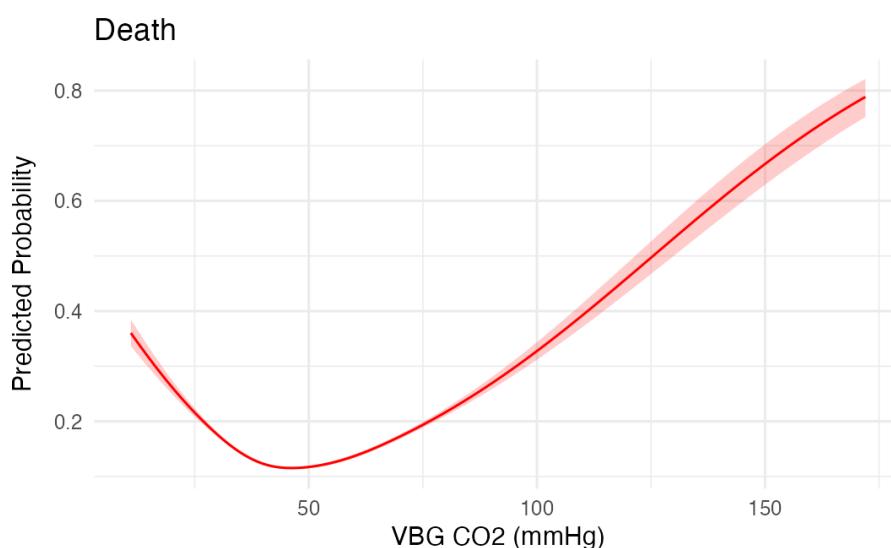
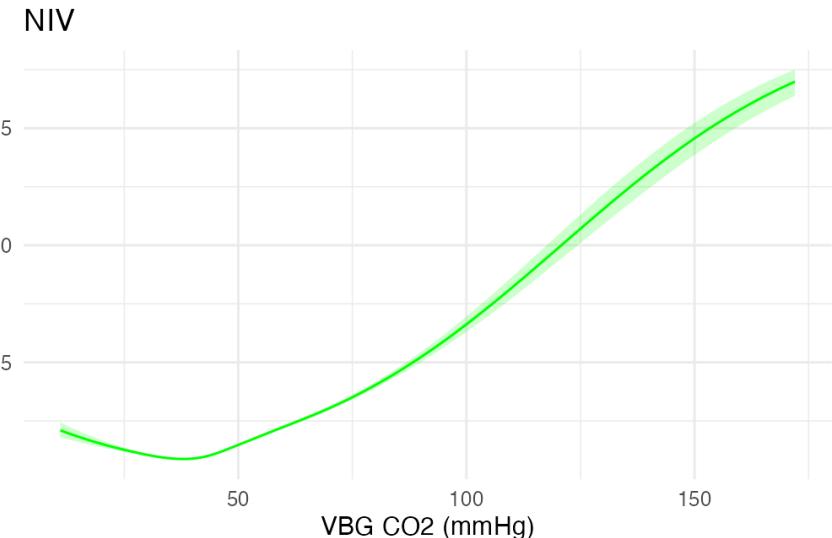
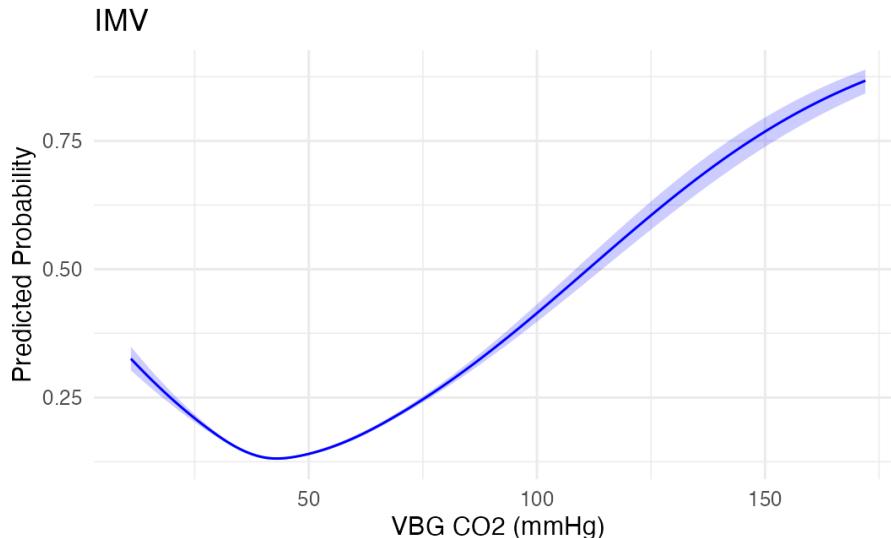
plot_hcrcf_vbg <- ggplot(pred_hcrcf_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "purple") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.2) +
  labs(title = "Hypercapnic RF", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +

```

```
theme_minimal()

((plot_imv_vbg | plot_niv_vbg) /
(plot_death_vbg | plot_hcrf_vbg)) +
plot_annotation(title = "Predicted Probability by VBG C02 (RCS Models)")
```

Predicted Probability by VBG CO₂ (RCS Models)



Chunk rcs-vbg-unweighted-models runtime: 2.14 s

3 Inverse Propensity Weighting

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

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

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

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

3.0.1 5.1 ABG IPW weighting and diagnostics

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

Chunk encode-encounter-type runtime: 0.01 s

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

```

# 1. fit GBM propensity model, ABG
set.seed(42)
weight_data <- normalize_types(subset_data, levels_ref)

weight_model <- do.call(
  weightit,
  c(
    list(
      formula_abg,
      data      = weight_data,
      method    = "gbm",
      estimand  = "ATE",
      missing   = "ind",
      include.obj = FALSE
    ),
    gbm_params
  )
)
w_abg_obj <- weight_model

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

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

```

Chunk ipw-abg-weighting runtime: 663.21 s

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

These are univariate outcome models ($\text{outcome} \sim \text{spline}(\text{CO}_2)$), fit separately for ABG and VBG cohorts using `survey::svyglm` with robust (design-based) SEs.

3.0.2 5.2 ABG IPW spline models

```
# set.seed(42) # reproducible GBM fit
#
# # 1. inverse-probability weights for receiving an ABG
#
# # done in the last block, so not needed
#
#
# Model diagrams: IPW ABG spline models
ipw_abg_rcs_forms <- list(
  "ABG IPW RCS: IMV ~ PaCO2"      = imv_proc ~ rcs(paco2, 4),
  "ABG IPW RCS: NIV ~ PaCO2"      = niv_proc ~ rcs(paco2, 4),
  "ABG IPW RCS: Death60d ~ PaCO2" = death_60d ~ rcs(paco2, 4),
  "ABG IPW RCS: HCRF ~ PaCO2"     = hypercap_resp_failure ~ rcs(paco2, 4)
)
register_model_diagrams(ipw_abg_rcs_forms)
```

ABG IPW RCS: IMV ~ PaCO₂

paco₂



imv_proc

ABG IPW RCS: NIV ~ PaCO₂



ABG IPW RCS: Death60d ~ PaCO2

paco2



death_60d

ABG IPW RCS: HCRF ~ PaCO2

paco2



hypercap_resp_failure

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

```

    hypercap_resp_failure, w_abg) %>%
filter(complete.cases(.))

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

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

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

# 4. prediction helper
mkpred <- function(fit, data_ref) {
  # 1. Grid of PaCO2 values
  newd <- data.frame(
    paco2 = seq(min(data_ref$paco2, na.rm = TRUE),
                max(data_ref$paco2, na.rm = TRUE),
                length.out = 200)
  )

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

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

```

```

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

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

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

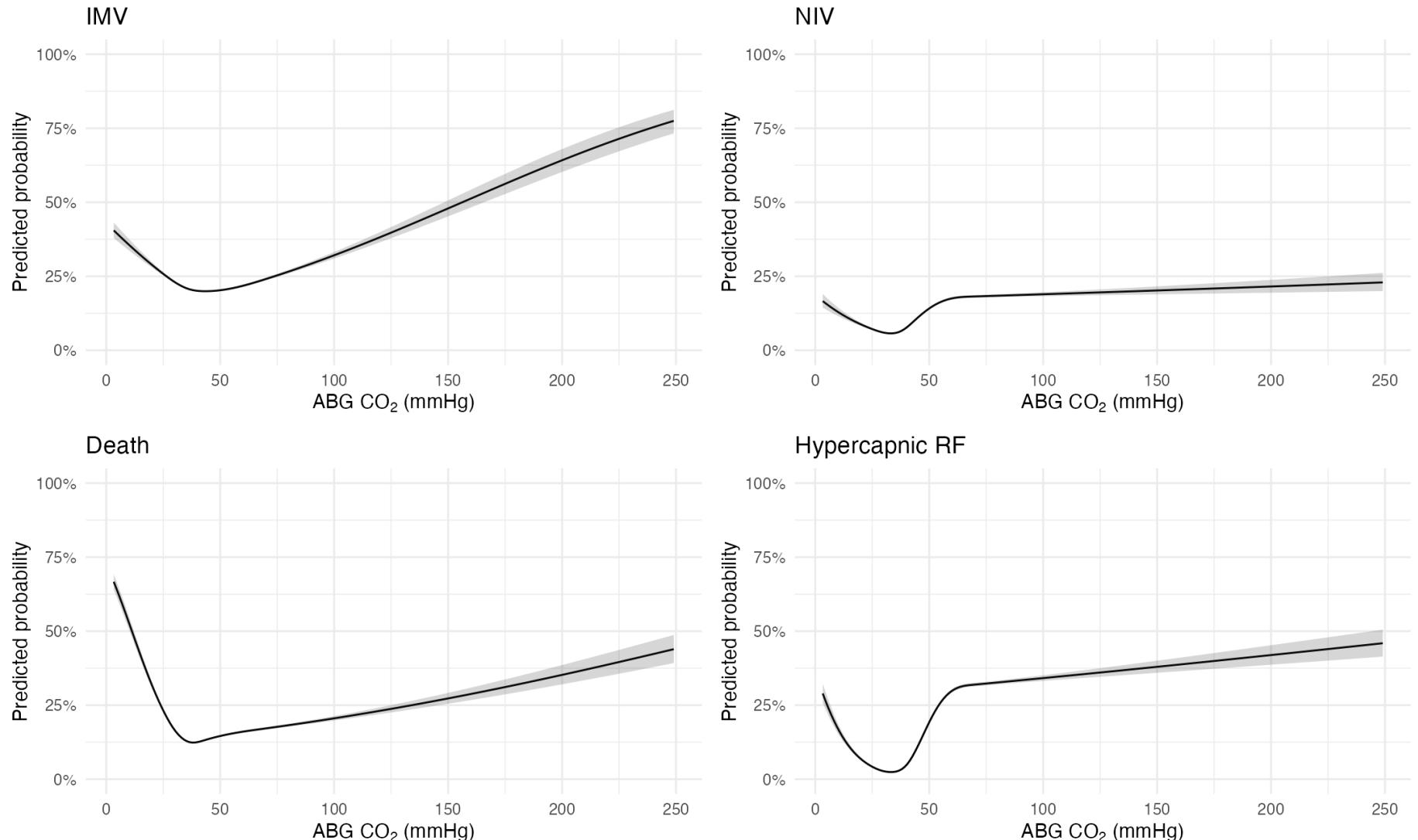
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

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

```

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

Propensity-weighted predicted probability by ABG CO₂ (restricted cubic spline)



Chunk ipw-abg-rcts-models runtime: 4.71 s

Restricting plots bewtween 0.02 and 0.98

3.0.3 5.3 ABG IPW spline models (2–98th percentile)

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

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

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

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

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

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

```

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

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

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

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

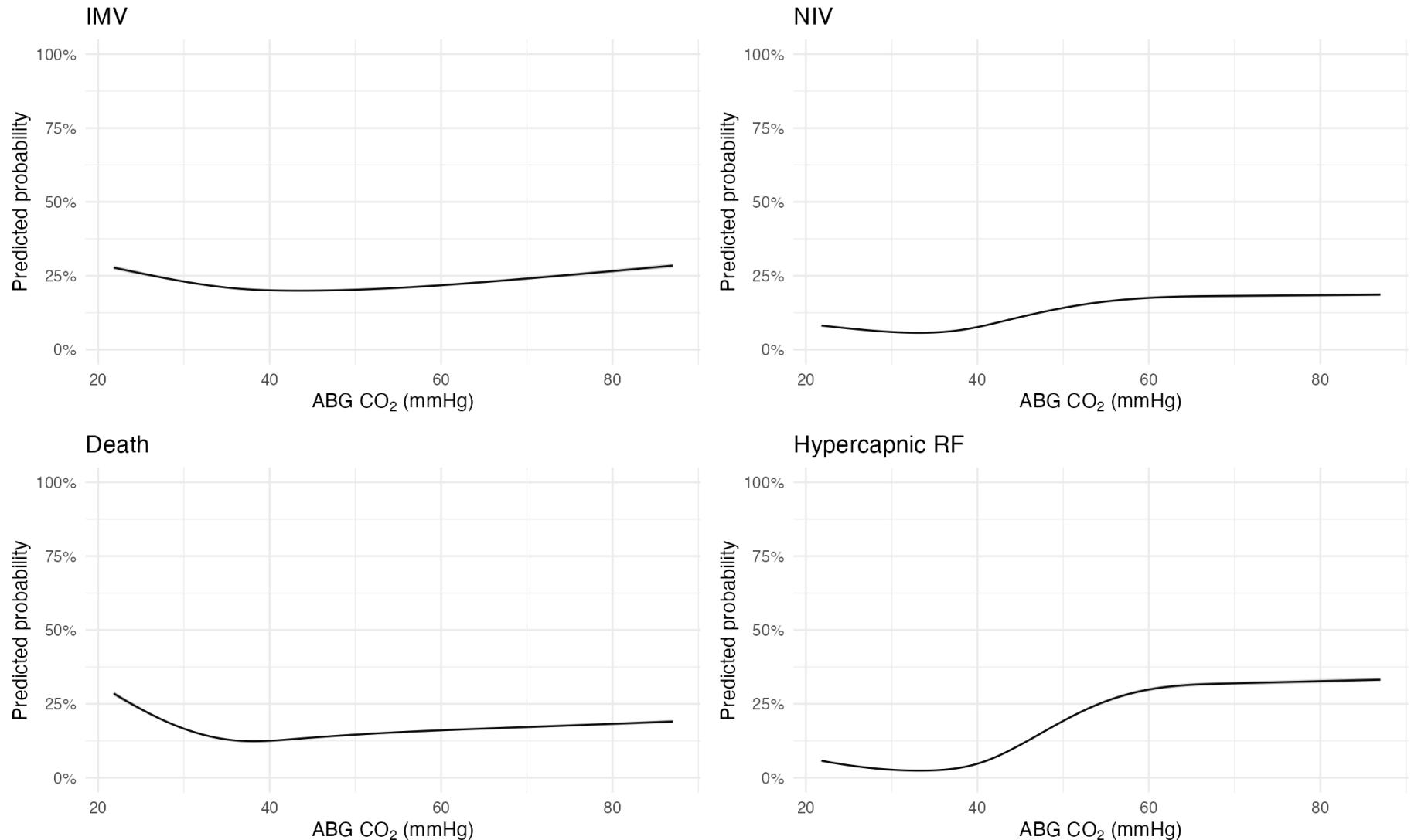
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

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

```

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

Propensity-weighted predicted probability by ABG CO₂ (restricted cubic spline)



Chunk ipw-abg-rcts-trimmed runtime: 4.29 s

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

3.0.4 5.4 VBG IPW weighting and spline models

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

set.seed(42)

# 1. IPW for VBG -----
set.seed(42)
weight_data <- normalize_types(subset_data, levels_ref)
w_vbg <- do.call(
  weightit,
  c(
    list(
      formula_vbg,
      data      = weight_data,
      method    = "gbm",
      estimand  = "ATE",
      missing   = "ind",
      include.obj = FALSE
    ),
    gbm_params
  )
)
w_vbg_obj <- w_vbg

# One-sided IPSW (VBG observed only) + truncation of small propensities
ipow_vbg <- compute_ipow_weights(
  w_vbg,
  treat = weight_data$has_vbg,
  ps_floor_quantile = ps_trunc_quantile,
  stabilize = TRUE
)
w_vbg_ipow <- ipow_vbg$weights
subset_data$ps_vbg <- ipow_vbg$ps
```

```

subset_data$w_vbg <- w_vbg_ipow
assert_finite_weights(w_vbg_ipow[subset_data$has_vbg == 1], "w_vbg")

# Balance diagnostics are handled later.

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

# Model diagrams: IPW VBG spline models
ipw_vbg_rcs_forms <- list(
  "VBG IPW RCS: IMV ~ CO2"      = imv_proc ~ rcs(vbg_co2, 4),
  "VBG IPW RCS: NIV ~ CO2"      = niv_proc ~ rcs(vbg_co2, 4),
  "VBG IPW RCS: Death60d ~ CO2" = death_60d ~ rcs(vbg_co2, 4),
  "VBG IPW RCS: HCRF ~ CO2"     = hypercap_resp_failure ~ rcs(vbg_co2, 4)
)
register_model_diagrams(ipw_vbg_rcs_forms)

```

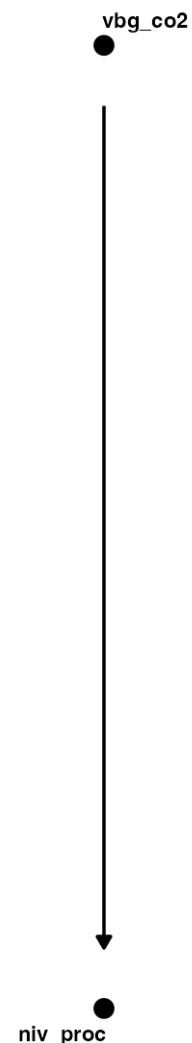
VBG IPW RCS: IMV ~ CO2

vbg_co2



imv_proc

VBG IPW RCS: NIV ~ CO2



VBG IPW RCS: Death60d ~ CO2

vbg_co2



death_60d

vbg_co2



hypercap_resp_failure

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

```

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

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

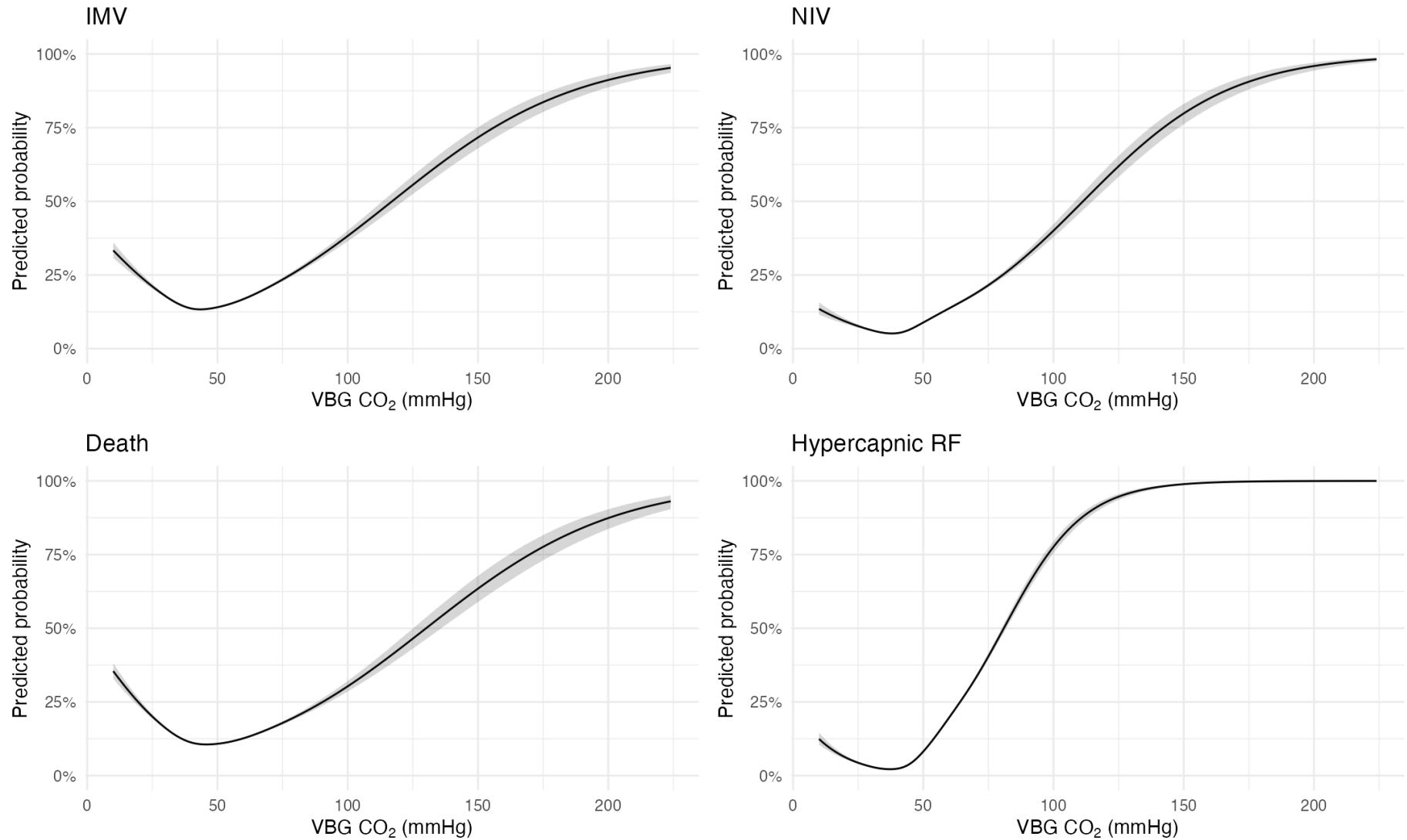
# 5. Plotting (gray scheme) -----
xlab <- expression(paste("VBG CO" [2], " (mmHg)"))

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

```

```
(patchwork::wrap_plots(  
  plt(pred_imv_vbg,    "IMV"),  
  plt(pred_niv_vbg,    "NIV"),  
  plt(pred_death_vbg, "Death"),  
  plt(pred_hcrcf_vbg, "Hypercapnic RF"),  
  ncol = 2  
)  
) +  
  plot_annotation(  
    title = expression(  
      paste("Propensity-weighted predicted probability by VBG CO"[2],  
            " (restricted cubic spline)"))  
)  
)
```

Propensity-weighted predicted probability by VBG CO₂ (restricted cubic spline)



Chunk ipw-vbg-workflow runtime: 664.28 s

Calculated VBG to ABG / Farkas

3.1 5) Weighted effect estimates

New weighted binary regression figures.

```
# IP-weighted odds-ratio plot (ABG, VBG, Calculated-ABG)
#   - exact analogue of the un-weighted figure
#
# Model diagrams: IPW binary hypercapnia models
ipw_hyper_forms <- list(
  "ABG IPW: IMV ~ hypercapnia"      = imv_proc ~ hypercap_on_abg,
  "ABG IPW: NIV ~ hypercapnia"      = niv_proc ~ hypercap_on_abg,
  "ABG IPW: Death60d ~ hypercapnia" = death_60d ~ hypercap_on_abg,
  "ABG IPW: HCRF ~ hypercapnia"     = hypercap_resp_failure ~ hypercap_on_abg,
  "VBG IPW: IMV ~ hypercapnia"      = imv_proc ~ hypercap_on_vbg,
  "VBG IPW: NIV ~ hypercapnia"      = niv_proc ~ hypercap_on_vbg,
  "VBG IPW: Death60d ~ hypercapnia" = death_60d ~ hypercap_on_vbg,
  "VBG IPW: HCRF ~ hypercapnia"     = hypercap_resp_failure ~ hypercap_on_vbg
)
register_model_diagrams(ipw_hyper_forms)
```

ABG IPW: IMV ~ hypercapnia

hypercap_on_abg



imv_proc

ABG IPW: NIV ~ hypercapnia

hypercap_on_abg



niv_proc

ABG IPW: Death60d ~ hypercapnia

hypercap_on_abg



death_60d

ABG IPW: HCRF ~ hypercapnia

hypercap_on_abg



hypercap_resp_failure

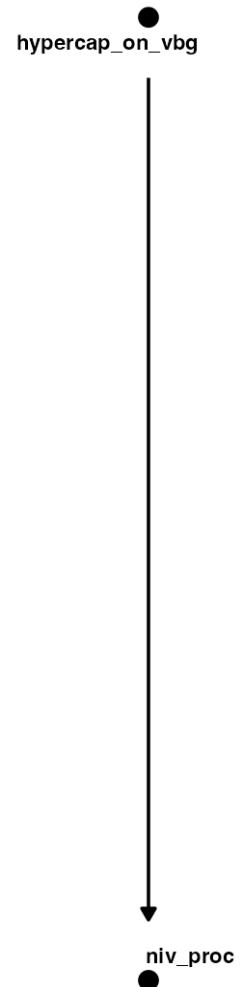
VBG IPW: IMV ~ hypercapnia

hypercap_on_vbg



imv_proc

VBG IPW: NIV ~ hypercapnia



VBG IPW: Death60d ~ hypercapnia

hypercap_on_vbg



death_60d

VBG IPW: HCRF ~ hypercapnia

hypercap_on_vbg



hypercap_resp_failure

```
# weights already attached earlier:  
#   • w_abg           - propensity for *ABG*    (column in subset_data)  
#   • w_vbg           - propensity for *VBG*    (column in subset_data)
```

```

#   •      - same weights, used for calculated ABG CO2

# 1. helper to fit an IP-weighted GLM and return tidy OR -----
tidy_ipw <- function(data, outcome, exposure, weight_var,
                      group_label, outcome_label) {
  des <- svydesign(ids = ~1, weights = as.formula(paste0("~", weight_var)),
                   data = data)
  mod <- svyglm(
    as.formula(paste0(outcome, " ~ ", exposure)),
    design = des,
    family = quasibinomial()
  )
  tidy(mod, exponentiate = TRUE, conf.int = TRUE) %>%
    filter(term == exposure) %>% # keep the exposure row
    mutate(group = group_label, outcome = outcome_label)
}

# 2. cohort-specific data frames -----
abg_df   <- subset_data %>% filter(has_abg == 1)
vbg_df   <- subset_data %>% filter(has_vbg == 1)

# 3. fit models & collect estimates -----
ipw_estimates <- bind_rows(
  # ABG
  tidy_ipw(abg_df, "imv_proc", "hypercap_on_abg", "w_abg", "ABG", "Intubation"),
  tidy_ipw(abg_df, "niv_proc", "hypercap_on_abg", "w_abg", "ABG", "NIV"),
  tidy_ipw(abg_df, "death_60d", "hypercap_on_abg", "w_abg", "ABG", "Death"),
  tidy_ipw(abg_df, "hypercap_resp_failure", "hypercap_on_abg", "w_abg", "ABG", "ICD Code"),

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

```

```

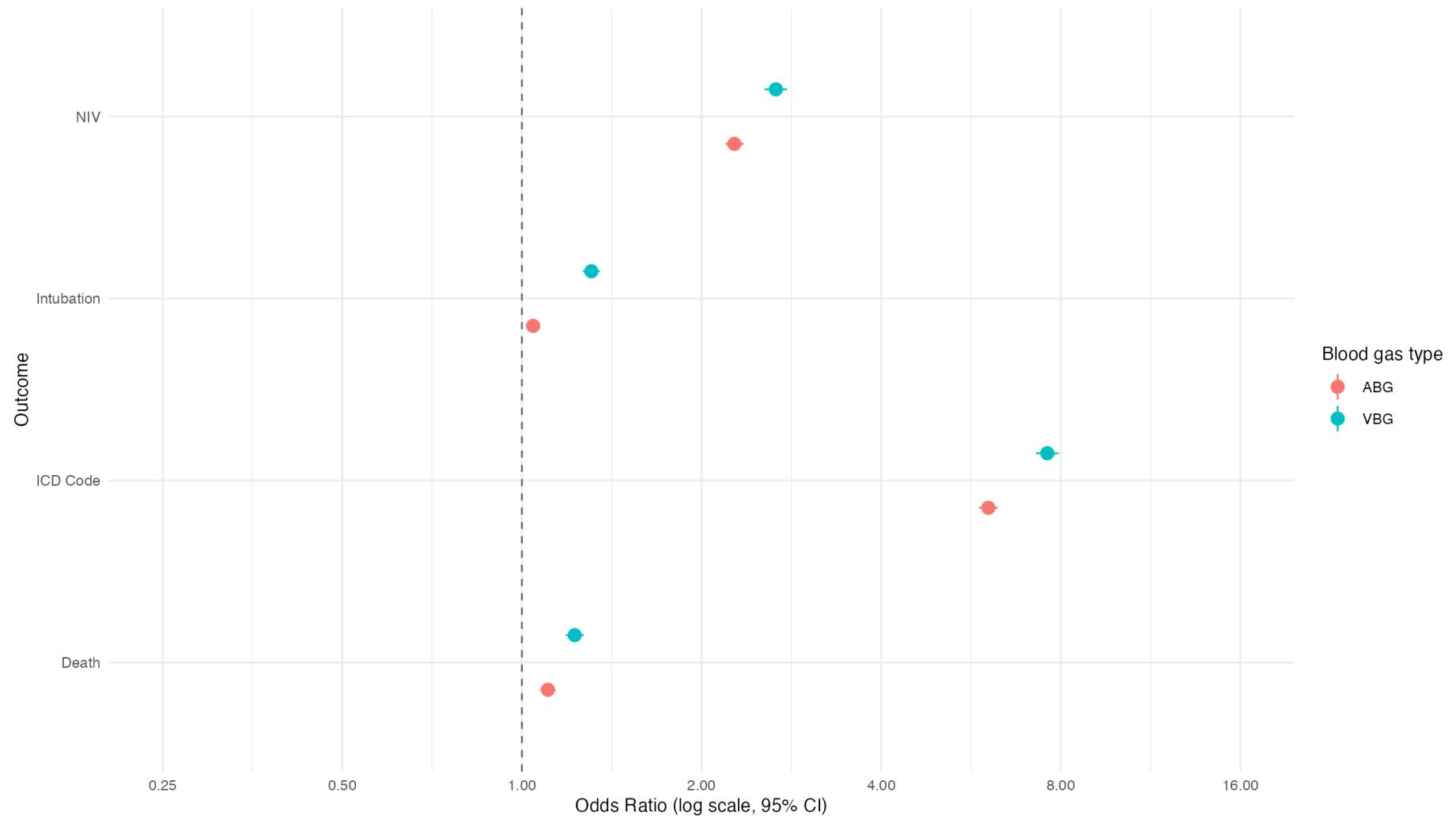
# 4. plotting -----
ipw_estimates$group <- factor(
  ipw_estimates$group,
  levels = c("ABG", "VBG")
)

ggplot(
  ipw_estimates,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group
  )
) +
  geom_pointrange(position = position_dodge(width = 0.6), size = 0.6) +
  geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
  scale_y_log10(
    breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
    limits = c(0.25, 16),
    labels = number_format(accuracy = 0.01)
  ) +
  coord_flip() +
  labs(
    title  = "IP Weighted Odds Ratio of Outcomes When Hypercapnia Present",
    x      = "Outcome",
    y      = "Odds Ratio (log scale, 95% CI)",
    color  = "Blood gas type",
    caption = paste(
      "Inverse-probability weights adjust for covariates associated with receiving each blood-gas.",
      "Models are fitted within their respective cohorts:",
      "ABG (weights = w_abg), VBG (w_vbg).",
      "Numerator = hypercapnic; denominator = normocapnic within cohort.",
      sep = "\n"
    )
  )

```

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

IP Weighted Odds Ratio of Outcomes When Hypercapnia Present



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

Models are fitted within their respective cohorts:

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

Numerator = hypercapnic; denominator = normocapnic within cohort.

Chunk ipw-binary-or-plot runtime: 14.31 s

3.1.1 5.5 Three-level PCO₂ categories (weighted; ABG, VBG)

Three Groups with Weights

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

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

# 2. Function: weighted logistic regression & OR extraction
run_weighted_or <- function(data, outcome, cat_var, weight_var, group_name) {
  dat <- data %>%
    filter(
      !is.na(.data[[cat_var]]),
      !is.na(.data[[outcome]]),
      !is.na(.data[[weight_var]]),
      .data[[weight_var]] > 0
    ) %>%
```

```

mutate(
  !cat_var := factor(.data[[cat_var]],
                      levels = c("Eucapnia", "Hypocapnia", "Hypercapnia"))
) %>%
  droplevels()

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

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

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

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

ipw_three_level_forms <- list(
  "ABG IPW 3-level: IMV ~ CO2 category"      = imv_proc ~ pco2_cat_abg,
  "ABG IPW 3-level: NIV ~ CO2 category"      = niv_proc ~ pco2_cat_abg,
  "ABG IPW 3-level: Death60d ~ CO2 category" = death_60d ~ pco2_cat_abg,
  "ABG IPW 3-level: HCRF ~ CO2 category"     = hypercap_resp_failure ~ pco2_cat_abg,
  "VBG IPW 3-level: IMV ~ CO2 category"      = imv_proc ~ pco2_cat_vbg,
  "VBG IPW 3-level: NIV ~ CO2 category"      = niv_proc ~ pco2_cat_vbg,
  "VBG IPW 3-level: Death60d ~ CO2 category" = death_60d ~ pco2_cat_vbg,
)

```

```
"VBG IPW 3-level: HCRF ~ CO2 category"      = hypercap_resp_failure ~ pco2_cat_vbg
)
register_model_diagrams(ipw_three_level_forms)
```

ABG IPW 3-level: IMV ~ CO2 category

pco2_cat_abg



imv_proc

ABG IPW 3-level: NIV ~ CO2 category

pco2_cat_abg



niv_proc

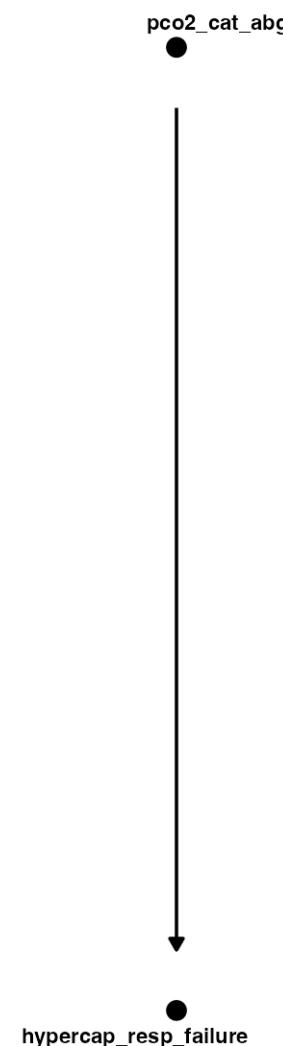
ABG IPW 3-level: Death60d ~ CO2 category

pco2_cat_abg



death_60d

ABG IPW 3-level: HCRF ~ CO2 category



VBG IPW 3-level: IMV ~ CO2 category

pco2_cat_vbg



imv_proc

VBG IPW 3-level: NIV ~ CO2 category

pco2_cat_vbg



niv_proc

VBG IPW 3-level: Death60d ~ CO2 category

pco2_cat_vbg



death_60d

VBG IPW 3-level: HCRF ~ CO2 category

pco2_cat_vbg



hypercap_resp_failure

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

```

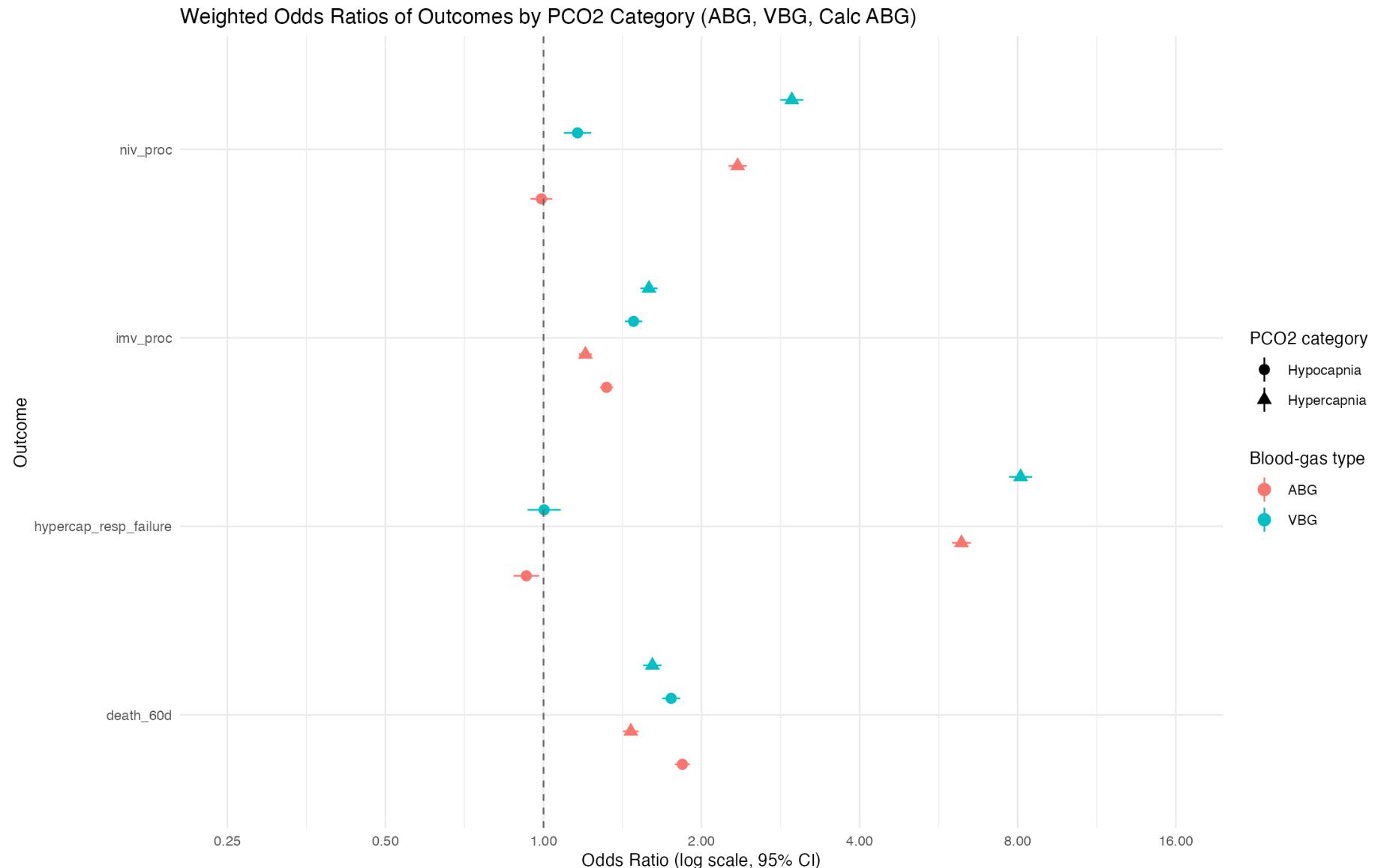
    run_weighted_or(subset_data, out, "pco2_cat_vbg", "w_vbg", "VBG"))
)

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

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

```

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



Chunk ipw-three-level-pco2-all runtime: 23.80 s

3.1.2 5.6 Three-level PCO₂ categories (weighted; ABG vs VBG only)

Three groups with weights: Just ABG and VBG

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

# 2. Function: weighted logistic regression & OR extraction
run_weighted_or <- function(data, outcome, cat_var, weight_var, group_name) {
  dat <- data %>%
    filter(
      !is.na(.data[[cat_var]]),
      !is.na(.data[[outcome]]),
      !is.na(.data[[weight_var]]),
      .data[[weight_var]] > 0
    ) %>%
    mutate(
      !!cat_var := factor(.data[[cat_var]],
                           levels = c("Eucapnia", "Hypocapnia", "Hypercapnia"))
    ) %>%
    droplevels()

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

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

```

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

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

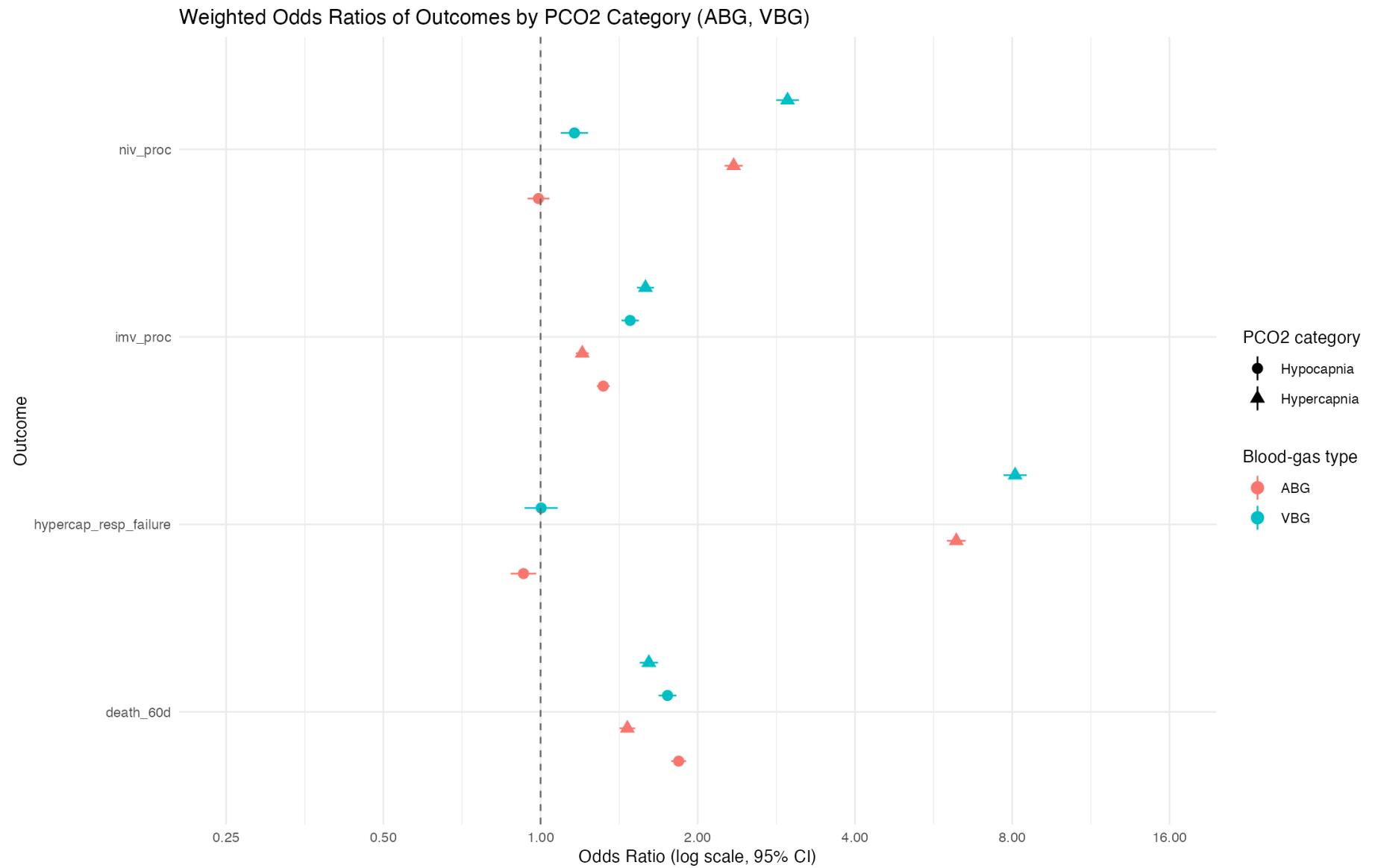
combined_or_df <- bind_rows(
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_abg", "w_abg", "ABG")),
  lapply(outcomes, function(out)
    run_weighted_or(subset_data, out, "pco2_cat_vbg", "w_vbg", "VBG"))
)

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

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

```

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



Chunk ipw-three-level-pco2-abg-vbg runtime: 20.74 s

3.2 6) Propensity score diagnostics

Plotting propensity scores

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

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

# Resolve WeightIt objects regardless of naming used upstream
w_abg_obj <- if (exists("w_abg") && inherits(w_abg, "weightit")) {
  w_abg
} else if (exists("w_abg_obj")) {
  w_abg_obj
} else if (exists("weight_model")) {
  weight_model
} else {
  NULL
}
w_vbg_obj <- if (exists("w_vbg") && inherits(w_vbg, "weightit")) {
  w_vbg
} else if (exists("w_vbg_obj")) {
  w_vbg_obj
} 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      = extract_weightit_ps(w_abg_obj),
```

```

    treat      = subset_data$has_abg,
    ScoreType = "ABG"
  )
)

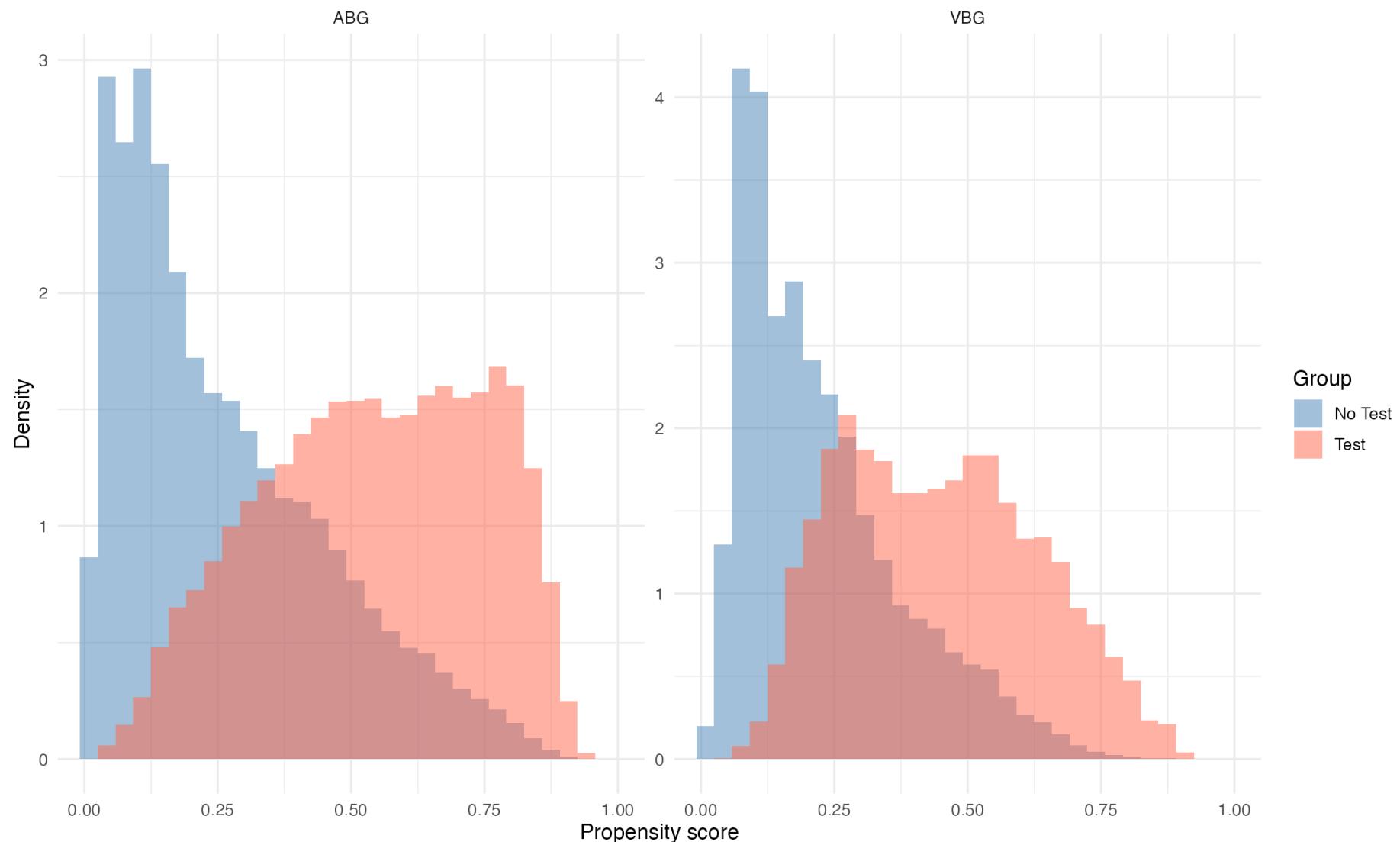
if (!is.null(w_vbg_obj) && "has_vbg" %in% names(subset_data)) {
  ps_dfs$VBG <- data.frame(
    ps        = extract_weightit_ps(w_vbg_obj),
    treat     = subset_data$has_vbg,
    ScoreType = "VBG"
  )
}

# 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"))
  )

# Plot
ggplot(df_ps, aes(x = ps, fill = treat)) +
  geom_histogram(aes(y = after_stat(density)), alpha = 0.5,
                 position = "identity", bins = 30) +
  scale_fill_manual(values = c("No Test" = "steelblue", "Test" = "tomato")) +
  facet_wrap(~ScoreType, scales = "free_y") +
  coord_cartesian(xlim = c(0, 1)) +
  labs(
    title = "Propensity Score Distributions",
    x     = "Propensity score",
    y     = "Density",
    fill  = "Group"
  ) +
  theme_minimal(base_size = 12)

