

ABG-VBG Analysis

Brian Locke, Anila Mehta

Table of contents

1 Data Pre-processing	1
1.1 1) Seed escrow (reproducibility anchors)	1
1.2 2) Baseline tables	9
1.2.1 2.1 Table 1A and 1B:	9
1.2.2 2.2 Table 1 (Overall ABG/VBG status)	12
1.2.3 2.3 Table 2 (Hypercapnia within cohorts)	16
1.2.4 2.4 Generating Word Docs for Table 1 and Table 2 (separate files)	19
2 Unweighted Binary Logistic Regressions	19
2.0.1 3.1 ABG: Binary hypercapnia models	19
2.0.2 3.2 VBG: Binary hypercapnia models	30
2.0.3 3.3 Display model coefficients for binary hypercapnia on VBG logistic regression	38
2.1 3) Three-level PCO ₂ categories (unweighted)	39
2.2 4) Restricted cubic spline regressions (unweighted)	53
2.2.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO ₂	53
2.2.2 4.2 Unweighted, Restricted Cubic Spline - VBG	60
3 Inverse Propensity Weighting	68
3.0.1 5.1 ABG IPW weighting and diagnostics	68
3.0.2 5.2 ABG IPW spline models	77
3.0.3 5.3 ABG IPW spline models (2–98th percentile)	87
3.0.4 5.4 VBG IPW weighting and spline models	91
3.1 5) Weighted effect estimates	102
3.1.1 5.5 Three-level PCO ₂ categories (weighted; ABG, VBG)	114

3.1.2	5.6 Three-level PCO2 categories (weighted; ABG vs VBG only)	127
3.2	6) Propensity score diagnostics	131
4	Multiple Imputation Analysis	136
4.1	7) Packages and reproducibility	136
4.1.1	Missing data / imputation summary (pre-imputation)	138
4.1.2	7.1 Missingness audit (what, where, how much)	138
4.1.3	7.2 Missingness structure and drivers	144
4.1.4	7.3 Monte Carlo error check after MI	163
4.2	8) Pre-imputation data prep (consistent types & predictors)	164
4.3	9) Imputation model specification (MICE)	167
4.3.1	9.1 Predictor matrix & methods. Run MICE (moderate settings for scale)	167
4.3.2	9.2 Convergence & plausibility checks	172
4.3.3	9.3 Observed vs imputed distributions (by strata)	181
4.4	10) Refit propensity models within each imputation	211
4.4.1	10.1 ABG propensity (has_abg)	211
4.4.2	10.2 Balance diagnostics across imputations	212
4.4.3	10.3 VBG propensity (has_vbg)	217
4.4.4	10.4 VBG balance	218
4.5	11) Weighted outcome models within each imputation + pooling	222
4.5.1	11.1 Helper: fit + extract log-OR and SE from svyglm	222
4.5.2	11.2 ABG: outcomes = IMV, NIV, Death(60d), Hypercapnic RF	224
4.5.3	11.3 Repeat for VBG	230
4.6	12) Explainability on one representative imputation	236
4.7	13) MI three-level PCO2 helpers and checks	248
4.8	14) MI + IPW three-level PCO2 (ABG & VBG)	251
4.8.1	14.1 ABG: MI + IPW, three-level PCO2 outcomes	251
4.8.2	14.2 VBG: MI + IPW, three-level PCO2 outcomes	257
4.8.3	14.3 Table 3: MI-pooled IPW associations (3-level CO)	263
4.8.4	14.3 Visualization: pooled three-level ORs	264
4.9	15) Imputed, weighted spline PCO2 (ABG & VBG)	269
4.9.1	15.1 ABG, imputed, weighted, spline outcome	269
4.9.2	15.2 VBG, imputed, weighted, spline outcome	270
4.9.3	15.3 Visualization	271
4.10	16) Save, export, and session info	278

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.

1.1 1) Seed escrow (reproducibility anchors)

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

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

Chunk seed-escrow runtime: 0.00 s

Chunk gt-pdf-helper runtime: 0.00 s

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

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

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

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

```
Directory 'data' already exists.
```

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

```
Loaded existing dataset from 'full_trinetx.rdata'.
```

Chunk load-trinetx-data runtime: 7.40 s

```
covars_gbm <- c(  
  "age_at_encounter", "sex", "race_ethnicity", "curr_bmi",  
  "copd", "asthma", "osa", "chf", "acute_nmd", "phtn", "ckd", "dm",  
  "location", "encounter_type", "temp_new", "sbp", "dbp", "hr", "spo2",  
  "sodium", "serum_cr", "serum_hco3", "serum_cl", "serum_lac", "serum_k",  
  "wbc", "plt", "bnp", "serum_phos", "serum_ca"  
)  
  
gbm_params <- list(
```

```

n.trees          = 1500,
interaction.depth = 3,
shrinkage        = 0.01,
bag.fraction     = 0.8,
cv.folds         = 5,
stop.method      = "es.mean",
n.cores          = parallel::detectCores()
)

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