```

Propensity Score Distributions



Chunk propensity-histograms-conditional runtime: 0.55 s

```
w_abg_obj <- if (exists("w_abg") && inherits(w_abg, "weightit")) {  
  w_abg
```

```

} else if (exists("w_abg_obj")) {
  w_abg_obj
} else if (exists("weight_model")) {
  weight_model
} else {
  NULL
}
w_vbg_obj <- if (exists("w_vbg") && inherits(w_vbg, "WeightIt")) {
  w_vbg
} else if (exists("w_vbg_obj")) {
  w_vbg_obj
} else {
  NULL
}

if (is.null(w_abg_obj)) stop("ABG WeightIt object not found for PS plot.")

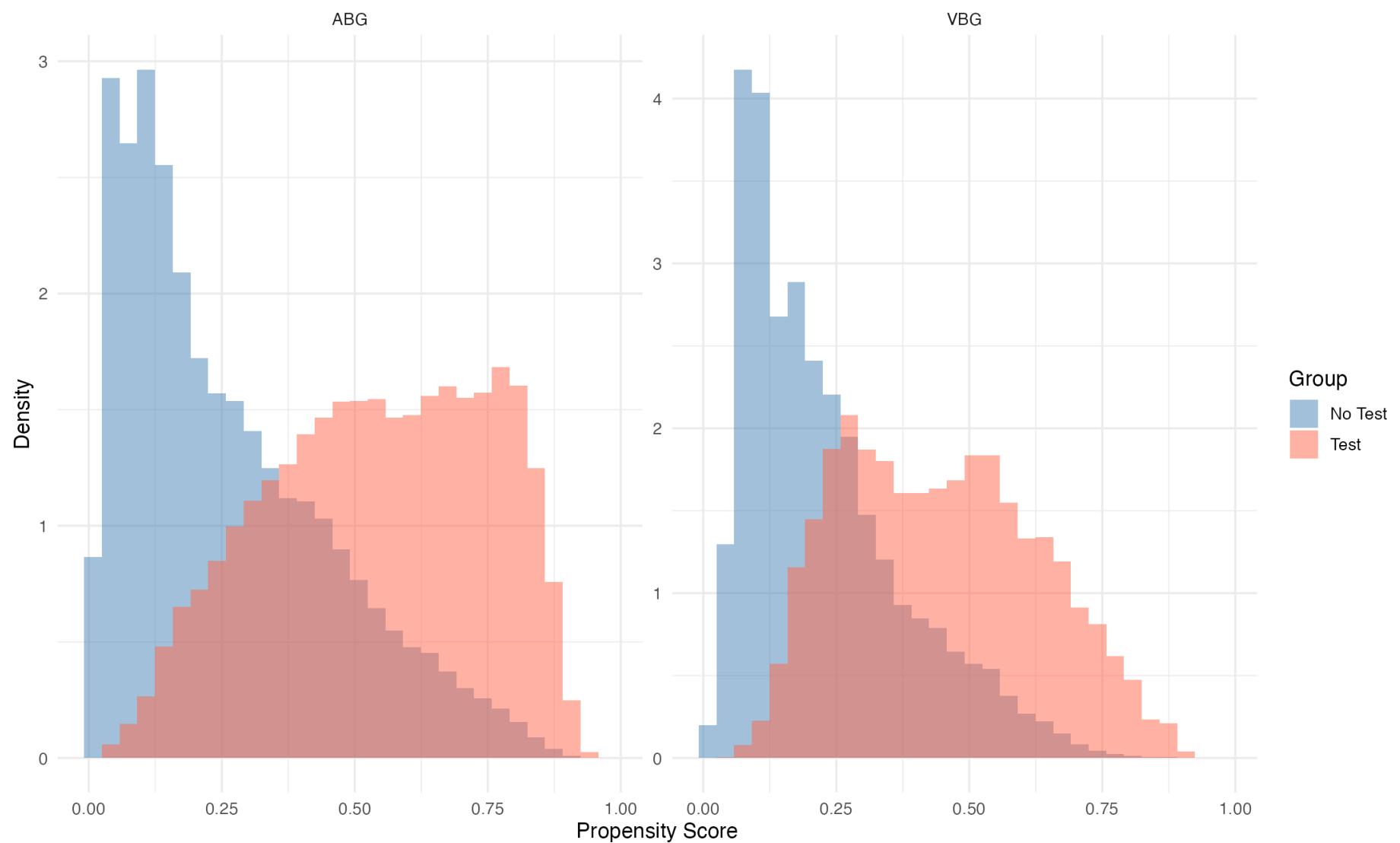
ps_dfs <- list(
  ABG = data.frame(
    ps      = extract_weightit_ps(w_abg_obj),
    treat   = subset_data$has_abg,
    ScoreType = "ABG"
  )
)
if (!is.null(w_vbg_obj) && "has_vbg" %in% names(subset_data)) {
  ps_dfs$VBG <- data.frame(
    ps      = extract_weightit_ps(w_vbg_obj),
    treat   = subset_data$has_vbg,
    ScoreType = "VBG"
  )
}

df_ps <- bind_rows(ps_dfs) %>%
  mutate(
    treat     = factor(treat, levels = c(0,1), labels = c("No Test", "Test")),
    ScoreType = factor(ScoreType, levels = c("ABG", "VBG"))

```

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

Propensity Score Distributions



Chunk propensity-histograms-all runtime: 0.32 s

4 Multiple Imputation Analysis

added 12/6/2025

4.1 7) Packages and reproducibility

```
# Core MI + diagnostics
library(mice)          # chained equations (MICE)
library(miceadds)       # pooling helpers & utilities
library(naniar)         # missingness summaries/plots
library(visdat)         # quick type/missingness viz
library(skimr)          # data skim for large frames
```

Attaching package: 'skimr'

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

n_complete

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

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

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

Attaching package: 'future'

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

cluster

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

mi_mids_file    <- results_path("mi_abg_vbg_mids.rds")
mi_w_abg_file   <- results_path("mi_W_abg_list.rds")
mi_w_vbg_file   <- results_path("mi_W_vbg_list.rds")
mi_pooled_file  <- results_path("mi_pooled_results.rds")

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

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

set.seed(20251206)

# ensure a writable figure dir + stable device on macOS
fs::dir_create(fig_dir, recurse = TRUE)
knitr::opts_chunk$set(fig.path = fig_path, dev = "png", dpi = 144)
options(bitmapType = "cairo") # prevents device issues on macOS
```

Chunk mi-packages runtime: 1.25 s

Chunk mi-settings runtime: 0.00 s

4.1.1 Missing data / imputation summary (pre-imputation)

Chunk mi-missing-summary runtime: 0.05 s

4.1.2 7.1 Missingness audit (what, where, how much)

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

# Use imputed data if available (preferred); otherwise skip
imp_obj <- NULL
if (exists("imp")) imp_obj <- imp
if (is.null(imp_obj) && file.exists(mi_mids_file)) {
  imp_obj <- tryCatch(readRDS(mi_mids_file), error = function(e) NULL)
}

if (is.null(imp_obj)) {
  # MI object not available; skip visuals
} else {
  dat_imp <- mice::complete(imp_obj, action = 1, include = FALSE)

  # 1) Tabular summary on completed data (should be near 0% by design)
  miss_tbl <- naniar::miss_var_summary(dat_imp) %>% arrange(desc(pct_miss))
  print(utils::head(miss_tbl, 40))  # show top 40 in the report

  # 2) Bar plot of top-K (mostly zeros after imputation)
  K <- 40
  top_vars <- miss_tbl$variable[seq_len(min(K, nrow(miss_tbl)))]
  p_top <- ggplot(miss_tbl[miss_tbl$variable %in% top_vars, ],
                  aes(x = reorder(variable, pct_miss), y = pct_miss)) +
```

Pre-imputation missingness

Variable	Missing (n)	Missing (%)
vbg_o2sat	444,799	86.3%
bnp	421,741	81.8%
vbg_co2	365,623	71.0%
spo2	363,720	70.6%
paco2	328,044	63.7%
serum_lac	308,187	59.8%
curr_bmi	293,545	57.0%
serum_phos	273,168	53.0%
temp_new	247,068	47.9%
hr	186,698	36.2%
dbp	154,565	30.0%
sbp	153,161	29.7%
wbc	91,444	17.7%
serum_ca	53,331	10.3%
serum_cr	48,785	9.5%
plt	41,393	8.0%
serum_k	41,266	8.0%
serum_cl	30,694	6.0%
serum_hco3	29,886	5.8%
sodium	27,486	5.3%
age_at_encounter	0	0.0%
sex	0	0.0%
race_ethnicity	0	0.0%
copd	0	0.0%
asthma	0	0.0%
osa	0	0.0%
chf	0	0.0%
acute_nmd	0	0.0%
phtn	0	0.0%
ckd	0	0.0%
dm	0	0.0%
location	0	0.0%
encounter_type	0	0.0%
has_abg	0	0.0%
has_vbg	0	0.0%
imv_proc	0	0.0%
	134	0.0%

```

geom_col() +
coord_flip() +
labs(title = "Top missing variables (after MI)", x = NULL, y = "% missing") +
theme_minimal()
print(p_top)

# 3) Small heatmap on imputed data (rows/cols sampled)
M <- 60
R <- min(1500, nrow(dat_imp))
cols_heat <- head(top_vars, M)
rows_heat <- dplyr::slice_sample(dat_imp, n = R)

p_heat <- naniar::vis_miss(rows_heat[, cols_heat, drop = FALSE]) +
  labs(title = sprintf("Missingness heatmap on imputed data (%d rows × %d cols)", R, length(cols_heat)))
print(p_heat)
}

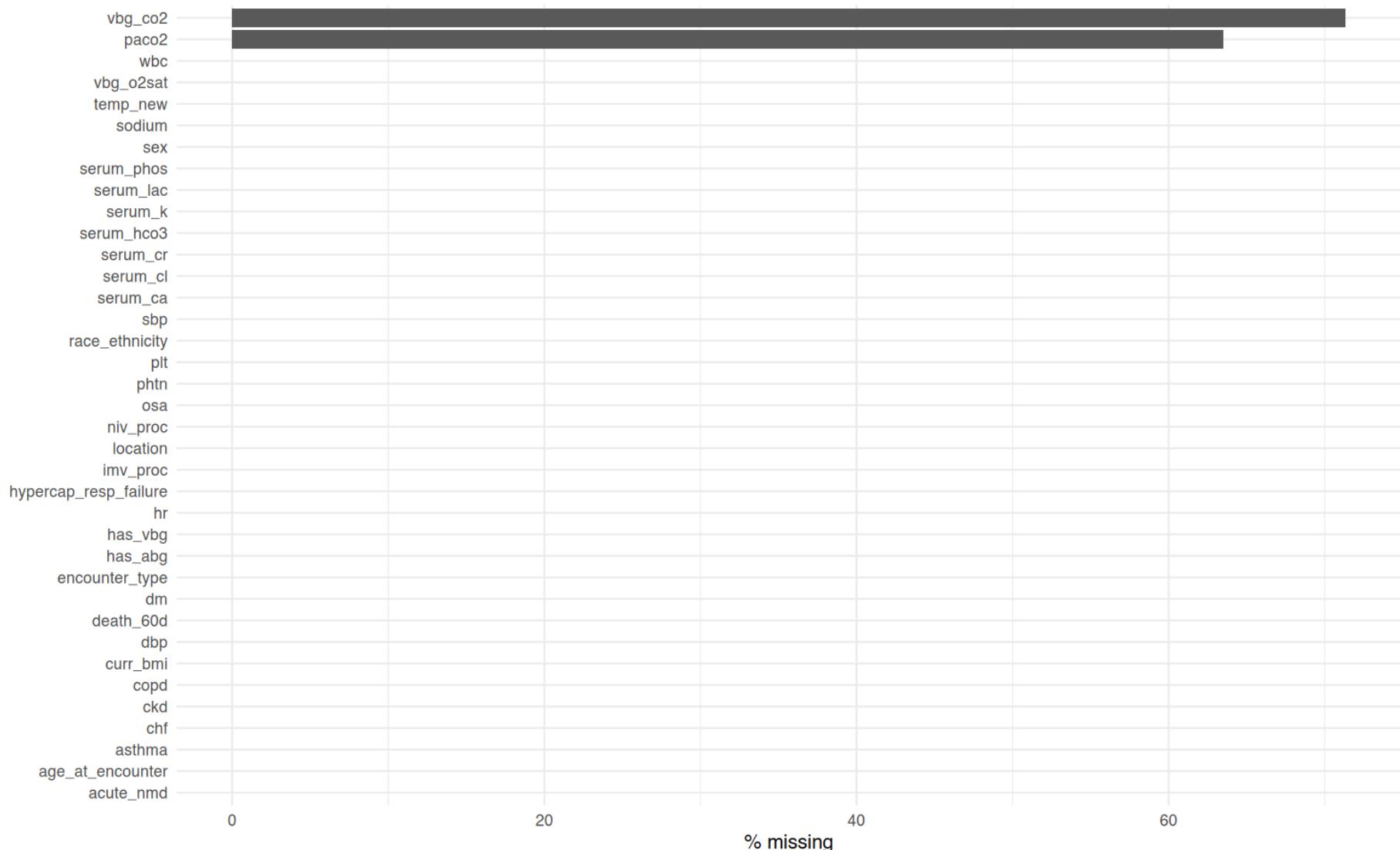
```

```

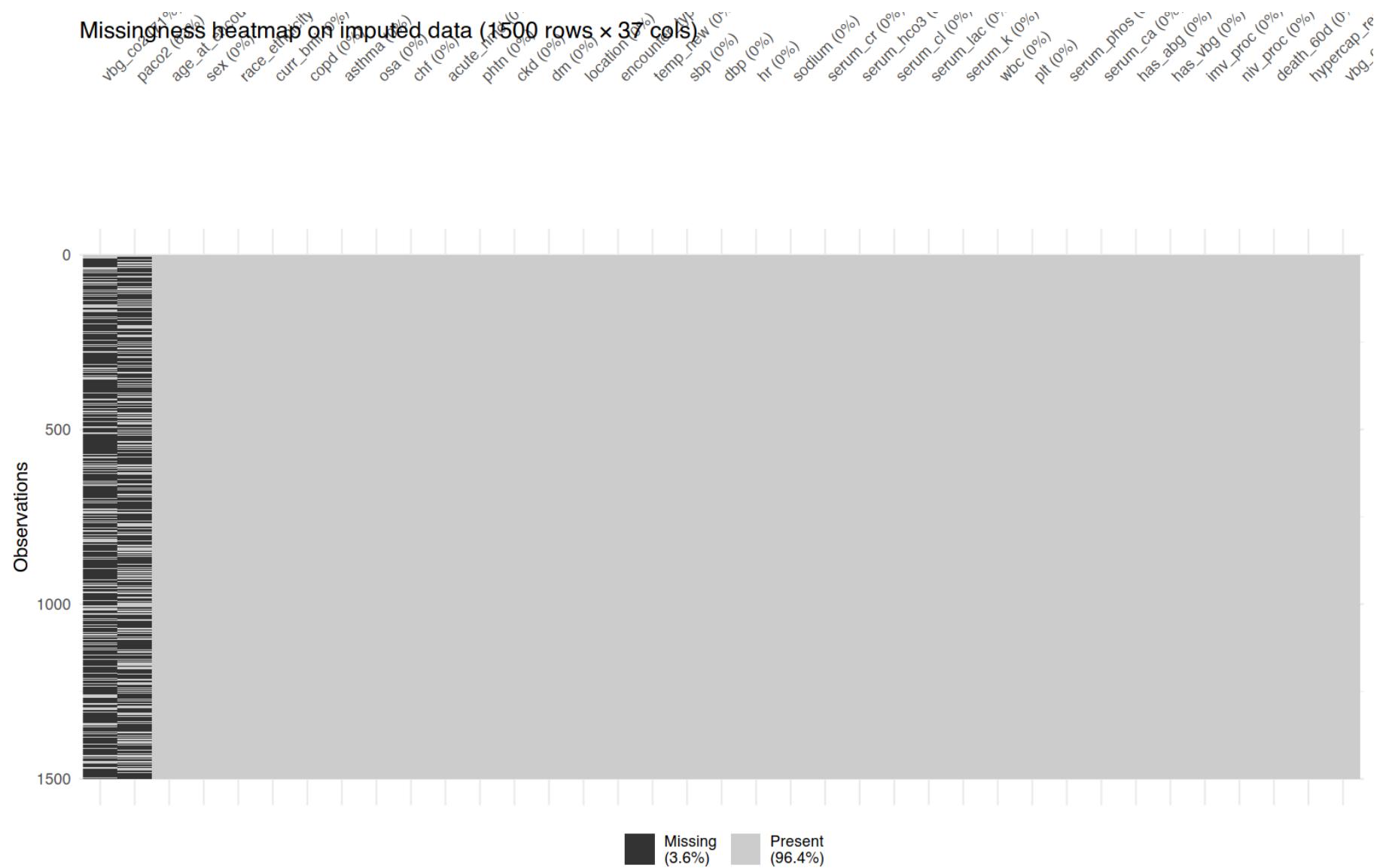
# A tibble: 37 x 3
  variable      n_miss pct_miss
  <chr>        <int>    <num>
1 vbg_co2       18342     71.3
2 paco2         16341     63.5
3 age_at_encounter     0      0
4 sex            0      0
5 race_ethnicity     0      0
6 curr_bmi        0      0
7 copd           0      0
8 asthma          0      0
9 osa             0      0
10 chf            0      0
# i 27 more rows

```

Top missing variables (after MI)



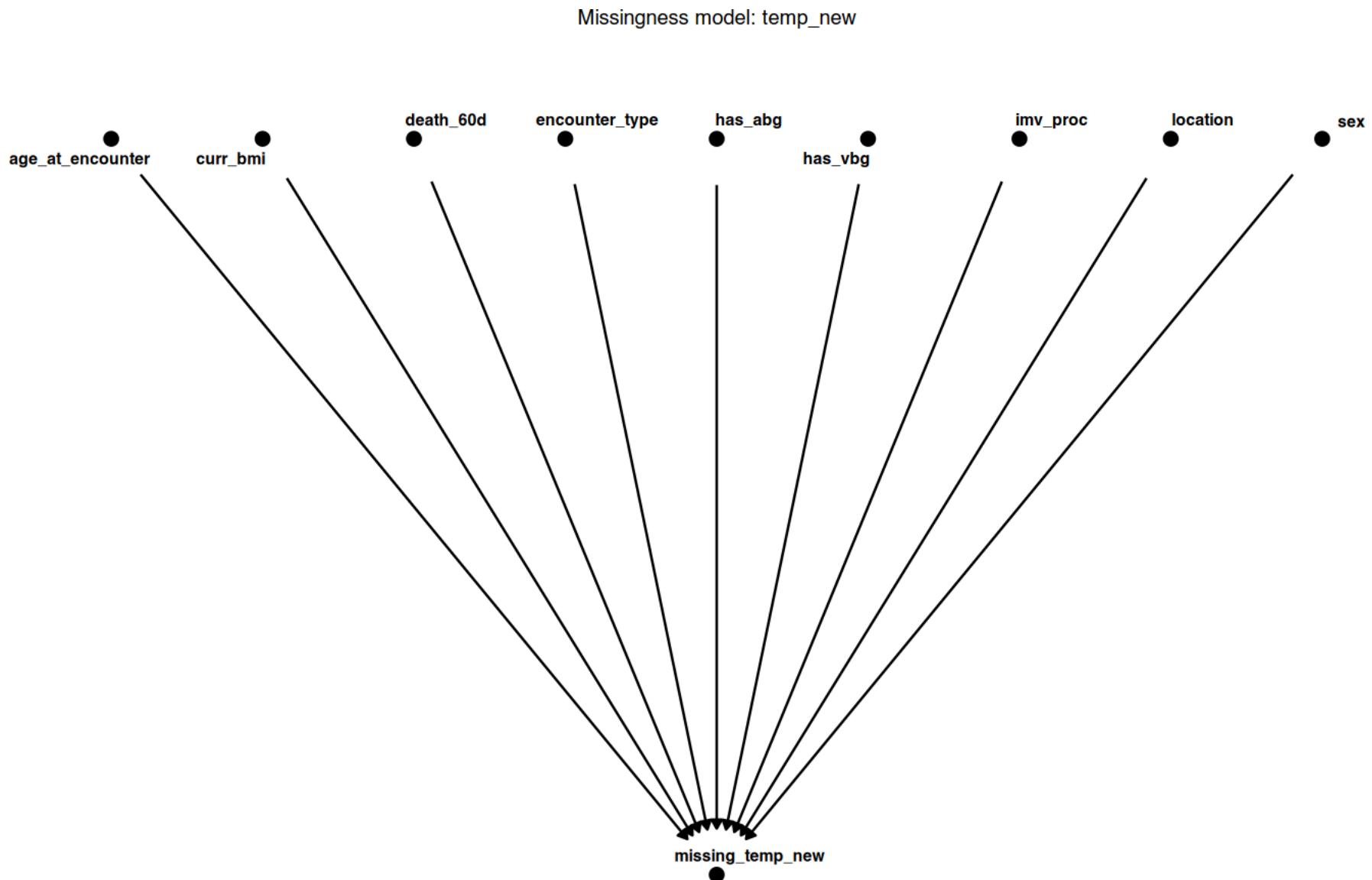
Missingness heatmap on imputed data (1500 rows x 37 cols)



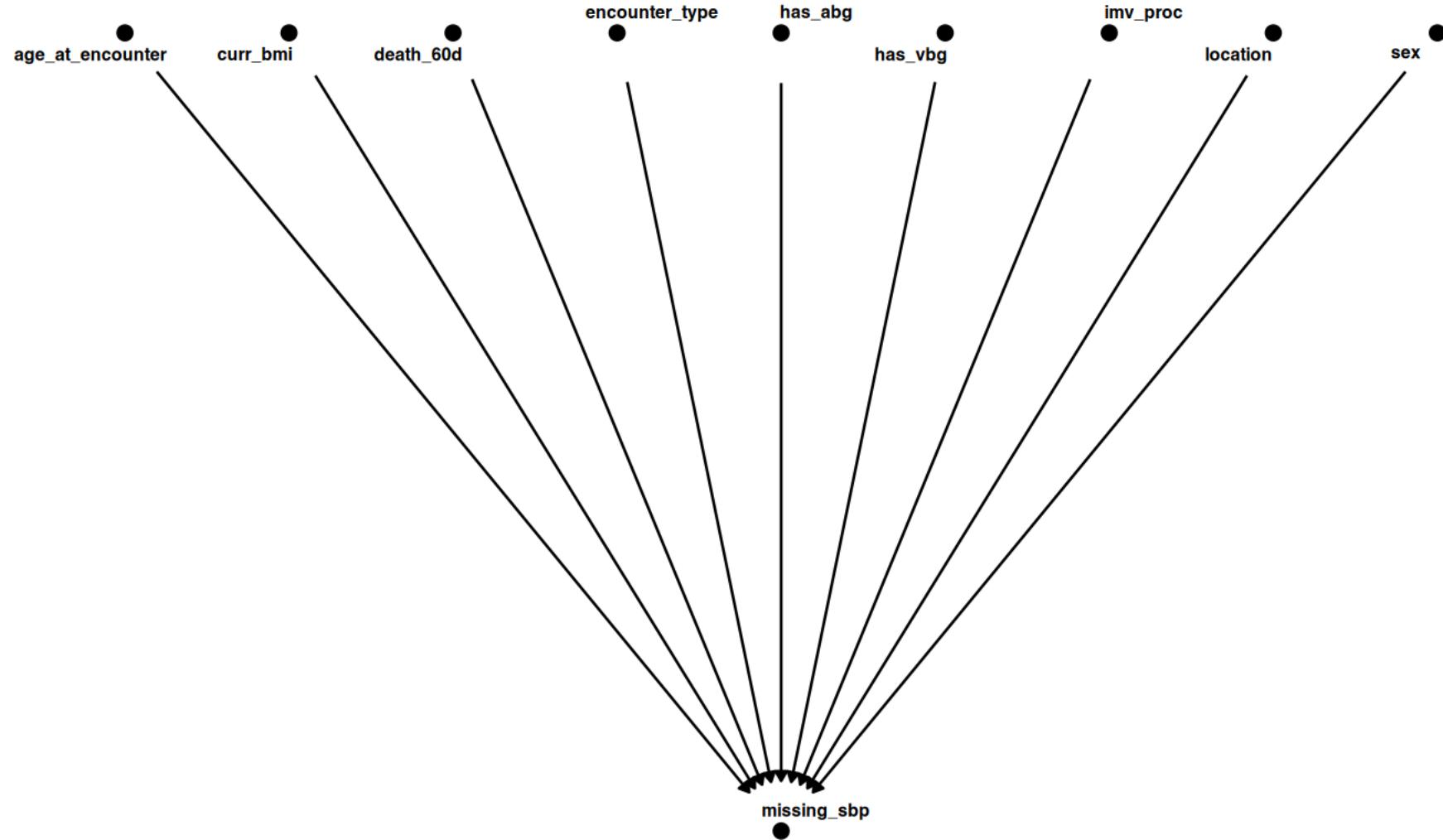
```
# 4) Optional, but often heavy: UpSet of co-missingness - skip by default.  
# If you really want it, do it for top 6-10 variables only:  
# naniar::gg_miss_upset(dat_imp[, head(top_vars, 8)], drop = FALSE))
```

Chunk mi-missing-audit runtime: 0.48 s

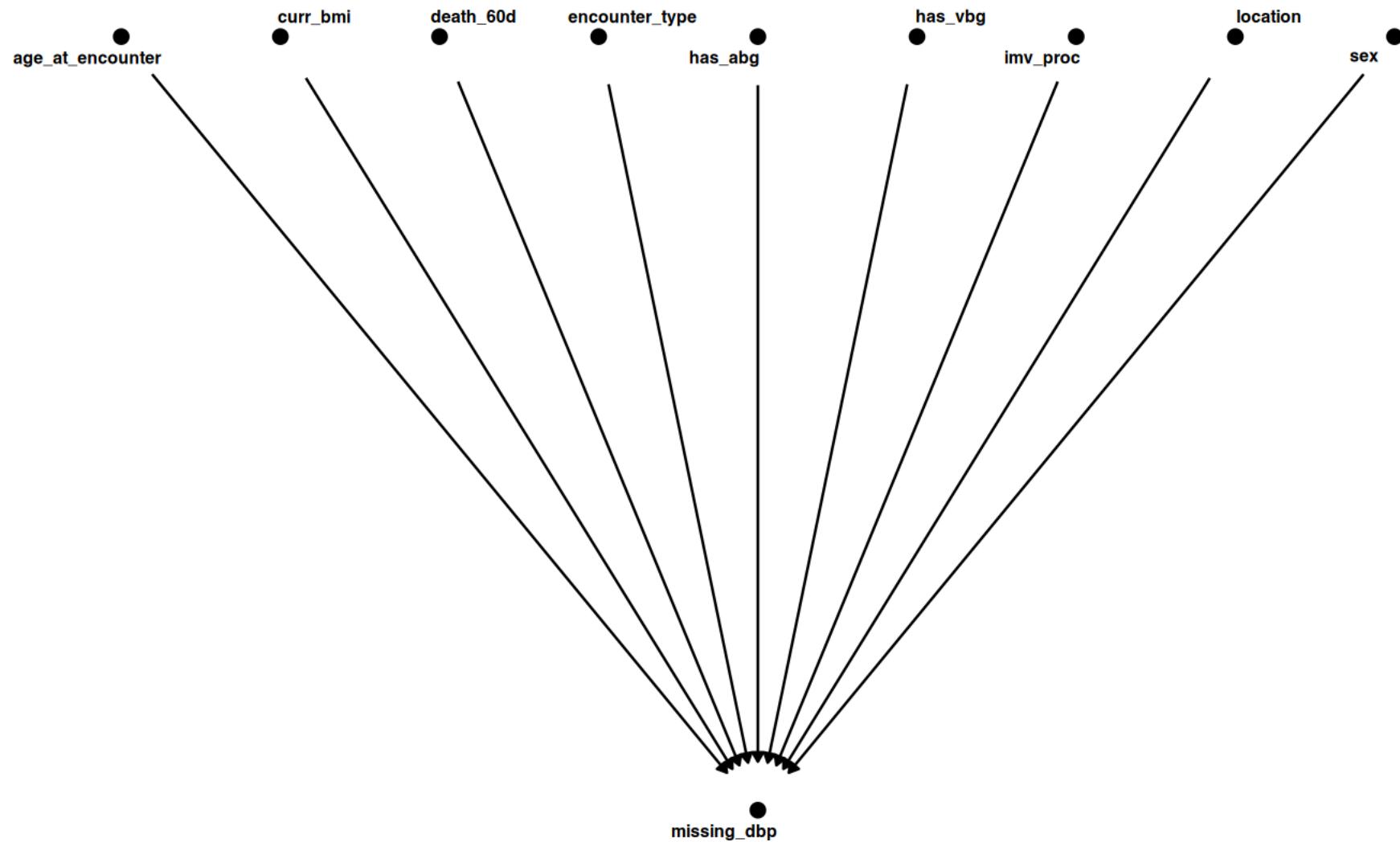
4.1.3 7.2 Missingness structure and drivers



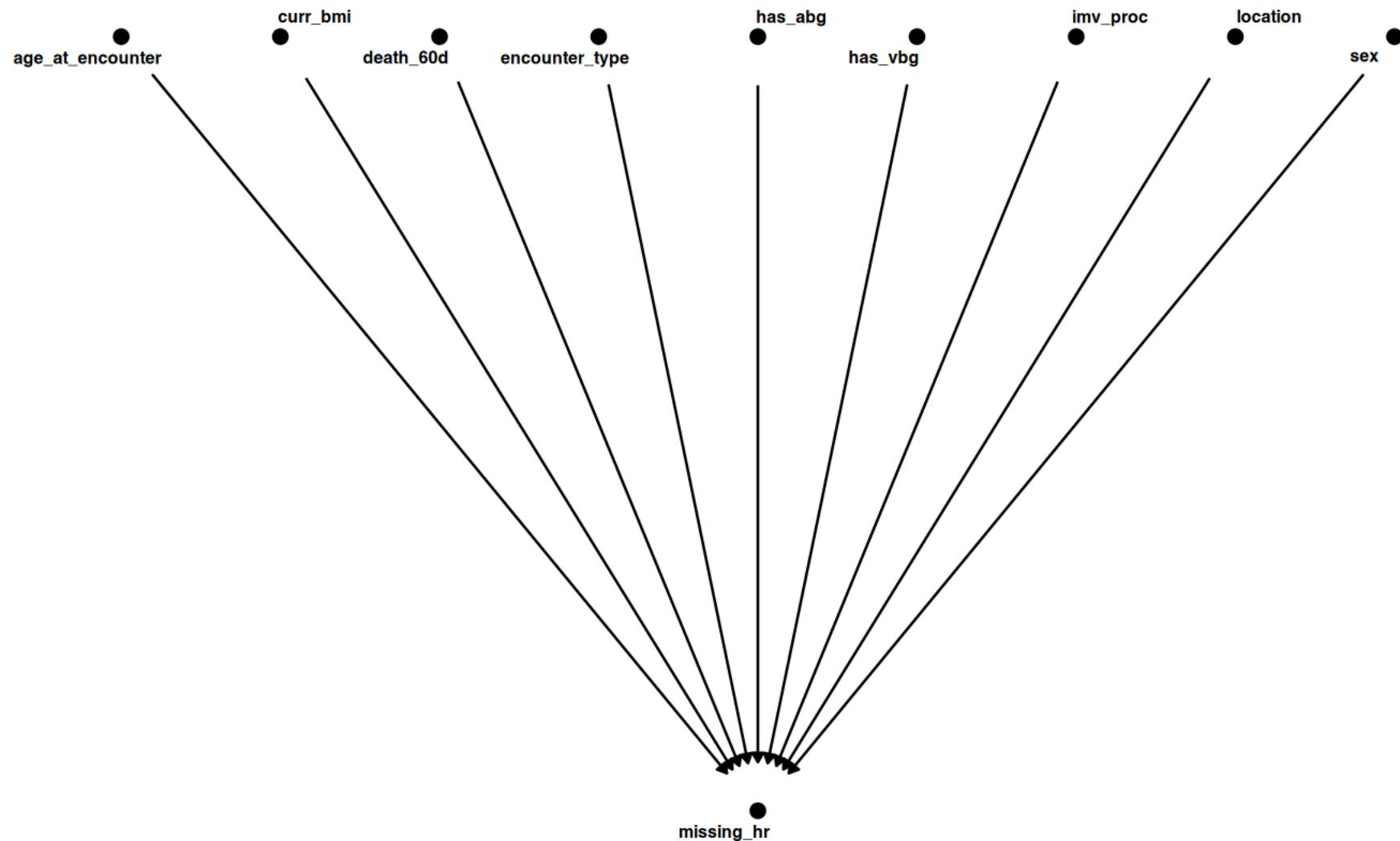
Missingness model: sbp



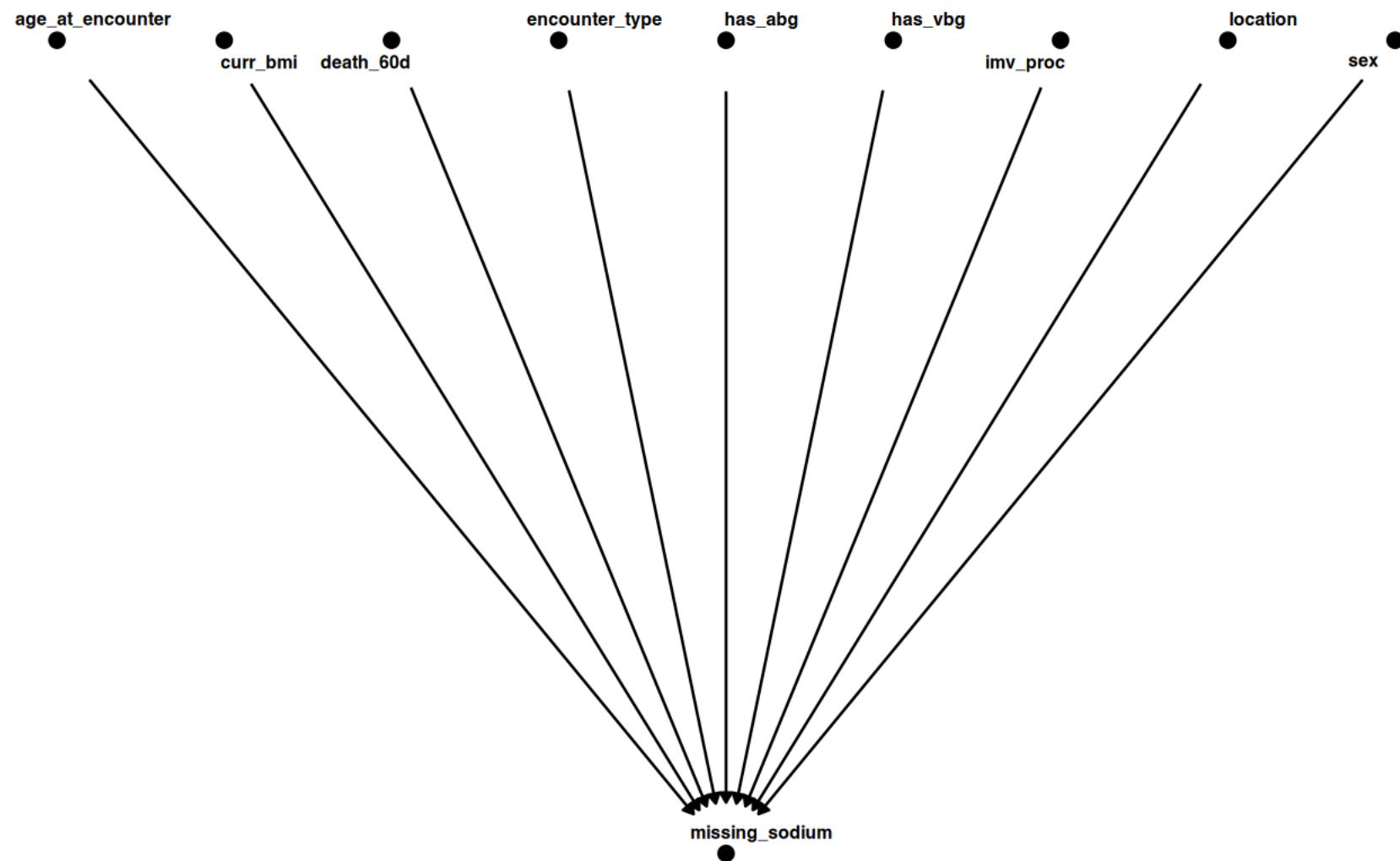
Missingness model: dbp



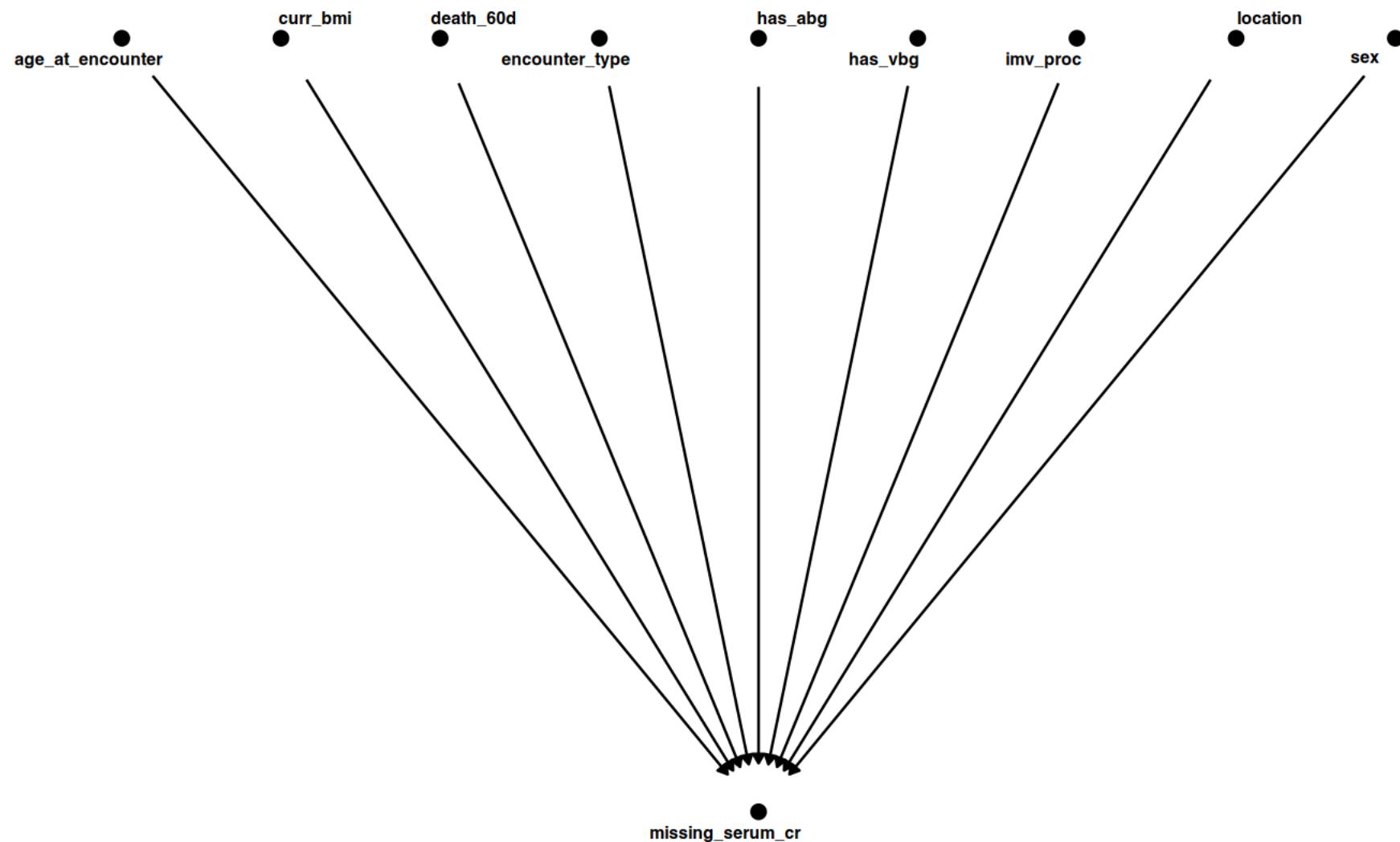
Missingness model: hr



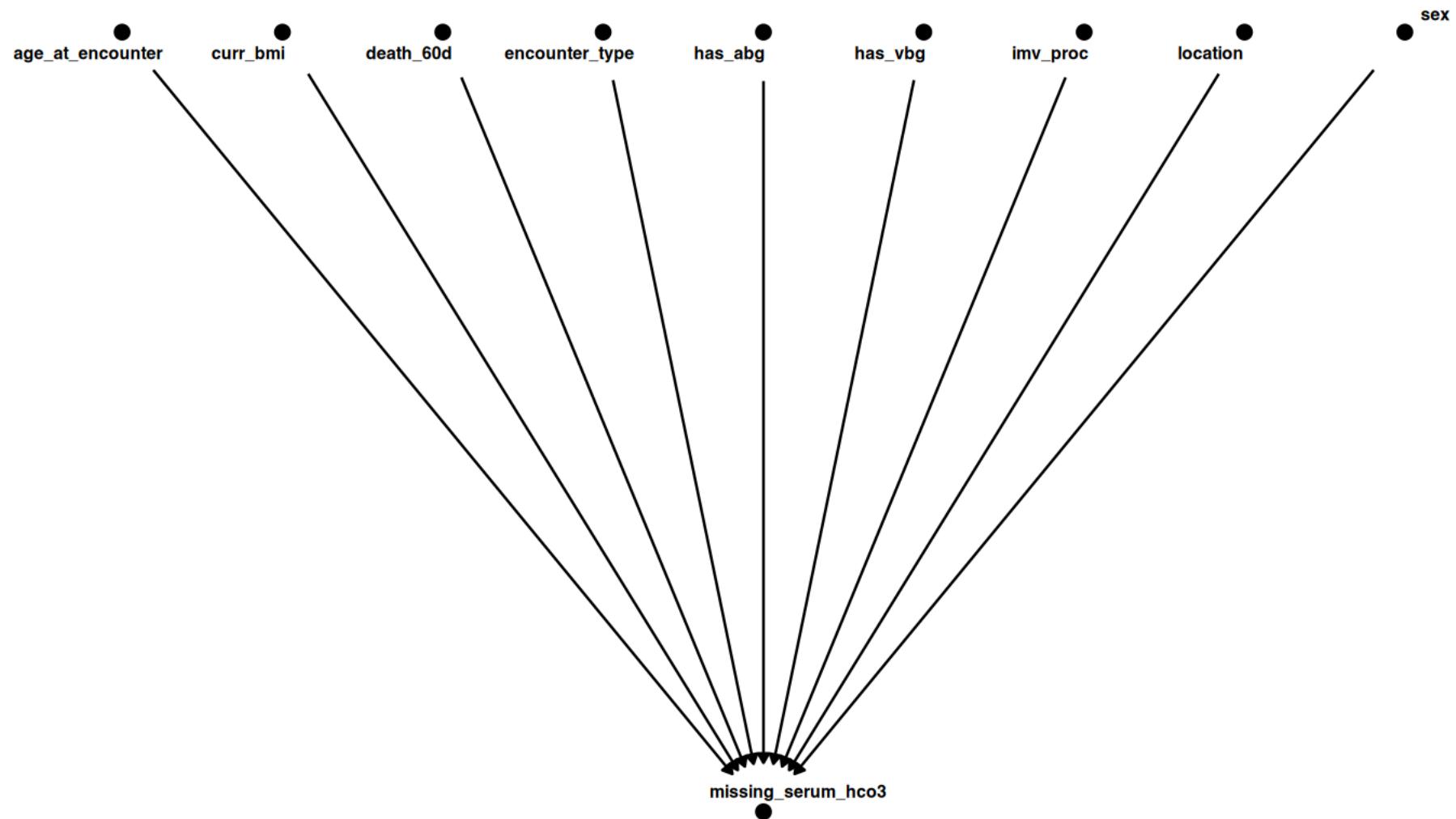
Missingness model: sodium



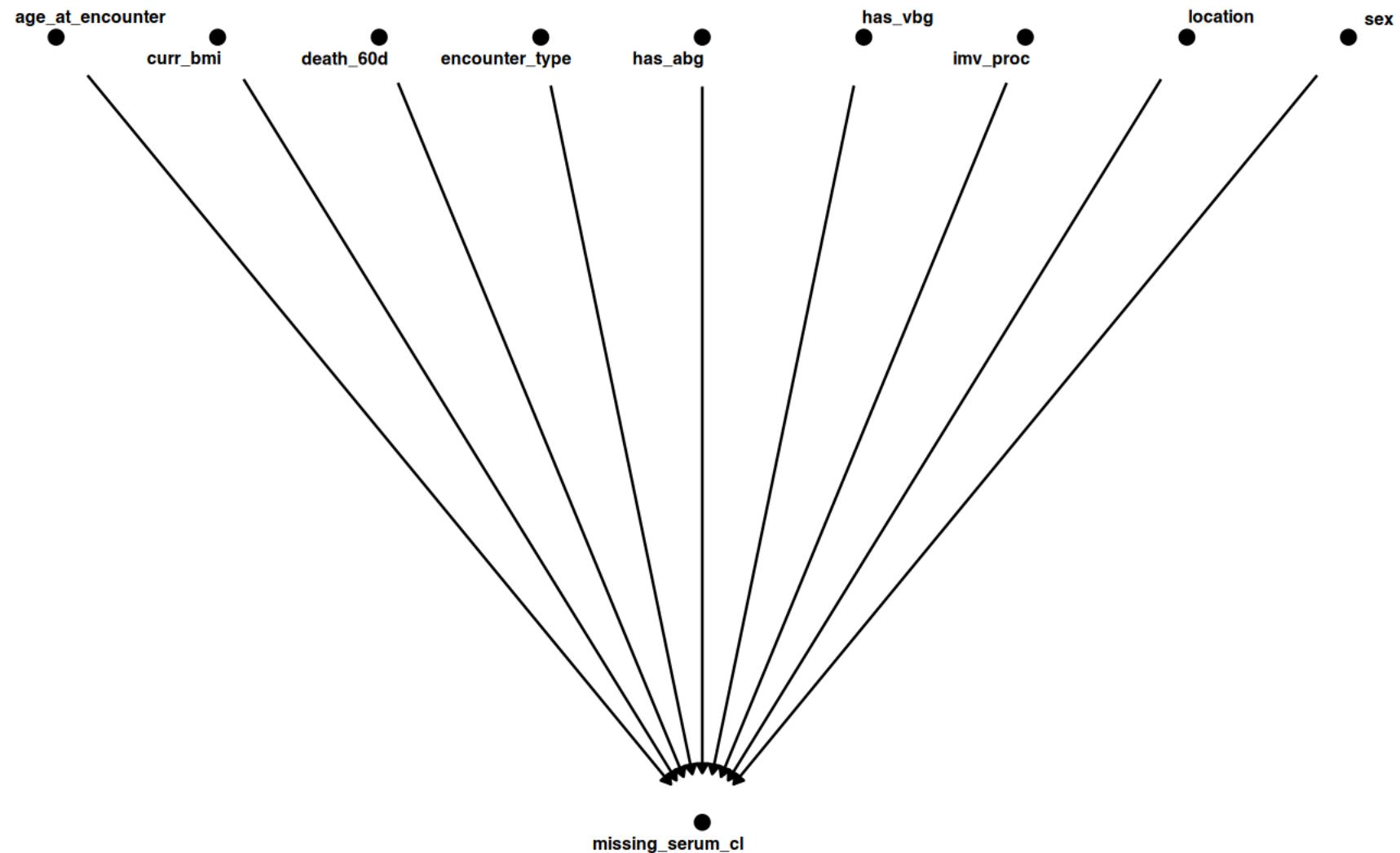
Missingness model: serum_cr



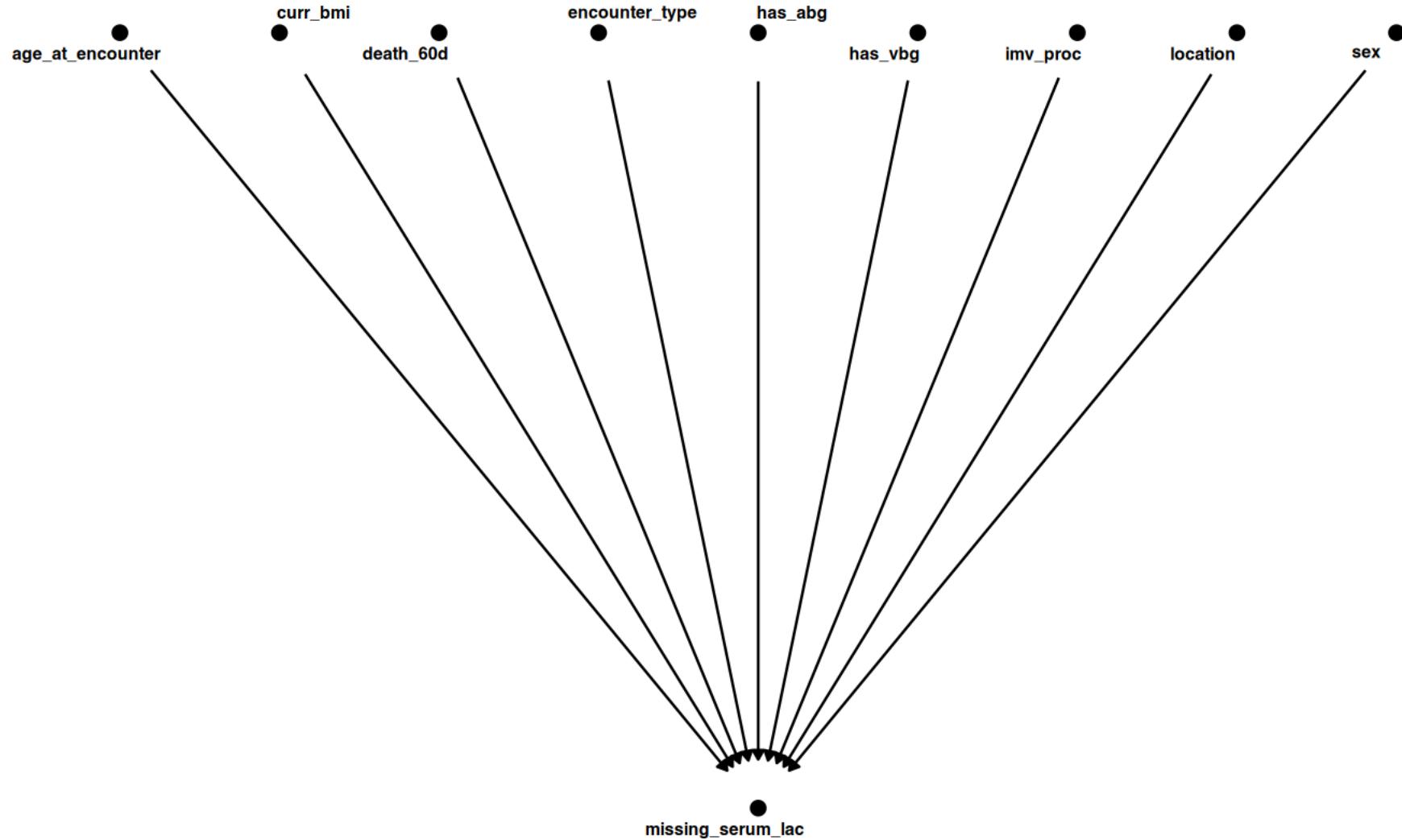
Missingness model: serum_hco3



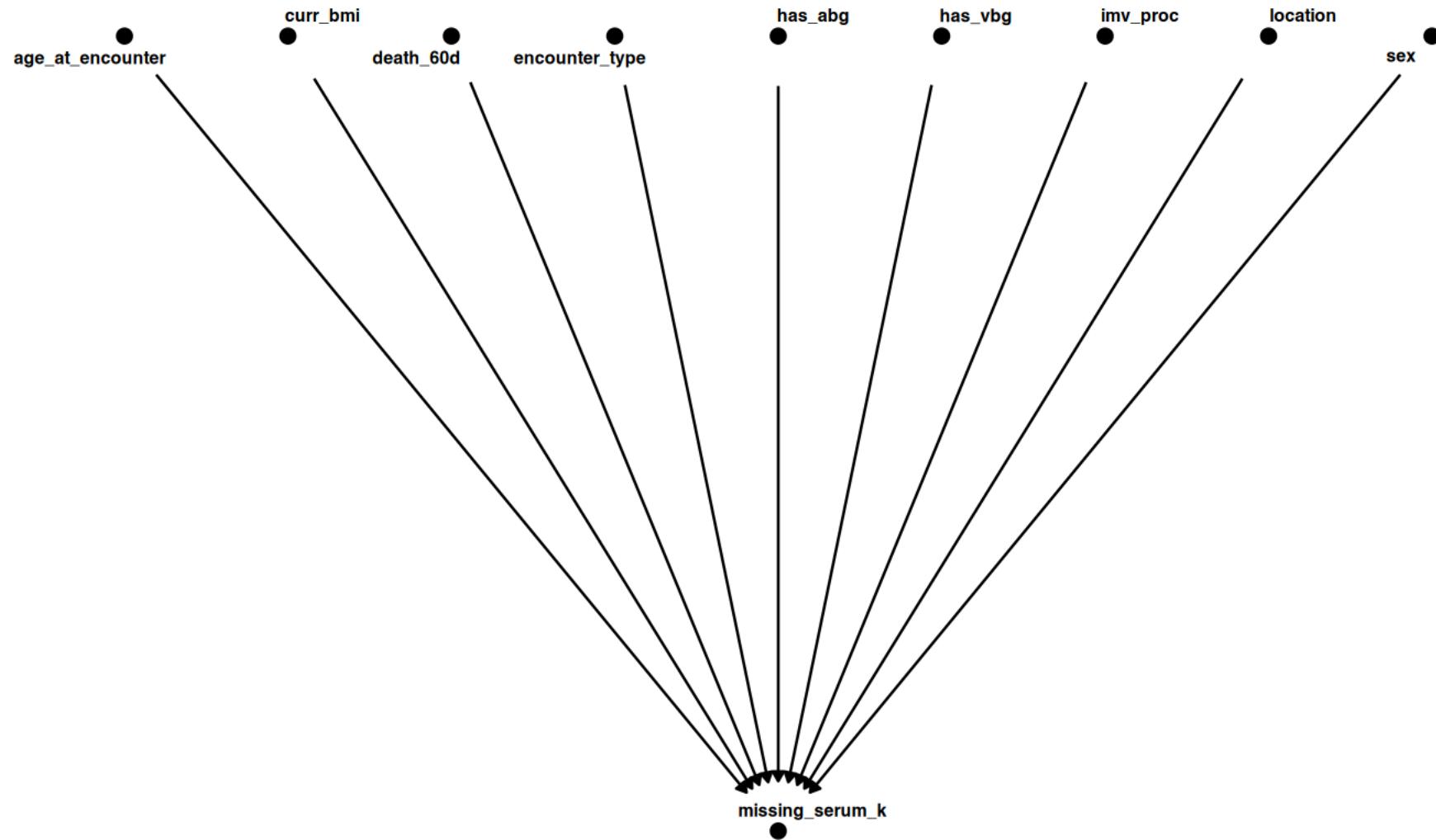
Missingness model: serum_cl



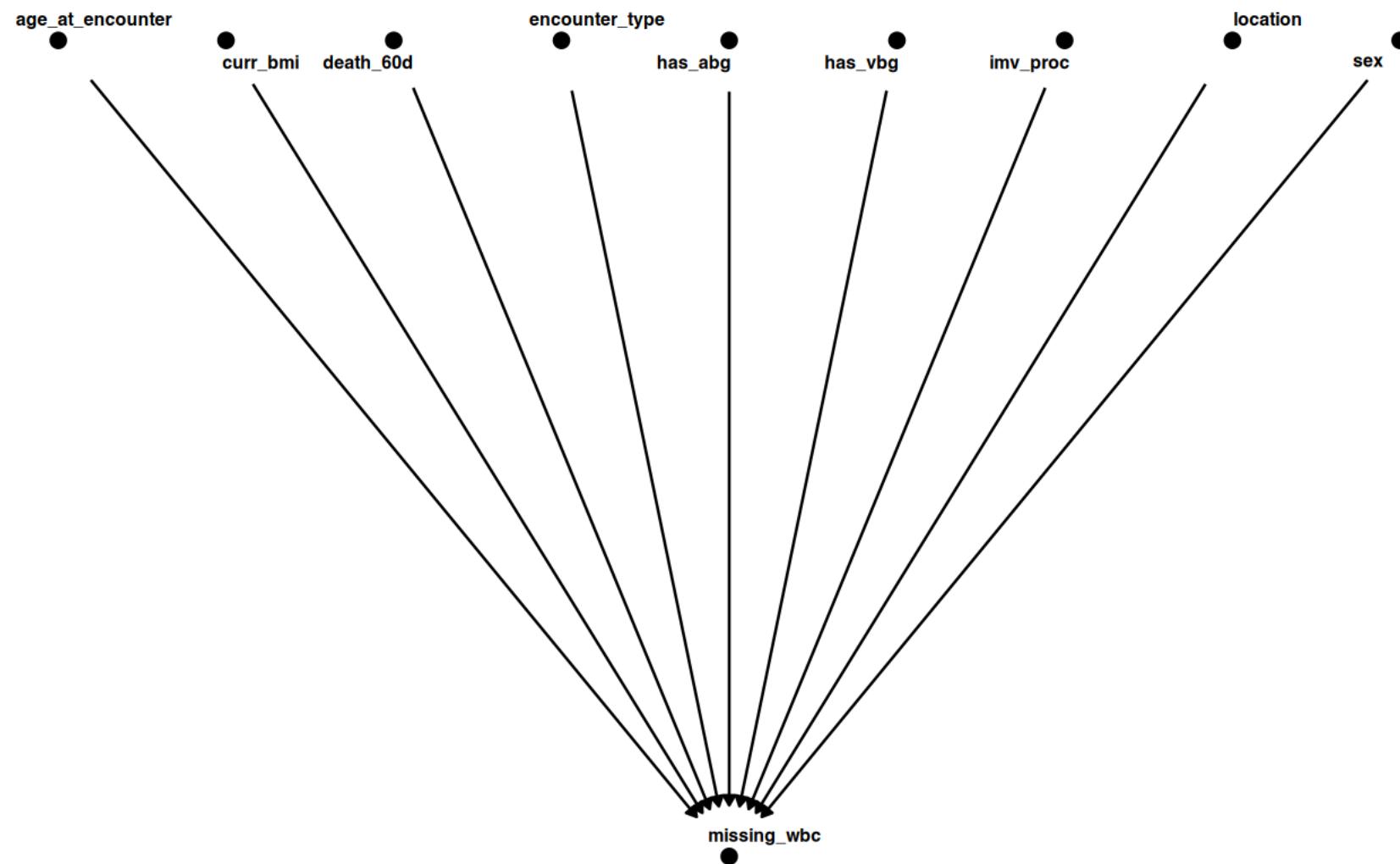
Missingness model: serum_lac



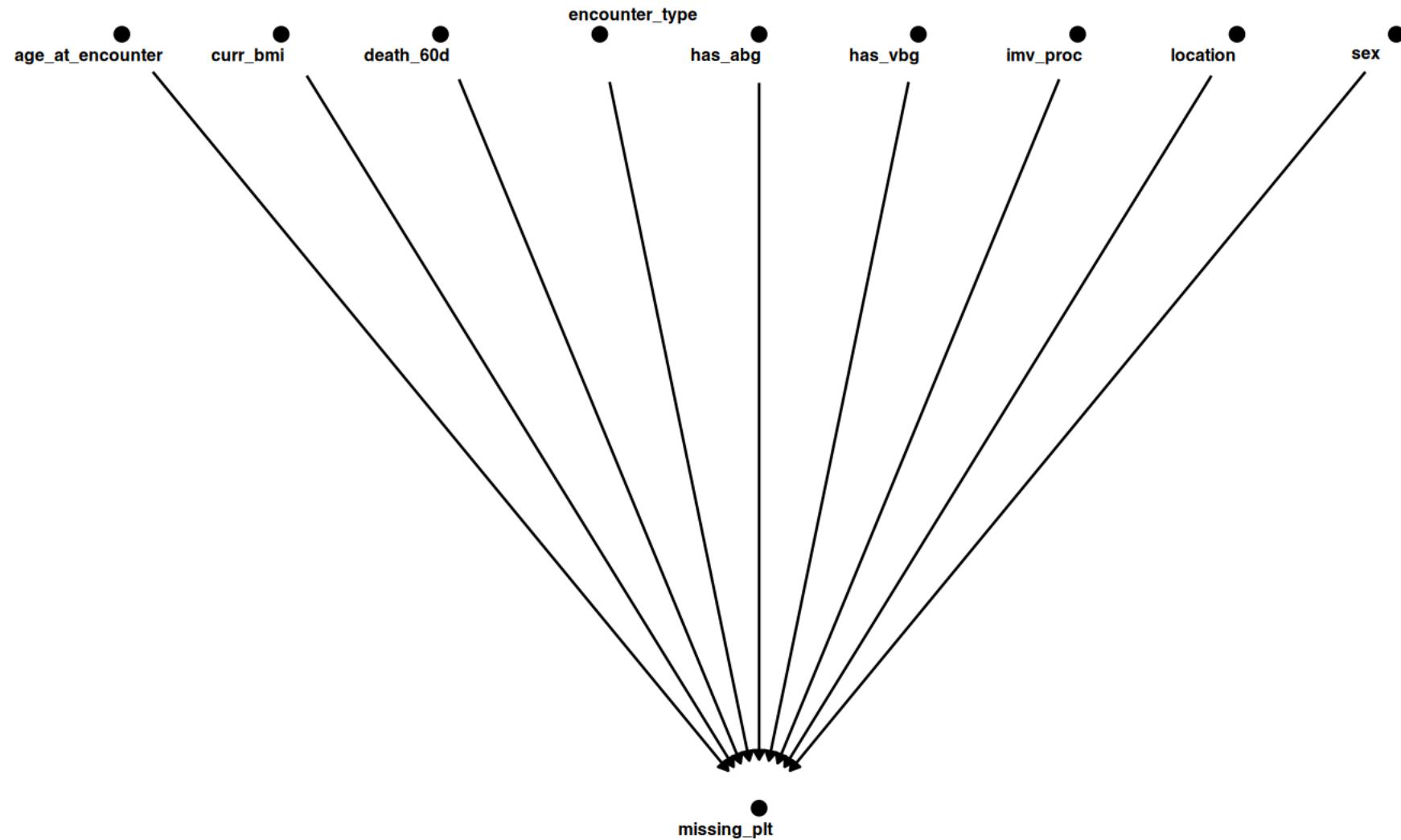
Missingness model: serum_k



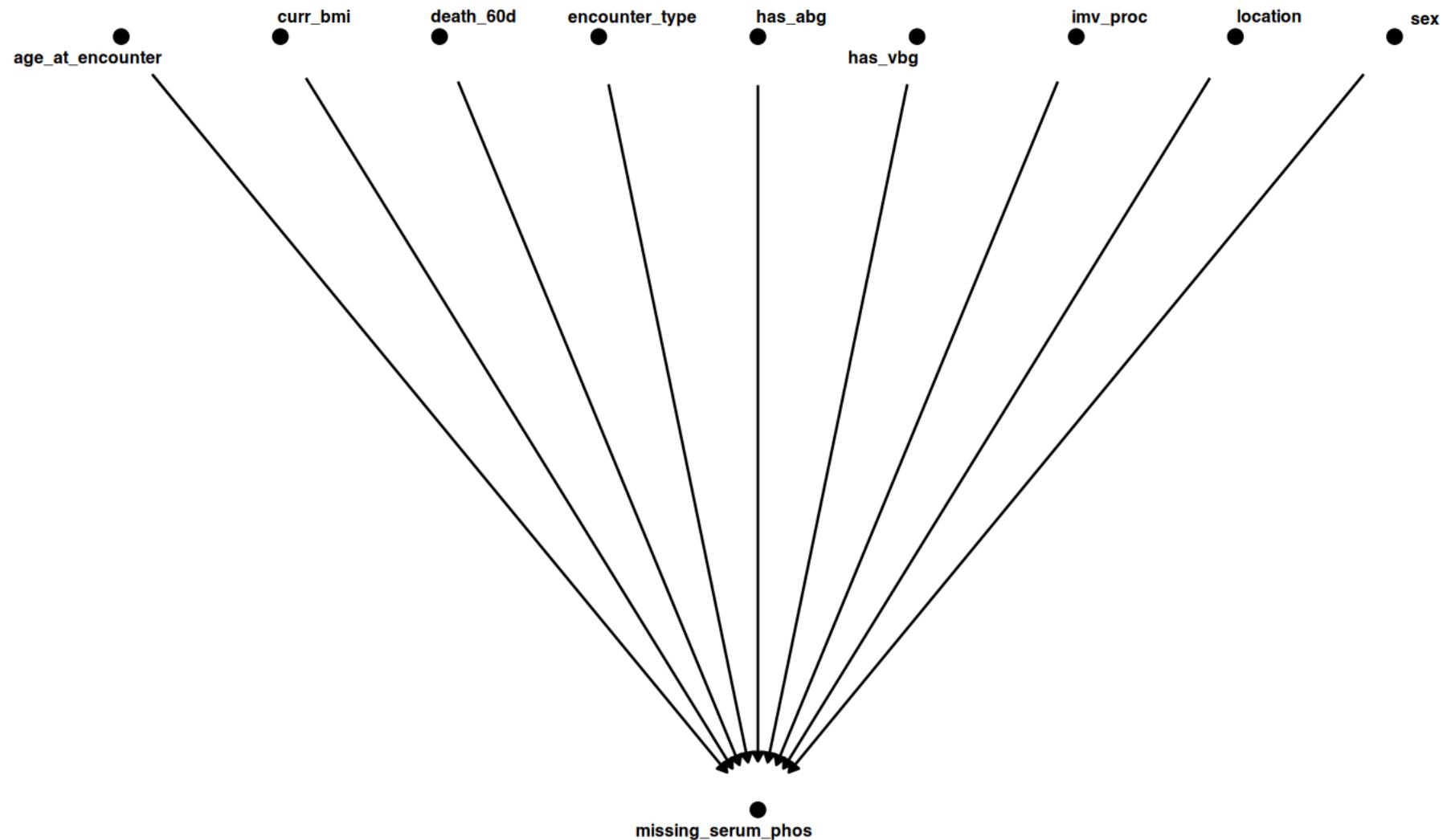
Missingness model: wbc



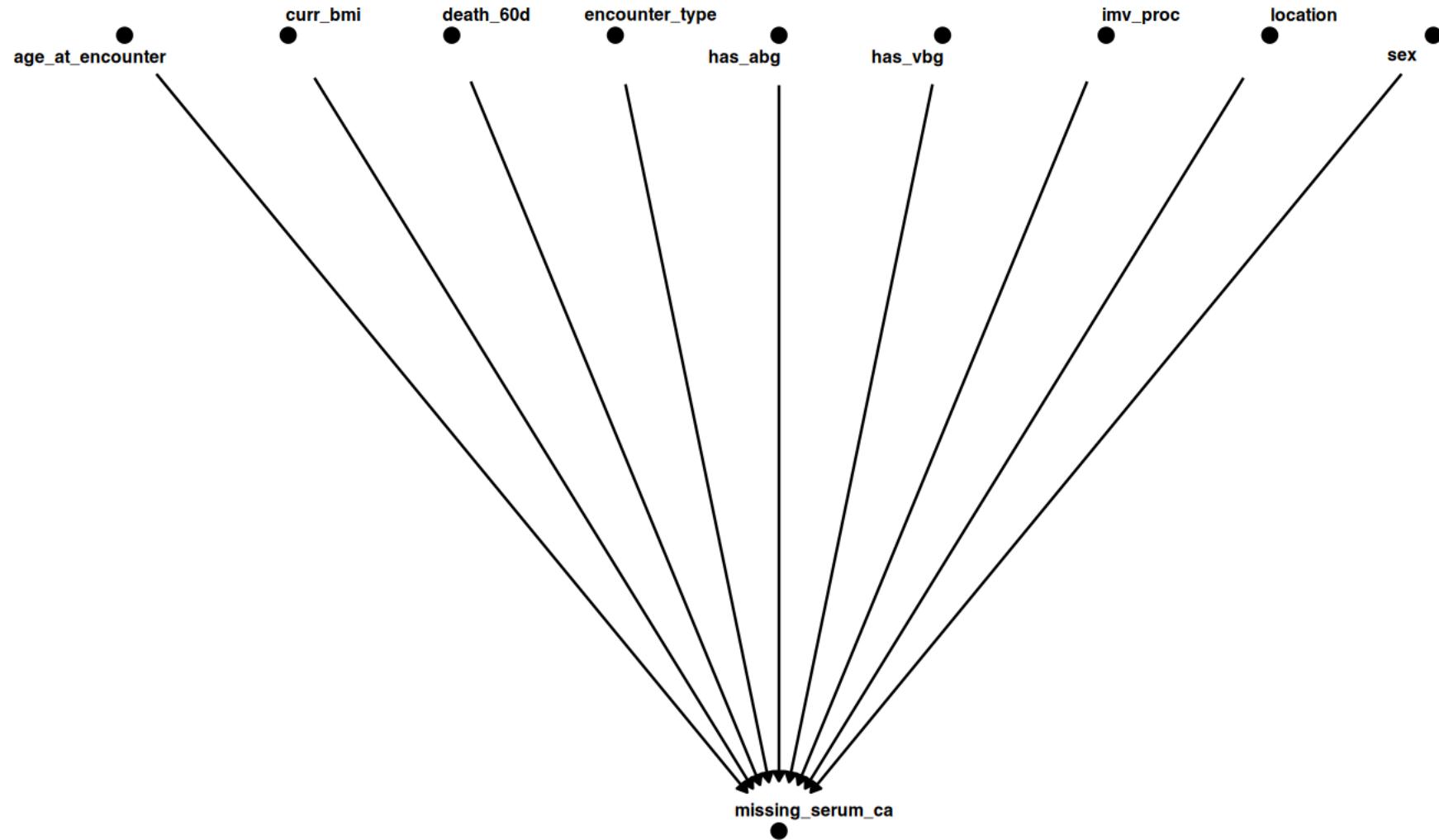
Missingness model: plt



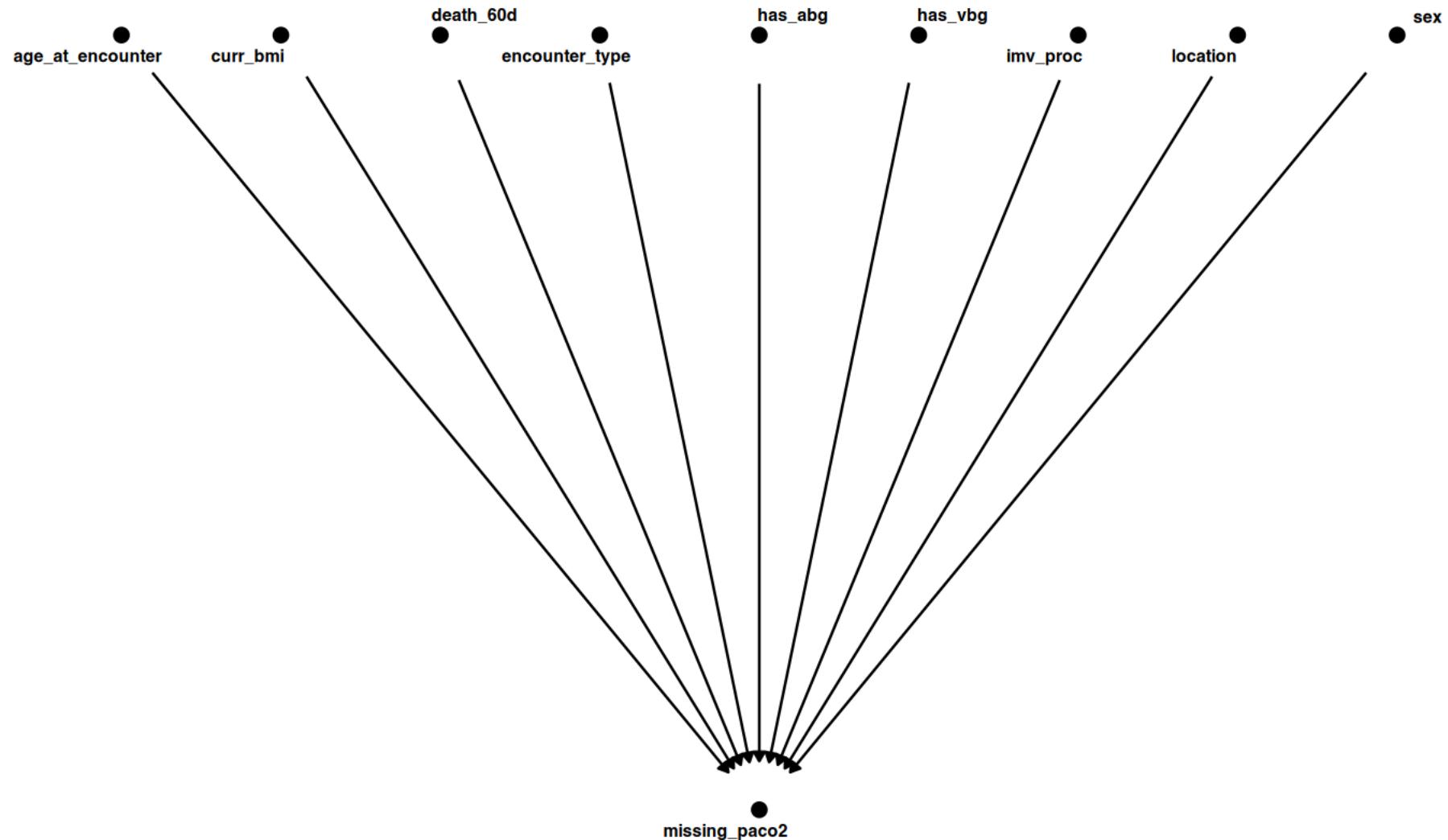
Missingness model: serum_phos



Missingness model: serum_ca

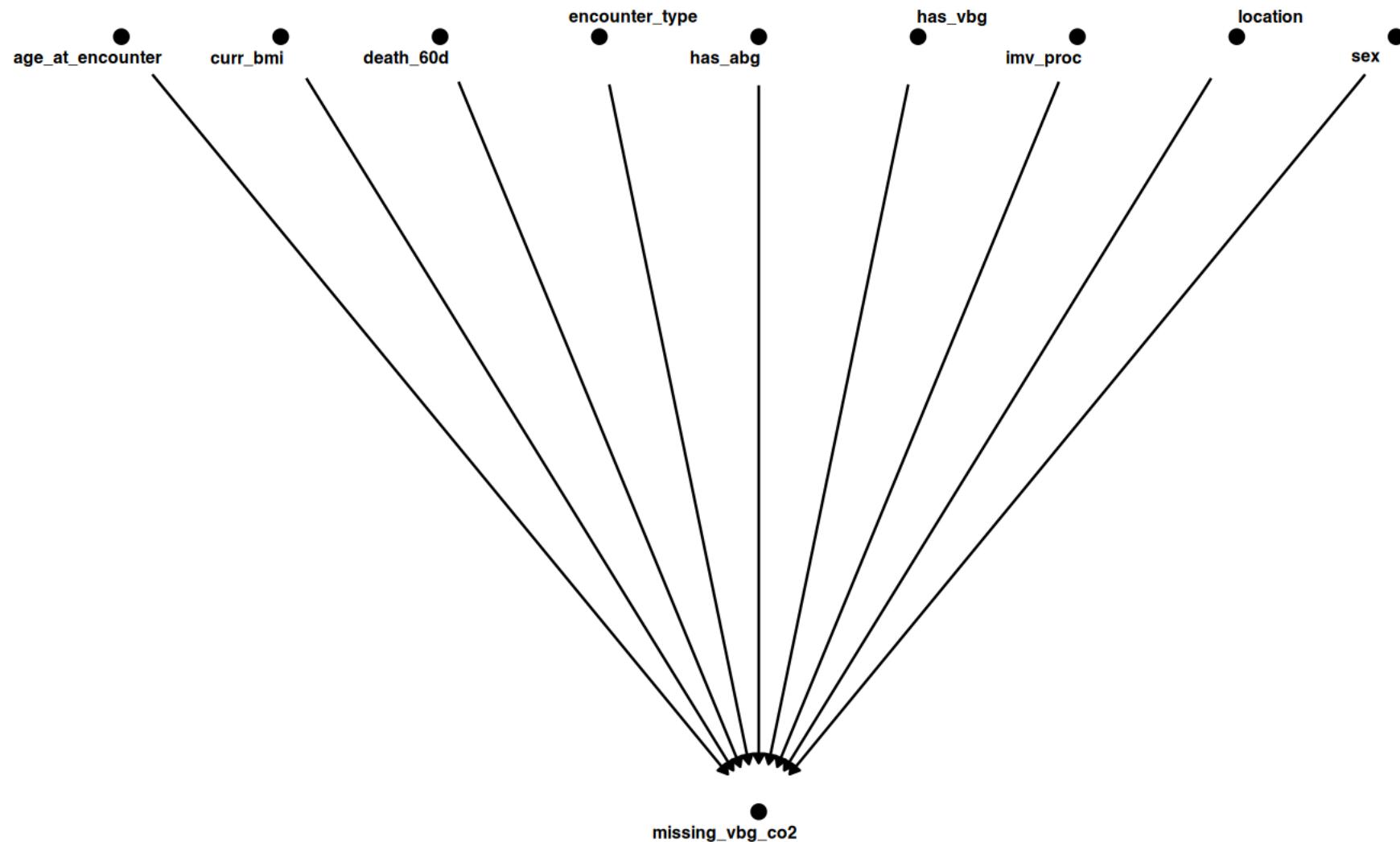


Missingness model: paco2



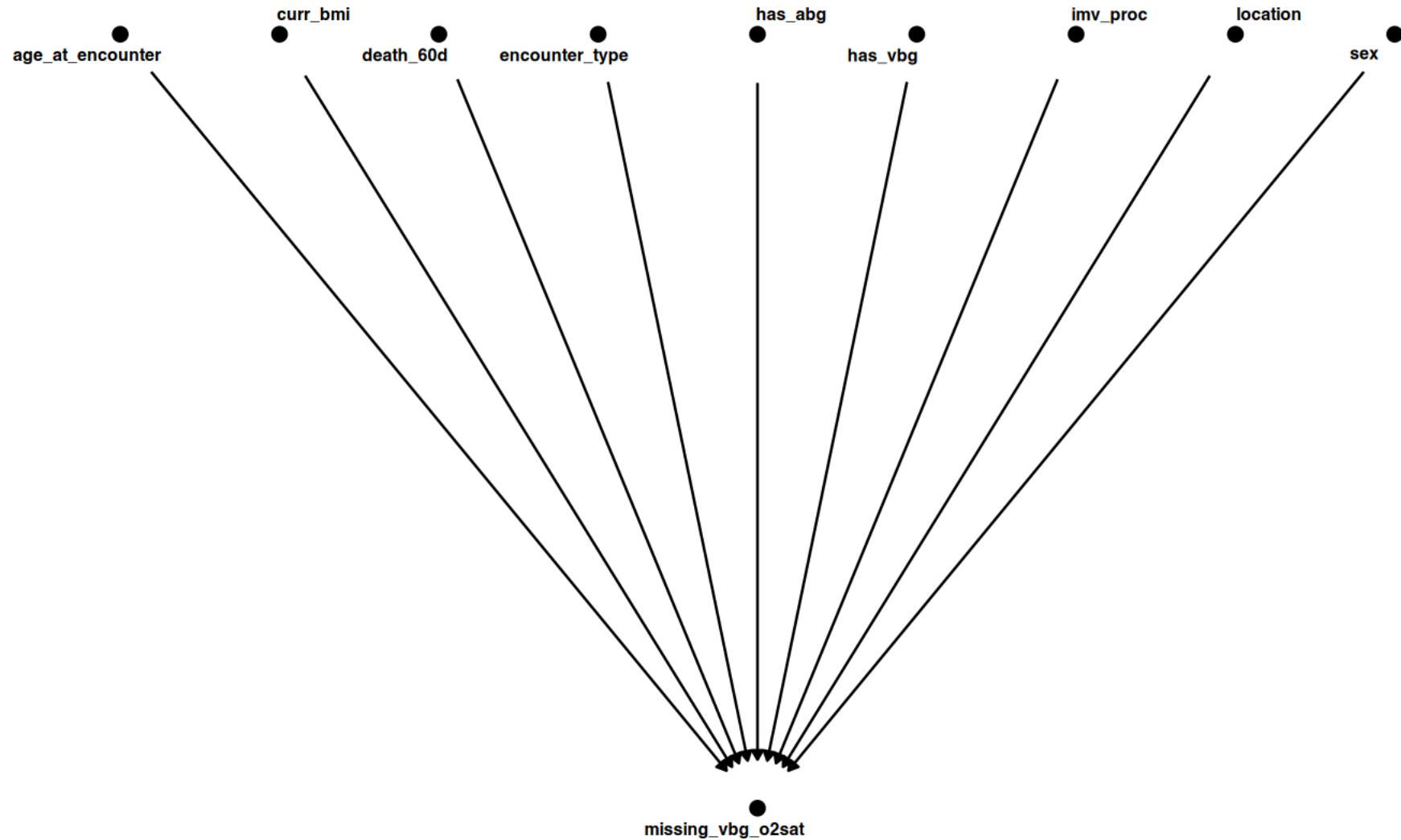
Warning: `glm.fit`: algorithm did not converge

Missingness model: vbg_co2



Warning: glm.fit: algorithm did not converge

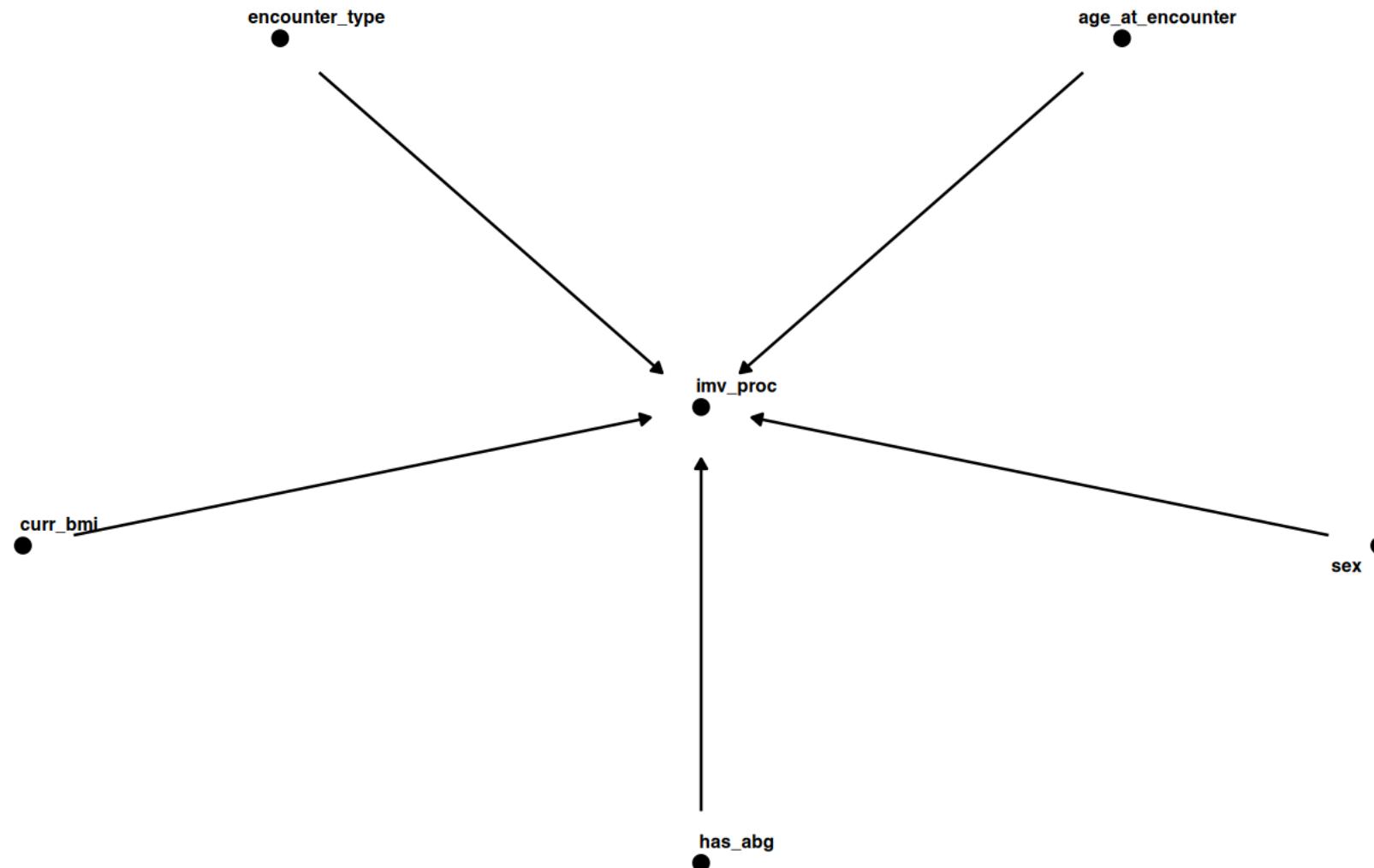
Missingness model: vbg_o2sat



Chunk mi-missing-structure runtime: 15.36 s

4.1.4 7.3 Monte Carlo error check after MI

MI MC error: IMV ~ ABG + covariates



Chunk mi-mcerror runtime: 5.29 s

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

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

```
# Types are normalized in the schema-normalize block.  
cat("encounter_type class:", paste(class(subset_data$encounter_type), collapse = ", "), "\n")
```

```
encounter_type class: factor
```

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

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

Chunk mi-prep runtime: 0.00 s

Chunk type-invariants runtime: 0.24 s

4.3 9) Imputation model specification (MICE)

4.3.1 9.1 Predictor matrix & methods. Run MICE (moderate settings for scale)

```
# --- variables for GBM propensity (kept identical to main analysis) ---  
# ----- MICE setup: include PaCO2/VBG CO2 as predictors but do not impute -----  
library(mice)  
library(dplyr)  
  
# --- add analysis targets and CO2 measures explicitly -----  
mi_vars <- setdiff(unique(c(  
  covars_gbm,  
  "has_abg", "has_vbg", # treatments (NOT imputed)  
  "imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure", # outcomes (NOT imputed)  
  co2_vars
```

```

)), drop_vars_ultra_missing)

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

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

```

MI data size (bytes): 131.7 Mb

```

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

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

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

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

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

```

```

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

# predictor matrix; force has_abg/has_vbg as predictors, but do not impute no_imp
pred <- mice:::quickpred(mi_df, mincor = 0.05, minpuc = 0.25)
for (nm in intersect(c("has_abg", "has_vbg"), colnames(pred))) {
  pred[, nm] <- 1
}
pred[intersect(rownames(pred), no_imp), ] <- 0

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

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

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

```

```
MICE: predictor column sums (pre-exclude) for paco2, vbg_co2: 5, 3
```

```
if (length(no_imp_with_missing)) {  
  pred[, intersect(no_imp_with_missing, colnames(pred))] <- 0  
  message("MICE: excluded non-imputed missing predictors: ",  
         paste(no_imp_with_missing, collapse = ", "))  
}
```

```
MICE: excluded non-imputed missing predictors: paco2, vbg_co2
```

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

```
MICE: predictor column sums (post-exclude) for paco2, vbg_co2: 0, 0
```

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

```

    ". Adding core predictors.")
pred[zero_pred, core_preds] <- 1
}
pred[intersect(rownames(pred), no_imp), ] <- 0
pred[c(zero_pred), c(zero_pred)] <- 0

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

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

# --- run MICE -----
set.seed(MI_SEED)
imp <- runtime_logger(
  "mice_imputation",
  {
    mice::mice(
      data          = mi_df,
      m             = M_IMP,
      maxit         = MAXIT_MI,
      predictorMatrix = pred,
      method        = meth,
      printFlag     = TRUE,
      seed          = MI_SEED
    )
  },
),

```

```
notes = paste0("m=", M_IMP, "; maxit=", MAXIT_MI)
```



```

20   60 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   61 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   62 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   63 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   64 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   65 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   66 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   67 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   68 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   69 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   70 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   71 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   72 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   73 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   74 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   75 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   76 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   77 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   78 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   79 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s
20   80 curr_bmi temp_new sbp dbp hr sodium serum_cr serum_hco3 serum_cl serum_lac serum_k wbc plt serum_phos s

```

```

saveRDS(imp, file = mi_mids_file)

# quick sanity: these must exist and be numeric in completed data
imp_n <- imp$m
get_imp <- function(i, imp_obj = imp) {
  normalize_types(mice:::complete(imp_obj, action = i), levels_ref)
}
d1 <- get_imp(1)
stopifnot(all(c("paco2", "vbg_co2") %in% names(d1)))
stopifnot(is.numeric(d1$paco2), is.numeric(d1$vbg_co2))

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

```

```

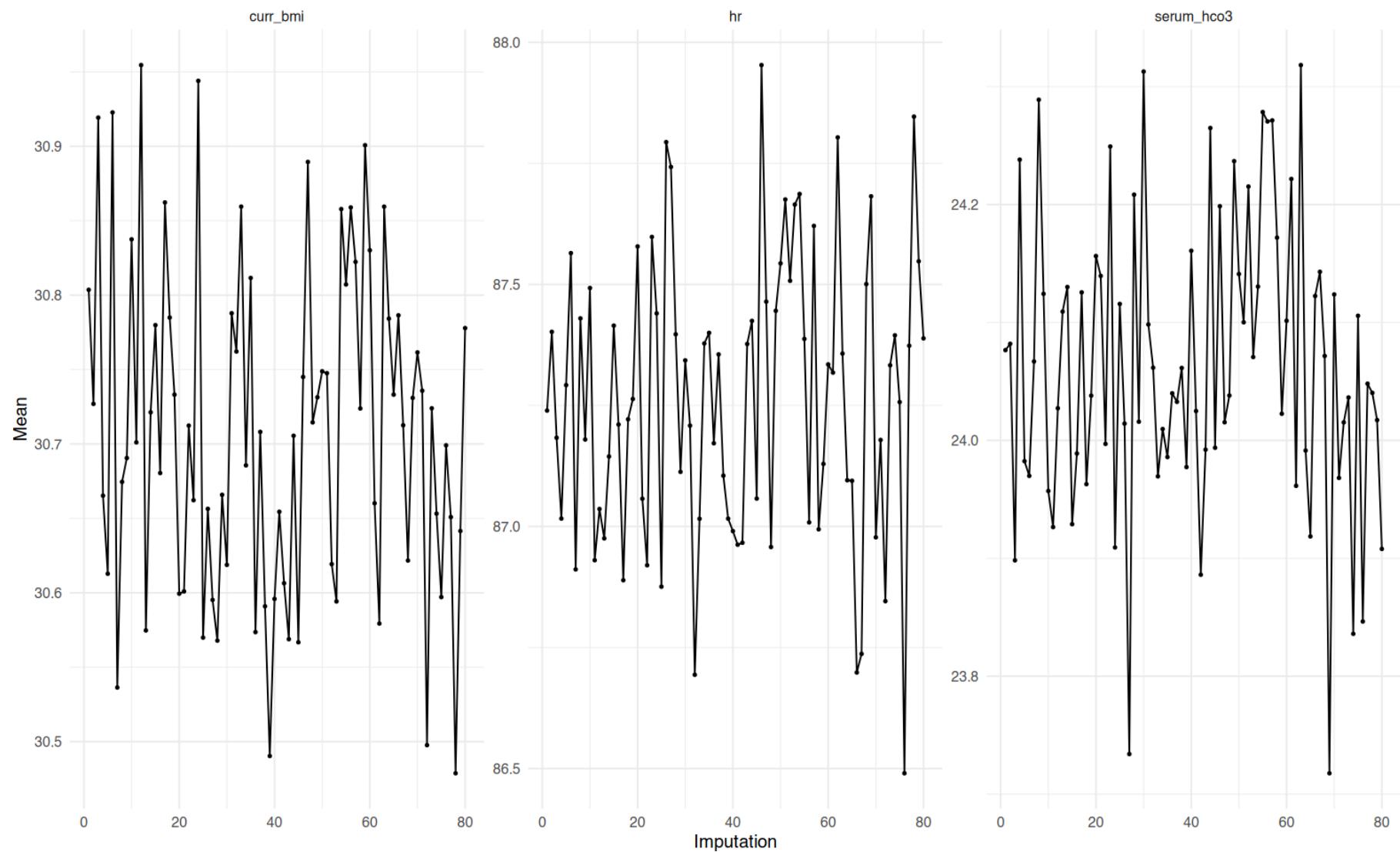
na_counts <- na_counts[na_counts > 0]
if (length(na_counts)) {
  message("Post-MICE NA counts (covars_gbm): ",
         paste(names(na_counts), na_counts, collapse = ", "))
  ev_sum <- summarize_logged_events(imp)
  if (nrow(ev_sum)) {
    ev_sub <- ev_sum[ev_sum$variable %in% names(na_counts), , drop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

```

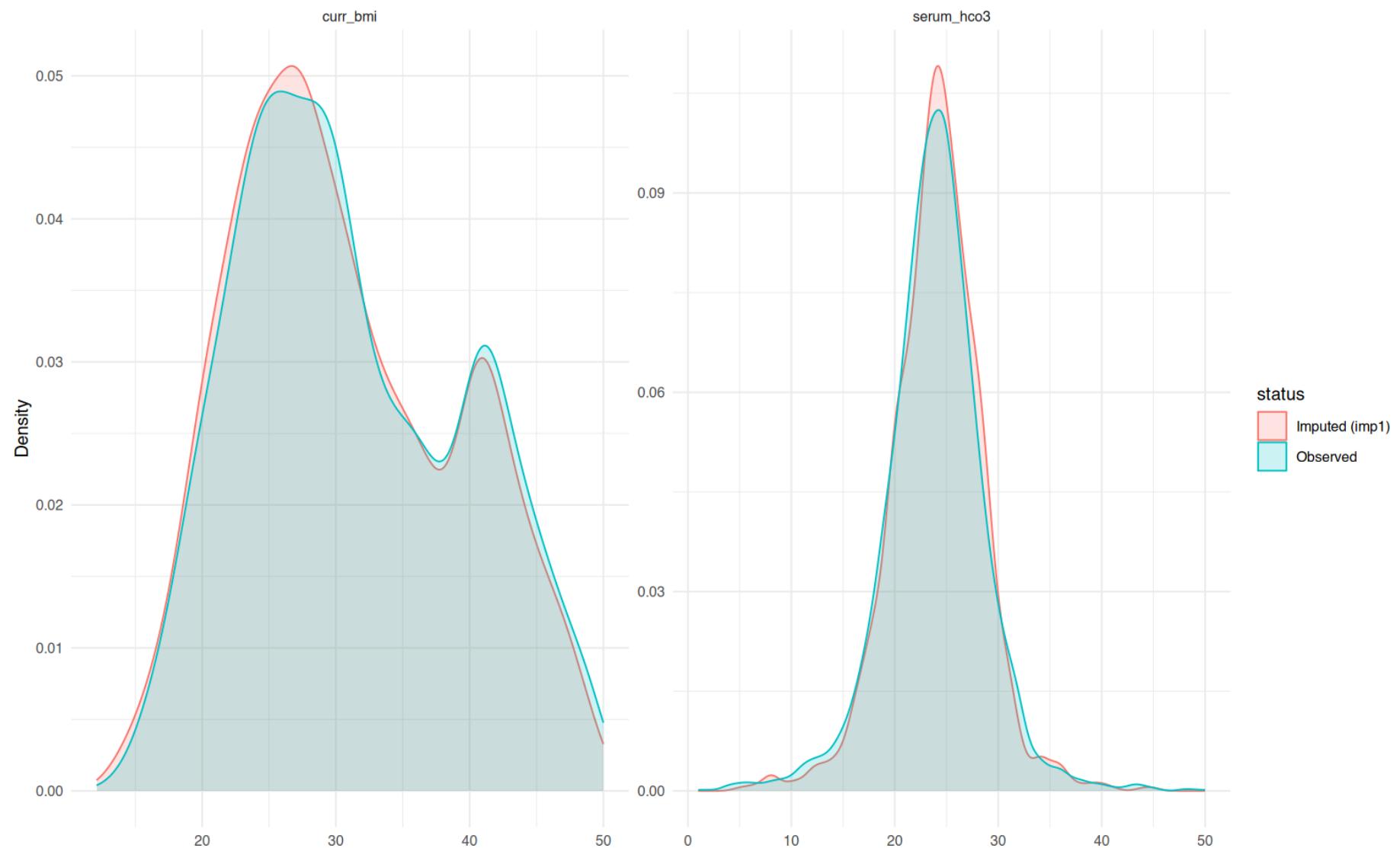
Chunk mi-exec runtime: 21533.71 s

4.3.2 9.2 Convergence & plausibility checks

Mean imputed value by imputation



Observed vs imputed distributions (imp1)



Observed vs imputed (imp1)

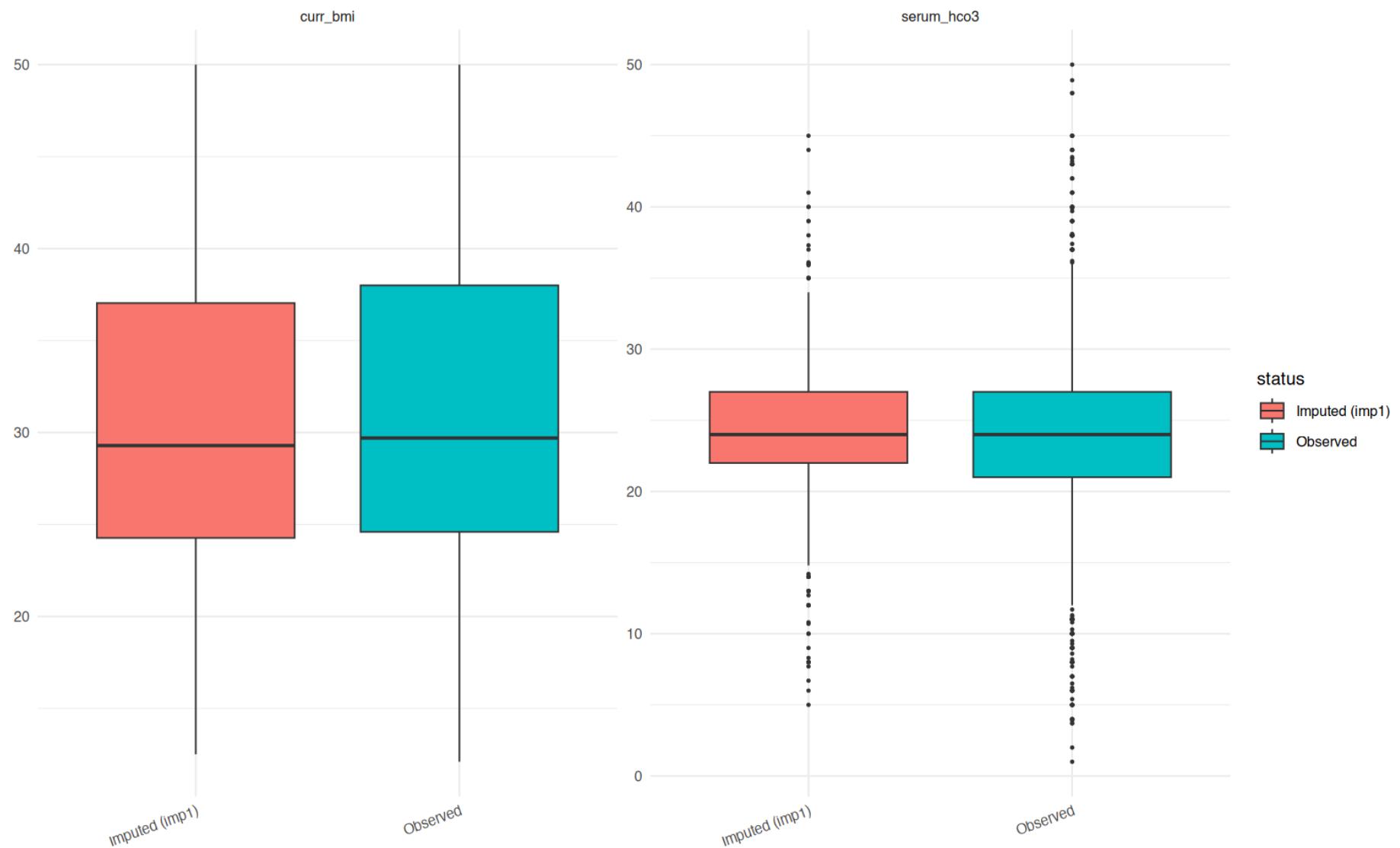


Table 12: Observed vs imputed summaries (first 20 rows)

variable	type	stat	observed	imputed
curr_bmi	numeric	mean	31.077456	30.777901
curr_bmi	numeric	sd	8.407062	8.299711
curr_bmi	numeric	p10	20.964000	20.900000
curr_bmi	numeric	p50	29.559999	29.230000
curr_bmi	numeric	p90	43.279999	43.000000
curr_bmi	numeric	n	11128.000000	50000.000000
temp_new	numeric	mean	97.903942	97.934582
temp_new	numeric	sd	1.549413	1.448177
temp_new	numeric	p10	96.812500	96.968750
temp_new	numeric	p50	98.000000	98.000000
temp_new	numeric	p90	99.000000	99.000000
temp_new	numeric	n	13307.000000	50000.000000
sbp	numeric	mean	128.177905	127.794560
sbp	numeric	sd	26.490862	26.029259
sbp	numeric	p10	97.000000	98.000000
sbp	numeric	p50	127.000000	126.000000
sbp	numeric	p90	161.000000	160.000000
sbp	numeric	n	18004.000000	50000.000000
dbp	numeric	mean	72.616409	72.770840
dbp	numeric	sd	16.171648	15.819780

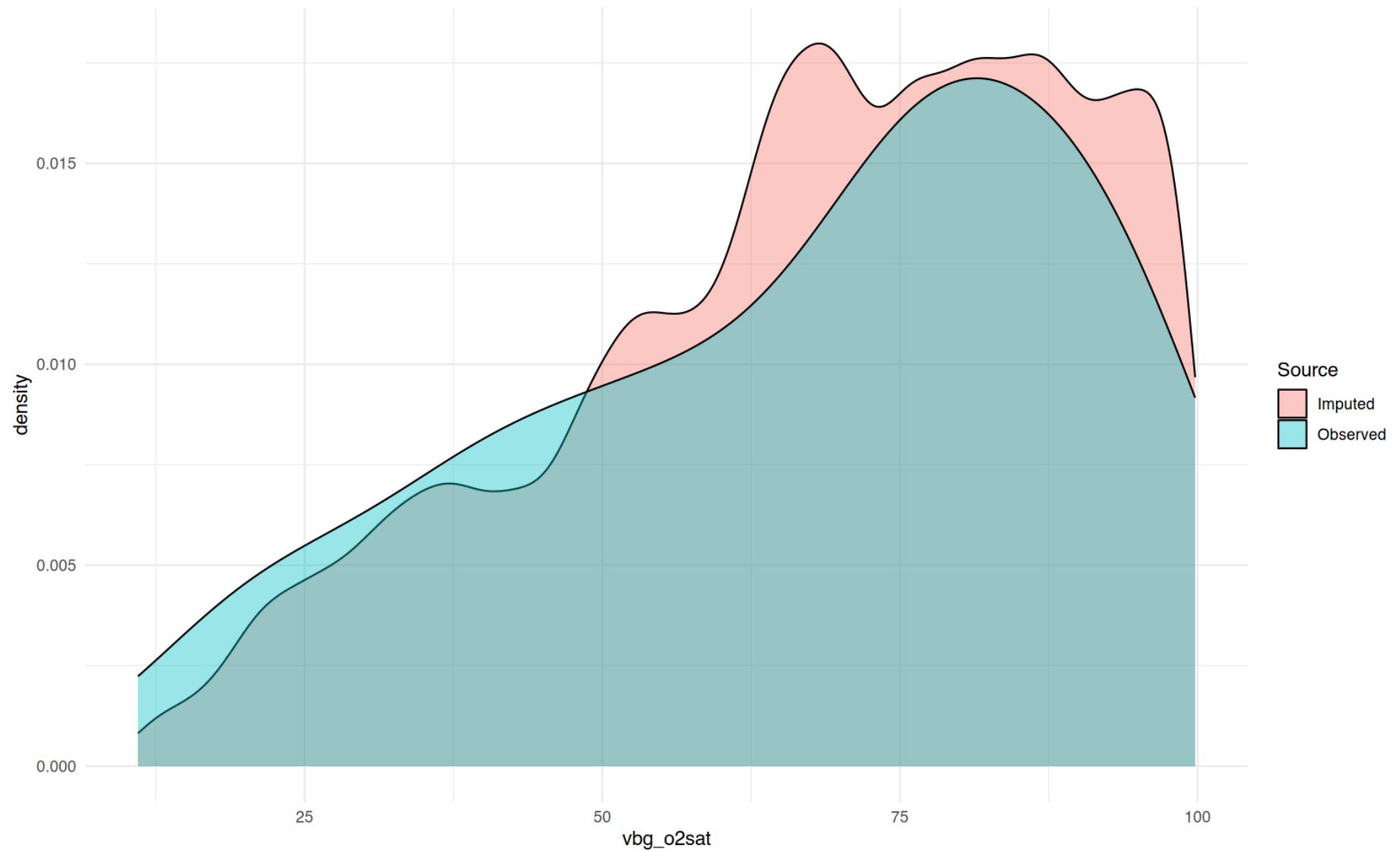
Chunk mi-diagnostics runtime: 5.90 s

Chunk mi-missing-pattern runtime: 0.83 s

4.3.3 9.3 Observed vs imputed distributions (by strata)

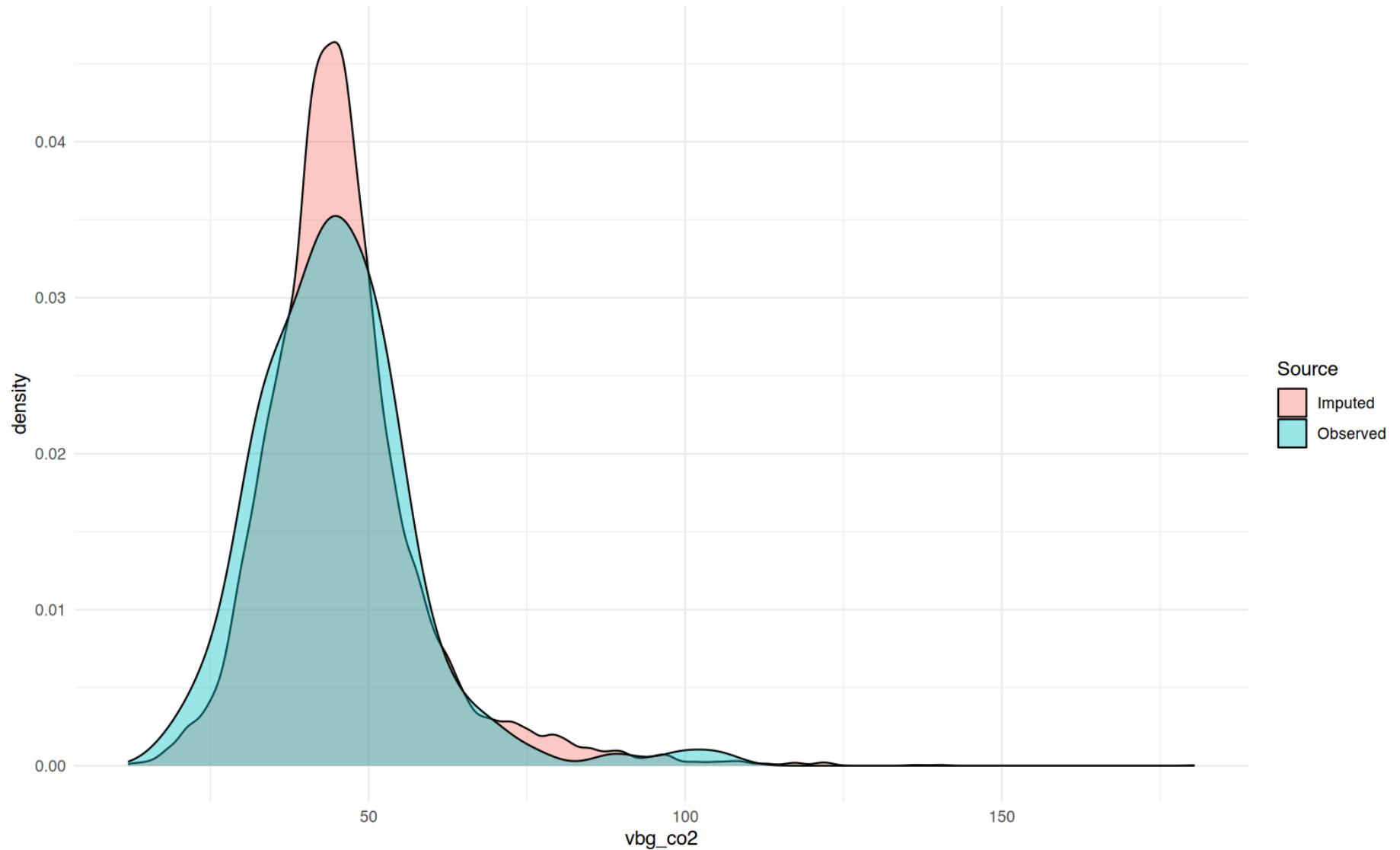
Warning: Removed 428 rows containing non-finite outside the scale range
(`stat_density()`).

Observed vs imputed density: vbg_o2sat



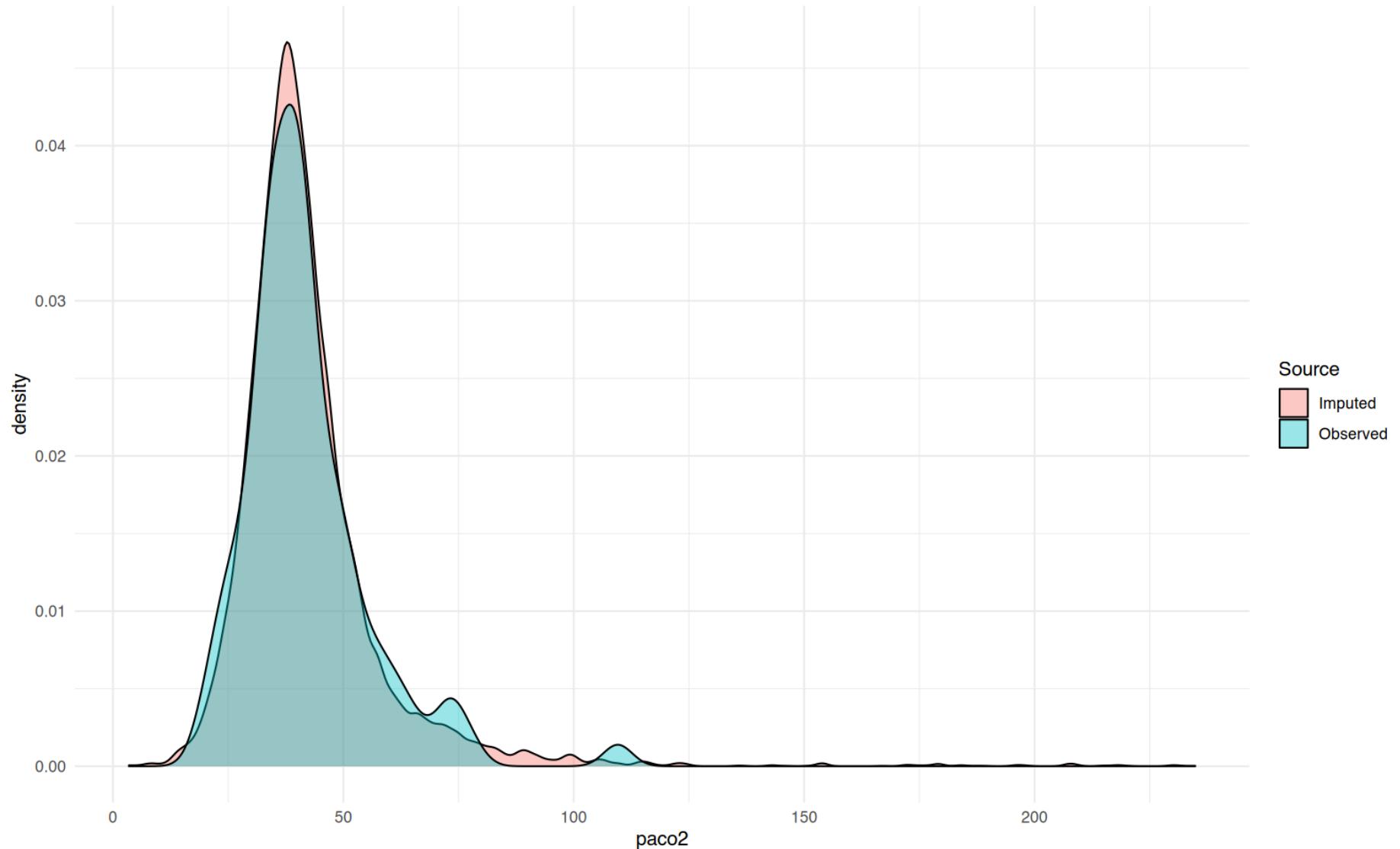
Warning: Removed 28880 rows containing non-finite outside the scale range
(`stat_density()`).

Observed vs imputed density: vbg_co2



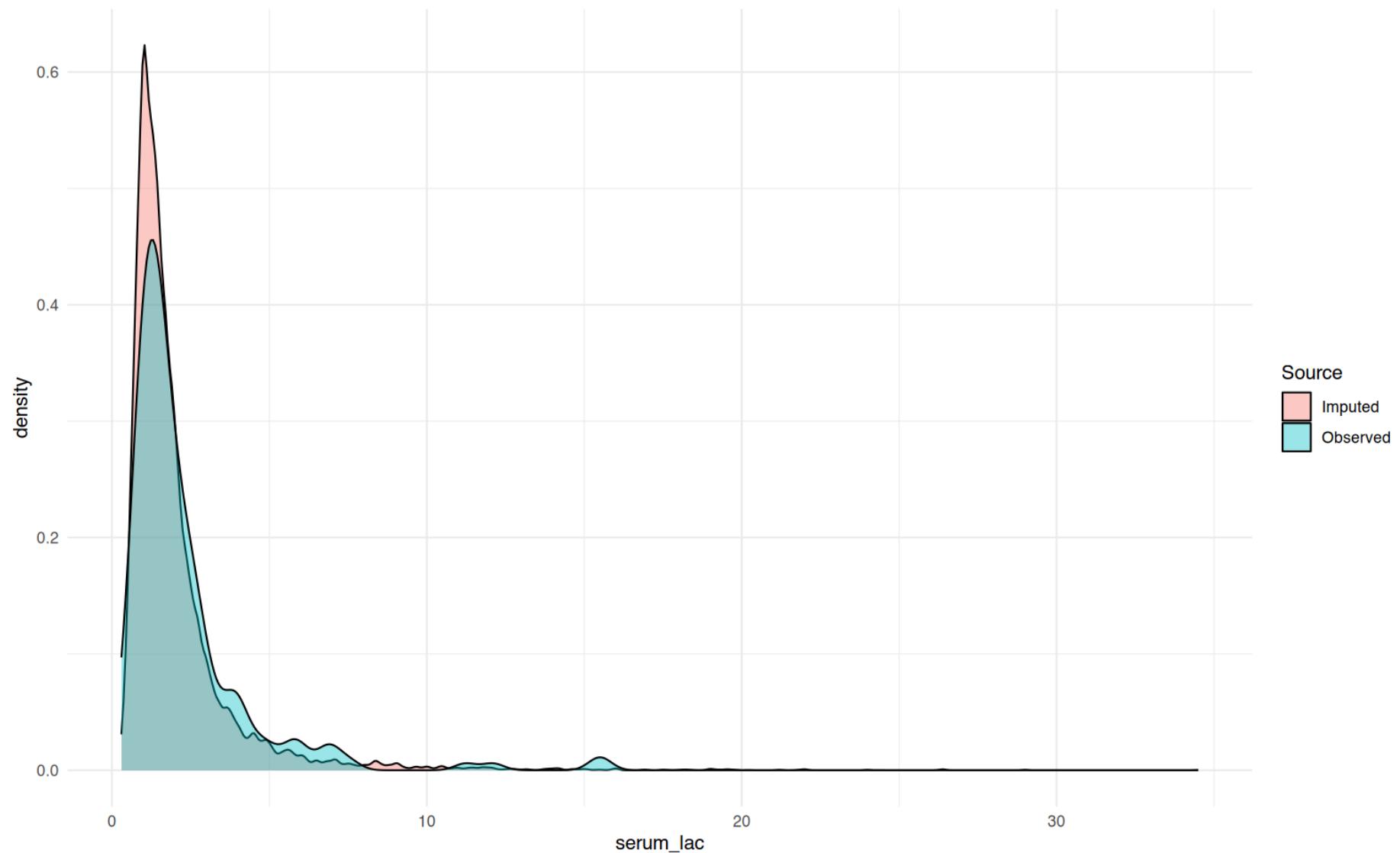
Warning: Removed 25772 rows containing non-finite outside the scale range
(`stat_density()`).

Observed vs imputed density: paco2



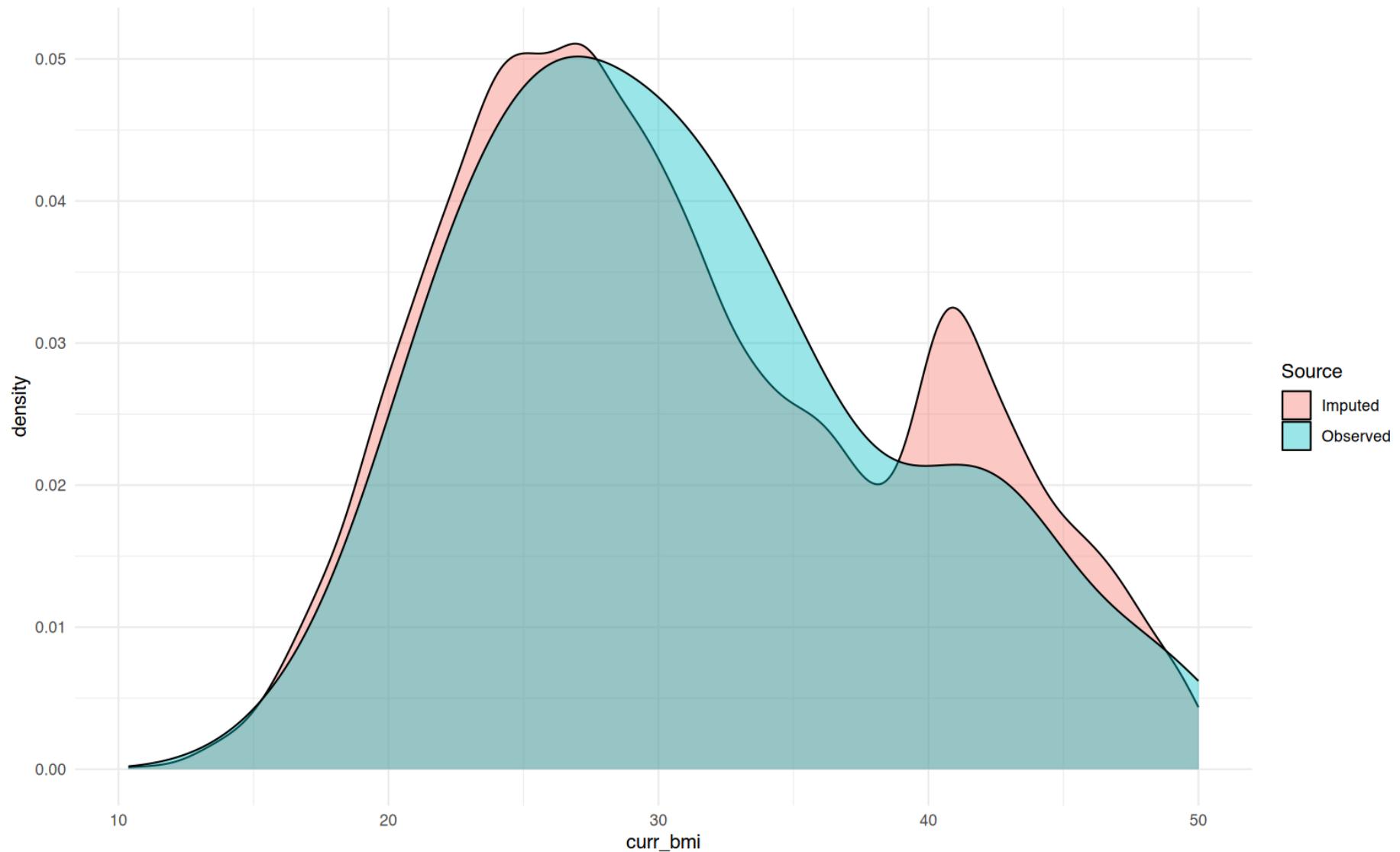
Warning: Removed 302 rows containing non-finite outside the scale range
(`stat_density()`).

Observed vs imputed density: serum_lac



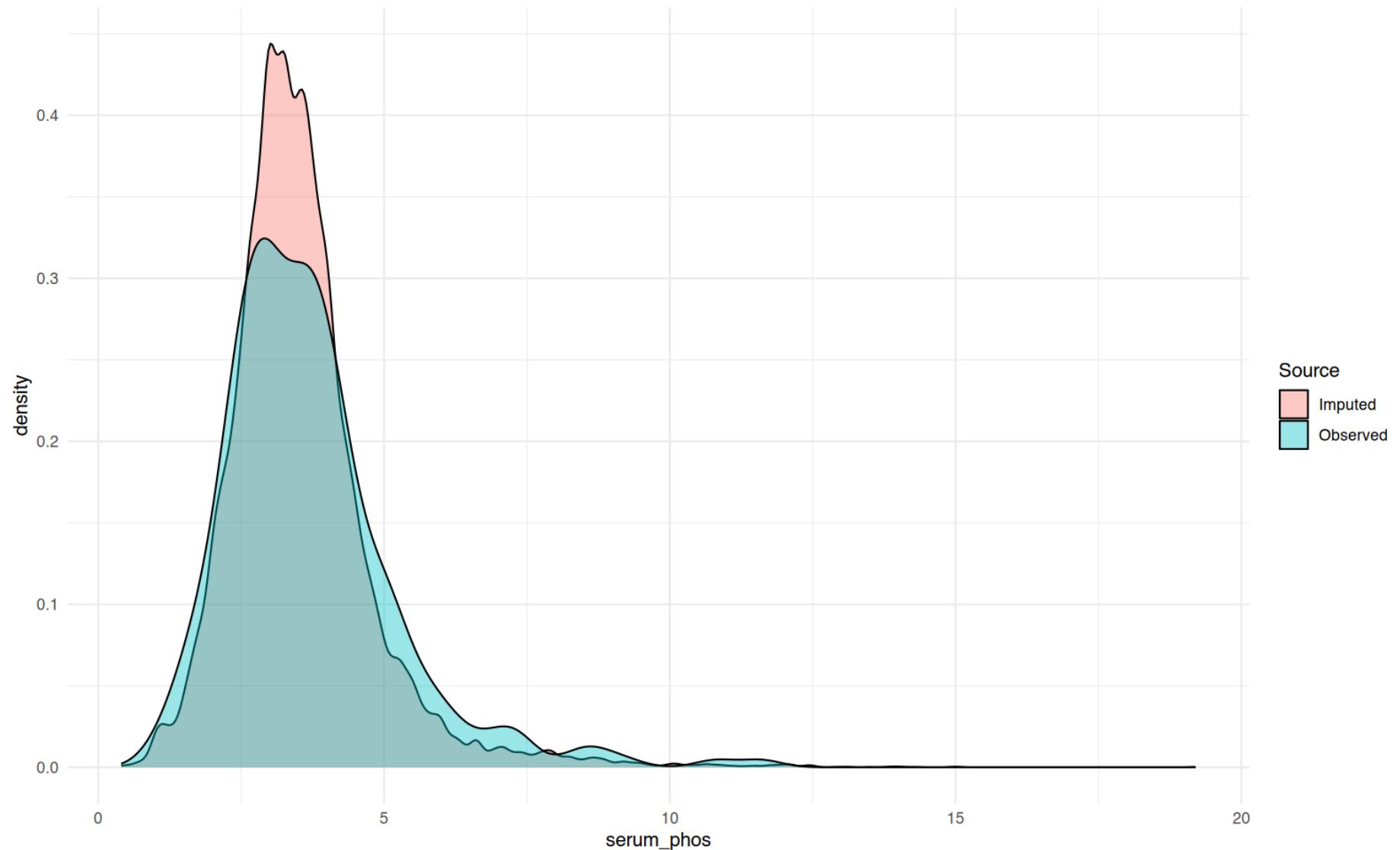
Warning: Removed 276 rows containing non-finite outside the scale range
(`stat_density()`).

Observed vs imputed density: curr_bmi



Warning: Removed 245 rows containing non-finite outside the scale range
(`stat_density()`).

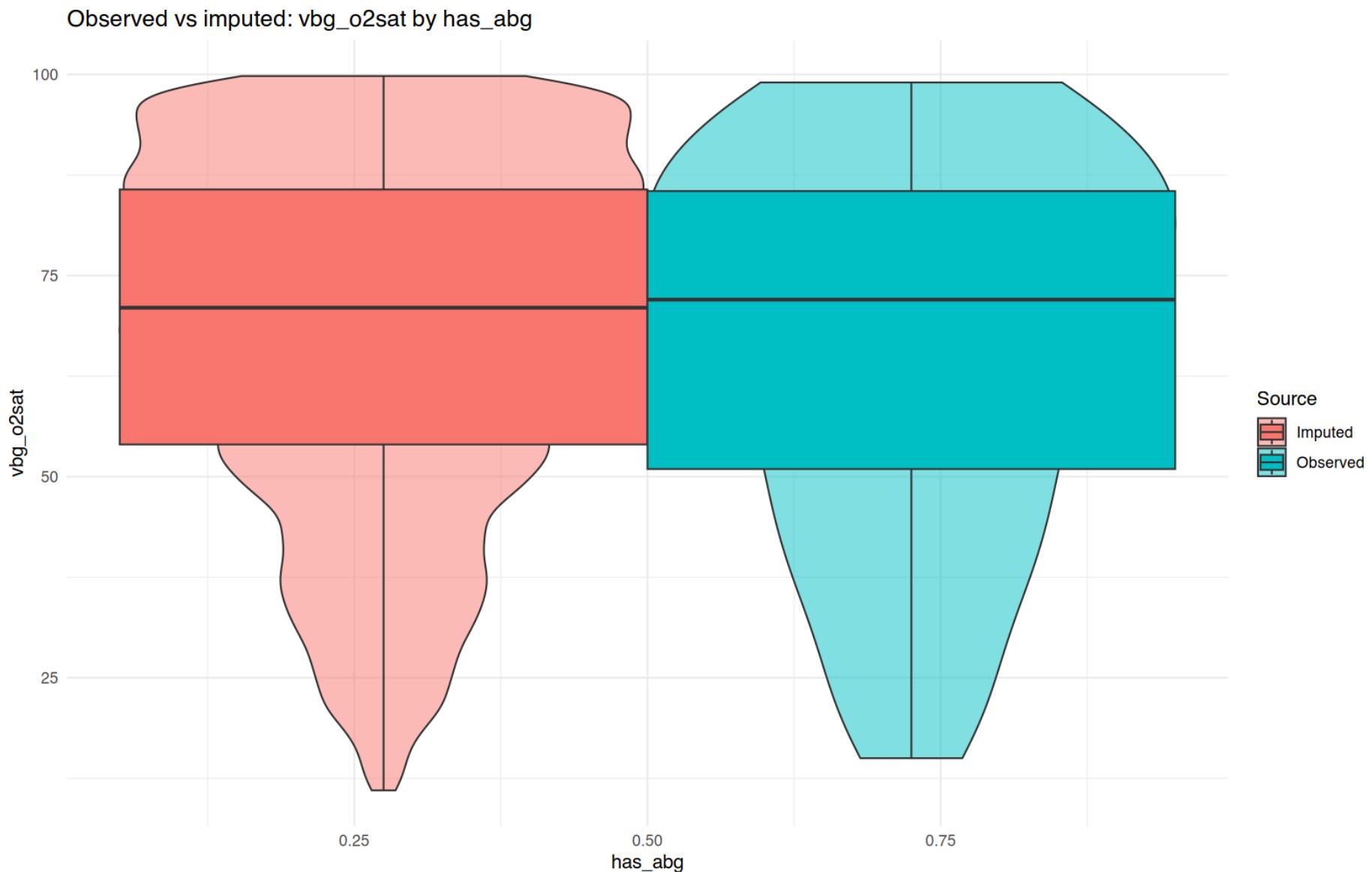
Observed vs imputed density: serum_phos



Warning: Removed 428 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 428 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

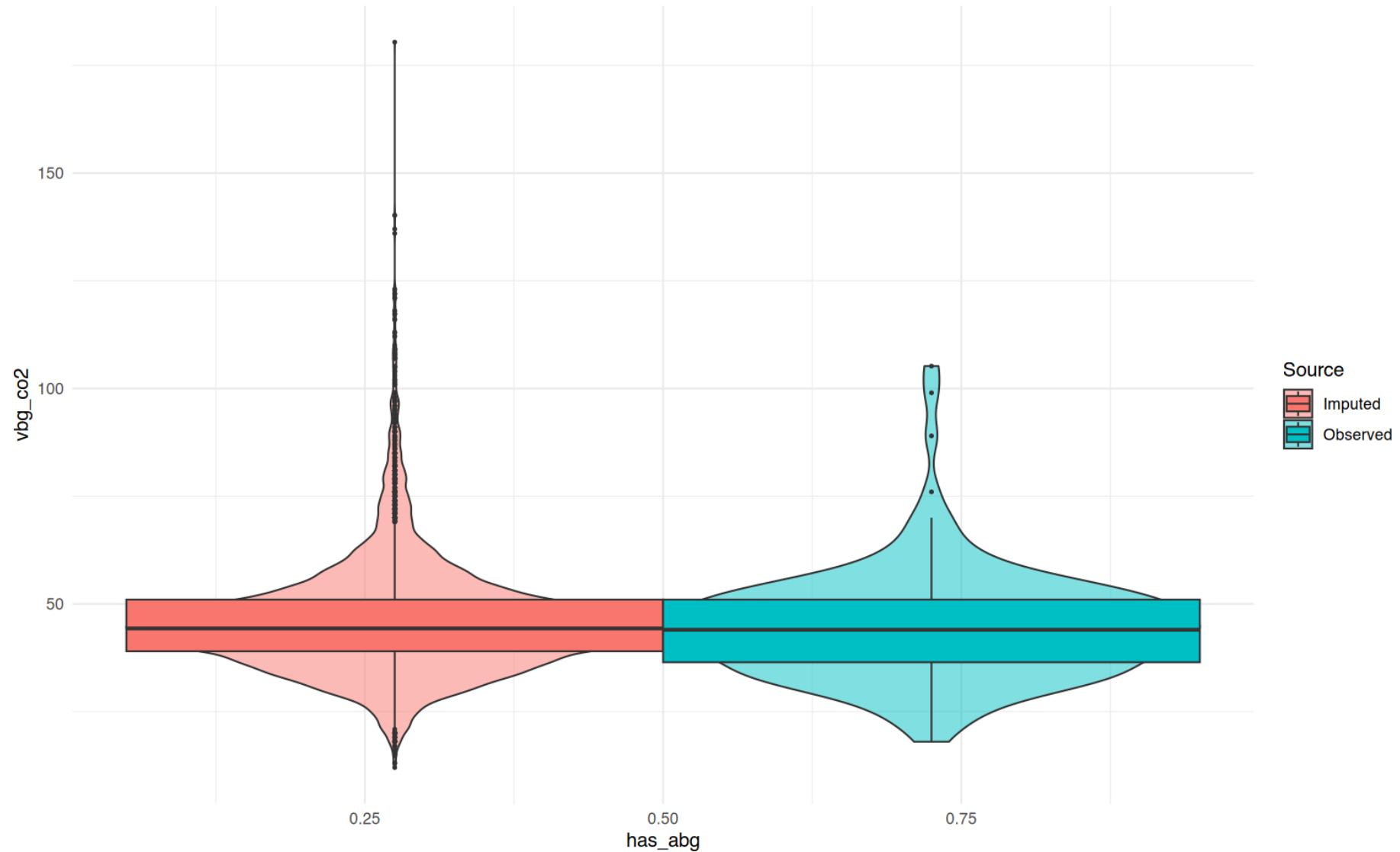


Warning: Removed 28880 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 28880 rows containing non-finite outside the scale range
(`stat_boxplot()`).

Observed vs imputed: vbg_co2 by has_abg

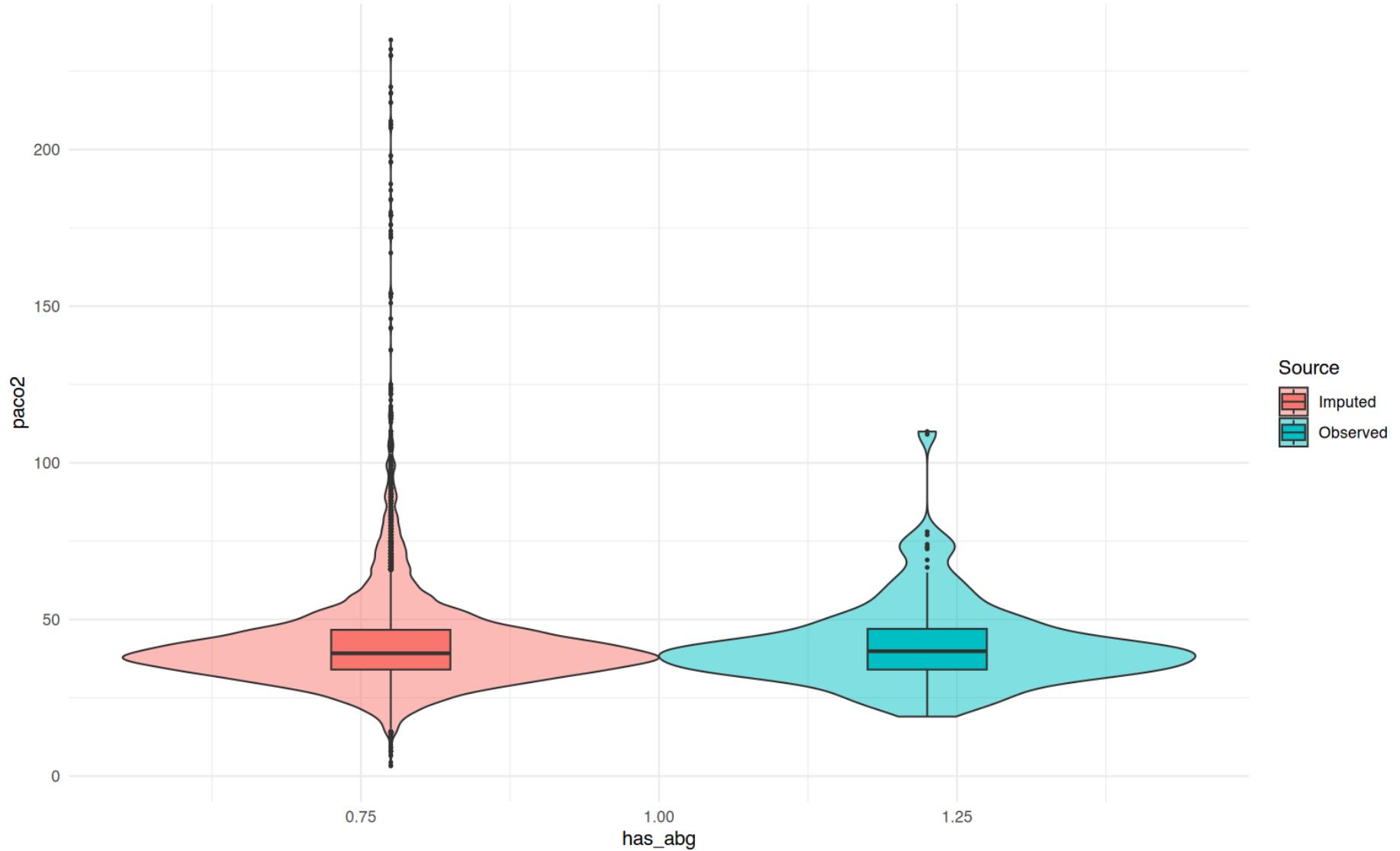


Warning: Removed 25772 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 25772 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

Observed vs imputed: paco2 by has_abg

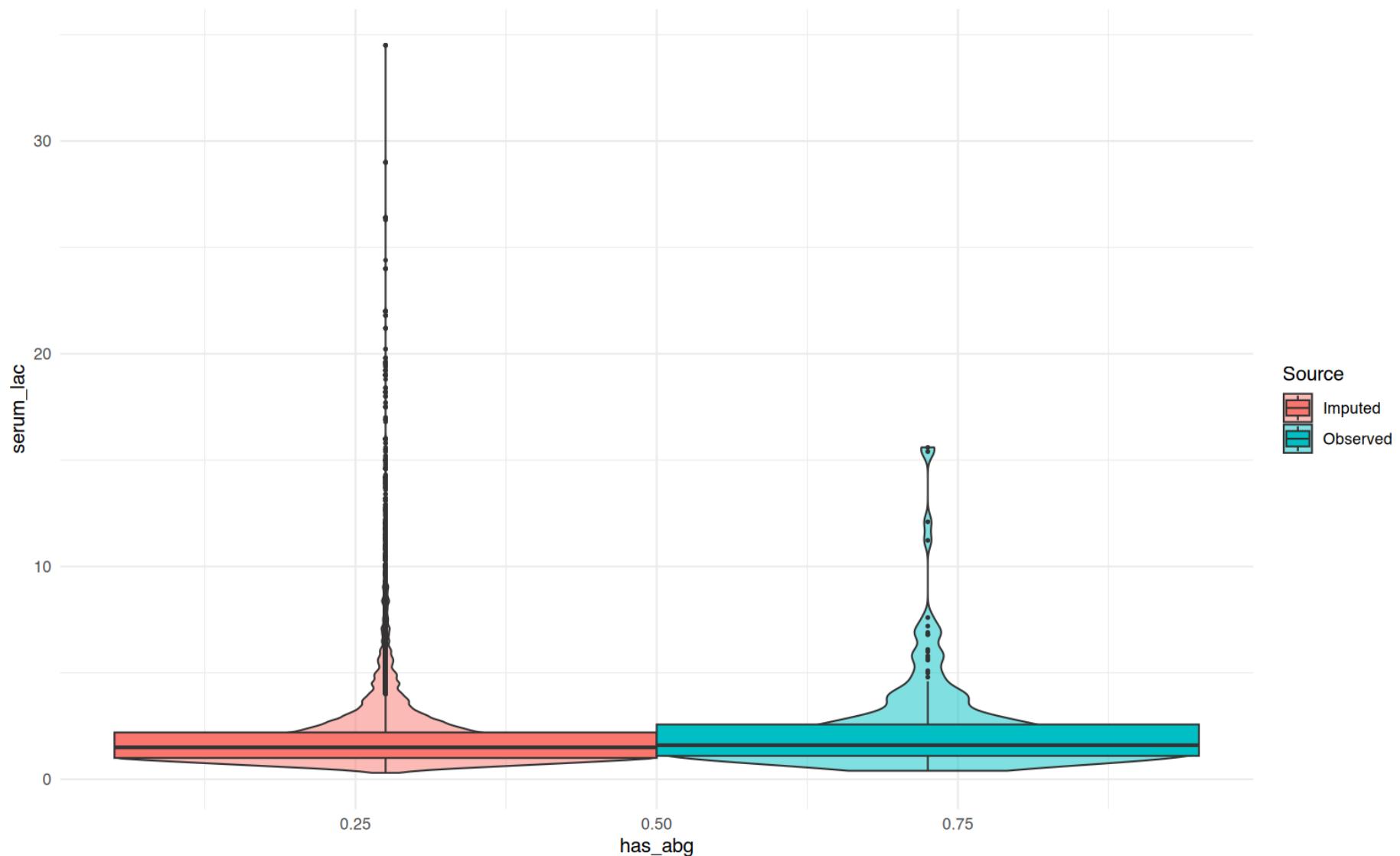


Warning: Removed 302 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 302 rows containing non-finite outside the scale range
(`stat_boxplot()`).

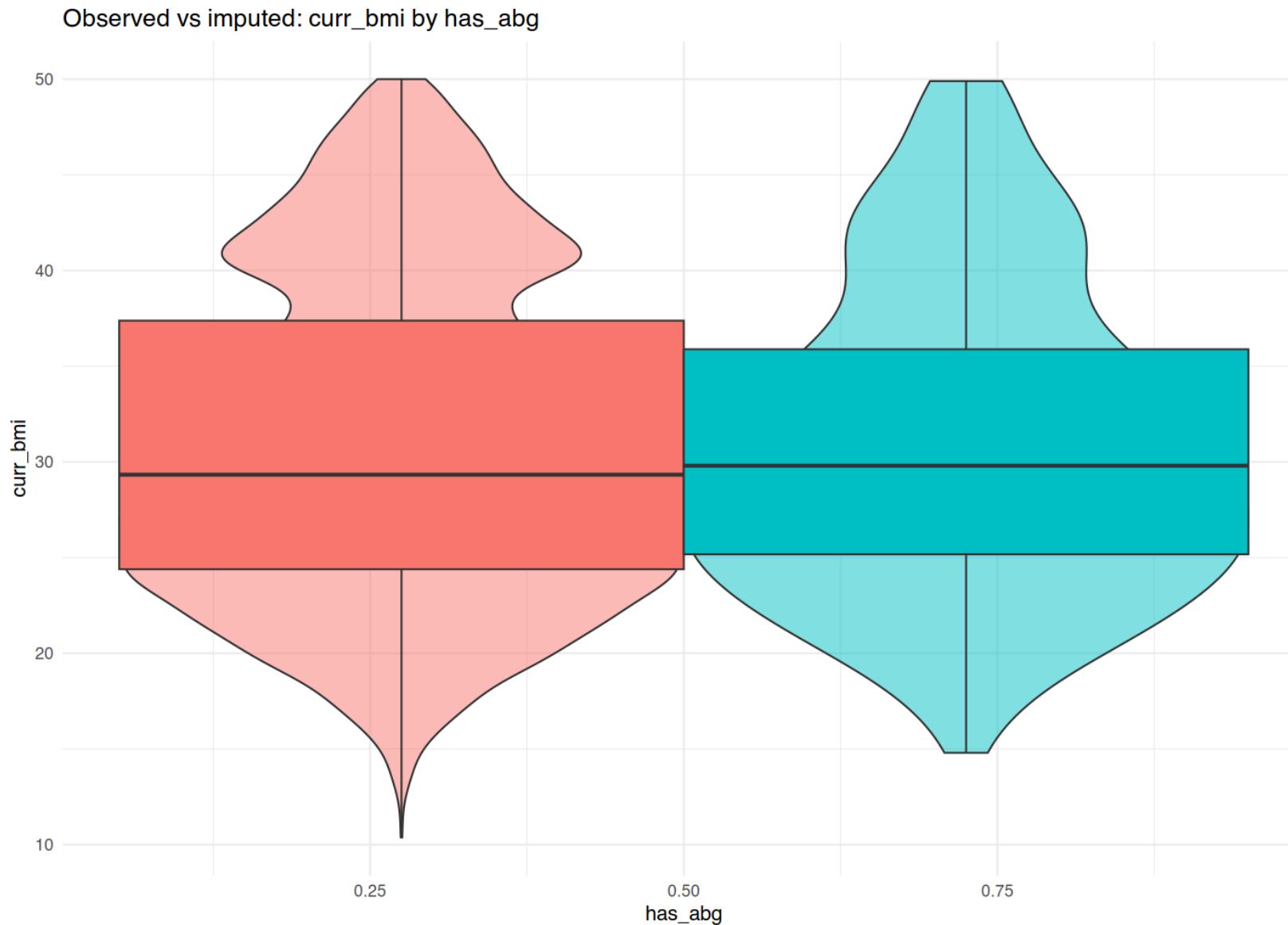
Observed vs imputed: serum_lac by has_abg



Warning: Removed 276 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 276 rows containing non-finite outside the scale range

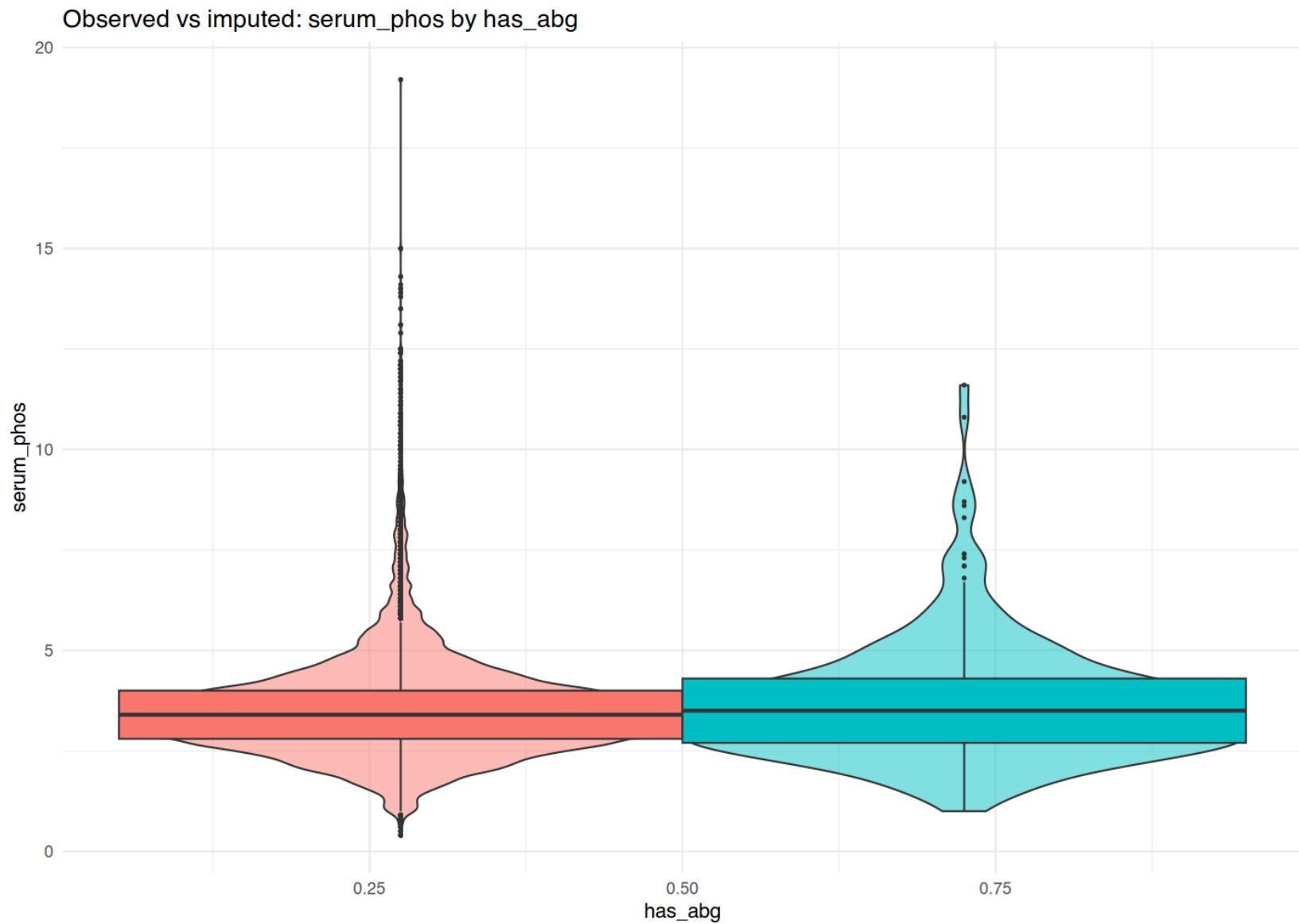
```
(`stat_boxplot()`).
```



Warning: Removed 245 rows containing non-finite outside the scale range

(`stat_ydensity()`).

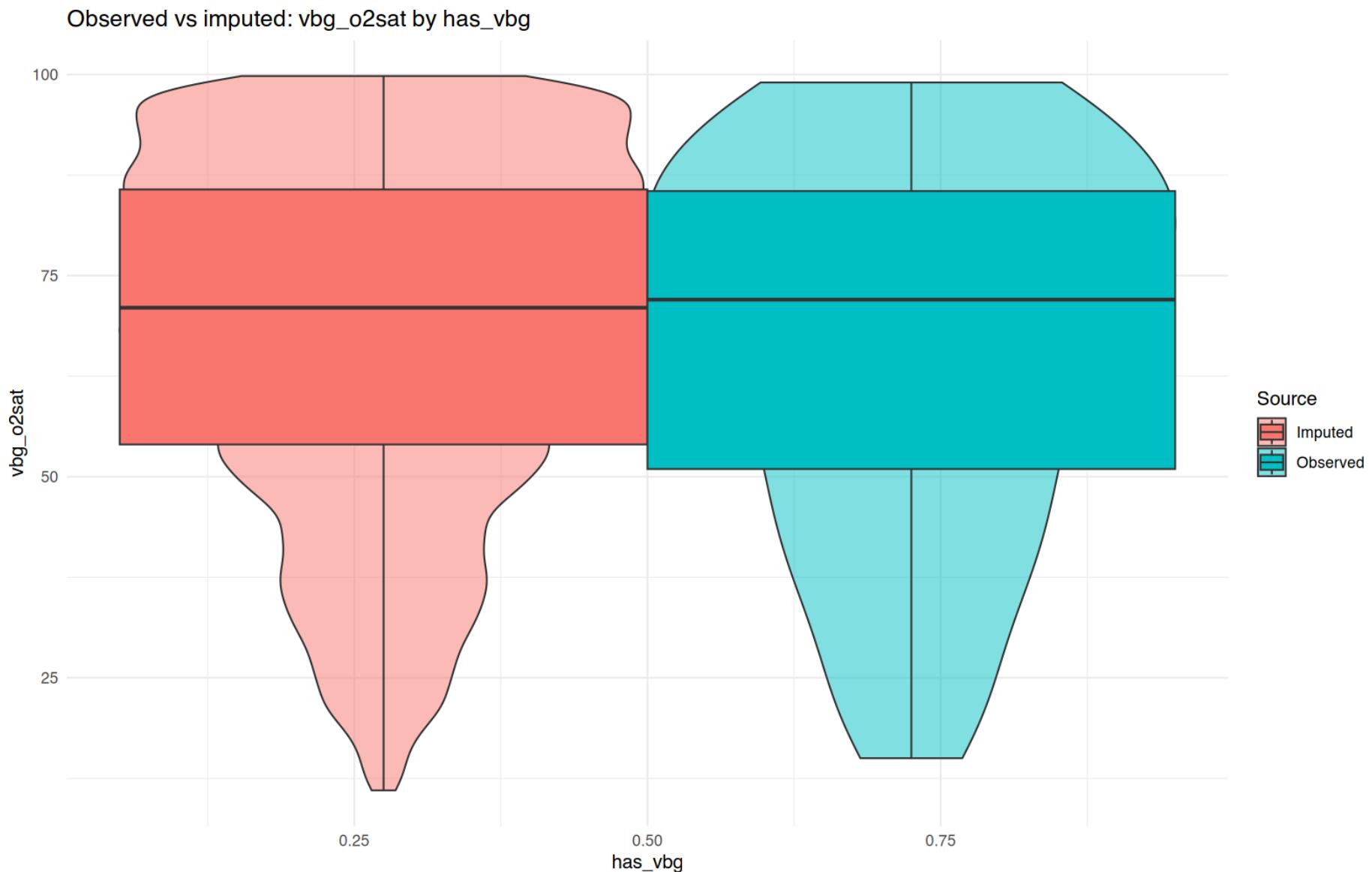
Warning: Removed 245 rows containing non-finite outside the scale range
(`stat_boxplot()`).



Warning: Removed 428 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 428 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

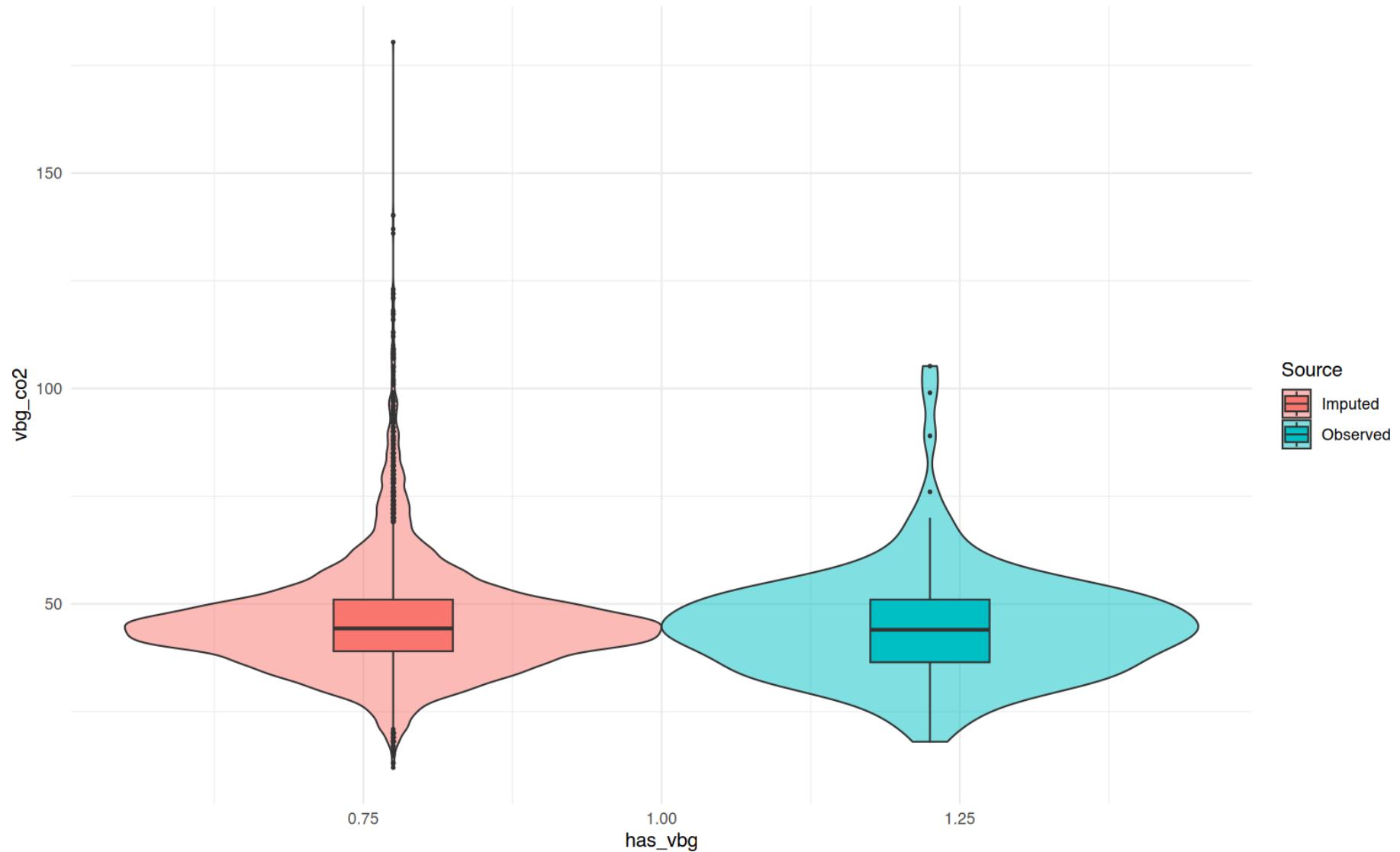


Warning: Removed 28880 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 28880 rows containing non-finite outside the scale range
(`stat_boxplot()`).

Observed vs imputed: vbg_co2 by has_vbg

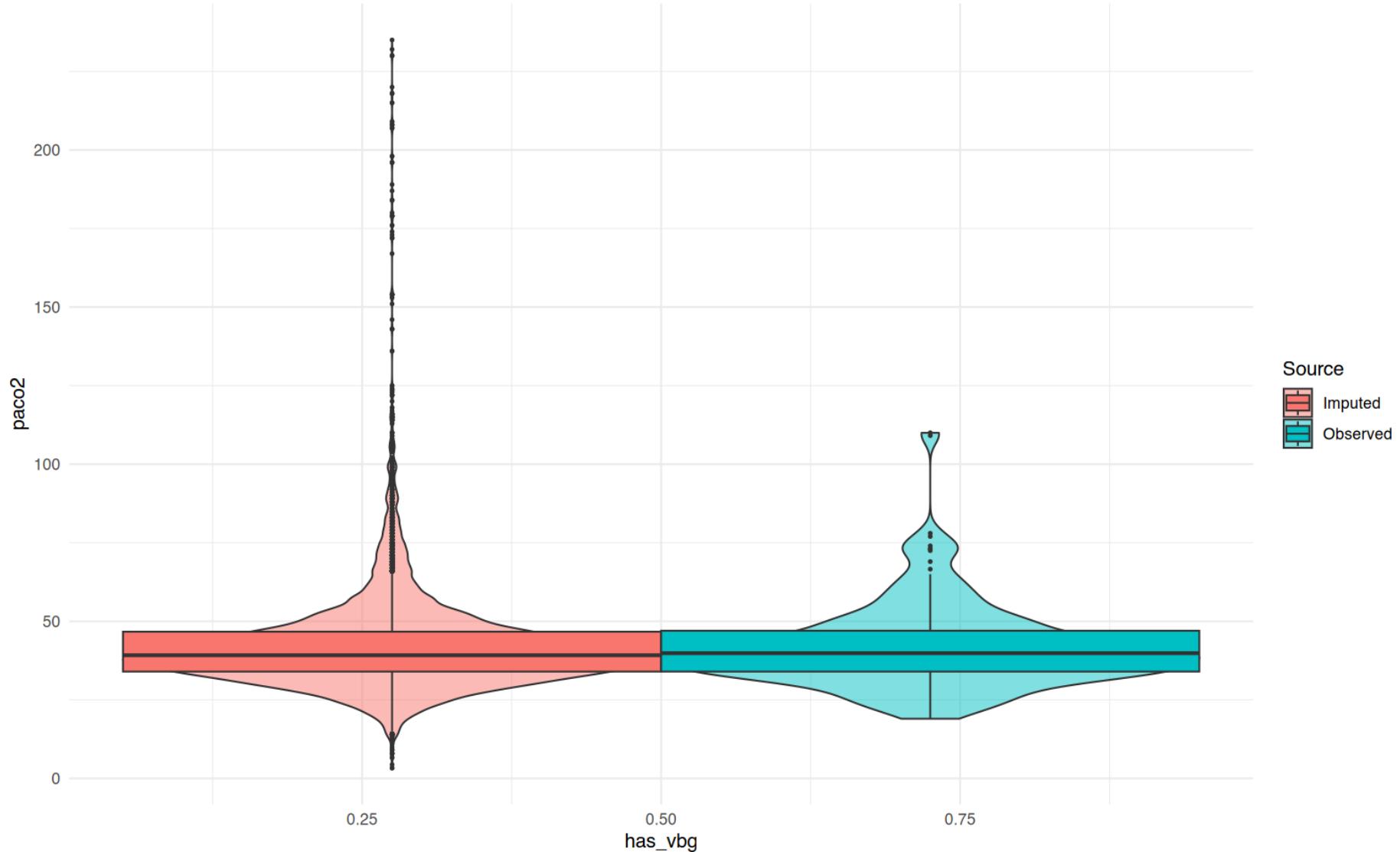


Warning: Removed 25772 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 25772 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

Observed vs imputed: paco2 by has_vbg

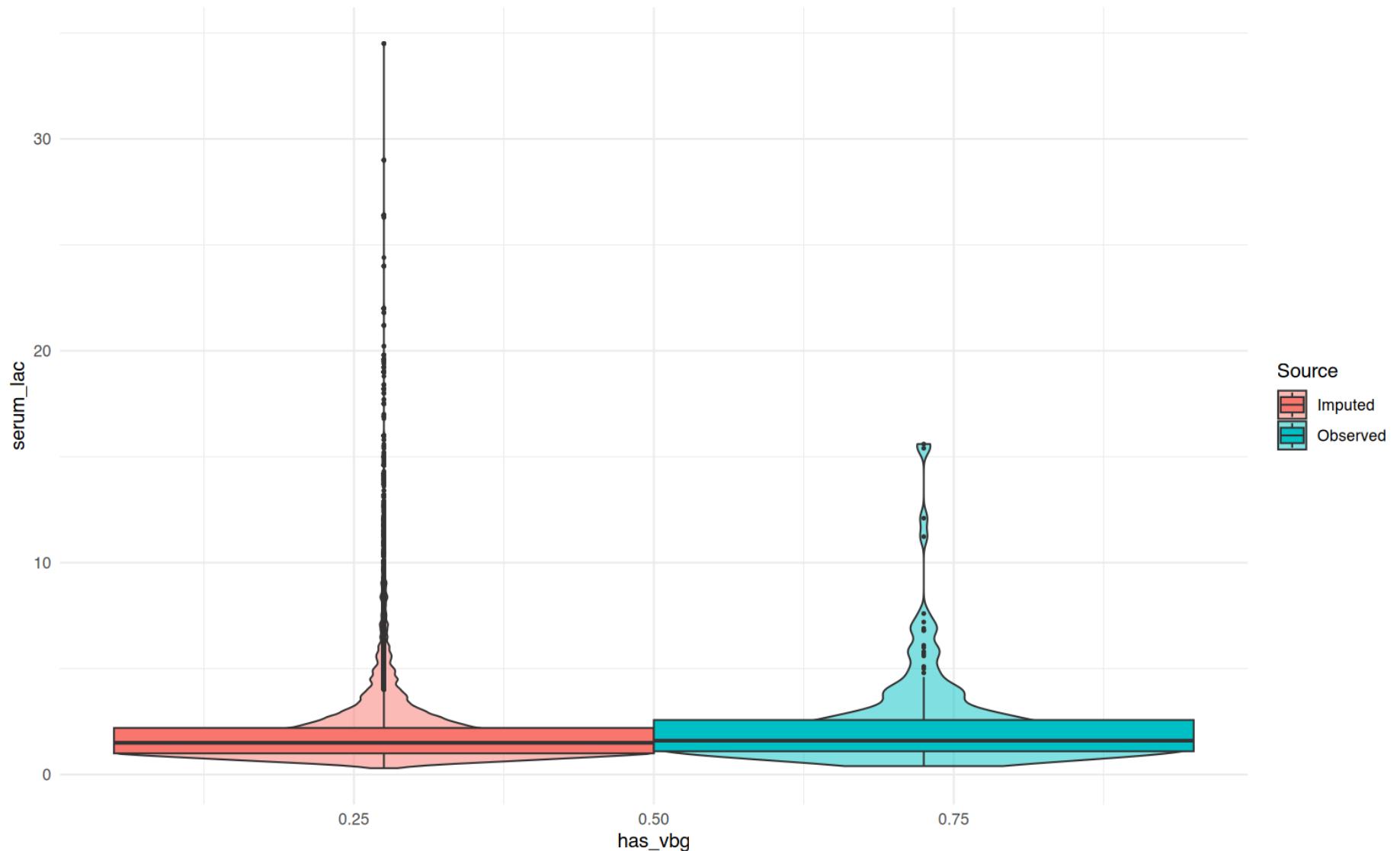


Warning: Removed 302 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 302 rows containing non-finite outside the scale range
(`stat_boxplot()`).

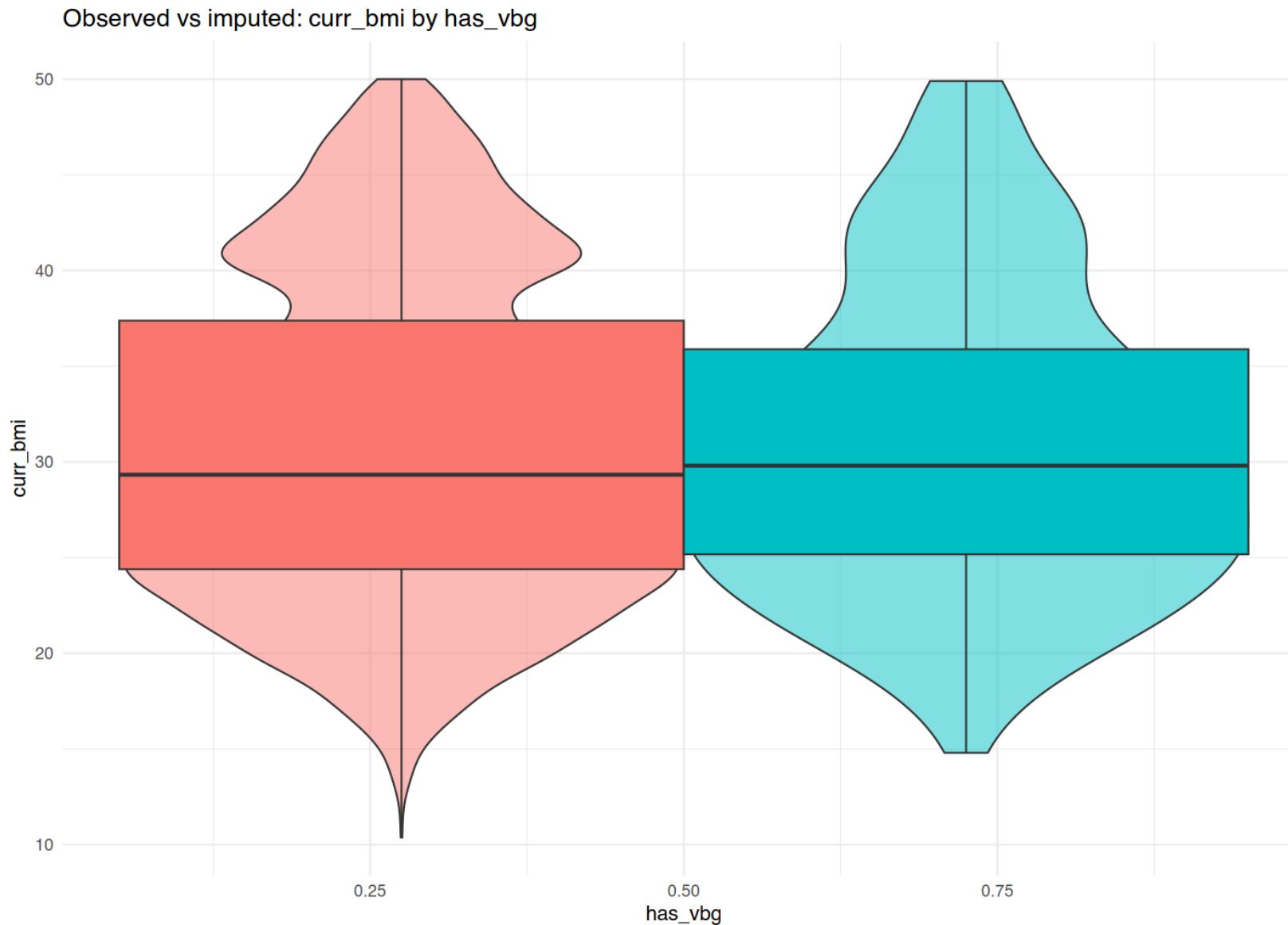
Observed vs imputed: serum_lac by has_vbg



Warning: Removed 276 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 276 rows containing non-finite outside the scale range

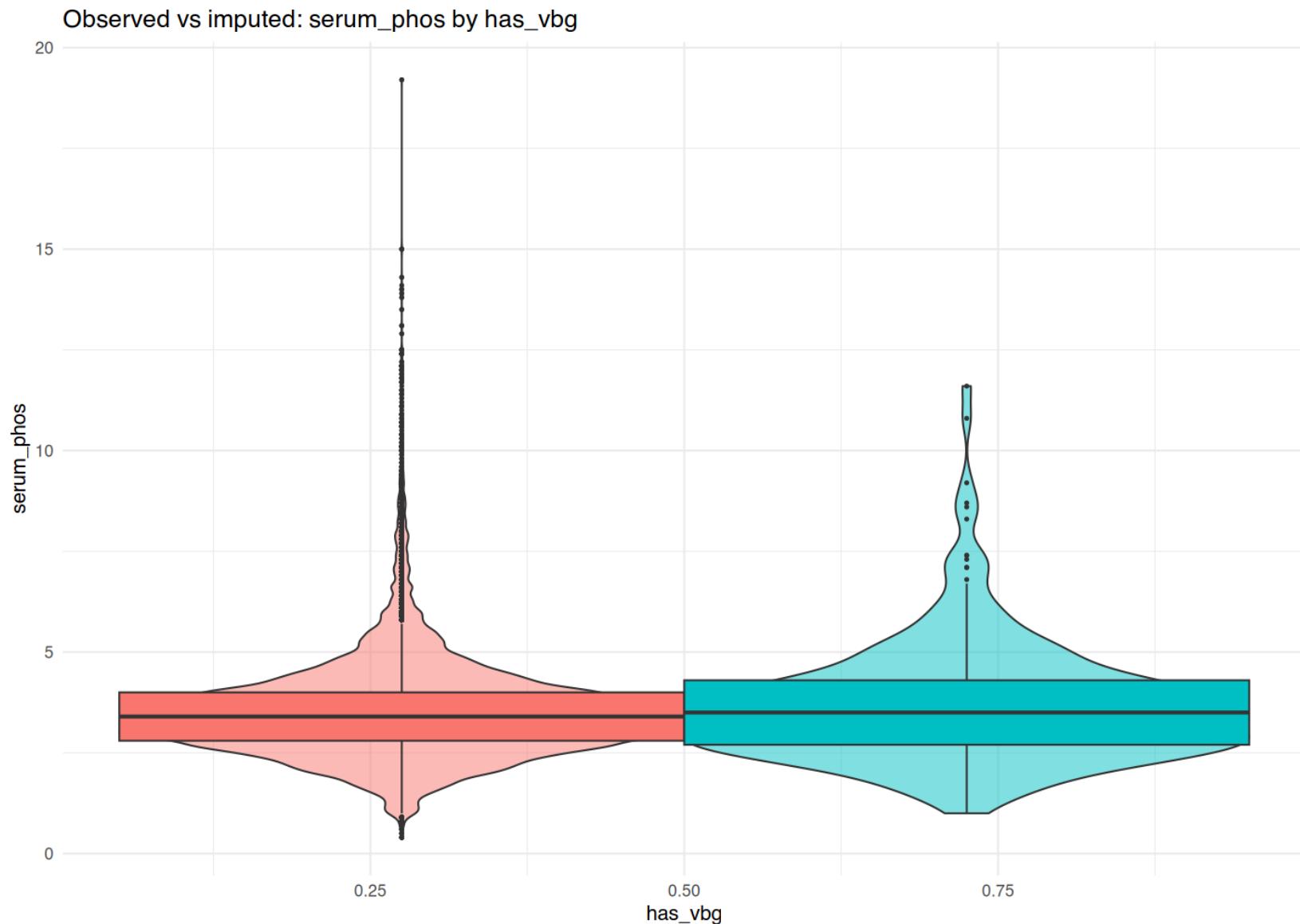
```
(`stat_boxplot()`).
```



Warning: Removed 245 rows containing non-finite outside the scale range

(`stat_ydensity()`).

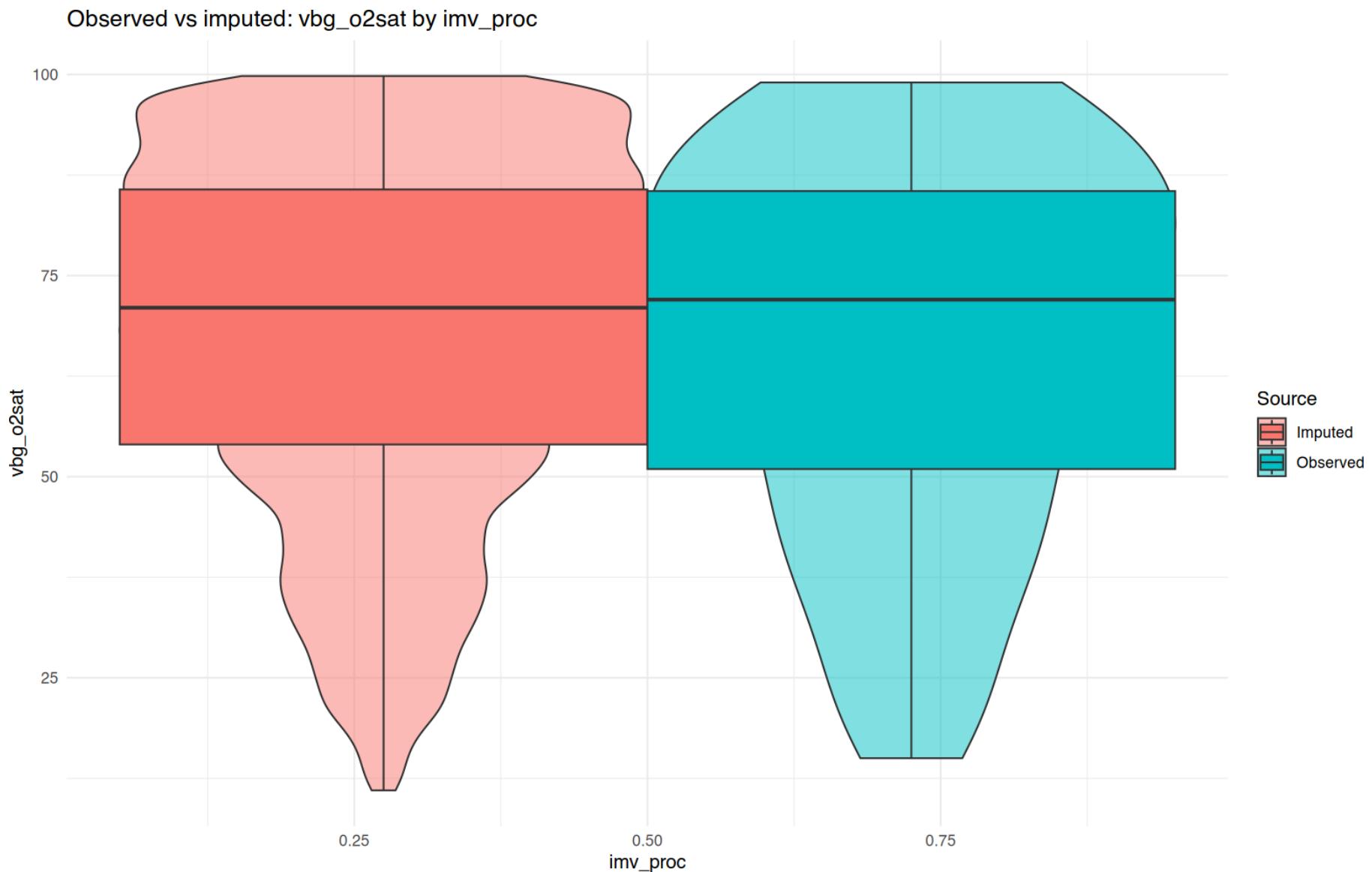
Warning: Removed 245 rows containing non-finite outside the scale range
(`stat_boxplot()`).



Warning: Removed 428 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 428 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

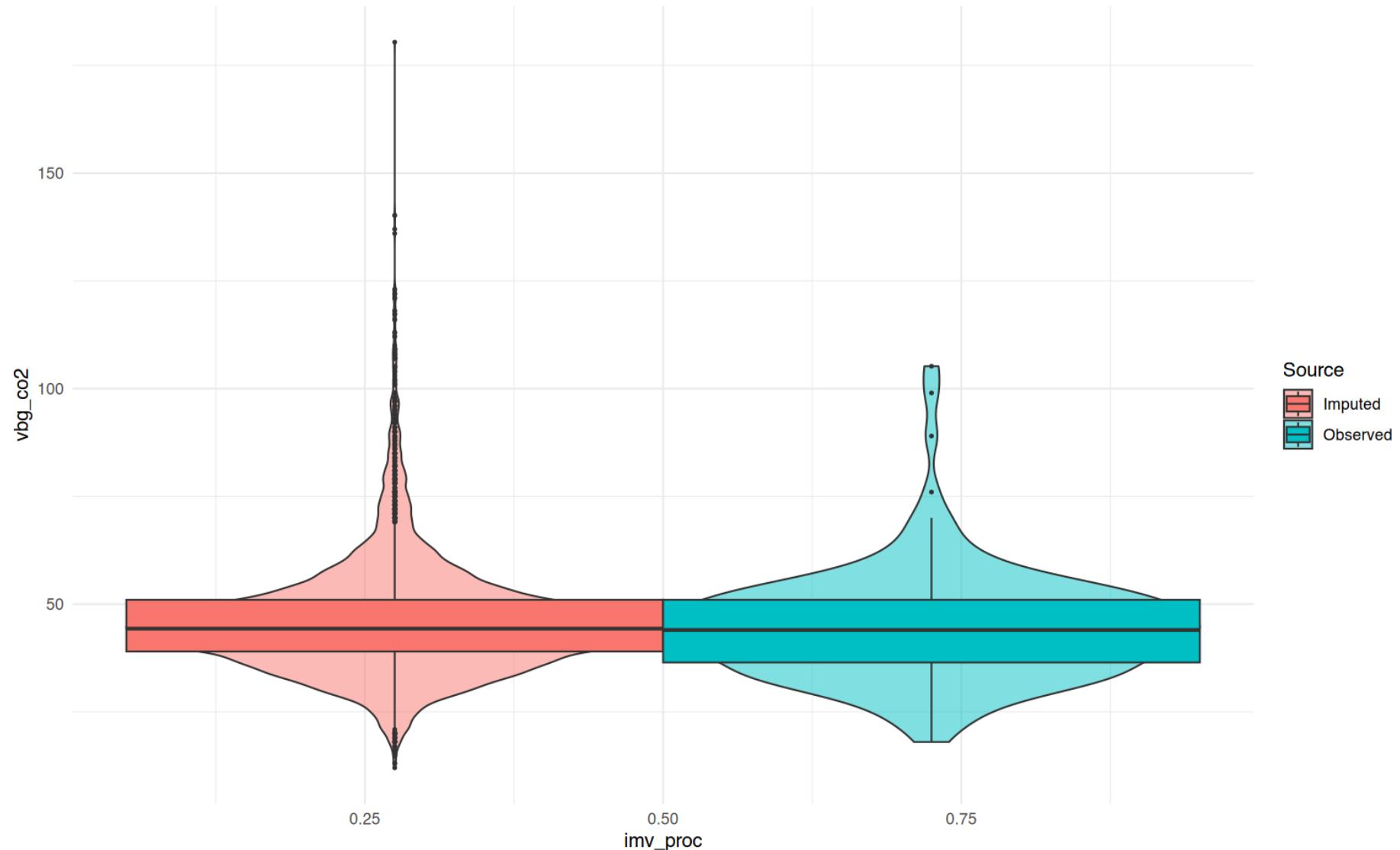


Warning: Removed 28880 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 28880 rows containing non-finite outside the scale range
(`stat_boxplot()`).

Observed vs imputed: vbg_co2 by imv_proc

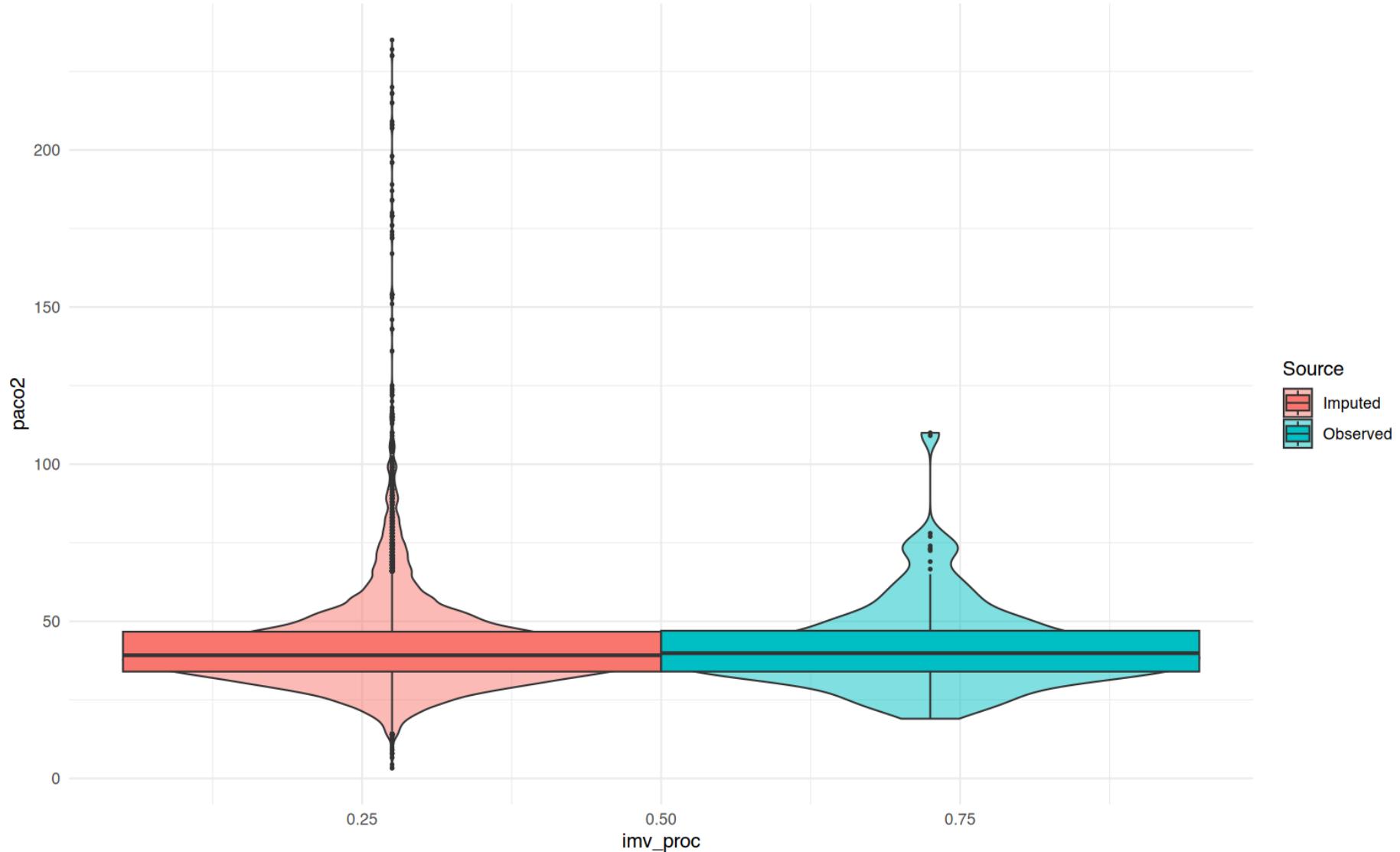


Warning: Removed 25772 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 25772 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

Observed vs imputed: paco2 by imv_proc

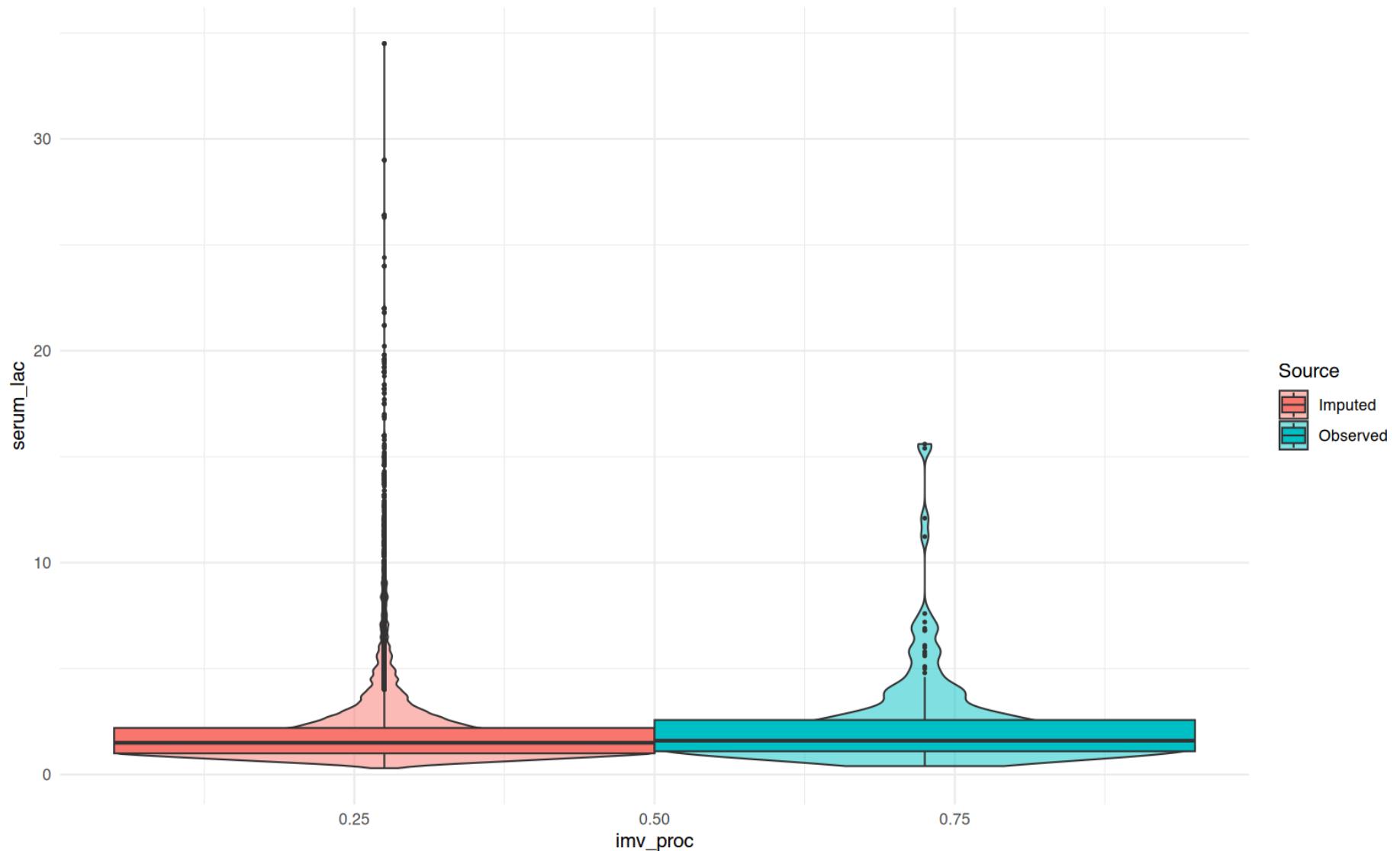


Warning: Removed 302 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 302 rows containing non-finite outside the scale range
(`stat_boxplot()`).

Observed vs imputed: serum_lac by imv_proc

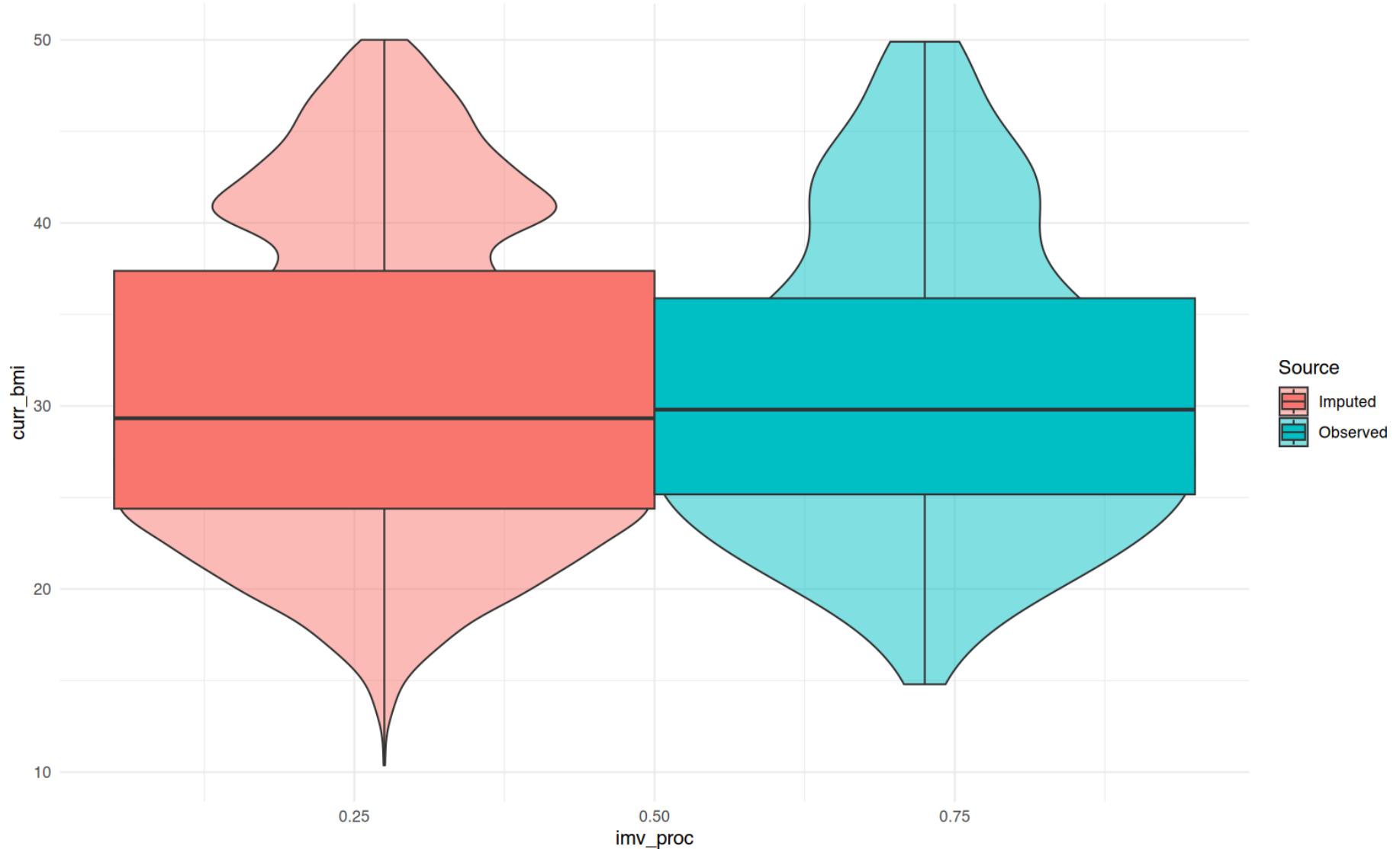


Warning: Removed 276 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 276 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

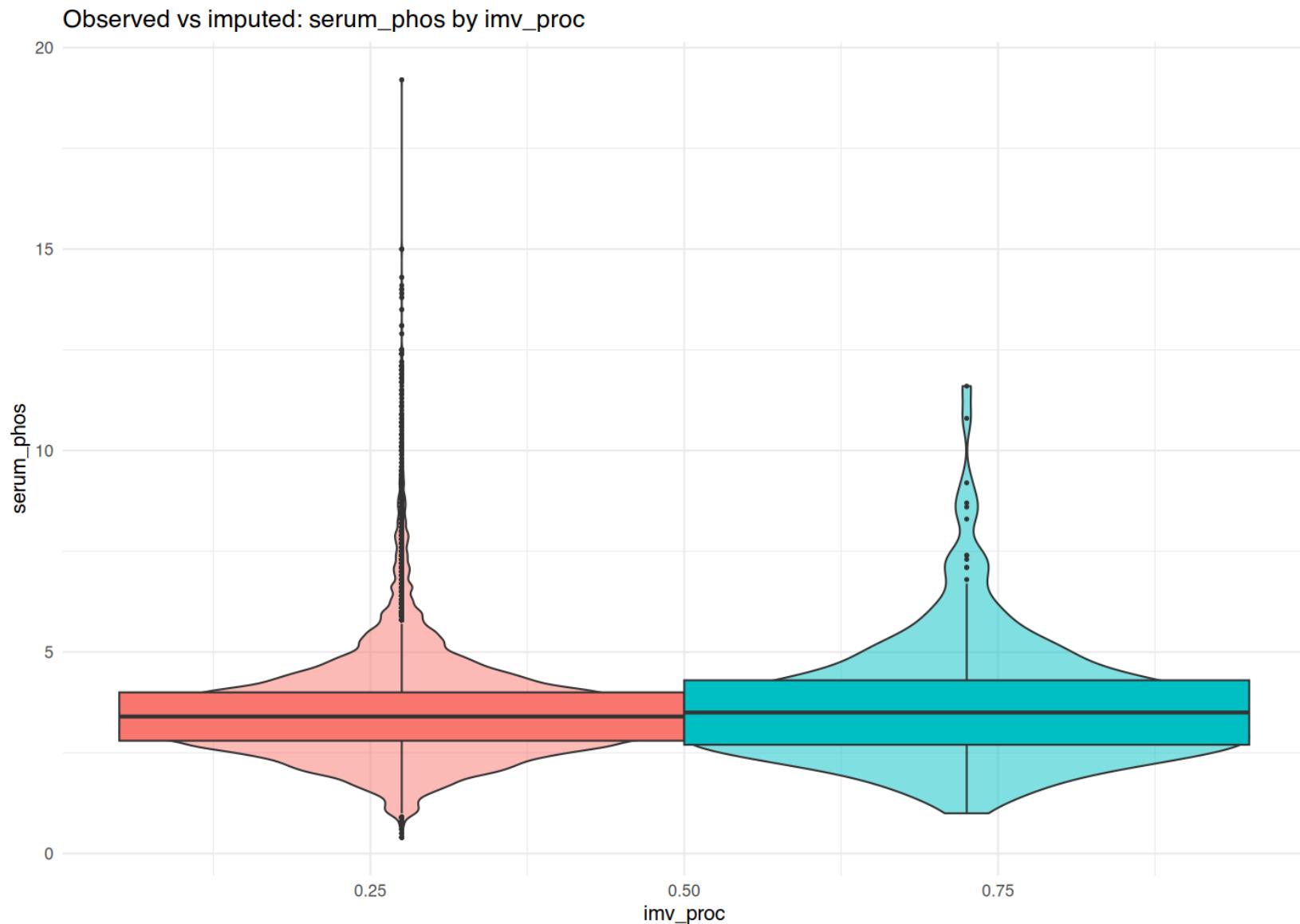
Observed vs imputed: curr_bmi by imv_proc



Warning: Removed 245 rows containing non-finite outside the scale range

(`stat_ydensity()`).

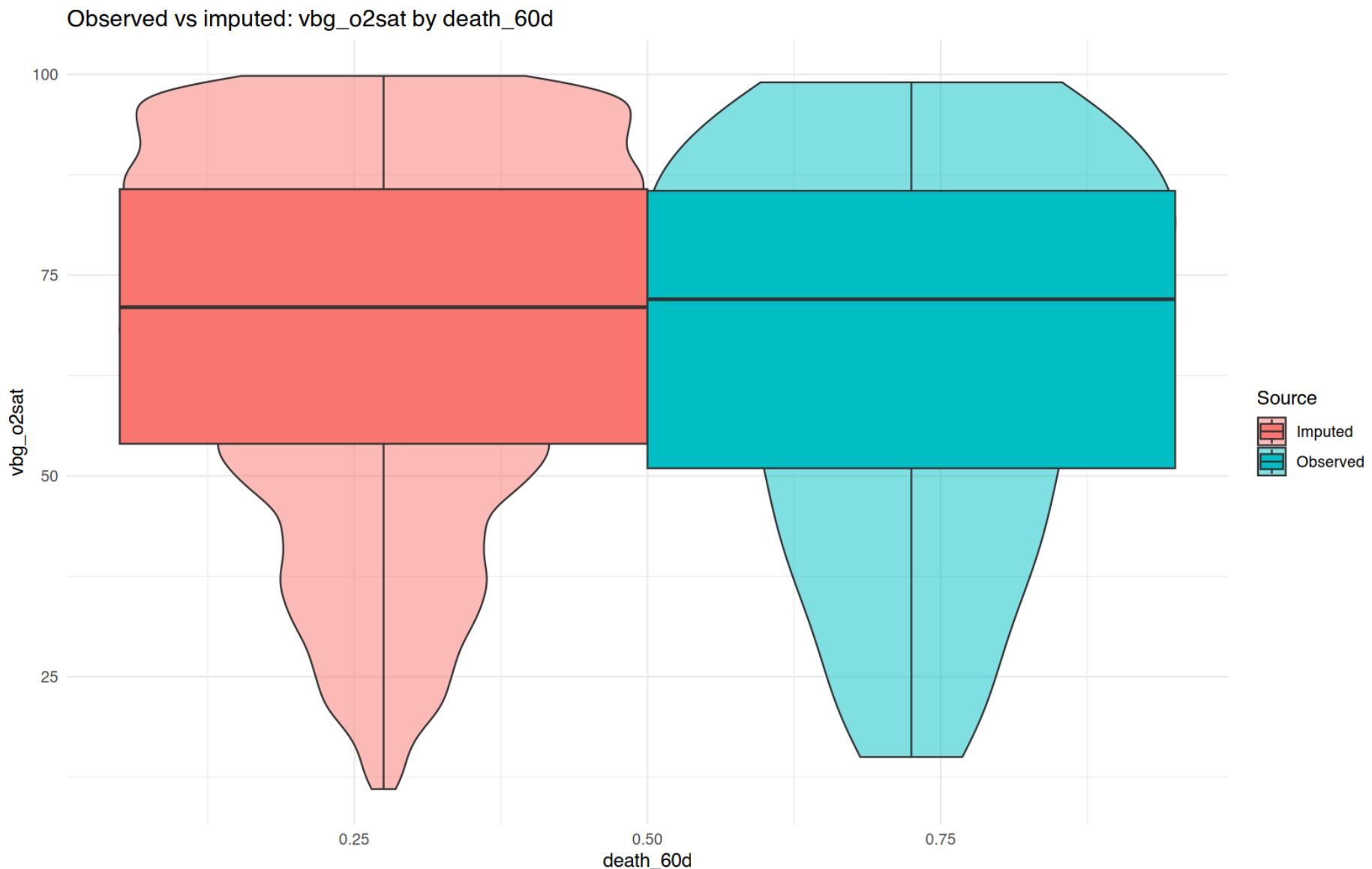
Warning: Removed 245 rows containing non-finite outside the scale range
(`stat_boxplot()`).



Warning: Removed 428 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 428 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

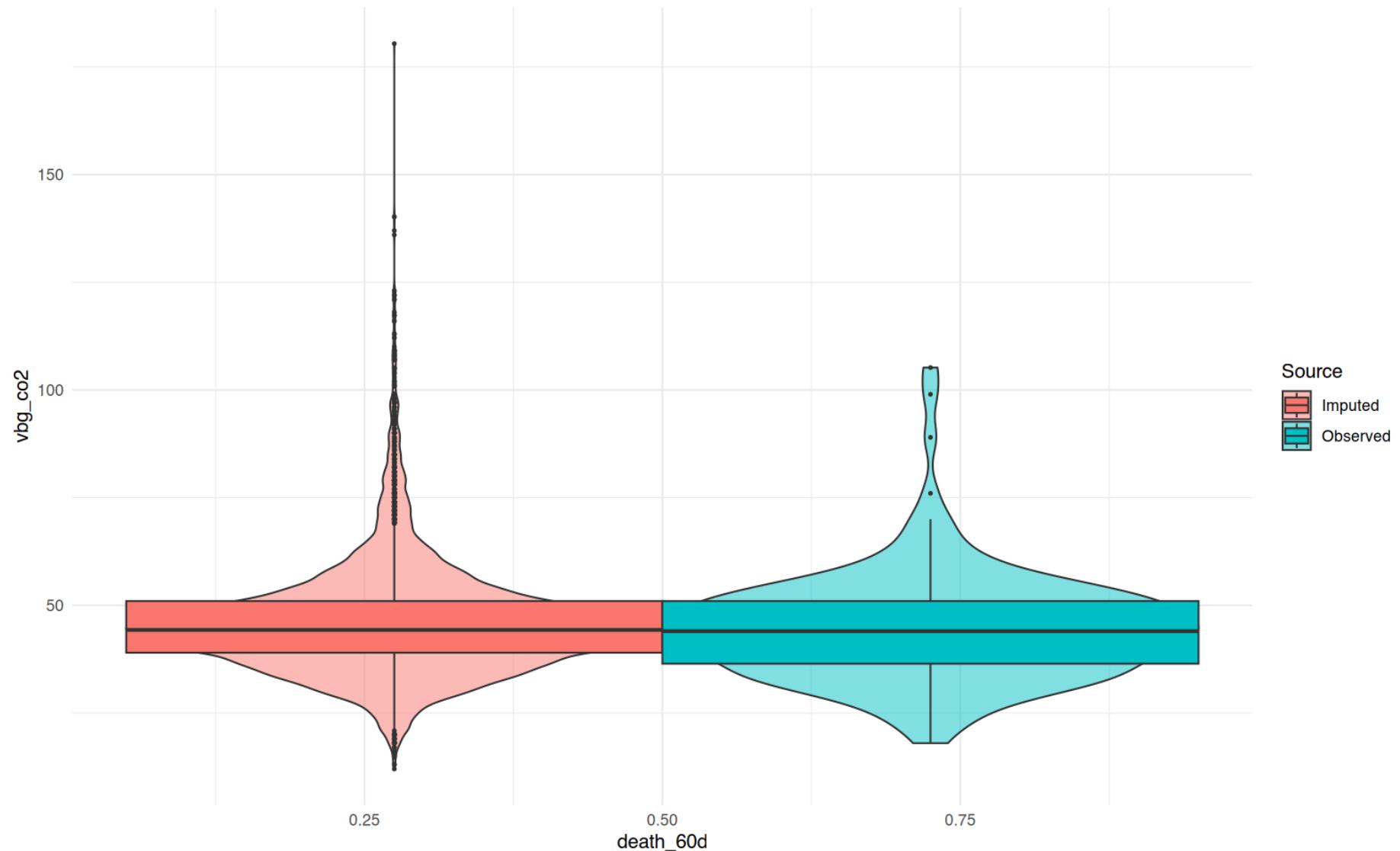


Warning: Removed 28880 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 28880 rows containing non-finite outside the scale range
(`stat_boxplot()`).

Observed vs imputed: vbg_co2 by death_60d

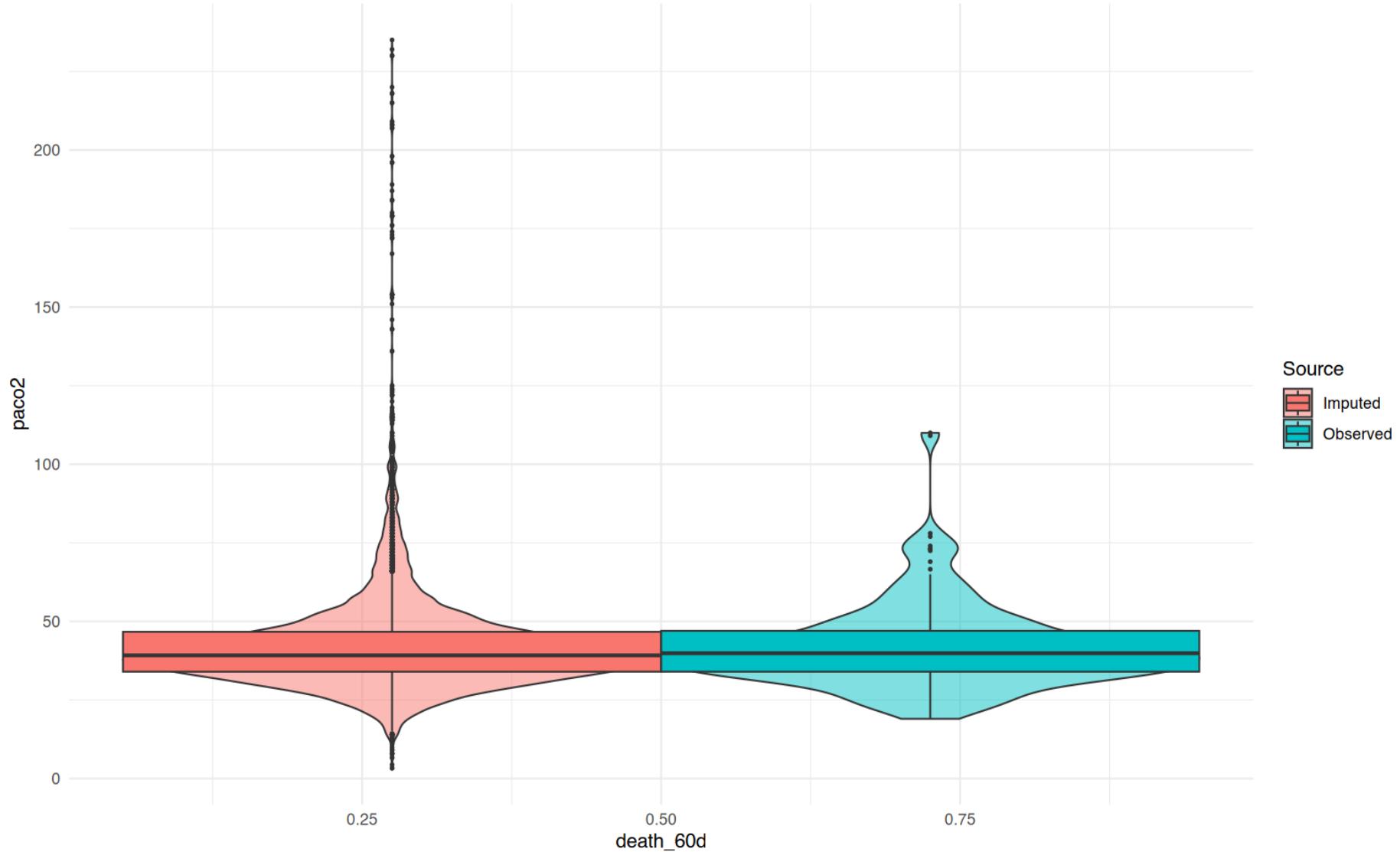


Warning: Removed 25772 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 25772 rows containing non-finite outside the scale range

```
(`stat_boxplot()`).
```

Observed vs imputed: paco2 by death_60d

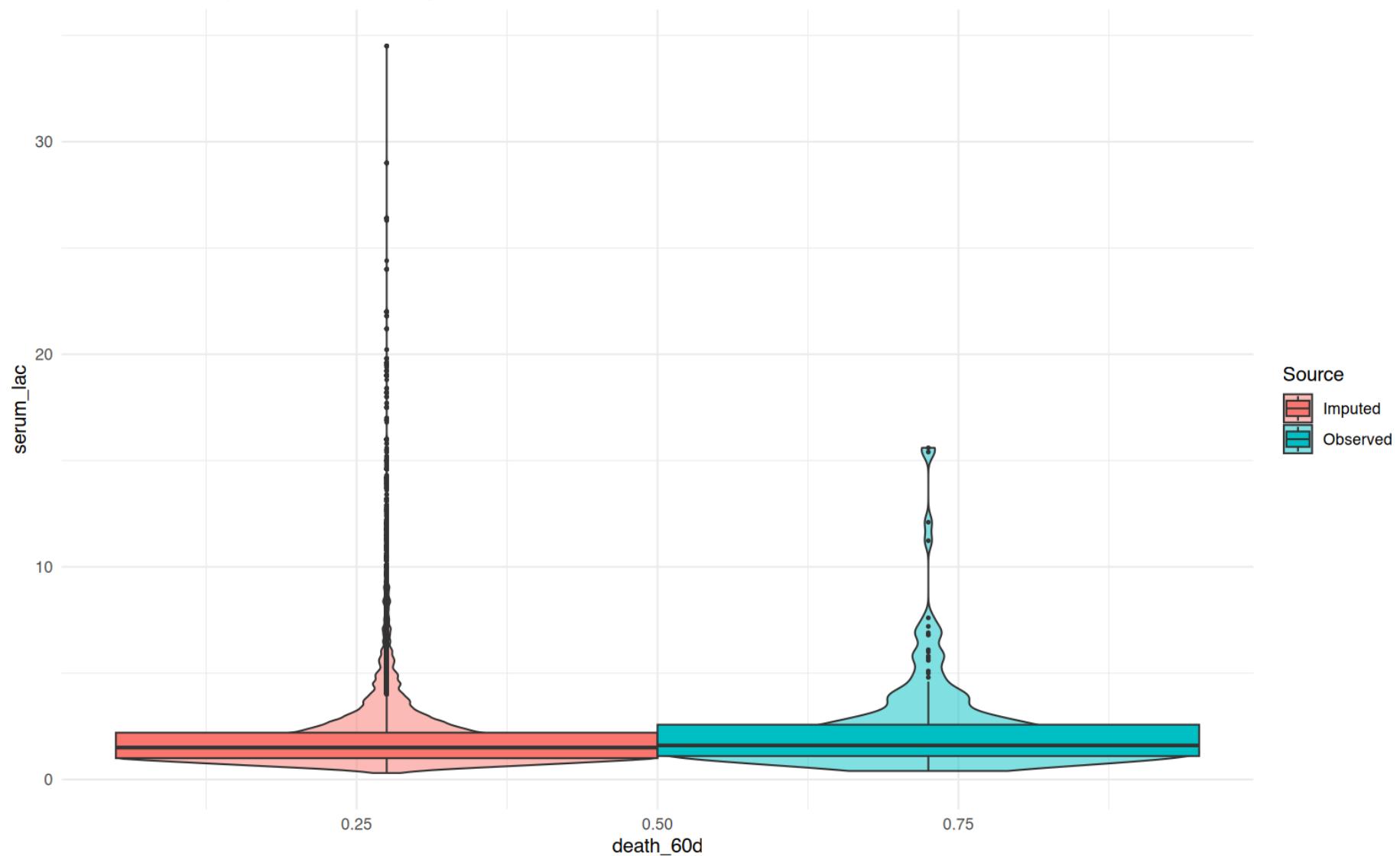


Warning: Removed 302 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 302 rows containing non-finite outside the scale range
(`stat_boxplot()`).

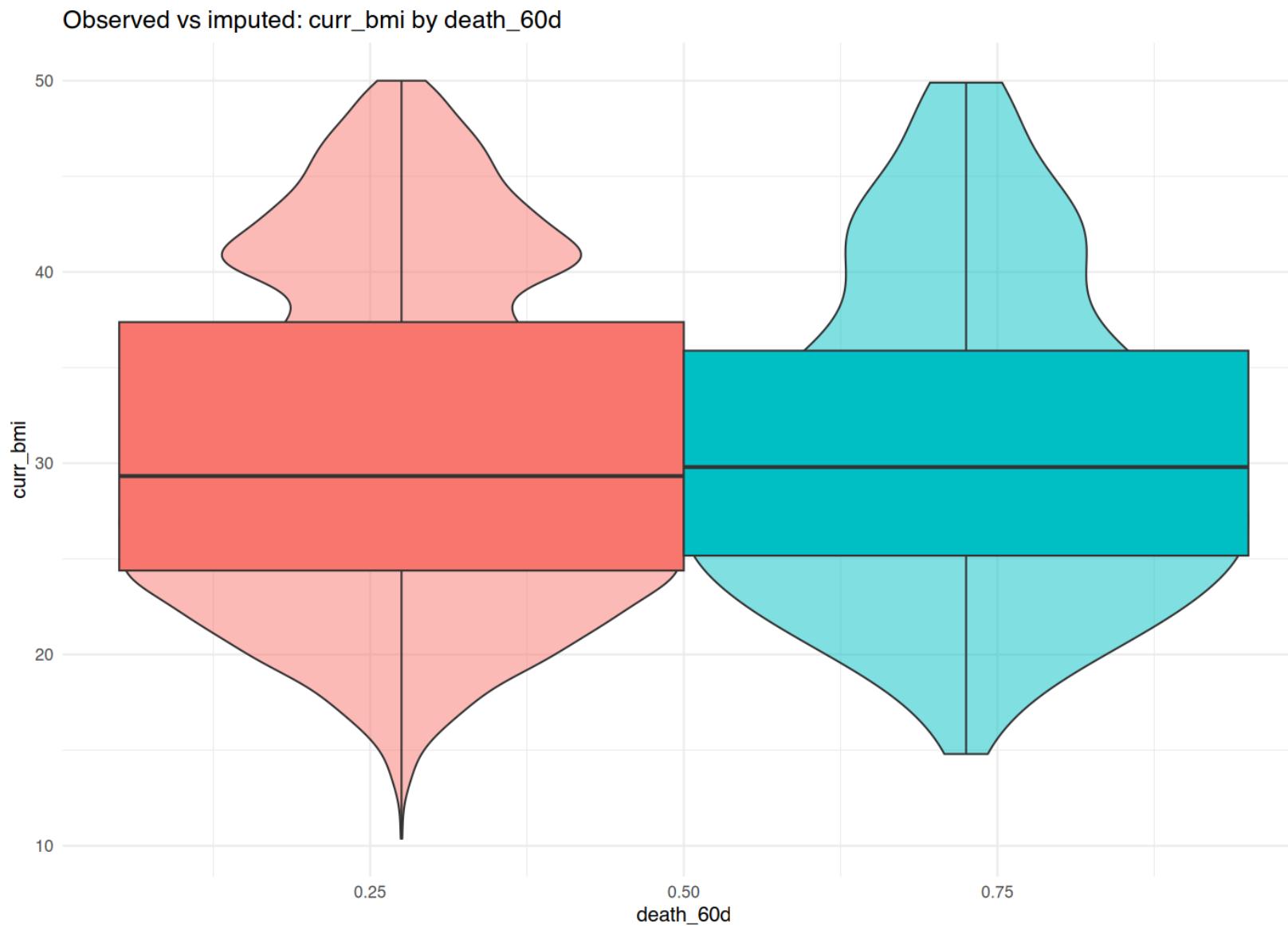
Observed vs imputed: serum_lac by death_60d



Warning: Removed 276 rows containing non-finite outside the scale range
(`stat_ydensity()`).

Warning: Removed 276 rows containing non-finite outside the scale range

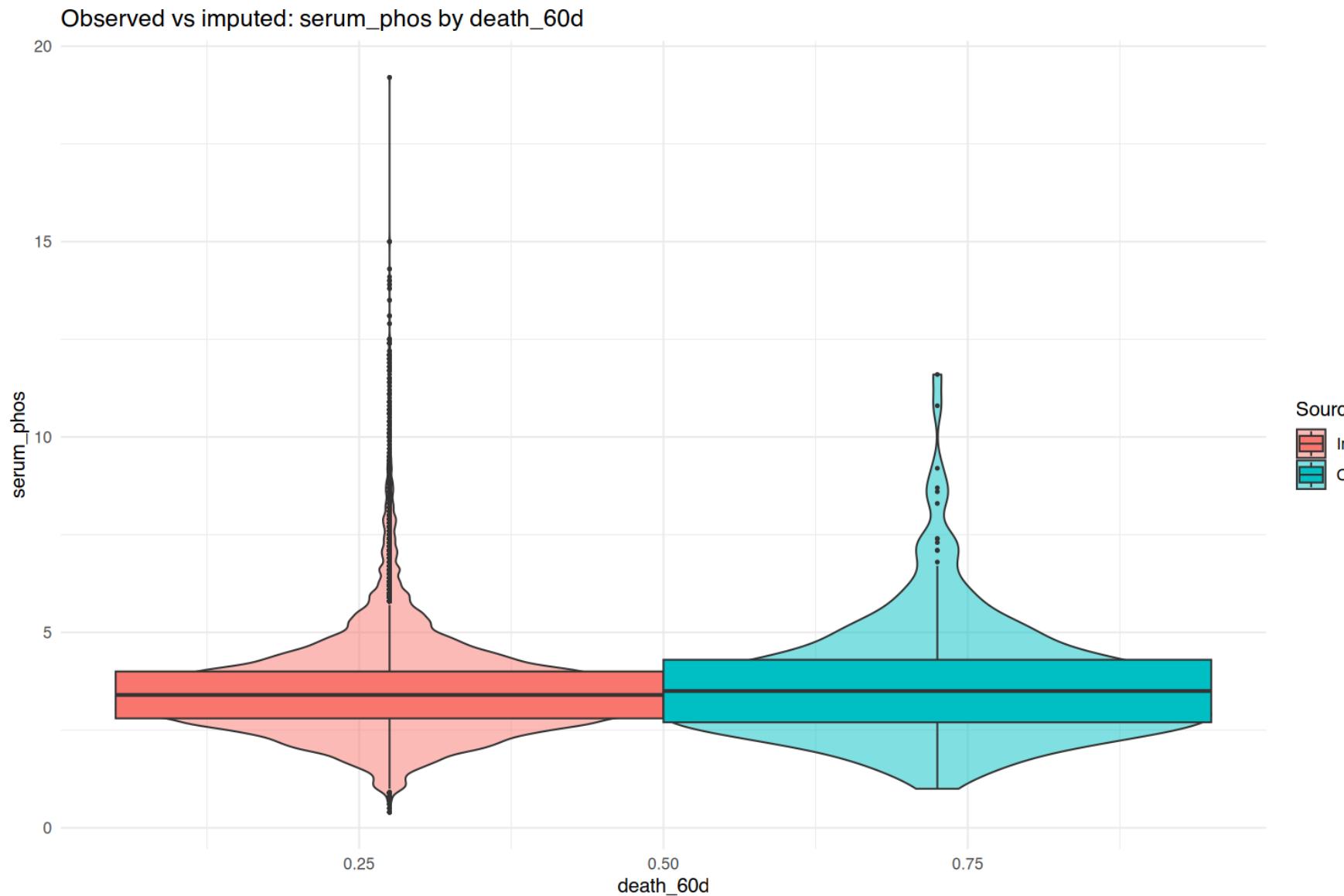
```
(`stat_boxplot()`).
```



Warning: Removed 245 rows containing non-finite outside the scale range

(`stat_ydensity()`).

Warning: Removed 245 rows containing non-finite outside the scale range
(`stat_boxplot()`).



Chunk mi-imputation-diagnostics runtime: 20.47 s

4.4 10) Refit propensity models within each imputation

GBM recipe matches the non-MI run (shared `gbm_params`, balance-based stopping).

4.4.1 FAIL-FAST CHECKS

Chunk fail-fast-checks runtime: 13.12 s

4.4.2 10.1 ABG propensity (`has_abg`)

```
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)

# Fit ABG propensity weights in each imputation
fit_abg_one <- function(d) {
  stopifnot(gbm_params$stop.method == "smd.max")
  d <- normalize_types(d, levels_ref)
  assert_no_na_covars(d, covars_gbm, context = "ABG WeightIt (MI)")
  w <- weightit(
    formula_abg,
    data    = d[, c("has_abg", covars_gbm)],
    method  = "gbm",
    estimand = "ATE",
    include.obj = FALSE,
    n.trees        = gbm_params$n.trees,
    interaction.depth = gbm_params$interaction.depth,
    shrinkage       = gbm_params$shrinkage,
    bag.fraction    = gbm_params$bag.fraction,
    cv.folds         = gbm_params$cv.folds,
    stop.method      = gbm_params$stop.method,
    n.cores          = gbm_params$n.cores
  )
  ipow <- compute_ipow_weights(
```

```

w,
treat = d$has_abg,
ps_floor_quantile = ps_trunc_quantile,
stabilize = TRUE
)
assert_finite_weights(ipow$weights[d$has_abg == 1], "w_abg")
w$weights <- ipow$weights
w$ipow_info <- list(
  ps_floor = ipow$ps_floor,
  cap      = ipow$cap,
  trunc_rate = mean(ipow$truncated, na.rm = TRUE)
)
w
}

imp_size <- utils::object.size(imp$data)
message("imp$data size (bytes): ", format(imp_size, units = "auto"))

```

imp\$data size (bytes): 5.8 Mb

```

if (imp_size > 4e8) {
  message("Large imp$data; switching to sequential futures to avoid memory pressure.")
  future::plan(sequential)
} else {
  options(future.globals.maxSize = max(8e9, imp_size * 2))
}

W_abg_list <- runtime_logger(
  "mi_weight_abg",
  {
    with_progress({
      p <- progressor(along = seq_len(imp_n))
      future_lapply(
        X = seq_len(imp_n),
        FUN = function(i) {

```

```

    p(sprintf("Fitting ABG on imputation %d", i))
    set.seed(20251206 + i)           # per-imputation seed for reproducibility
    fit_abg_one(get_imp(i))
  },
  future.seed = TRUE               # reproducible RNG across workers
)
}
},
notes = paste0("m=", imp_n)
)

saveRDS(W_abg_list, mi_w_abg_file)

```

Chunk *mi-propensity-abg* runtime: 2298.64 s

	n	min	p99.99%	max	ess
[1,]	9383	0.435	3.502	3.502	7283.295
[2,]	9383	0.441	3.592	3.594	7319.678
[3,]	9383	0.446	3.411	3.416	7335.708
[4,]	9383	0.436	3.607	3.607	7232.413
[5,]	9383	0.439	3.560	3.564	7277.458
[6,]	9383	0.434	3.472	3.473	7269.294
[7,]	9383	0.442	3.529	3.531	7311.400
[8,]	9383	0.443	3.489	3.491	7375.207
[9,]	9383	0.439	3.534	3.534	7319.441
[10,]	9383	0.437	3.527	3.528	7293.432
[11,]	9383	0.442	3.631	3.632	7235.230
[12,]	9383	0.434	3.555	3.555	7285.191
[13,]	9383	0.435	3.505	3.505	7331.890
[14,]	9383	0.438	3.656	3.656	7203.257
[15,]	9383	0.433	3.598	3.598	7252.950
[16,]	9383	0.441	3.484	3.485	7310.786
[17,]	9383	0.428	3.552	3.554	7235.379
[18,]	9383	0.441	3.491	3.491	7302.538
[19,]	9383	0.438	3.508	3.509	7313.775

[20,]	9383	0.433	3.580	3.581	7273.928
[21,]	9383	0.437	3.576	3.576	7286.771
[22,]	9383	0.436	3.527	3.528	7319.183
[23,]	9383	0.430	3.420	3.420	7363.999
[24,]	9383	0.441	3.601	3.603	7300.832
[25,]	9383	0.436	3.532	3.537	7316.187
[26,]	9383	0.442	3.539	3.544	7339.014
[27,]	9383	0.433	3.502	3.503	7329.765
[28,]	9383	0.437	3.559	3.560	7258.800
[29,]	9383	0.429	3.576	3.578	7316.048
[30,]	9383	0.439	3.547	3.548	7310.895
[31,]	9383	0.436	3.501	3.502	7330.787
[32,]	9383	0.432	3.497	3.497	7303.682
[33,]	9383	0.433	3.588	3.592	7270.843
[34,]	9383	0.448	3.519	3.520	7360.520
[35,]	9383	0.444	3.590	3.590	7261.493
[36,]	9383	0.442	3.530	3.532	7378.776
[37,]	9383	0.438	3.541	3.542	7305.847
[38,]	9383	0.441	3.581	3.583	7293.418
[39,]	9383	0.439	3.619	3.620	7261.002
[40,]	9383	0.436	3.468	3.469	7318.859
[41,]	9383	0.440	3.545	3.548	7312.841
[42,]	9383	0.439	3.512	3.513	7335.967
[43,]	9383	0.440	3.444	3.444	7339.476
[44,]	9383	0.431	3.573	3.573	7276.767
[45,]	9383	0.439	3.430	3.431	7393.351
[46,]	9383	0.436	3.577	3.577	7278.690
[47,]	9383	0.441	3.684	3.684	7245.857
[48,]	9383	0.443	3.513	3.518	7297.805
[49,]	9383	0.437	3.615	3.616	7255.737
[50,]	9383	0.433	3.557	3.560	7262.731
[51,]	9383	0.438	3.475	3.475	7328.915
[52,]	9383	0.448	3.466	3.467	7346.556
[53,]	9383	0.438	3.464	3.464	7346.756
[54,]	9383	0.432	3.527	3.527	7258.673
[55,]	9383	0.433	3.590	3.590	7244.224

```
[56,] 9383 0.441  3.579 3.581 7302.181
[57,] 9383 0.432  3.662 3.662 7262.370
[58,] 9383 0.437  3.445 3.449 7327.199
[59,] 9383 0.432  3.511 3.512 7300.004
[60,] 9383 0.440  3.616 3.616 7221.657
[61,] 9383 0.446  3.410 3.412 7381.299
[62,] 9383 0.437  3.488 3.488 7302.232
[63,] 9383 0.436  3.510 3.510 7299.038
[64,] 9383 0.435  3.492 3.493 7303.627
[65,] 9383 0.435  3.553 3.555 7256.098
[66,] 9383 0.435  3.627 3.629 7239.454
[67,] 9383 0.436  3.573 3.574 7241.593
[68,] 9383 0.440  3.624 3.625 7264.935
[69,] 9383 0.437  3.513 3.513 7320.392
[70,] 9383 0.440  3.504 3.504 7326.019
[71,] 9383 0.433  3.542 3.543 7317.388
[72,] 9383 0.436  3.545 3.547 7293.962
[73,] 9383 0.429  3.629 3.631 7241.429
[74,] 9383 0.436  3.543 3.543 7296.594
[75,] 9383 0.439  3.580 3.580 7278.415
[76,] 9383 0.434  3.491 3.492 7303.227
[77,] 9383 0.442  3.561 3.561 7285.834
[78,] 9383 0.432  3.581 3.581 7264.942
[79,] 9383 0.440  3.548 3.550 7288.726
[80,] 9383 0.431  3.579 3.579 7274.014
```

Chunk mi-weight-diagnostics-abg runtime: 12.21 s

4.4.3 10.2 Balance diagnostics across imputations

```
# Vars you intended to use (from your earlier code)
vars0 <- covars_gbm
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
```

```

if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)

# Which factors collapse to 1 level AFTER complete-case filtering (per imputation, per arm)?
find_offenders_post_cc <- function(d, treat_var, vars) {
  keep <- c(treat_var, vars)
  dd   <- d[, keep, drop = FALSE]
  dd   <- dd[stats::complete.cases(dd), , drop = FALSE] # mimic cobalt's CC
  if (!nrow(dd)) return(character(0))

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

  names(bad)[bad]
}

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

```

Offenders (post CC): <none>

```

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

```

Chunk unnamed-chunk-1 runtime: 13.47 s

```

# Build a variable set that has 2 levels in *every* imputation (prevents contrasts errors)
vary_ok <- function(z) {
  nz <- z[!is.na(z)]
  if (is.factor(nz)) nlevels(droplevels(nz)) > 1 else dplyr::n_distinct(nz) > 1
}

vars_keep <- Reduce(intersect, lapply(seq_len(imp_n), function(i) {
  d <- get_imp(i)
  keep <- vapply(d[, covars_gbm, drop = FALSE], vary_ok, logical(1))
  names(keep)[keep]
}))

# Long data for cobalt with weights and imputation id
make_long_for_cobalt <- function(imp_n, get_imp, W_list, treat_var, covars) {
  stopifnot(length(W_list) == imp_n)
  do.call(rbind, lapply(seq_len(imp_n), function(i) {
    di <- get_imp(i)[, c(treat_var, covars), drop = FALSE]
    di$.imp <- i
    di$.w <- W_list[[i]]$weights
    di$.w[di[[treat_var]] == 0] <- 1
    di
  }))
}
dlong_abg <- make_long_for_cobalt(imp_n, get_imp, W_abg_list, "has_abg", vars_keep)

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

# Fail fast if any covariate remains missing
assert_no_na_covars(dlong_abg, vars_keep, context = "ABG balance (MI)")

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

```

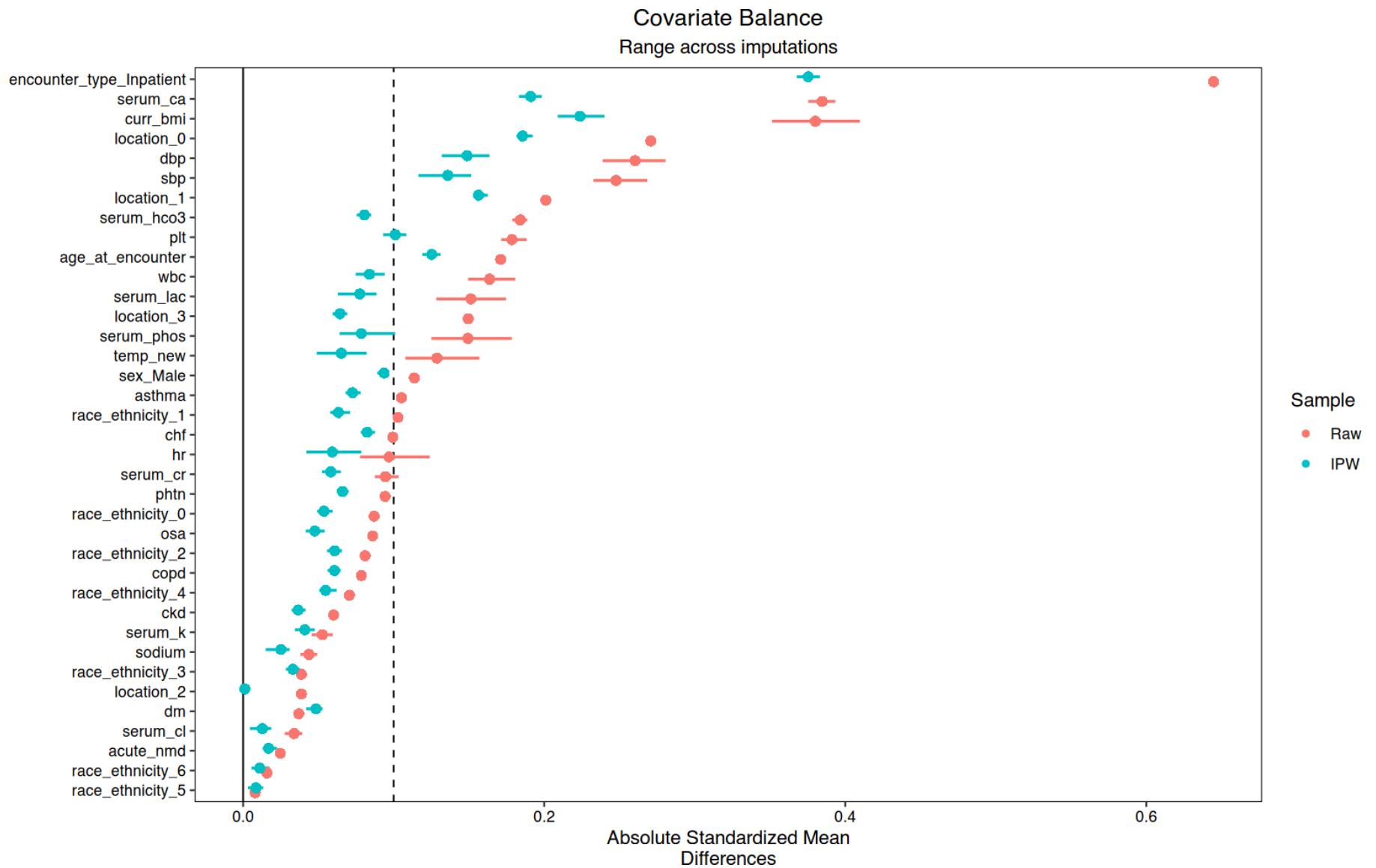
```

}

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

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

```



Chunk mi-balance-abg runtime: 55.38 s

4.4.4 10.3 VBG propensity (has_vbg)

```
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)

fit_vbg_one <- function(d) {
  stopifnot(gbm_params$stop.method == "smd.max")
  d <- normalize_types(d, levels_ref)
  assert_no_na_covars(d, covars_gbm, context = "VBG WeightIt (MI)")
  w <- weightit(
    formula_vbg,
    data   = d[, c("has_vbg", covars_gbm)],
    method = "gbm",
    estimand      = "ATE",
    include.obj = FALSE,
    n.trees       = gbm_params$n.trees,
    interaction.depth = gbm_params$interaction.depth,
    shrinkage     = gbm_params$shrinkage,
    bag.fraction  = gbm_params$bag.fraction,
    cv.folds      = gbm_params$cv.folds,
    stop.method    = gbm_params$stop.method,
    n.cores        = gbm_params$n.cores
  )
  ipow <- compute_ipow_weights(
    w,
    treat = d$has_vbg,
    ps_floor_quantile = ps_trunc_quantile,
    stabilize = TRUE
  )
  assert_finite_weights(ipow$weights[d$has_vbg == 1], "w_vbg")
  w$weights <- ipow$weights
  w$ipow_info <- list(
    ps_floor = ipow$ps_floor,
    cap      = ipow$cap,
```

```

    trunc_rate = mean(ipow$truncated, na.rm = TRUE)
  )
  w
}

imp_size <- utils::object.size(imp$data)
message("imp$data size (bytes): ", format(imp_size, units = "auto"))

```

imp\$data size (bytes): 5.8 Mb

```

if (imp_size > 4e8) {
  message("Large imp$data; switching to sequential futures to avoid memory pressure.")
  future::plan(sequential)
} else {
  options(future.globals.maxSize = max(8e9, imp_size * 2))
}

W_vbg_list <- runtime_logger(
  "mi_weight_vbg",
  {
    with_progress({
      p <- progressor(along = seq_len(imp_n))
      future_lapply(
        X = seq_len(imp_n),
        FUN = function(i) {
          p(sprintf("Fitting VBG on imputation %d", i))
          set.seed(30251206 + i)
          fit_vbg_one(get_imp(i))
        },
        future.seed = TRUE
      )
    })
  },
  notes = paste0("m=", imp_n)
)

```

```
saveRDS(W_vbg_list, mi_w_vbg_file)
```

Chunk mi-propensity-vbg runtime: 2289.60 s

	n	min	p99.99%	max	ess
[1,]	7382	0.340	3.336	3.336	5425.509
[2,]	7382	0.345	3.131	3.132	5492.014
[3,]	7382	0.341	3.237	3.239	5448.339
[4,]	7382	0.339	3.171	3.172	5489.566
[5,]	7382	0.342	3.097	3.100	5510.581
[6,]	7382	0.335	3.075	3.076	5557.593
[7,]	7382	0.337	3.193	3.194	5467.227
[8,]	7382	0.343	3.172	3.173	5512.546
[9,]	7382	0.343	3.194	3.194	5500.075
[10,]	7382	0.337	3.138	3.139	5520.540
[11,]	7382	0.340	3.216	3.217	5502.355
[12,]	7382	0.341	3.109	3.109	5481.081
[13,]	7382	0.342	3.147	3.148	5485.374
[14,]	7382	0.346	3.027	3.028	5540.302
[15,]	7382	0.339	3.085	3.086	5530.594
[16,]	7382	0.345	3.148	3.148	5518.761
[17,]	7382	0.341	3.145	3.146	5479.508
[18,]	7382	0.342	3.169	3.169	5513.643
[19,]	7382	0.340	3.097	3.097	5519.700
[20,]	7382	0.340	3.236	3.237	5467.045
[21,]	7382	0.343	3.126	3.126	5508.413
[22,]	7382	0.337	3.114	3.116	5502.812
[23,]	7382	0.338	3.149	3.149	5477.077
[24,]	7382	0.343	3.205	3.206	5459.859
[25,]	7382	0.342	3.161	3.161	5495.114
[26,]	7382	0.345	3.278	3.278	5478.543
[27,]	7382	0.343	3.016	3.016	5547.913
[28,]	7382	0.342	3.143	3.145	5495.371
[29,]	7382	0.342	3.119	3.120	5532.211
[30,]	7382	0.339	3.121	3.121	5515.132

[31,]	7382	0.341	3.342	3.343	5424.981
[32,]	7382	0.337	2.982	2.983	5568.680
[33,]	7382	0.340	3.151	3.151	5498.577
[34,]	7382	0.341	3.189	3.189	5453.290
[35,]	7382	0.341	3.130	3.130	5497.220
[36,]	7382	0.338	3.229	3.230	5488.474
[37,]	7382	0.345	3.095	3.096	5513.266
[38,]	7382	0.341	3.180	3.182	5475.572
[39,]	7382	0.344	3.167	3.168	5490.957
[40,]	7382	0.337	3.253	3.255	5426.224
[41,]	7382	0.339	3.163	3.164	5480.086
[42,]	7382	0.343	3.250	3.253	5477.702
[43,]	7382	0.341	3.153	3.155	5480.391
[44,]	7382	0.339	3.102	3.105	5489.054
[45,]	7382	0.336	3.139	3.141	5501.589
[46,]	7382	0.339	3.289	3.289	5436.395
[47,]	7382	0.339	3.203	3.203	5499.140
[48,]	7382	0.339	3.121	3.126	5496.643
[49,]	7382	0.341	3.207	3.207	5467.616
[50,]	7382	0.339	3.134	3.135	5516.394
[51,]	7382	0.340	3.102	3.102	5543.553
[52,]	7382	0.337	3.109	3.110	5496.464
[53,]	7382	0.340	3.317	3.317	5419.556
[54,]	7382	0.343	3.159	3.159	5496.759
[55,]	7382	0.341	3.117	3.117	5503.331
[56,]	7382	0.338	3.275	3.276	5445.136
[57,]	7382	0.341	3.135	3.136	5514.246
[58,]	7382	0.339	3.025	3.025	5558.910
[59,]	7382	0.335	3.089	3.091	5505.873
[60,]	7382	0.346	3.005	3.007	5559.061
[61,]	7382	0.339	3.079	3.080	5514.907
[62,]	7382	0.345	3.212	3.212	5463.967
[63,]	7382	0.339	3.084	3.086	5503.498
[64,]	7382	0.338	3.083	3.084	5535.122
[65,]	7382	0.348	3.314	3.314	5433.386
[66,]	7382	0.337	3.123	3.125	5540.048

```
[67,] 7382 0.337 3.258 3.262 5450.410
[68,] 7382 0.338 3.114 3.115 5503.915
[69,] 7382 0.336 3.053 3.054 5534.963
[70,] 7382 0.340 3.188 3.188 5474.863
[71,] 7382 0.336 3.108 3.109 5521.299
[72,] 7382 0.331 3.097 3.097 5506.504
[73,] 7382 0.334 3.055 3.055 5528.957
[74,] 7382 0.329 3.149 3.153 5468.109
[75,] 7382 0.336 3.098 3.100 5523.354
[76,] 7382 0.341 3.121 3.121 5520.079
[77,] 7382 0.338 2.998 2.998 5545.518
[78,] 7382 0.343 3.127 3.130 5507.203
[79,] 7382 0.337 3.204 3.207 5457.686
[80,] 7382 0.342 3.200 3.200 5491.311
```

Chunk mi-weight-diagnostics-vbg runtime: 12.30 s

4.4.5 10.4 VBG balance

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

if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)
stopifnot(length(W_vbg_list) == imp_n)

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

# 1) Keep only covariates that vary (2 levels for factors) in every imputation
vars_keep_vbg <- Reduce(intersect, lapply(seq_len(imp_n), function(i) {
  d <- get_imp(i)
```

```

keep <- vapply(d[, covars_gbm, drop = FALSE], vary_ok, logical(1))
names(keep)[keep]
}))

# 2) Build per-imputation lists; align weights and drop non-finite rows
X_list_vbg <- vector("list", imp_n)
t_list_vbg <- vector("list", imp_n)
w_list_vbg <- vector("list", imp_n)
for (i in seq_len(imp_n)) {
  d_imp <- get_imp(i)
  di <- d_imp[, vars_keep_vbg, drop = FALSE]
  assert_no_na_covars(di, vars_keep_vbg, context = paste0("VBG balance (MI) imp ", i))
  ids <- rownames(di)
  ti <- d_imp[["has_vbg"]]
  wi <- W_vbg_list[[i]]$weights
  if (!is.null(names(wi))) {
    wi <- wi[ids]
  } else if (length(wi) != length(ids)) {
    stop("Length mismatch weights vs data in VBG imp ", i)
  }
  wi[ti == 0] <- 1
  keep <- is.finite(wi)
  if (!all(keep)) {
    di <- di[keep, , drop = FALSE]
    ti <- ti[keep]
    wi <- wi[keep]
  }
  X_list_vbg[[i]] <- di
  t_list_vbg[[i]] <- ti
  w_list_vbg[[i]] <- wi
}

# 3) Per-imputation SMDs (mean diffs only; quick path)
one_imp_bal_vbg <- function(i) {
  b <- cobalt::bal.tab(
    x           = X_list_vbg[[i]],

```

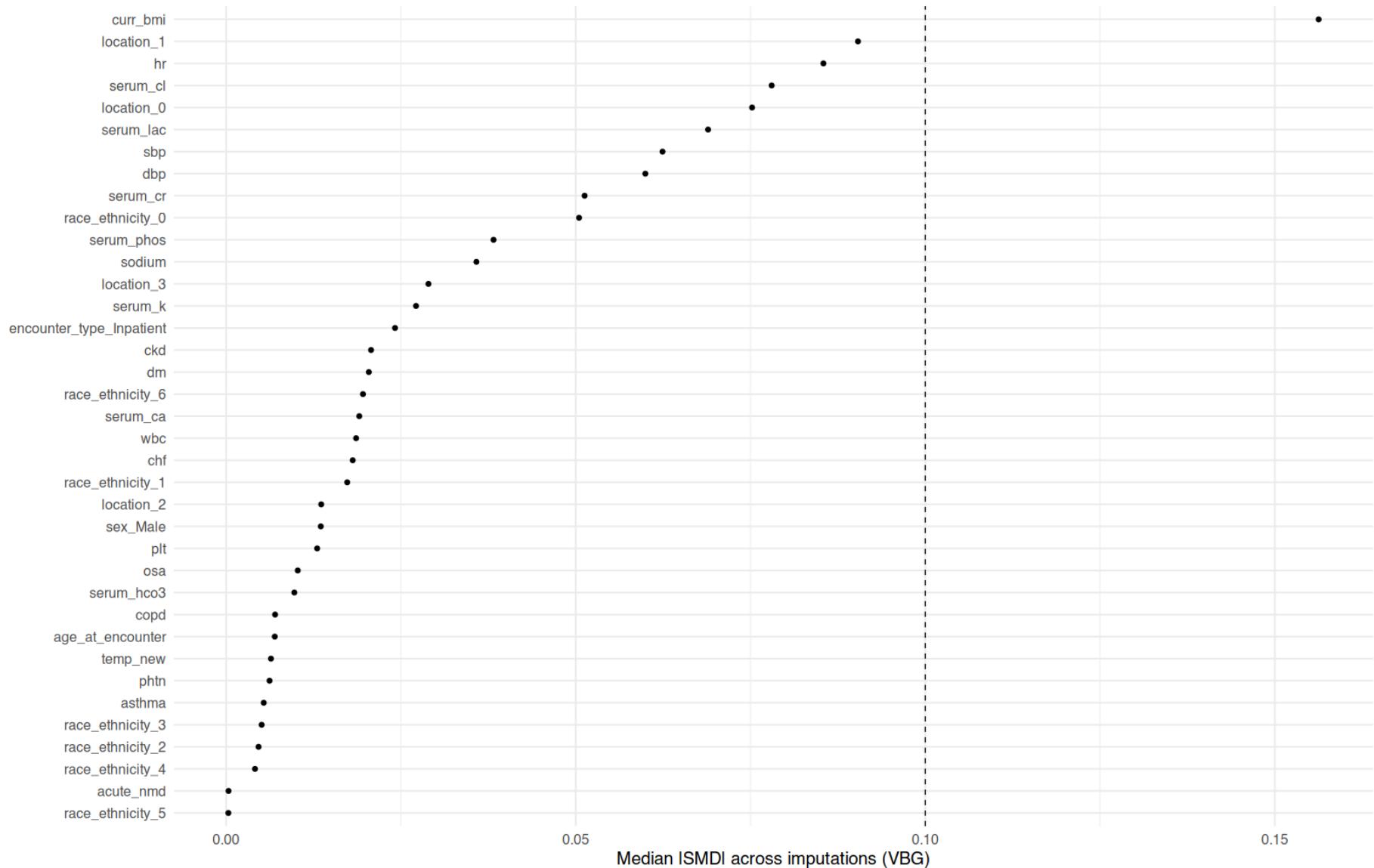
```

treat      = t_list_vbg[[i]],
weights    = w_list_vbg[[i]],
stats      = "m",           # mean diffs only
s.d.denom  = "pooled",
quick      = TRUE,
int        = FALSE,
disp.v.ratio = FALSE,
disp.ks     = FALSE,
un         = TRUE
)
S <- as.data.frame(b$Balance)
setNames(S[["Diff.Adj"]], rownames(S))
}

smd_list_vbg <- lapply(seq_len(imp_n), one_imp_bal_vbg)
all_rows_vbg <- Reduce(union, lapply(smd_list_vbg, names))
SMD_mat_vbg  <- do.call(cbind, lapply(smd_list_vbg, function(v) v[match(all_rows_vbg, names(v))]))
smd_abs_vbg  <- abs(SMD_mat_vbg)
smd_pooled_vbg <- data.frame(
  covariate = all_rows_vbg,
  smd_med   = apply(smd_abs_vbg, 1, median, na.rm = TRUE),
  smd_mean  = rowMeans(smd_abs_vbg, na.rm = TRUE),
  smd_max   = apply(smd_abs_vbg, 1, max,    na.rm = TRUE),
  stringsAsFactors = FALSE
)

# Optional quick plot of pooled median |SMD|
ggplot(smd_pooled_vbg, aes(x = reorder(covariate, smd_med), y = smd_med)) +
  geom_hline(yintercept = 0.1, linetype = 2, linewidth = 0.3) +
  geom_point(size = 1) +
  coord_flip() +
  labs(x = NULL, y = "Median |SMD| across imputations (VBG)") +
  theme_minimal(base_size = 10)

```



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

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

Chunk mi-balance-vbg runtime: 50.00 s

4.5 11) Weighted outcome models within each imputation + pooling

Within each imputation, fit CO2 spline outcome models **only in the measured cohort** (has_abg==1 for PaCO2; has_vbg==1 for VBG CO2), using IPSW weights to standardize the treated cohort to the full analytic sample. Pool the spline curves pointwise across imputations (Rubin's rules on the link scale), then transform to probability scale for plotting and inference.

4.5.1 11.1 Helper: fit + extract log-OR and SE from svyglm

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

# --- Fit one weighted GLM on one completed dataset and extract log-OR + var ---
fit_and_extract <- function(data, weights, formula, treat_name) {
  stopifnot(length(weights) == nrow(data))
  yname <- all.vars(formula)[1L]

  # keep only rows with finite weights and non-missing outcome/treatment
  data[[yname]] <- to01(data[[yname]])
  data[[treat_name]] <- to01(data[[treat_name]])
```

```

data$w <- as.numeric(weights)
keep <- is.finite(data$w) & !is.na(data[[treat_name]]) & !is.na(data[[yname]])
data <- data[keep, , drop = FALSE]
if (nrow(data) == 0L) stop("No rows with finite weights/outcome/treatment.")

# basic guards: variation in outcome and treatment (after filtering)
if (dplyr::n_distinct(data[[yname]]) < 2L) stop("Outcome '", yname, "' has one level.")
if (dplyr::n_distinct(data[[treat_name]]) < 2L) stop("Treatment '", treat_name, "' has one level.")

# design + fit
des <- survey::svydesign(ids = ~1, weights = ~w, data = data)
fit <- survey::svyglm(formula, design = des, family = quasibinomial())
if (!inherits(fit, "svyglm")) stop("Expected svyglm fit for robust variance pooling.")
cv <- coef_var_from_svy(fit, treat_name)
list(coef = cv$b, vcov = cv$V)
}

# --- Pool across imputations; tolerate failures; report how many used -----
pool_logOR <- function(est_list, term) {
  ok <- vapply(est_list, function(x) is.list(x) && is.finite(x$coef) && is.finite(x$vcov), logical(1))
  est_list <- est_list[ok]
  m_ok <- length(est_list); m_tot <- length(ok)
  if (m_ok == 0L) {
    return(data.frame(term = term, logOR = NA_real_, SE = NA_real_, OR = NA_real_,
                      LCL = NA_real_, UCL = NA_real_, m_used = 0L, m_total = m_tot))
  }
  results <- lapply(est_list, function(x) setNames(c(x$coef), term))
  variances <- lapply(est_list, function(x) { M <- matrix(x$vcov, 1, 1); dimnames(M) <- list(term, term); M })
  pooled <- mitools::MIcombine(results = results, variances = variances)
  coefs <- vapply(est_list, function(x) x$coef, numeric(1))
  within <- mean(vapply(est_list, function(x) x$vcov, numeric(1)))
  between <- stats::var(coefs)
  attr(pooled, "within") <- within
  attr(pooled, "between") <- between
  pooled_mi_vcov_check(pooled)
  est <- as.numeric(coef(pooled))
}

```

```

se <- sqrt(diag(pooled$variance))
data.frame(term = term, logOR = est, SE = se, OR = exp(est),
           LCL = exp(est - 1.96 * se), UCL = exp(est + 1.96 * se),
           m_used = m_ok, m_total = m_tot, row.names = NULL)
}

# --- Fit spline model on treated cohort only (IPSW) -----
fit_spline_imp <- function(data, weights, outcome, co2_var, treat_var,
                           spline_df = 4,
                           spline_basis = c("ns", "rcs"),
                           grid_df = NULL) {
  spline_basis <- match.arg(spline_basis)
  stopifnot(is.data.frame(data))
  stopifnot(length(weights) == nrow(data))
  stopifnot(all(c(outcome, co2_var, treat_var) %in% names(data)))

  data[[treat_var]] <- as.integer(data[[treat_var]] == 1L)
  data[[outcome]] <- to01(data[[outcome]])
  data[[co2_var]] <- suppressWarnings(as.numeric(data[[co2_var]]))
  w <- suppressWarnings(as.numeric(weights))

  keep <- (data[[treat_var]] == 1L) &
    is.finite(data[[co2_var]]) &
    !is.na(data[[outcome]]) &
    is.finite(w) & (w > 0)

  if (!any(keep)) {
    return(list(error = "No eligible treated rows after filtering (treated==1, finite CO2, non-missing outcome, finite positive weight)"))
  }

  d2 <- data[keep, , drop = FALSE]
  w2 <- w[keep]
  d2$w <- w2

  if (length(unique(d2[[outcome]])) < 2L) {
    return(list(error = paste0("Outcome has one level in treated sample after filtering: outcome=", unique(d2[[outcome]]))))
  }
}

```

```

        outcome, ", n=", nrow(d2),
        ", events=", sum(d2[[outcome]] == 1, na.rm = TRUE))))
}

des <- survey::svydesign(ids = ~1, weights = ~w, data = d2)
if (spline_basis == "ns") {
  fml <- stats::as.formula(sprintf("%s ~ splines::ns(%s, %d)", outcome, co2_var, spline_df))
} else {
  if (!requireNamespace("rms", quietly = TRUE)) {
    return(list(error = "Package 'rms' is required for rcs()."))
  }
  dd <- rms::datadist(d2)
  old_opt <- options(datadist = ".__dd_tmp__")
  assign(".__dd_tmp__", dd, envir = .GlobalEnv)
  on.exit({
    options(old_opt)
    rm(list = ".__dd_tmp__", envir = .GlobalEnv)
  }, add = TRUE)
  fml <- stats::as.formula(sprintf("%s ~ rms::rcs(%s, %d)", outcome, co2_var, spline_df))
}

fit <- tryCatch(
  survey::svyglm(fml, design = des, family = quasibinomial(),
  error = function(e) e
)
if (inherits(fit, "error")) return(list(error = paste0("svyglm error: ", conditionMessage(fit)))

if (!is.null(grid_df)) {
  pr <- tryCatch({
    mm <- stats::model.matrix(stats::delete.response(stats::terms(fit)), grid_df)
    beta <- stats::coef(fit)
    V     <- stats::vcov(fit)
    eta   <- as.numeric(mm %*% beta)
    var_eta <- rowSums((mm %*% V) * mm)
    list(eta = eta, var_eta = var_eta)
  }, error = function(e) list(error = paste0("predict error: ", conditionMessage(e))))
}

```

```

if (!is.null(pr$error)) return(list(error = pr$error))
return(list(fit = fit, eta = pr$eta, var_eta = pr$var_eta,
           n = nrow(d2), events = sum(d2[[outcome]] == 1, na.rm = TRUE)))
}

list(fit = fit, n = nrow(d2), events = sum(d2[[outcome]] == 1, na.rm = TRUE))
}

# --- Extract svyglm from wrapper or bare object -----
extract_svyglm <- function(x) {
  if (inherits(x, "svyglm")) return(x)
  if (is.list(x) && !is.null(x$fit) && inherits(x$fit, "svyglm")) return(x$fit)
  NULL
}

extract_error <- function(x) {
  if (is.list(x) && !is.null(x$error)) return(as.character(x$error))
  paste("class:", paste(class(x), collapse = "/"))
}

# --- Summarize per-imputation errors -----
summarize_fit_errors <- function(fit_list, n_grid = NULL) {
  ok <- vapply(fit_list, function(x) {
    if (!is.list(x) || !is.null(x$error)) return(FALSE)
    if (!is.null(n_grid)) {
      return(!is.null(x$eta) && !is.null(x$var_eta) &&
             length(x$eta) == n_grid && length(x$var_eta) == n_grid)
    }
    TRUE
  }, logical(1))

  err <- vapply(fit_list, function(x) {
    if (is.list(x) && !is.null(x$error)) return(as.character(x$error))
    if (!is.null(n_grid) && is.list(x) && is.null(x$error)) return("Missing eta/var_eta")
    paste("class:", paste(class(x), collapse = "/"))
  }, character(1))
}

```

```

err[ok] <- NA_character_

df <- data.frame(
  imputation = seq_along(fit_list),
  ok = ok,
  error = err,
  stringsAsFactors = FALSE
)

err_counts <- sort(table(df$error, useNA = "no"), decreasing = TRUE)
if (length(err_counts)) {
  print(utils::head(err_counts, 10))
  top_errs <- names(err_counts)[seq_len(min(10, length(err_counts)))]
  idx_by_err <- lapply(top_errs, function(msg) utils::head(df$imputation[df$error == msg], 5))
  names(idx_by_err) <- top_errs
  print(idx_by_err)
} else {
  message("No errors to summarize.")
}
df
}

# --- Predict on link scale with SE; fallback to model.matrix if needed -----
predict_link_svyglm <- function(fit, newdata) {
  pr <- try(predict(fit, newdata = newdata, type = "link", se.fit = TRUE), silent = TRUE)
  if (!inherits(pr, "try-error") && !is.null(pr$fit) && !is.null(pr$se.fit)) {
    return(list(fit = as.numeric(pr$fit), se.fit = as.numeric(pr$se.fit)))
  }
  mm <- model.matrix(stats::delete.response(terms(fit)), newdata)
  eta <- as.numeric(mm %*% coef(fit))
  se <- sqrt(rowSums((mm %*% vcov(fit)) * mm))
  list(fit = eta, se.fit = se)
}

# --- Pool spline coefficients across imputations -----
pool_spline_coefs <- function(fit_list, min_ok_frac = 0.9) {

```

```

fits <- lapply(fit_list, extract_svyglm)
ok <- !vapply(fits, is.null, logical(1))
m_tot <- length(ok)
m_ok <- sum(ok)
if (m_ok < ceiling(min_ok_frac * m_tot)) {
  summarize_fit_errors(fit_list)
  stop("Too many failed spline fits: m_ok=", m_ok, " / m_total=", m_tot)
}
fits <- fits[ok]
results <- lapply(fits, coef)
variances <- lapply(fits, vcov)
pooled <- mitools::MIcombine(results = results, variances = variances)
pooled_mi_vcov_check(pooled)
est <- as.numeric(coef(pooled))
se <- sqrt(diag(pooled$variance))
data.frame(
  term = names(coef(pooled)),
  estimate = est,
  SE = se,
  LCL = est - 1.96 * se,
  UCL = est + 1.96 * se,
  m_used = m_ok,
  m_total = m_tot,
  row.names = NULL
)
}

# --- Pool spline curves (pointwise Rubin pooling on link scale) -----
pool_spline_curve <- function(fit_list, grid_df, min_ok_frac = 0.9) {
  n_grid <- nrow(grid_df)
  ok <- vapply(fit_list, function(x) {
    is.list(x) && is.null(x$error) &&
      !is.null(x$eta) && !is.null(x$var_eta) &&
      length(x$eta) == n_grid && length(x$var_eta) == n_grid
  }, logical(1))
  m_tot <- length(ok)
}

```

```

m_ok <- sum(ok)
if (m_ok < ceiling(min_ok_frac * m_tot)) {
  summarize_fit_errors(fit_list, n_grid = n_grid)
  stop("Too many failed spline fits: m_ok=", m_ok, " / m_total=", m_tot)
}

eta_mat <- matrix(NA_real_, nrow = n_grid, ncol = m_ok)
var_mat <- matrix(NA_real_, nrow = n_grid, ncol = m_ok)
ok_idx <- which(ok)
for (i in seq_along(ok_idx)) {
  fit_i <- fit_list[[ok_idx[i]]]
  eta_mat[, i] <- fit_i$eta
  var_mat[, i] <- fit_i$var_eta
}
Qbar <- rowMeans(eta_mat, na.rm = TRUE)
Ubar <- rowMeans(var_mat, na.rm = TRUE)
B <- apply(eta_mat, 1, stats::var, na.rm = TRUE)
Tvar <- Ubar + (1 + 1 / m_ok) * B
se <- sqrt(Tvar)
data.frame(
  grid_df,
  eta = Qbar,
  SE = se,
  prob = plogis(Qbar),
  LCL = plogis(Qbar - 1.96 * se),
  UCL = plogis(Qbar + 1.96 * se),
  m_used = m_ok,
  m_total = m_tot,
  row.names = NULL
)
}

```

Chunk mi-pool-helpers runtime: 0.01 s

4.5.2 11.2 ABG: MI pooled spline models (treated cohort only)

```
library(future.apply)
library(progressr)

# Inputs assumed present: imp, W_abg_list
stopifnot(exists("imp"), exists("W_abg_list"))
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice:::complete(imp_obj, action = i), levels_ref)
stopifnot(length(W_abg_list) == imp_n)

min_ok_frac <- 0.9
weights_abg <- lapply(W_abg_list, `[[`, "weights")
w_abg_size <- utils::object.size(weights_abg)
imp_size <- utils::object.size(imp)
max_glob <- getOption("future.globals.maxSize", 5e8)
max_glob_sz <- structure(max_glob, class = "object_size")
total_sz <- w_abg_size + imp_size
use_future <- as.numeric(total_sz) <= max_glob
if (use_future) {
  old_plan <- future::plan()
  workers <- max(1L, as.integer(future::availableCores() - 1L))
  future::plan(multisession, workers = workers)
  on.exit(future::plan(old_plan), add = TRUE)
  options(future.globals.maxSize = max_glob)
  message("MI ABG outcomes: using futures (weights-only list); globals size = ",
    format(total_sz, units = "auto"),
    " (limit ", format(max_glob_sz, units = "auto"), ".)")
} else {
  message("MI ABG outcomes: sequential fits (globals size = ",
    format(total_sz, units = "auto"),
    " > limit ", format(max_glob_sz, units = "auto"), ".)")
}
```

MI ABG outcomes: using futures (weights-only list); globals size = 122 Mb (limit 7.5 Gb).

```

abg_co2_obs <- subset_data$paco2[subset_data$has_abg == 1 & !is.na(subset_data$paco2)]
if (length(abg_co2_obs) < 10) stop("ABG spline: not enough observed PaCO2 values.")
q_abg <- stats::quantile(abg_co2_obs, probs = c(0.02, 0.98), na.rm = TRUE)
grid_abg <- data.frame(paco2 = seq(q_abg[1], q_abg[2], length.out = 200))

spline_basis <- "ns"
spline_df <- 4
imp_n_use <- if (DEBUG_SPLINE) min(3L, imp_n) else imp_n
use_future <- use_future && !DEBUG_SPLINE
if (DEBUG_SPLINE) {
  message("DEBUG_SPLINE=TRUE: running sequentially on first outcome and first ", imp_n_use, " imputations.")
}

outcomes_abg <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")
if (DEBUG_SPLINE) outcomes_abg <- outcomes_abg[1]
purrr::walk(outcomes_abg, function(nm) {
  register_model_diagram(paste("MI IPW ABG:", nm, "~ ns(paco2,4)"),
                         stats::as.formula(paste0(nm, " ~ splines::ns(paco2, 4)")))
})

```

MI IPW ABG: imv_proc ~ ns(paco2,4)

paco2

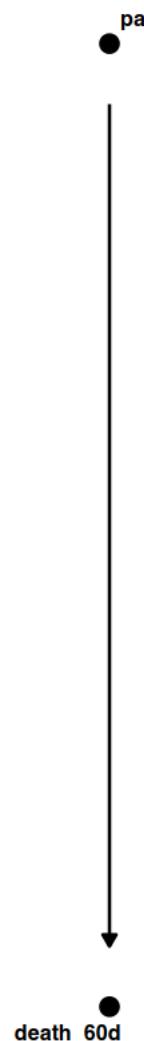


imv_proc

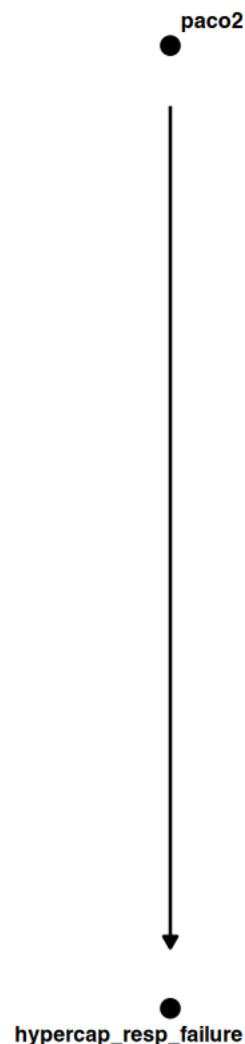
MI IPW ABG: niv_proc ~ ns(paco2,4)



MI IPW ABG: death_60d ~ ns(paco2,4)



MI IPW ABG: hypercap_resp_failure ~ ns(paco2,4)



```
abg_spline <- runtime_logger(  
  "mi_spline_abg",  
 {
```

```

res <- NULL
progressr::with_progress({
  p <- progressr::progressor(steps = length(outcomes_abg) * imp_n_use)
  res <- lapply(outcomes_abg, function(outcome) {
    fit_list <- if (use_future) {
      future.apply::future_lapply(
        X = seq_len(imp_n_use),
        FUN = function(i) {
          tryCatch({
            p(sprintf("ABG spline: %s (imp %d)", outcome, i))
            d <- get_imp(i)
            d <- d[, c(outcome, "has_abg", "paco2"), drop = FALSE]
            fit_spline_imp(
              d, weights_abg[[i]], outcome, "paco2", "has_abg",
              spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_abg
            )
          }, error = function(e) list(error = conditionMessage(e)))
        },
        future.seed = TRUE,
        future.packages = c("survey", "mice", "splines", "dplyr", if (spline_basis == "rcs") "rms")
      )
    } else {
      lapply(seq_len(imp_n_use), function(i) {
        if (DEBUG_SPLINE) {
          p(sprintf("ABG spline: %s (imp %d)", outcome, i))
          d <- get_imp(i)
          d <- d[, c(outcome, "has_abg", "paco2"), drop = FALSE]
          fit_spline_imp(
            d, weights_abg[[i]], outcome, "paco2", "has_abg",
            spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_abg
          )
        } else {
          tryCatch({
            p(sprintf("ABG spline: %s (imp %d)", outcome, i))
            d <- get_imp(i)
            d <- d[, c(outcome, "has_abg", "paco2"), drop = FALSE]
          }
        }
      })
    }
  })
}

```

```

    fit_spline_imp(
      d, weights_abg[[i]], outcome, "paco2", "has_abg",
      spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_abg
    )
  }, error = function(e) list(error = conditionMessage(e)))
}
})
}

curve <- pool_spline_curve(fit_list, grid_abg, min_ok_frac = min_ok_frac) |>
  mutate(outcome = outcome, group = "ABG")
coef_tbl <- pool_spline_coefs(fit_list, min_ok_frac = min_ok_frac) |>
  mutate(outcome = outcome, group = "ABG")
list(curve = curve, coef = coef_tbl)
})
}
list(
  curves = dplyr::bind_rows(lapply(res, `[[`, "curve")),
  coefs = dplyr::bind_rows(lapply(res, `[[`, "coef")))
)
},
notes = paste0("m=", imp_n)
)

abg_curves <- abg_spline$curves
abg_coefs <- abg_spline$coefs
utils::write.csv(abg_curves, results_path("mi_spline_curve_abg.csv"), row.names = FALSE)
utils::write.csv(abg_coefs, results_path("mi_spline_coef_abg.csv"), row.names = FALSE)

```

Chunk mi-abg-outcomes runtime: 64.37 s

4.5.3 11.3 VBG: MI pooled spline models (treated cohort only)

```

library(future.apply)
library(progressr)

# Inputs assumed present: imp, W_vbg_list
stopifnot(exists("imp"), exists("W_vbg_list"))
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)
stopifnot(length(W_vbg_list) == imp_n)

min_ok_frac <- 0.9
weights_vbg <- lapply(W_vbg_list, `[[`, "weights")
w_vbg_size <- utils::object.size(weights_vbg)
imp_size <- utils::object.size(imp)
max_glob <- getOption("future.globals.maxSize", 5e8)
max_glob_sz <- structure(max_glob, class = "object_size")
total_sz <- w_vbg_size + imp_size
use_future <- as.numeric(total_sz) <= max_glob
if (use_future) {
  old_plan <- future::plan()
  workers <- max(1L, as.integer(future::availableCores() - 1L))
  future::plan(multisession, workers = workers)
  on.exit(future::plan(old_plan), add = TRUE)
  options(future.globals.maxSize = max_glob)
  message("MI VBG outcomes: using futures (weights-only list); globals size = ",
         format(total_sz, units = "auto"),
         " (limit ", format(max_glob_sz, units = "auto"), ".)")
} else {
  message("MI VBG outcomes: sequential fits (globals size = ",
         format(total_sz, units = "auto"),
         " > limit ", format(max_glob_sz, units = "auto"), ".)")
}

```

MI VBG outcomes: using futures (weights-only list); globals size = 122 Mb (limit 7.5 Gb).

```

vbg_co2_obs <- subset_data$vbg_co2[subset_data$has_vbg == 1 & !is.na(subset_data$vbg_co2)]
if (length(vbg_co2_obs) < 10) stop("VBG spline: not enough observed VBG CO2 values.")
q_vbg <- stats::quantile(vbg_co2_obs, probs = c(0.02, 0.98), na.rm = TRUE)
grid_vbg <- data.frame(vbg_co2 = seq(q_vbg[1], q_vbg[2], length.out = 200))

spline_basis <- "ns"
spline_df <- 4
imp_n_use <- if (DEBUG_SPLINE) min(3L, imp_n) else imp_n
use_future <- use_future && !DEBUG_SPLINE
if (DEBUG_SPLINE) {
  message("DEBUG_SPLINE=TRUE: running sequentially on first outcome and first ", imp_n_use, " imputations.")
}

outcomes_vbg <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")
if (DEBUG_SPLINE) outcomes_vbg <- outcomes_vbg[1]
purrr::walk(outcomes_vbg, function(nm) {
  register_model_diagram(paste("MI IPW VBG:", nm, "~ ns(vbg_co2,4))",
                         stats::as.formula(paste0(nm, " ~ splines::ns(vbg_co2, 4)")))
})

```

MI IPW VBG: imv_proc ~ ns(vbg_co2,4)

vbg_co2



imv_proc

MI IPW VBG: niv_proc ~ ns(vbg_co2,4)

vbg_co2



niv_proc

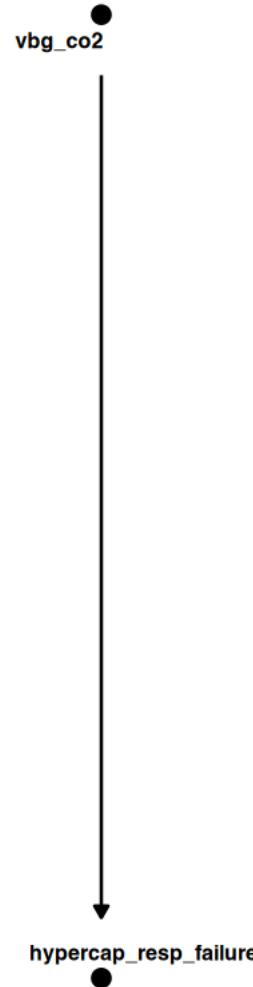
MI IPW VBG: $\text{death}_{-60d} \sim \text{ns}(\text{vbg_co2}, 4)$

vbg_co2



death_60d

MI IPW VBG: hypercap_resp_failure ~ ns(vbg_co2,4)



```
vbg_spline <- runtime_logger(  
  "mi_spline_vbg",  
  {  
    res <- NULL
```

```

progressr::with_progress({
  p <- progressr::progressor(steps = length(outcomes_vbg) * imp_n_use)
  res <- lapply(outcomes_vbg, function(outcome) {
    fit_list <- if (use_future) {
      future.apply::future_lapply(
        X = seq_len(imp_n_use),
        FUN = function(i) {
          tryCatch({
            p(sprintf("V рг spline: %s (imp %d)", outcome, i))
            d <- get_imp(i)
            d <- d[, c(outcome, "has_vbg", "vbg_co2"), drop = FALSE]
            fit_spline_imp(
              d, weights_vbg[[i]], outcome, "vbg_co2", "has_vbg",
              spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_vbg
            )
          }, error = function(e) list(error = conditionMessage(e)))
        },
        future.seed = TRUE,
        future.packages = c("survey", "mice", "splines", "dplyr", if (spline_basis == "rcs") "rms")
      )
    } else {
      lapply(seq_len(imp_n_use), function(i) {
        if (DEBUG_SPLINE) {
          p(sprintf("V рг spline: %s (imp %d)", outcome, i))
          d <- get_imp(i)
          d <- d[, c(outcome, "has_vbg", "vbg_co2"), drop = FALSE]
          fit_spline_imp(
            d, weights_vbg[[i]], outcome, "vbg_co2", "has_vbg",
            spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_vbg
          )
        } else {
          tryCatch({
            p(sprintf("V рг spline: %s (imp %d)", outcome, i))
            d <- get_imp(i)
            d <- d[, c(outcome, "has_vbg", "vbg_co2"), drop = FALSE]
            fit_spline_imp(

```

```

        d, weights_vbg[[i]], outcome, "vbg_co2", "has_vbg",
        spline_df = spline_df, spline_basis = spline_basis, grid_df = grid_vbg
    )
}, error = function(e) list(error = conditionMessage(e)))
}
})
}

curve <- pool_spline_curve(fit_list, grid_vbg, min_ok_frac = min_ok_frac) |>
  mutate(outcome = outcome, group = "VBG")
coef_tbl <- pool_spline_coefs(fit_list, min_ok_frac = min_ok_frac) |>
  mutate(outcome = outcome, group = "VBG")
list(curve = curve, coef = coef_tbl)
})
})
list(
  curves = dplyr::bind_rows(lapply(res, `[[`, "curve")),
  coefs = dplyr::bind_rows(lapply(res, `[[`, "coef")))
)
},
notes = paste0("m=", imp_n)
)

vbg_curves <- vbg_spline$curves
vbg_coefs <- vbg_spline$coefs
utils::write.csv(vbg_curves, results_path("mi_spline_curve_vbg.csv"), row.names = FALSE)
utils::write.csv(vbg_coefs, results_path("mi_spline_coef_vbg.csv"), row.names = FALSE)

```

Chunk mi-vbg-outcomes runtime: 61.95 s

4.6 12) Explainability on one representative imputation

To manage runtime, compute SHAP summaries (importance + dependence) on the first imputed dataset and its fitted GBM(s).

Chunk shap-axis-labels runtime: 0.00 s

```

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

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

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

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

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

  cn     <- colnames(X_raw)
}

```

```

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

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

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

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

  # Case 2: derive dummy = 1{ base == level } from a base column prefix
  # Find the longest raw name that is a prefix of v (sanitized comparison)
  cand <- which(startsWith(v_s, cn_s))
  if (length(cand)) {
    j <- cand[which.max(nchar(cn_s[cand]))]
    base <- cn[j]
    # level is the suffix of v after the base name (unsanitized; preserves case)
    lev <- sub(paste0("^", base), "", v)
    x   <- X_raw[[base]]

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

  stop("Cannot construct design column for '", v, "'.
}

```

```

        "No matching base variable found in raw covariates.")
}

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

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

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

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

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

  # Path B: dummy-coded training (var.names are one-hot)
}

```

```

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

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

```

Chunk unnamed-chunk-2 runtime: 0.00 s

```

# Choose one completed dataset's fit
# Device safety (once per doc)
fs::dir_create(fig_dir, recurse = TRUE)
knitr::opts_chunk$set(fig.path = fig_path, dev = "ragg_png", dpi = 200)

# SHAP throttle knobs (single imputation, subsample rows, fewer sims)
shap_cfg <- list(
  frac_rows = 0.03,
  nsim      = 8,
  top_k     = 20,
  seed       = 123
)
shap_t0 <- Sys.time()

add_smooth_safe <- function(p, x_vals, k = 4) {
  x_vals <- suppressWarnings(as.numeric(as.character(x_vals)))
  x_vals <- x_vals[is.finite(x_vals)]
  n_uniq <- length(unique(x_vals))
  if (n_uniq < 4) return(p)
  k_use <- min(k, n_uniq - 1)
}

```

```

p + ggplot2::geom_smooth(method = "gam",
                         formula = y ~ s(x, bs = "cs", k = k_use),
                         se = FALSE, linewidth = 0.5)
}

# --- ABG explainability on one representative imputation -----
stopifnot(exists("imp"))
rep_imp_idx <- 1
d_rep <- get_imp(rep_imp_idx)
assert_no_na_covars(d_rep, covars_gbm, context = "SHAP ABG")
w_abg_shap <- weightit(
  formula_abg,
  data      = d_rep[, c("has_abg", covars_gbm)],
  method    = "gbm",
  estimand  = "ATE",
  include.obj = TRUE,
  n.trees     = gbm_params$n.trees,
  interaction.depth = gbm_params$interaction.depth,
  shrinkage   = gbm_params$shrinkage,
  bag.fraction = gbm_params$bag.fraction,
  cv.folds    = gbm_params$cv.folds,
  stop.method = gbm_params$stop.method,
  n.cores     = gbm_params$n.cores
)
bk_abg <- make_shap_backend_any(w_abg_shap)

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

```

```

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

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

# Stability check: rerun SHAP with a different seed
set.seed(shap_cfg$seed + 1)
S_abg_b <- fastshap::explain(
  object      = bk_abg$gbm,
  X           = X_abg,
  pred_wrapper = bk_abg$pred,
  nsim        = shap_cfg$nsim,
  adjust       = FALSE
)

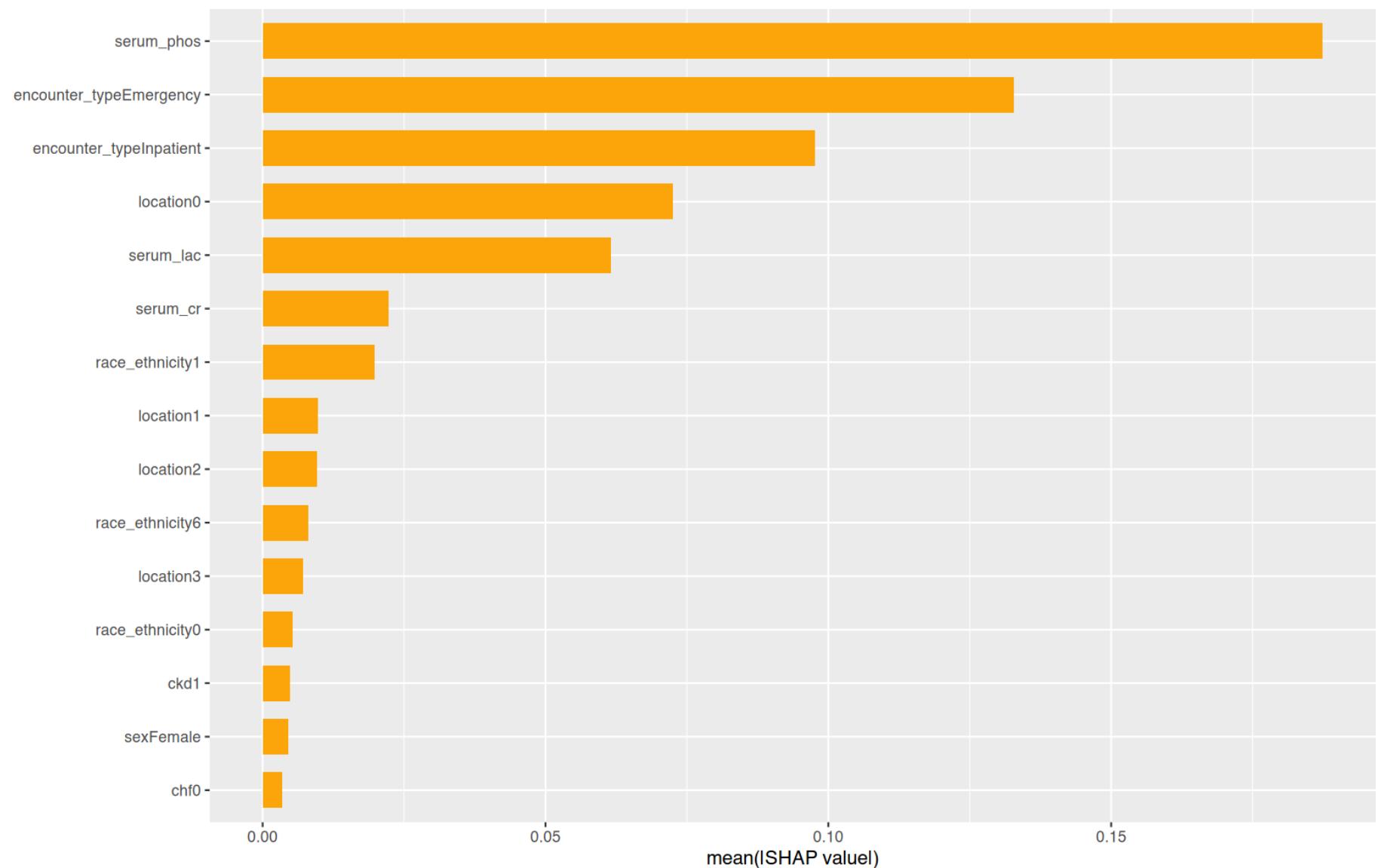
shap_stability_abg <- tibble::tibble(
  cohort = "ABG",
  feature = colnames(S_abg),
  mean_abs_seed1 = colMeans(abs(S_abg), na.rm = TRUE),
  mean_abs_seed2 = colMeans(abs(S_abg_b), na.rm = TRUE)
)

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

# Bar importance (top K)

```

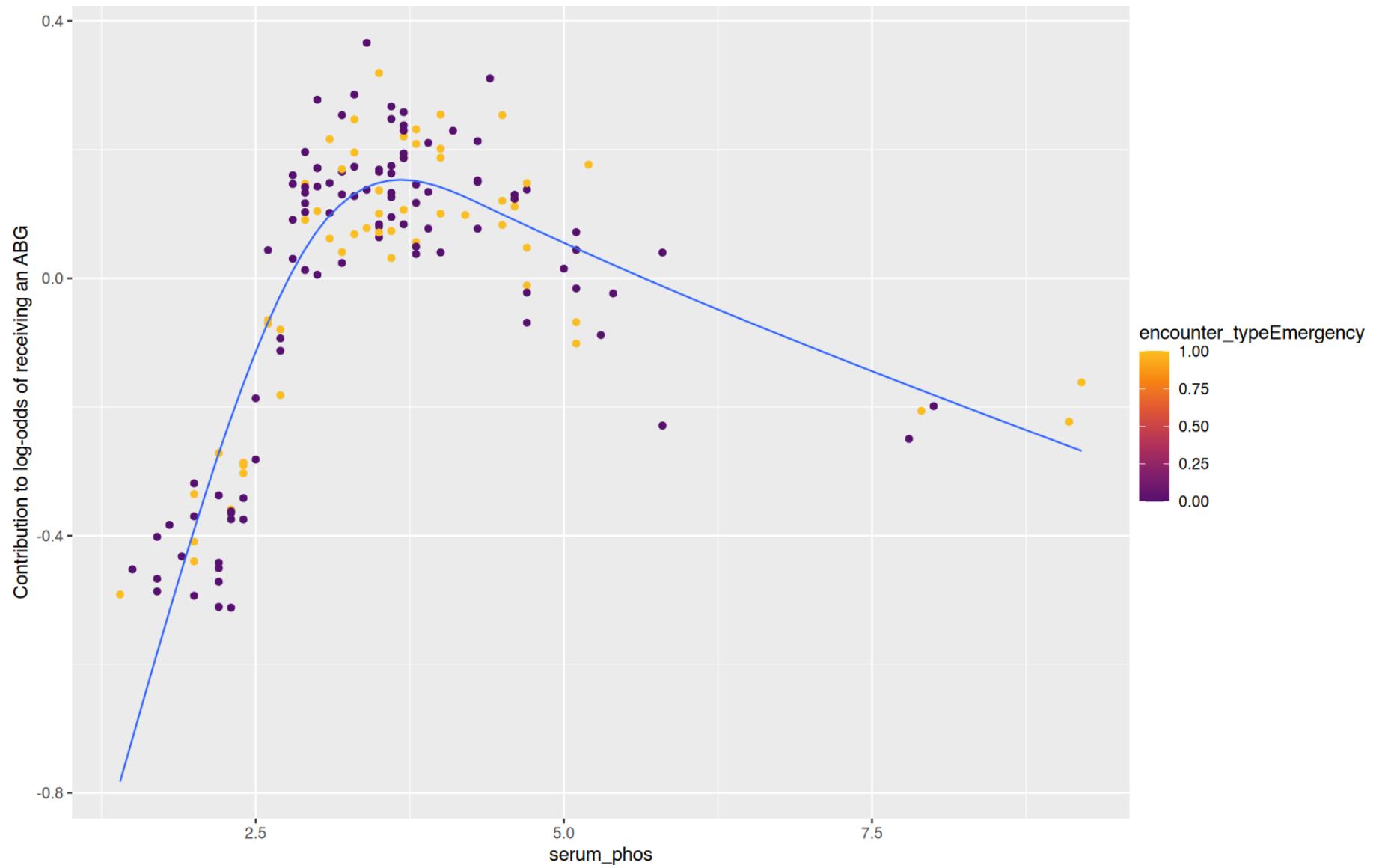
```
ord_abg    <- order(colMeans(abs(S_abg), na.rm = TRUE), decreasing = TRUE)
topK_abg   <- colnames(S_abg)[ord_abg[1:min(shap_cfg$top_k, ncol(S_abg))]]
p_bar_abg <- suppressWarnings(shapviz::sv_importance(sv_abg, kind = "bar", v = topK_abg))
suppressWarnings(print(p_bar_abg))
```



```
# Dependence: top feature colored by second (add smoother explicitly)
pri_abg <- topK_abg[1]
aux_abg <- topK_abg[2]
p_dep_abg <- suppressWarnings(shapviz::sv_dependence(sv_abg, v = pri_abg, color_var = aux_abg))
```

```
p_dep_abg <- add_smooth_safe(p_dep_abg, X_abg[, pri_abg])
p_dep_abg <- p_dep_abg + ggplot2::labs(y = shap_y_abg, x = pri_abg)

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



Chunk shap-abg-vbg runtime: 11.07 s

```
# Repeat for VBG
```

```

# --- VBG explainability on one representative imputation -----
assert_no_na_covars(d_rep, covars_gbm, context = "SHAP VBG")
w_vbg_shap <- weightit(
  formula_vbg,
  data      = d_rep[, c("has_vbg", covars_gbm)],
  method    = "gbm",
  estimand  = "ATE",
  include.obj = TRUE,
  n.trees     = gbm_params$n.trees,
  interaction.depth = gbm_params$interaction.depth,
  shrinkage   = gbm_params$shrinkage,
  bag.fraction = gbm_params$bag.fraction,
  cv.folds    = gbm_params$cv.folds,
  stop.method  = gbm_params$stop.method,
  n.cores     = gbm_params$n.cores
)
bk_vbg <- make_shap_backend_any(w_vbg_shap)

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

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

set.seed(shap_cfg$seed + 1)
S_vbg_b <- fastshap::explain(
  object      = bk_vbg$gbm,
  X           = X_vbg,
  pred_wrapper = bk_vbg$pred,
  nsim        = shap_cfg$nsim,

```

```

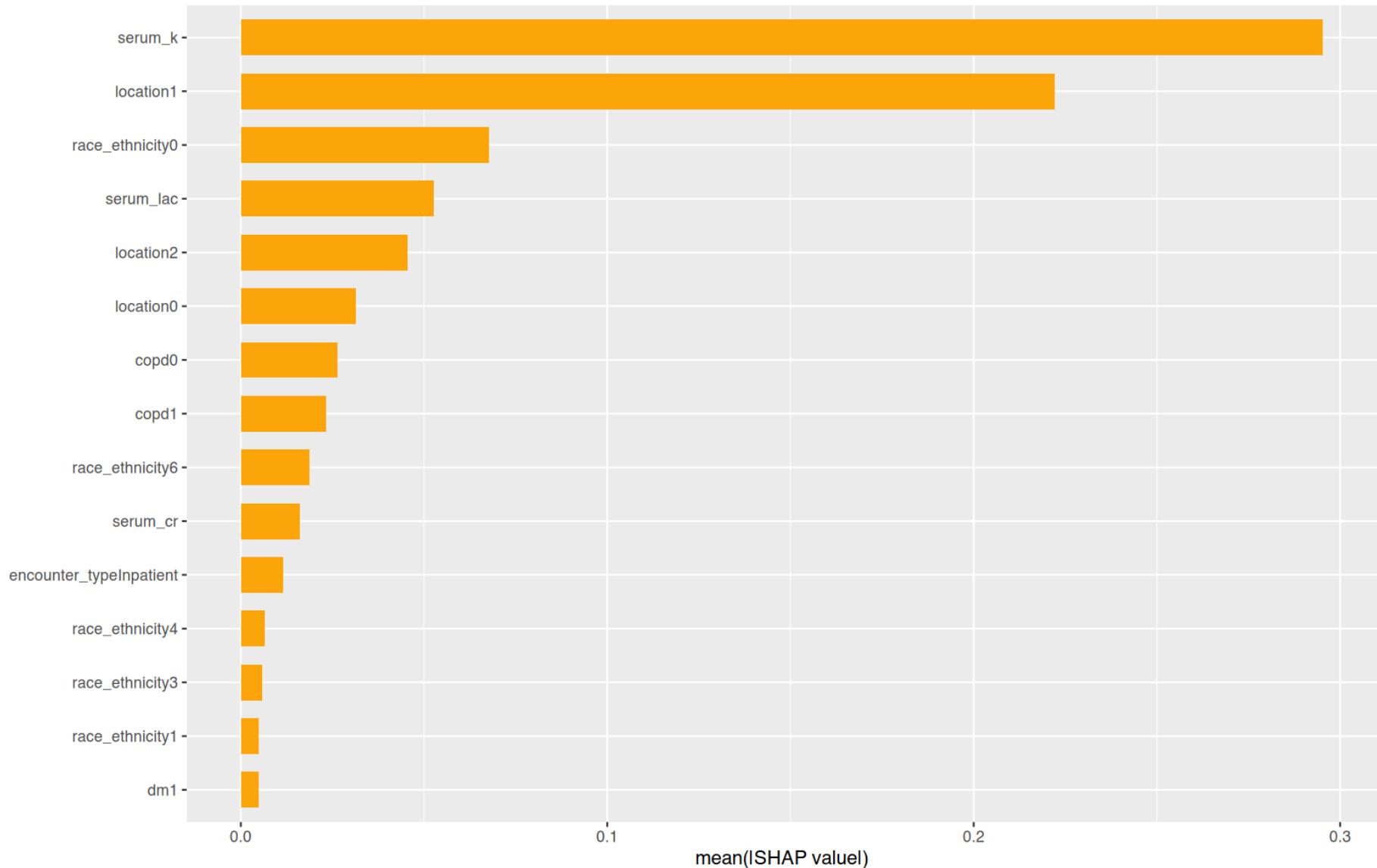
    adjust      = FALSE
)

shap_stability_vbg <- tibble::tibble(
  cohort = "VBG",
  feature = colnames(S_vbg),
  mean_abs_seed1 = colMeans(abs(S_vbg), na.rm = TRUE),
  mean_abs_seed2 = colMeans(abs(S_vbg_b), na.rm = TRUE)
)

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

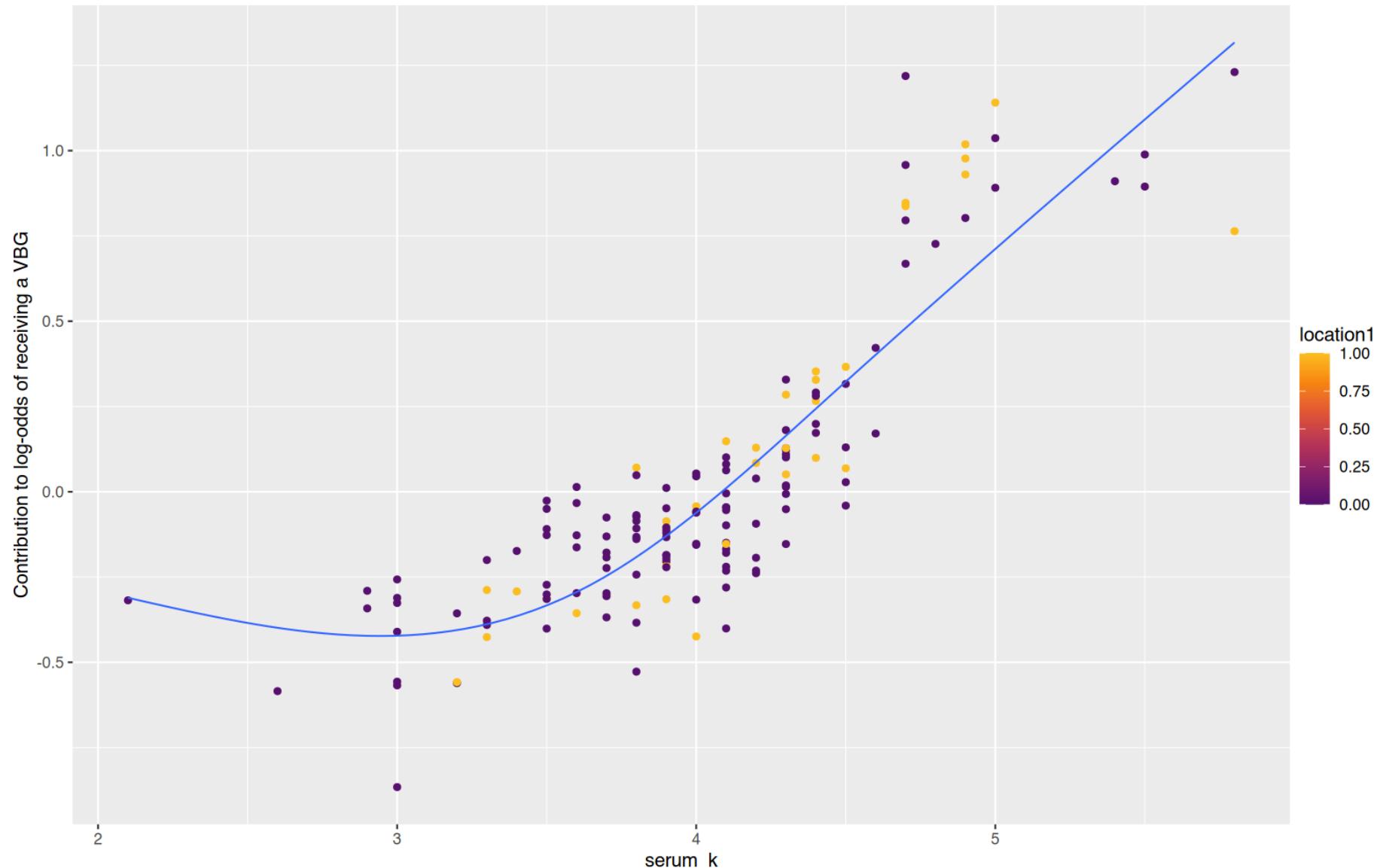
ord_vbg   <- order(colMeans(abs(S_vbg), na.rm = TRUE), decreasing = TRUE)
topK_vbg  <- colnames(S_vbg)[ord_vbg[1:min(shap_cfg$top_k, ncol(S_vbg))]]
p_bar_vbg <- suppressWarnings(shapviz::sv_importance(sv_vbg, kind = "bar", v = topK_vbg))
suppressWarnings(print(p_bar_vbg))

```



```
pri_vbg <- topK_vbg[1]
aux_vbg <- topK_vbg[2]
p_dep_vbg <- suppressWarnings(shapviz::sv_dependence(sv_vbg, v = pri_vbg, color_var = aux_vbg))
p_dep_vbg <- add_smooth_safe(p_dep_vbg, X_vbg[, pri_vbg])
```

```
p_dep_vbg <- p_dep_vbg + ggplot2::labs(y = shap_y_vbg, x = pri_vbg)
suppressWarnings(print(p_dep_vbg))
```



```

shap_stability <- bind_rows(shap_stability_abg, shap_stability_vbg)
shap_stability_file <- results_path("diag-ps-shap-stability.csv")
utils::write.csv(shap_stability, shap_stability_file, row.names = FALSE)

shap_corr <- shap_stability |>
  group_by(cohort) |>
  summarise(rank_corr = cor(mean_abs_seed1, mean_abs_seed2, method = "spearman"),
            .groups = "drop")

p_shap_stab <- ggplot(shap_stability, aes(x = mean_abs_seed1, y = mean_abs_seed2)) +
  geom_point(alpha = 0.6, size = 1) +
  geom_smooth(method = "lm", se = FALSE, linewidth = 0.4) +
  facet_wrap(~cohort, scales = "free") +
  labs(
    title = "SHAP stability (mean |SHAP|, two seeds)",
    x = "Mean |SHAP| (seed 1)",
    y = "Mean |SHAP| (seed 2)"
  ) +
  theme_minimal(base_size = 10)

save_diag_plot(p_shap_stab, results_path("figs", "diag-ps-shap-stability.png"),
               width = 8, height = 4.5)

```

`geom_smooth()` using formula = 'y ~ x'

```

shap_t1 <- Sys.time()
if (exists("runtime_log")) {
  runtime_log <- rbind(
    runtime_log,
    data.frame(
      step_name = "shap_summary",
      seconds = as.numeric(difftime(shap_t1, shap_t0, units = "secs")),
      notes = paste0("frac_rows=", shap_cfg$frac_rows, "; nsim=", shap_cfg$nsim),
      stringsAsFactors = FALSE
    )
}

```

```
)  
}
```

Chunk unnamed-chunk-3 runtime: 11.75 s

4.7 13) MI three-level PCO2 helpers and checks

```
# --- helpers -----  
library(splines)  
library(mitoools)  
library(survey)  
library(dplyr)  
  
# Pool (Rubin) any subset of coefficients across imputed fits (glm/svyglm)  
pool_terms <- function(fits, term_prefix = NULL, term_pattern = NULL) {  
  fits <- Filter(Negate(is.null), fits)  
  if (!length(fits)) return(  
    data.frame(term = character(), logOR = numeric(), SE = numeric(),  
               OR = numeric(), LCL = numeric(), UCL = numeric())  
)  
  if (!all(vapply(fits, inherits, logical(1), "svyglm"))){  
    stop("pool_terms expects svyglm fits so pooled vcov is robust.")  
  }  
  
  coef_names <- lapply(fits, function(f) names(stats::coef(f)))  
  common <- Reduce(intersect, coef_names)  
  if (!is.null(term_prefix))  common <- common[startsWith(common, term_prefix)]  
  if (!is.null(term_pattern)) common <- common[grep1(term_pattern, common)]  
  if (!length(common)) return(  
    data.frame(term = character(), logOR = numeric(), SE = numeric(),  
               OR = numeric(), LCL = numeric(), UCL = numeric())  
)  
  
  rows <- lapply(common, function(tt) {
```

```

results <- lapply(fits, function(f) setNames(c(stats::coef(f)[tt]), tt))
variances <- lapply(fits, function(f) {
  v <- stats::vcov(f)[tt, tt, drop = TRUE]
  m <- matrix(v, 1, 1); dimnames(m) <- list(tt, tt); m
})
pooled <- mitools::MIcombine(results = results, variances = variances)
coefs <- vapply(results, function(x) unname(x[[1]]), numeric(1))
within <- mean(vapply(variances, function(x) x[1, 1], numeric(1)))
between <- stats::var(coefs)
attr(pooled, "within") <- within
attr(pooled, "between") <- between
pooled_mi_vcov_check(pooled)
est <- as.numeric(coef(pooled))
se <- sqrt(diag(pooled$variance))
data.frame(term = tt,
           logOR = est, SE = se,
           OR = exp(est),
           LCL = exp(est - 1.96*se),
           UCL = exp(est + 1.96*se),
           row.names = NULL)
})
dplyr::bind_rows(rows)
}

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

```

```

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

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

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

# (Optional) explicit clinical cutpoints - override by setting use_fixed_* = TRUE
use_fixed_abg <- TRUE
co2_breaks_abg <- c(-Inf, 35, 45, Inf)
co2_labels_abg <- c("Hypocapnia", "Eucapnia", "Hypercapnia")
ref_label_abg <- "Eucapnia"

use_fixed_vbg <- TRUE
co2_breaks_vbg <- c(-Inf, 40, 50, Inf)
co2_labels_vbg <- c("Hypocapnia", "Eucapnia", "Hypercapnia")

```

```

ref_label_vbg      <- "Eucapnia"

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

```

Chunk mi_pco2_threel level runtime: 0.01 s

Chunk mi-co2-cat-checks runtime: 25.25 s

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

4.8.1 14.1 ABG: MI + IPW, three-level PCO2 outcomes

```

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

if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)

mi_abg_cat_forms <- list(
  "MI IPW ABG 3-level: IMV ~ CO2 category"      = imv_proc ~ co2_cat,
  "MI IPW ABG 3-level: NIV ~ CO2 category"        = niv_proc ~ co2_cat,
  "MI IPW ABG 3-level: Death60d ~ CO2 category"   = death_60d ~ co2_cat,
  "MI IPW ABG 3-level: HCRF ~ CO2 category"       = hypercap_resp_failure ~ co2_cat
)
register_model_diagrams(mi_abg_cat_forms)

```

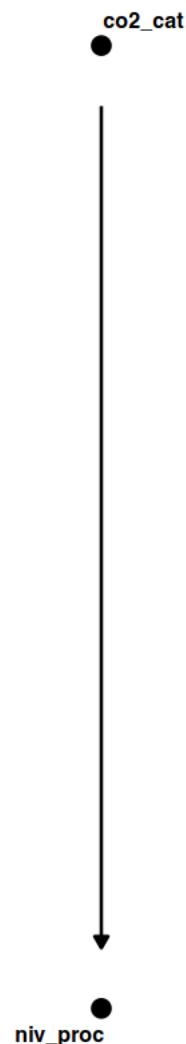
MI IPW ABG 3-level: IMV ~ CO2 category

co2_cat

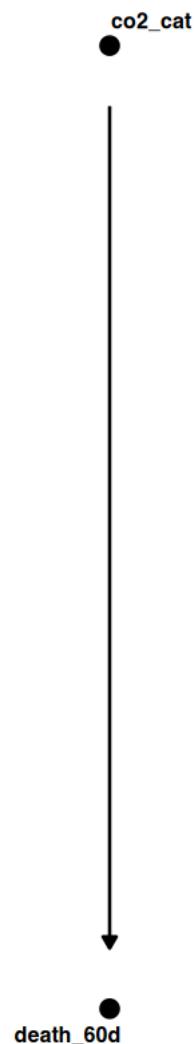


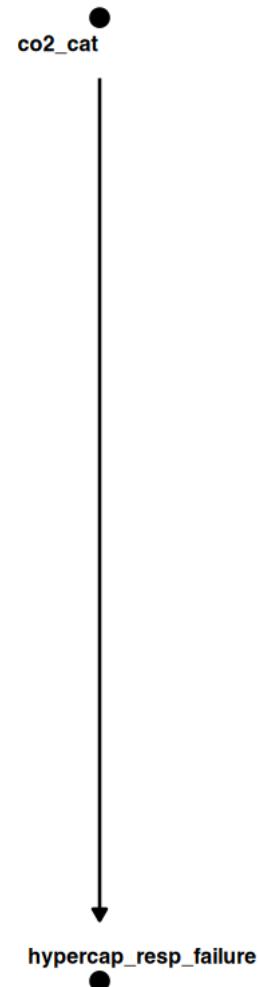
imv_proc

MI IPW ABG 3-level: NIV ~ CO2 category



MI IPW ABG 3-level: Death60d ~ CO2 category





```
fit_abg_cat <- function(outcome_var) {  
  fits <- vector("list", imp_n)  
  for (i in seq_len(imp_n)) {  
    d <- get_imp(i)
```

```

if (!("paco2" %in% names(d))) { fits[[i]] <- NULL; next }
d$paco2 <- suppressWarnings(as.numeric(d$paco2))

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

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

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

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

abg_cat_results <- dplyr::bind_rows(
  dplyr::mutate(fit_abg_cat("imv_proc"),           outcome = "IMV"),
  dplyr::mutate(fit_abg_cat("niv_proc"),           outcome = "NIV"),

```

```

dplyr::mutate(fit_abg_cat("death_60d"),           outcome = "Death60d"),
dplyr::mutate(fit_abg_cat("hypercap_resp_failure"), outcome = "HCRF")
) |>
dplyr::relocate(outcome)

```

Chunk unnamed-chunk-4 runtime: 64.25 s

4.8.2 14.2 VBG: MI + IPW, three-level PCO2 outcomes

```

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

if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice:::complete(imp_obj, action = i), levels_ref)

mi_vbg_cat_forms <- list(
  "MI IPW VBG 3-level: IMV ~ CO2 category"      = imv_proc ~ co2_cat,
  "MI IPW VBG 3-level: NIV ~ CO2 category"       = niv_proc ~ co2_cat,
  "MI IPW VBG 3-level: Death60d ~ CO2 category" = death_60d ~ co2_cat,
  "MI IPW VBG 3-level: HCRF ~ CO2 category"     = hypercap_resp_failure ~ co2_cat
)
register_model_diagrams(mi_vbg_cat_forms)

```

MI IPW VBG 3-level: IMV ~ CO2 category

co2_cat



imv_proc

MI IPW VBG 3-level: NIV ~ CO2 category

co2_cat



niv_proc

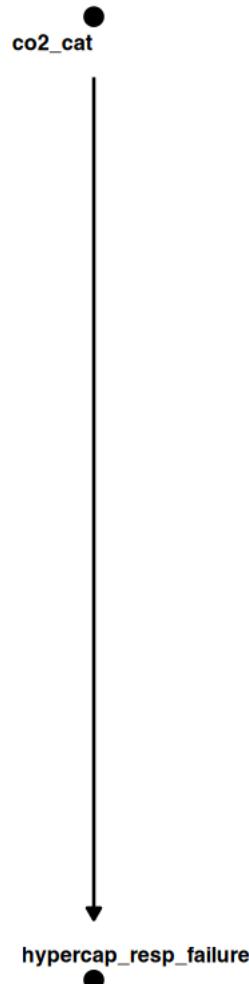
MI IPW VBG 3-level: Death60d ~ CO2 category

co2_cat



death_60d

MI IPW VBG 3-level: HCRF ~ CO2 category



```
fit_vbg_cat <- function(outcome_var) {  
  fits <- vector("list", imp_n)  
  for (i in seq_len(imp_n)) {  
    d <- get_imp(i)
```

```

if (!("vbg_co2" %in% names(d))) { fits[[i]] <- NULL; next }
d$vbg_co2 <- suppressWarnings(as.numeric(d$vbg_co2))

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

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

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

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

vbg_cat_results <- dplyr::bind_rows(
  dplyr::mutate(fit_vbg_cat("imv_proc"), outcome = "IMV"),
  dplyr::mutate(fit_vbg_cat("niv_proc"), outcome = "NIV"),

```

```

dplyr::mutate(fit_vbg_cat("death_60d"), outcome = "Death60d"),
dplyr::mutate(fit_vbg_cat("hypercap_resp_failure"), outcome = "HCRF")
) |>
dplyr::relocate(outcome)

```

Chunk unnamed-chunk-5 runtime: 60.44 s

4.8.3 14.3 Table 3: MI-pooled IPW associations (3-level CO)

Chunk table3-ipw-mi-co2 runtime: 0.02 s

```

# After re-running MICE:
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)

# 1) must exist and be numeric
d1 <- get_imp(1)
stopifnot(all(c("paco2", "vbg_co2") %in% names(d1)))
stopifnot(is.numeric(d1$paco2), is.numeric(d1$vbg_co2))

# 2) confirm at least two PaCO2 levels among those with ABG in each imputation
table(vapply(seq_len(imp_n), function(i) {
  d <- get_imp(i)
  dplyr::n_distinct(d$paco2[d$has_abg == 1 & is.finite(d$paco2)])
}), integer(1)) > 1)

```

TRUE

80

```

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

```

	term	logOR	SE	OR	LCL	UCL
1	co2_catHypocapnia	0.1815481	0.06431886	1.199072	1.057051	1.360175
2	co2_catHypercapnia	0.1396242	0.06564564	1.149842	1.011019	1.307726

Chunk unnamed-chunk-6 runtime: 28.01 s

4.8.4 14.3 Visualization: pooled three-level ORs

```

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

# --- Pre-flight -----
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) { normalize_types(mice::complete(imp_obj, action = i), levels_ref)
stopifnot(length(W_abg_list) == imp_n,
          length(W_vbg_list) == imp_n)

# --- Pooling helper for term-level log-ORs across imputations -----
pool_terms <- function(fits, term_prefix) {
  fits <- Filter(Negate(is.null), fits)
  if (length(fits) == 0L) {
    return(tibble::tibble(term=character(), logOR=numeric(), SE=numeric(),
                          OR=numeric(), LCL=numeric(), UCL=numeric()))
  }
  terms_list <- lapply(fits, function(f) names(stats::coef(f)))
  common      <- Reduce(intersect, terms_list)
  keep_terms <- grep(paste0("^", term_prefix), common, value = TRUE)
  if (!length(keep_terms)) {
    return(tibble::tibble(term=character(), logOR=numeric(), SE=numeric(),
                          OR=numeric(), LCL=numeric(), UCL=numeric()))
  }
  else {
    pool_tibble <- tibble::tibble(term=character(), logOR=numeric(), SE=numeric(),
                                  OR=numeric(), LCL=numeric(), UCL=numeric())
    for (term in keep_terms) {
      pool_tibble <- pool_tibble %>%
        mutate_at(.vars = term, .funs = list(logOR = ~sum(logOR),
                                             SE = ~sqrt(sum(SE^2)),
                                             OR = ~prod(OR),
                                             LCL = ~prod(OR) * exp(-1.96 * sum(SE)),
                                             UCL = ~prod(OR) * exp(1.96 * sum(SE))))
    }
    return(pool_tibble)
  }
}

```

```

          OR=numeric(), LCL=numeric(), UCL=numeric())))
}

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

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

  for (i in seq_len(imp_n)) {
    d <- get_imp(i)

    if (group == "ABG") {
      if (!("paco2" %in% names(d))) { fits[[i]] <- NULL; next }
      d$paco2 <- suppressWarnings(as.numeric(d$paco2))
      g <- with(d, has_abg == 1 & is.finite(paco2))
      if (!any(g)) { fits[[i]] <- NULL; next }
    }
  }
}

```

```

d2 <- d[g, , drop = FALSE]
d2$co2_cat <- make_co2_cat(
  d2$paco2,
  fixed_breaks = if (exists("use_fixed_abg", inherits = TRUE) && isTRUE(use_fixed_abg)) co2_breaks_abg else NULL,
  labels       = if (exists("co2_labels_abg", inherits = TRUE)) co2_labels_abg else c("Eucapnia","Hypocapnia","Hypercapnia"),
  normal       = if (exists("co2_breaks_abg", inherits = TRUE)) co2_breaks_abg else c(35, 45)
)
w <- W_abg_list[[i]]$weights[g]
w[!is.finite(w)] <- NA_real_

} else {
  if (!("vbg_co2" %in% names(d))) { fits[[i]] <- NULL; next }
  d$vbg_co2 <- suppressWarnings(as.numeric(d$vbg_co2))
  g <- with(d, has_vbg == 1 & is.finite(vbg_co2))
  if (!any(g)) { fits[[i]] <- NULL; next }
  d2 <- d[g, , drop = FALSE]
  d2$co2_cat <- make_co2_cat(
    d2$vbg_co2,
    fixed_breaks = if (exists("use_fixed_vbg", inherits = TRUE) && isTRUE(use_fixed_vbg)) co2_breaks_vbg else NULL,
    labels       = if (exists("co2_labels_vbg", inherits = TRUE)) co2_labels_vbg else c("Eucapnia","Hypocapnia","Hypercapnia"),
    normal       = if (exists("co2_breaks_vbg", inherits = TRUE)) co2_breaks_vbg else c(40, 50)
  )
  w <- W_vbg_list[[i]]$weights[g]
  w[!is.finite(w)] <- NA_real_
}

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

okw <- is.finite(w)

```

```

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

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

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

# --- Run & plot -----
outs <- c("imv_proc", "niv_proc", "death_60d", "hypercap_resp_failure")

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

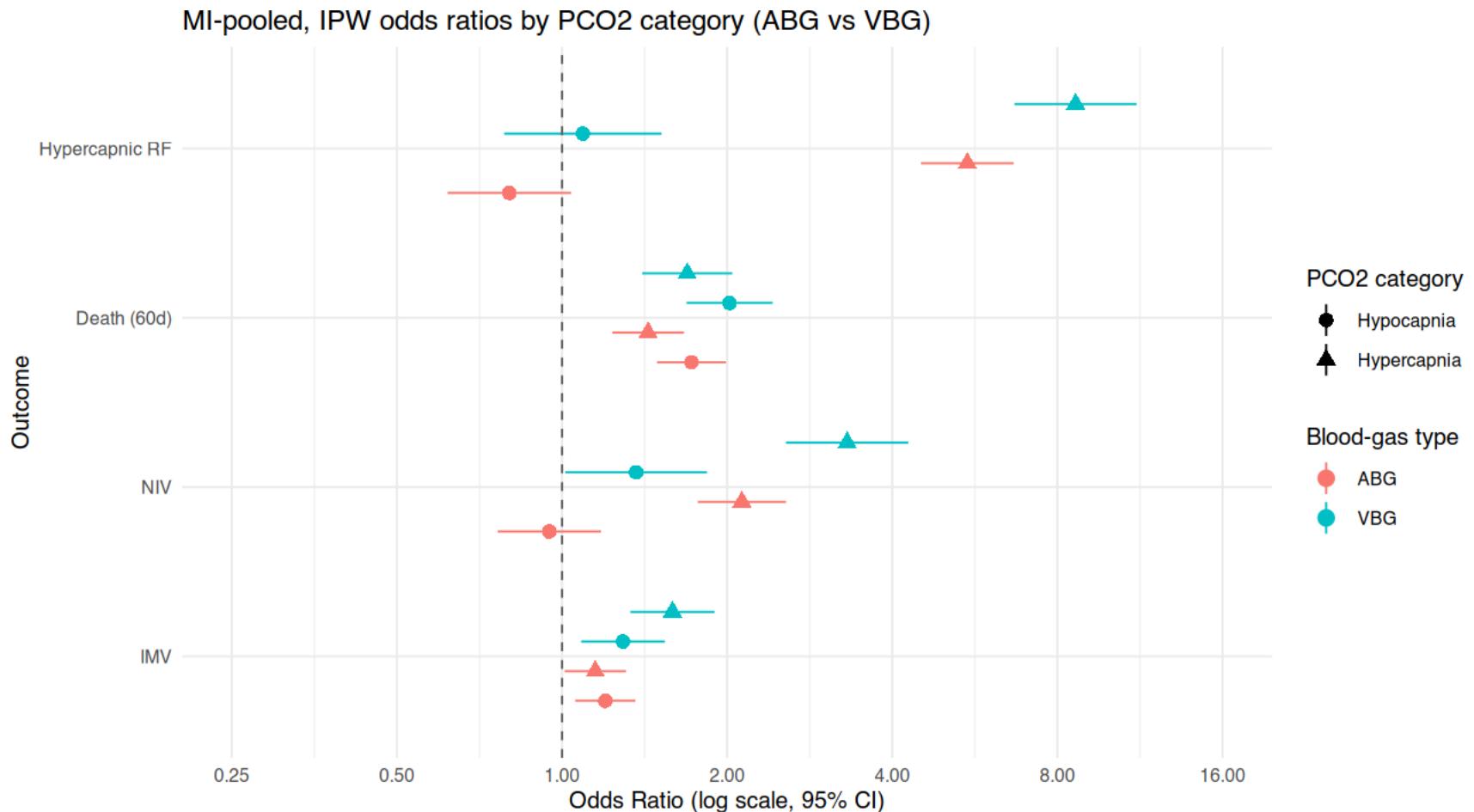
# decode "co2_cat<level>" → exposure
combined <- combined |>
  dplyr::mutate(exposure = gsub("^co2_cat", "", term),
                exposure = factor(exposure, levels = c("Eucapnia", "Hypocapnia", "Hypercapnia")),
                outcome  = factor(outcome,
                                  levels = c("imv_proc",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

```

```

combined,
aes(x = outcome, y = OR, ymin = LCL, ymax = UCL, color = group, shape = exposure)
) +
geom_pointrange(position = position_dodge(width = 0.7), size = 0.6) +
geom_hline(yintercept = 1, linetype = "dashed", colour = "grey40") +
scale_y_log10(
  breaks = c(0.25, 0.5, 1, 2, 4, 8, 16),
  limits = c(0.25, 16),
  labels = scales::number_format(accuracy = 0.01)
) +
coord_flip() +
labs(
  title = "MI-pooled, IPW odds ratios by PCO2 category (ABG vs VBG)",
  x = "Outcome",
  y = "Odds Ratio (log scale, 95% CI)",
  color = "Blood-gas type",
  shape = "PCO2 category"
) +
theme_minimal(base_size = 10)

```



Chunk ipw-three-level-pco2-mi-abg-vbg runtime: 124.39 s

4.9 15) Imputed, weighted spline PCO₂ (ABG & VBG)

MI-pooled IPW fits use univariate logistic splines ($\text{outcome} \sim \text{spline}(\text{CO}_2)$) within ABG and VBG separately, with robust variance from `svyglm`.

4.9.1 15.1 ABG, imputed, weighted, spline outcome

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

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

outcome	term	logOR	SE	OR	LCL	UCL	group
imv_proc	co2_catHypocapnia	0.1815481	0.0643189	1.1990722	1.0570512	1.360175	ABG
imv_proc	co2_catHypercapnia	0.1396242	0.0656456	1.1498416	1.0110190	1.307726	ABG
niv_proc	co2_catHypocapnia	-0.0534225	0.1105215	0.9479794	0.7633455	1.177272	ABG
niv_proc	co2_catHypercapnia	0.7549138	0.0945619	2.1274280	1.7675111	2.560635	ABG
death_60d	co2_catHypocapnia	0.5436555	0.0738001	1.7222912	1.4903446	1.990336	ABG
death_60d	co2_catHypercapnia	0.3614758	0.0767099	1.4354462	1.2350658	1.668337	ABG
hypercap_resp_failure	co2_catHypocapnia	-0.2211551	0.1322949	0.8015924	0.6185031	1.038880	ABG
hypercap_resp_failure	co2_catHypercapnia	1.7014151	0.0992745	5.4816990	4.5124353	6.659159	ABG

Chunk unnamed-chunk-7 runtime: 62.81 s

4.9.2 15.2 VBG, imputed, weighted, spline outcome

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

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

outcome	term	logOR	SE	OR	LCL	UCL	group
imv_proc	co2_catHypocapnia	0.2559543	0.0896778	1.291694	1.0834883	1.539908	VBG
imv_proc	co2_catHypercapnia	0.4636803	0.0902458	1.589915	1.3321557	1.897547	VBG
niv_proc	co2_catHypocapnia	0.3110623	0.1516738	1.364874	1.0138771	1.837384	VBG
niv_proc	co2_catHypercapnia	1.1970167	0.1311369	3.310227	2.5599516	4.280394	VBG
death_60d	co2_catHypocapnia	0.7032389	0.0921993	2.020286	1.6862857	2.420441	VBG
death_60d	co2_catHypercapnia	0.5257237	0.0965631	1.691683	1.3999829	2.044161	VBG
hypercap_resp_failure	co2_catHypocapnia	0.0873238	0.1684523	1.091250	0.7843949	1.518147	VBG
hypercap_resp_failure	co2_catHypercapnia	2.1557842	0.1307953	8.634659	6.6820540	11.157846	VBG

Chunk unnamed-chunk-8 runtime: 58.72 s

4.9.3 15.3 Visualization

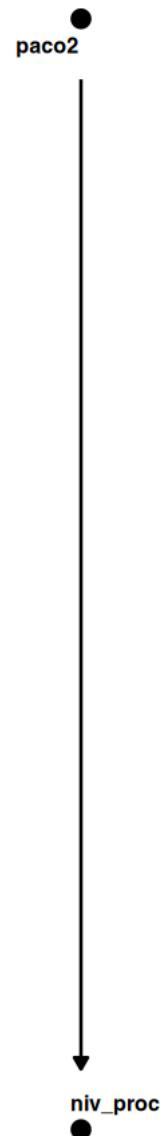
```
library(dplyr)
library(ggplot2)
library(patchwork)
library(purrr)

mi_ipw_rcs_forms <- list(
  "MI IPW RCS: IMV ~ PaCO2 (ABG)" = imv_proc ~ rcs(paco2, 4),
  "MI IPW RCS: NIV ~ PaCO2 (ABG)" = niv_proc ~ rcs(paco2, 4),
  "MI IPW RCS: Death60d ~ PaCO2 (ABG)" = death_60d ~ rcs(paco2, 4),
  "MI IPW RCS: HCRF ~ PaCO2 (ABG)" = hypercap_resp_failure ~ rcs(paco2, 4),
  "MI IPW RCS: IMV ~ CO2 (VBG)" = imv_proc ~ rcs(vbg_co2, 4),
  "MI IPW RCS: NIV ~ CO2 (VBG)" = niv_proc ~ rcs(vbg_co2, 4),
  "MI IPW RCS: Death60d ~ CO2 (VBG)" = death_60d ~ rcs(vbg_co2, 4),
  "MI IPW RCS: HCRF ~ CO2 (VBG)" = hypercap_resp_failure ~ rcs(vbg_co2, 4)
)
register_model_diagrams(mi_ipw_rcs_forms)
```

MI IPW RCS: IMV ~ PaCO₂ (ABG)



MI IPW RCS: NIV ~ PaCO2 (ABG)



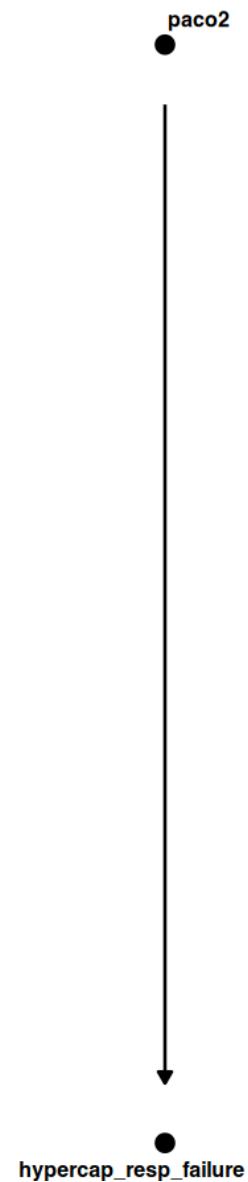
MI IPW RCS: Death60d ~ PaCO2 (ABG)

paco2



death_60d

MI IPW RCS: HCRF ~ PaCO2 (ABG)



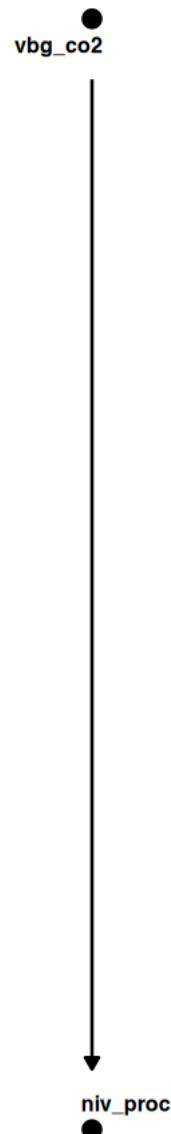
MI IPW RCS: IMV ~ CO2 (VBG)

vbg_co2



imv_proc

MI IPW RCS: NIV ~ CO2 (VBG)



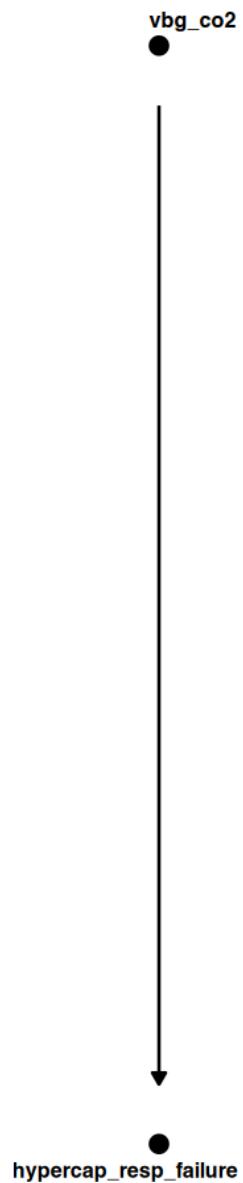
MI IPW RCS: Death60d ~ CO2 (VBG)

vbg_co2



death_60d

MI IPW RCS: HCRF ~ CO2 (VBG)



```

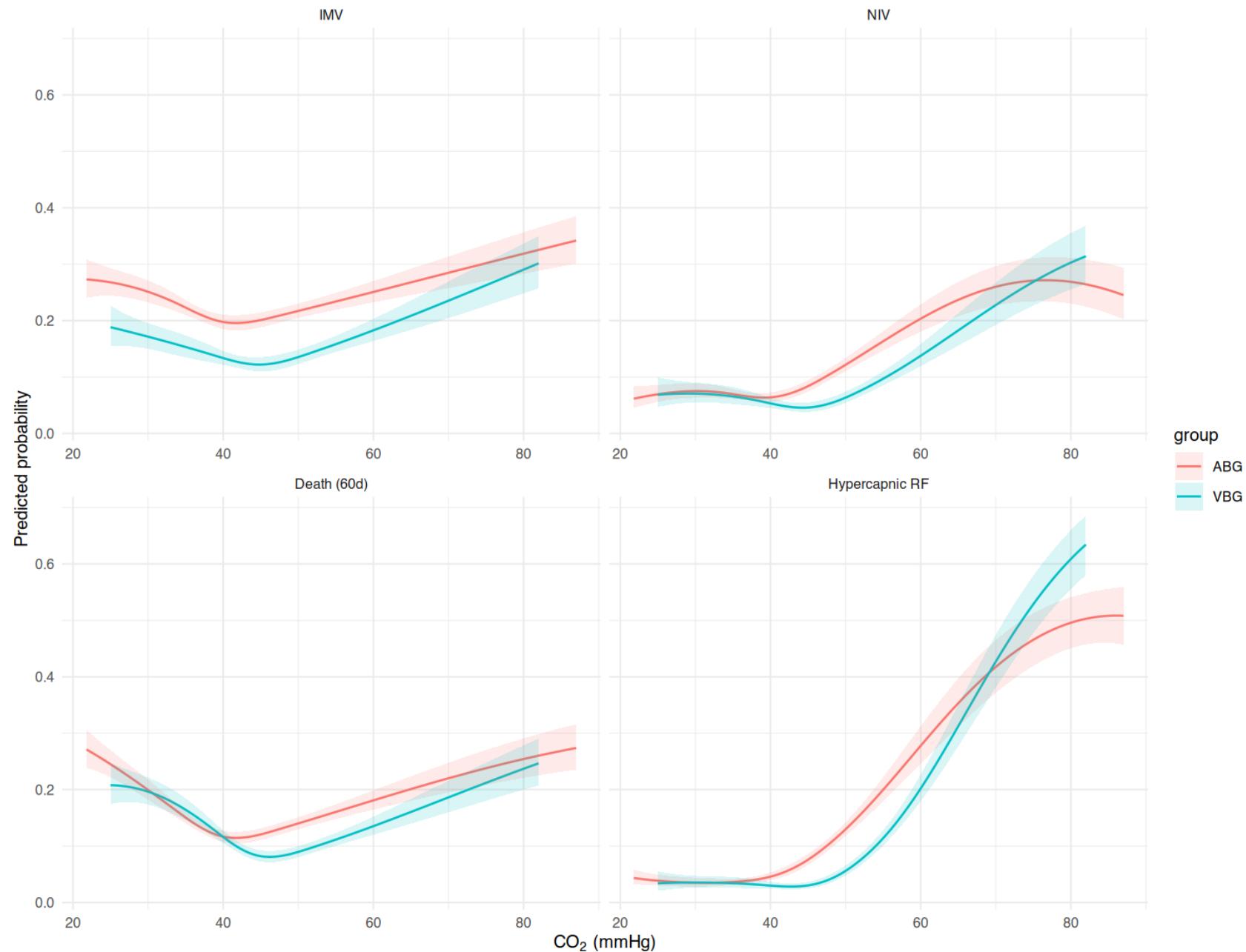
stopifnot(exists("abg_curves"), exists("vbg_curves"))

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

ggplot(curve_all, aes(x = co2, y = prob, color = group, fill = group)) +
  geom_line(linewidth = 0.6) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), alpha = 0.15, color = NA) +
  facet_wrap(~ outcome, scales = "free_x") +
  labs(
    title = expression(
      paste("MI-pooled, IPSW spline curves: ABG vs VBG CO[2]"))
  ),
  x = expression(CO[2]~"(mmHg)"),
  y = "Predicted probability"
) +
  theme_minimal(base_size = 10)

```

MI-pooled, IPSW spline curves: ABG vs VBG CO₂



Chunk ipw-rcs-overlay-mi-abg-vbg runtime: 1.97 s

4.10 Diagnostics

4.10.1 Diagnostics inputs and settings

Chunk diagnostics-inputs runtime: 0.01 s

4.10.2 Missingness diagnostics

png
2

Chunk diagnostics-missingness runtime: 0.86 s

4.10.3 MI convergence and mixing

png
2

Chunk diagnostics-mi-convergence runtime: 0.02 s

4.10.4 MI stability across m

Chunk diagnostics-mi-stability runtime: 64.56 s

4.10.5 MI maxit sensitivity (sampled)

Chunk diagnostics-mi-maxit runtime: 35.07 s

4.10.6 Propensity and weight diagnostics

Chunk diagnostics-weights runtime: 1.87 s

4.10.7 Balance diagnostics

Table 15: Target balance (top 10 by max |SMD|)

group	variable	max_abs_pre	max_abs_post
ABG	encounter_type	0.3782737	0.1727255
ABG	curr_bmi	0.2558666	0.1644248
ABG	location	0.1646756	0.0967483
ABG	age_at_encounter	0.1095048	0.0887311
ABG	dbp	0.1734467	0.0802226
ABG	sbp	0.1761077	0.0730417
ABG	serum_ca	0.2241445	0.0642001
ABG	chf	0.0647444	0.0579903
ABG	copd	0.0487148	0.0540523
ABG	sex	0.0788375	0.0499286
VBG	curr_bmi	0.2728350	0.1834602
VBG	location	0.4053569	0.1482032
VBG	hr	0.1409193	0.0887390
VBG	serum_cl	0.1393215	0.0751048
VBG	race_ethnicity	0.1945666	0.0679005
VBG	dm	0.0665887	0.0505101
VBG	ckd	0.0627237	0.0500826
VBG	sbp	0.1303943	0.0495423
VBG	chf	0.0508431	0.0469251
VBG	dbp	0.0871613	0.0449695

Warning: Target balance: max |SMD| > 0.10 in at least one imputation.

Chunk diagnostics-balance runtime: 33.35 s

4.10.8 Outcome diagnostics

Chunk diagnostics-outcome runtime: 0.24 s

4.10.9 Diagnostics summary and audit

Table 16: Diagnostics summary (IPSW + MI)

block	pi-block	run_mod	max	lot_size	fractile	top_methode	ps_floor_quan-	ps_floor_weight	weight_w1	weight_w2	weight_w3	weight_w4	weight_w5	sum_maxes	top1_weight	n_shares	target_max	target_min	target_mean	target_std	time_proj_to_max	run-time
ABG weights	full	80	20	1	smd.max	0.01	0.10390818632851287431754216298916300872432980.0416207.0100031292882	0.1364083	193.5442													
VBG weights	full	80	20	1	smd.max	0.01	0.12418312993847901235703219356269356824966318610.0293445.0100025561108	0.0694772	193.5442													
ABG outcomes	full	80	20	1	smd.max	0.01	0.10390818632851287431754216298916300872432980.0416207.0100031292882	0.1364083	193.5442													
VBG outcomes	full	80	20	1	smd.max	0.01	0.12418312993847901235703219356269356824966318610.0293445.0100025561108	0.0694772	193.5442													

Chunk diagnostics-summary runtime: 0.02 s

4.10.10 Performance / runtime log

Warning: Projected full-run time exceeds 10 hours (193.5h).

Chunk runtime-log-export runtime: 0.01 s

4.11 16) Save, export, and session info

```
saveRDS(  
  list(  
    abg_curves = if (exists("abg_curves")) abg_curves else NULL,  
    vbg_curves = if (exists("vbg_curves")) vbg_curves else NULL,  
    abg_coefs = if (exists("abg_coefs")) abg_coefs else NULL,  
    vbg_coefs = if (exists("vbg_coefs")) vbg_coefs else NULL  
,  
    mi_pooled_file  
)
```

Chunk *mi-save-exports* runtime: 0.01 s

```
sessionInfo()
```

```
R version 4.5.0 (2025-04-11)  
Platform: aarch64-apple-darwin20  
Running under: macOS 26.2  
  
Matrix products: default  
BLAS: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib  
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3.12.1  
  
locale:  
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8  
  
time zone: America/Denver  
tzcode source: internal  
  
attached base packages:  
[1] splines   grid     parallel  stats      graphics  grDevices utils  
[8] datasets  methods  base  
  
other attached packages:  
[1] rlang_1.1.6       progressr_0.18.0    future.apply_1.20.0  
[4] future_1.67.0     mitools_2.4       skimr_2.2.1
```

```
[7] visdat_0.6.0      tidyverse_1.3.1    digest_0.6.37  
[10] ggrepel_0.9.6    shapviz_0.10.2    fastshap_0.1.1  
[13] miceadds_3.18-36 mice_3.18.0      naniar_1.1.0  
[16] ggdag_0.2.13    dagitty_0.3-4    fs_1.6.6  
[19] here_1.0.2      sensitivitymw_2.1 lubridate_1.9.4  
[22] tibble_3.3.0    survey_4.4-8    survival_3.8-3  
[25] Matrix_1.7-4    rms_8.0-0       Hmisc_5.2-3  
[28] patchwork_1.3.2 officer_0.7.0    modelsummary_2.5.0  
[31] scales_1.4.0    labelled_2.15.0 haven_2.5.5  
[34] gt_1.1.0        ggplot2_4.0.0    gbm_2.2.2  
[37] flextable_0.9.10 dplyr_1.1.4    codebookr_0.1.8  
[40] cobalt_4.6.1    broom_1.0.11   WeightIt_1.5.0  
[43] purrr_1.2.0     gtsummary_2.4.0 kableExtra_1.4.0
```

loaded via a namespace (and not attached):

```
[1] polspline_1.1.25      datawizard_1.2.0      polyclip_1.10-7  
[4] rpart_4.1.24          lifecycle_1.0.4      Rdpack_2.6.4  
[7] rprojroot_2.1.1       globals_0.18.0       lattice_0.22-7  
[10] MASS_7.3-65          insight_1.4.2       backports_1.5.0  
[13] magrittr_2.0.4        rmarkdown_2.30      yaml_2.3.10  
[16] zip_2.3.3            askpass_1.2.1      DBI_1.2.3  
[19] minqa_1.2.8          RColorBrewer_1.1-3  multcomp_1.4-28  
[22] ggraph_2.2.2         nnet_7.3-20       TH.data_1.1-4  
[25] tweenr_2.0.3         sandwich_3.1-1    gdtools_0.4.3  
[28] listenv_0.9.1        cards_0.7.0       performance_0.15.1  
[31] MatrixModels_0.5-4   parallelly_1.45.1  svglite_2.2.1  
[34] commonmark_2.0.0     codetools_0.2-20   xml2_1.4.0  
[37] ggforce_0.5.0        tidyselect_1.2.1   shape_1.4.6.1  
[40] farver_2.1.2         effectsize_1.0.1   lme4_1.1-38  
[43] viridis_0.6.5        base64enc_0.1-3   jsonlite_2.0.0  
[46] mitml_0.4-5          tidygraph_1.3.1   Formula_1.2-5  
[49] emmeans_1.11.2-8     iterators_1.0.14  systemfonts_1.2.3  
[52] foreach_1.5.2         tools_4.5.0       ragg_1.5.0  
[55] Rcpp_1.1.0            glue_1.8.0       gridExtra_2.3  
[58] pan_1.9               mgcv_1.9-3       chk_0.10.0  
[61] xfun_0.53            withr_3.0.2      fastmap_1.2.0
```

[64] boot_1.3-32	SparseM_1.84-2	openssl_2.3.3
[67] litedown_0.7	estimability_1.5.1	timechange_0.3.0
[70] R6_2.6.1	textshaping_1.0.3	colorspace_2.1-2
[73] markdown_2.0	utf8_1.2.6	generics_0.1.4
[76] fontLiberation_0.1.0	data.table_1.17.8	graphlayouts_1.2.2
[79] htmlwidgets_1.6.4	parameters_0.28.2	pkgconfig_2.0.3
[82] gtable_0.3.6	lmtest_0.9-40	S7_0.2.0
[85] htmltools_0.5.8.1	fontBitstreamVera_0.1.1	reformulas_0.4.2
[88] knitr_1.50	rstudioapi_0.17.1	uuid_1.2-1
[91] coda_0.19-4.1	checkmate_2.3.3	nlme_3.1-168
[94] curl_7.0.0	nloptr_2.2.1	repr_1.1.7
[97] zoo_1.8-14	cachem_1.1.0	stringr_1.6.0
[100] foreign_0.8-90	pillar_1.11.1	vctrs_0.6.5
[103] jomo_2.7-6	xtable_1.8-4	cluster_2.1.8.1
[106] htmlTable_2.4.3	evaluate_1.0.5	tinytex_0.57
[109] magick_2.9.0	mvtnorm_1.3-3	cli_3.6.5
[112] compiler_4.5.0	crayon_1.5.3	labeling_0.4.3
[115]forcats_1.0.1	stringi_1.8.7	viridisLite_0.4.2
[118] tables_0.9.31	bayestestR_0.17.0	glmnet_4.1-10
[121] V8_7.0.0	quantreg_6.1	fontquiver_0.2.1
[124] hms_1.1.4	rbibutils_2.4	igraph_2.2.1
[127] memoise_2.0.1	xgboost_1.7.11.1	

Chunk mi-session runtime: 0.07 s

Table 10: Missingness by key strata (pre-imputation; top 10 variables; full table saved to /Users/reblocke/Research/abg-vbg-project/Results/missingness-by-strata.csv).

Variable	Stratum	Level	% missing in level	% missing overall
0	vbg_o2sat	86.3	has_abg	86.3
1	vbg_o2sat	86.4	has_abg	86.3
0	vbg_o2sat	99.1	has_vbg	86.3
1	vbg_o2sat	55.1	has_vbg	86.3
0	vbg_o2sat	86.8	imv_proc	86.3
1	vbg_o2sat	82.3	imv_proc	86.3
0	bnp	84.2	has_abg	81.8
1	bnp	77.7	has_abg	81.8
0	bnp	82.7	has_vbg	81.8
1	bnp	79.7	has_vbg	81.8
0	bnp	82.3	imv_proc	81.8
1	bnp	78.2	imv_proc	81.8
0	vbg_co2	71.2	has_abg	71.0
1	vbg_co2	70.6	has_abg	71.0
0	vbg_co2	100.0	has_vbg	71.0
1	vbg_co2	0.0	has_vbg	71.0
0	vbg_co2	72.3	imv_proc	71.0
1	vbg_co2	60.0	imv_proc	71.0
0	spo2	69.7	has_abg	70.6
1	spo2	72.1	has_abg	70.6
0	spo2	71.3	has_vbg	70.6
1	spo2	69.0	has_vbg	70.6
0	spo2	71.3	imv_proc	70.6
1	spo2	64.6	imv_proc	70.6
0	paco2	100.0	has_abg	63.7
1	paco2	0.0	has_abg	63.7
0	paco2	63.8	has_vbg	63.7
1	paco2	63.2	has_vbg	63.7
0	paco2	69.6	imv_proc	63.7
1	paco2	16.9	imv_proc	63.7
0	serum_lac	71.8	has_abg	59.8
1	serum_lac	38.9	has_abg	59.8
0	serum_lac	64.9	has_vbg	59.8
1	serum_lac	47.3	has_vbg	59.8
0	serum_lac	63.0	imv_proc	59.8
1	serum_lac	34.4	imv_proc	59.8
0	curr_bmi	56.2	has_abg	57.0
1	curr_bmi	58.4	has_abg	57.0
0	curr_bmi	52.8	has_vbg	57.0
1	curr_bmi	67.3	has_vbg	57.0
0	curr_bmi	56.0	imv_proc	57.0
1	curr_bmi	64.4	imv_proc	57.0
0	serum_phos	64.9	has_abg	53.0
1	serum_phos	32.2	has_abg	53.0
0	serum_phos	58.3	349 has_vbg	53.0

Table 11: Predictors of missingness (logit OR; top 50 by p-value; full table saved to /Users/reblocke/Research/abg-vbg-project/Results/missingness-drivers.csv).

Target	Predictor	OR	LCL	UCL	p
temp_new	encounter_typeInpatient	0.63	0.61	0.64	0
temp_new	location1	18.13	17.55	18.72	0
temp_new	location2	4.05	3.90	4.21	0
temp_new	location3	22.32	21.71	22.95	0
temp_new	has_abg	0.52	0.50	0.53	0
sbp	encounter_typeInpatient	0.04	0.04	0.05	0
sbp	location3	2728.33	2283.90	3259.24	0
dbp	encounter_typeInpatient	0.05	0.04	0.05	0
dbp	location3	1223.50	1080.16	1385.87	0
hr	location1	12.03	11.65	12.41	0
hr	location2	4.53	4.36	4.70	0
hr	has_abg	0.55	0.53	0.56	0
sodium	age_at_encounter	0.97	0.97	0.97	0
sodium	has_abg	0.13	0.12	0.14	0
sodium	has_vbg	0.09	0.08	0.10	0
serum_cr	age_at_encounter	0.97	0.97	0.97	0
serum_cr	has_abg	0.21	0.20	0.22	0
serum_cr	has_vbg	0.21	0.19	0.23	0
serum_hco3	age_at_encounter	0.97	0.97	0.97	0
serum_hco3	has_abg	0.12	0.11	0.13	0
serum_hco3	has_vbg	0.17	0.16	0.19	0
serum_cl	age_at_encounter	0.97	0.97	0.97	0
serum_cl	has_abg	0.16	0.15	0.17	0
serum_cl	has_vbg	0.11	0.10	0.12	0
serum_lac	location3	3.22	3.14	3.30	0
serum_lac	has_abg	0.24	0.23	0.24	0
serum_lac	has_vbg	0.46	0.45	0.48	0
serum_k	age_at_encounter	0.97	0.97	0.97	0
serum_k	has_abg	0.14	0.13	0.15	0
serum_k	has_vbg	0.13	0.11	0.14	0
wbc	age_at_encounter	0.98	0.98	0.98	0
wbc	location1	2.96	2.85	3.07	0
wbc	location2	3.35	3.21	3.50	0
wbc	location3	3.11	3.02	3.21	0
wbc	has_vbg	0.49	0.47	0.51	0
plt	age_at_encounter	0.98	0.98	0.98	0
plt	encounter_typeInpatient	0.43	0.41	0.45	0
plt	has_abg	0.25	0.24	0.27	0
plt	has_vbg	0.17	0.16	0.18	0
serum_phos	encounter_typeInpatient	0.13	0.13	0.13	0
serum_phos	location3	0.50	0.49	0.51	0
serum_phos	has_abg	0.34	0.33	0.35	0
serum_phos	has_vbg	0.40	0.39	0.41	0
serum_phos	imv_proc	0.37	0.36	0.39	0
serum_ca	age_at_encounter	350	0.97	0.97	0

Monte Carlo error vs SE (should be much smaller)

Term	Estimate	SE	MC error	MC error / SE	2.5%	97.5%
(Intercept)	-3.339	0.182	—	—	-3.696618386	-2.982280295
has_abg	2.162	0.053	—	—	2.058162757	2.266461350
age_at_encounter	-0.007	0.001	—	—	-0.009861604	-0.004866263
curr_bmi	-0.020	0.004	—	—	-0.028644205	-0.011154982
sexMale	0.314	0.043	—	—	0.228543353	0.398609672
encounter_typeInpatient	1.100	0.068	—	—	0.967084587	1.233849604

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

Cohort	Outcome	Low vs normal OR (95% CI)	High vs normal OR (95% CI)
ABG	IMV	1.20 (1.06, 1.36)	1.15 (1.01, 1.31)
ABG	NIV	0.95 (0.76, 1.18)	2.13 (1.77, 2.56)
ABG	Death (60d)	1.72 (1.49, 1.99)	1.44 (1.24, 1.67)
ABG	Hypercapnic RF	0.80 (0.62, 1.04)	5.48 (4.51, 6.66)
VBG	IMV	1.29 (1.08, 1.54)	1.59 (1.33, 1.90)
VBG	NIV	1.36 (1.01, 1.84)	3.31 (2.56, 4.28)
VBG	Death (60d)	2.02 (1.69, 2.42)	1.69 (1.40, 2.04)
VBG	Hypercapnic RF	1.09 (0.78, 1.52)	8.63 (6.68, 11.16)

Weighted survey GLMs; weights = MI-specific GBM IPW; m = 80 imputations (seed 20251206); reference = Eucapnia.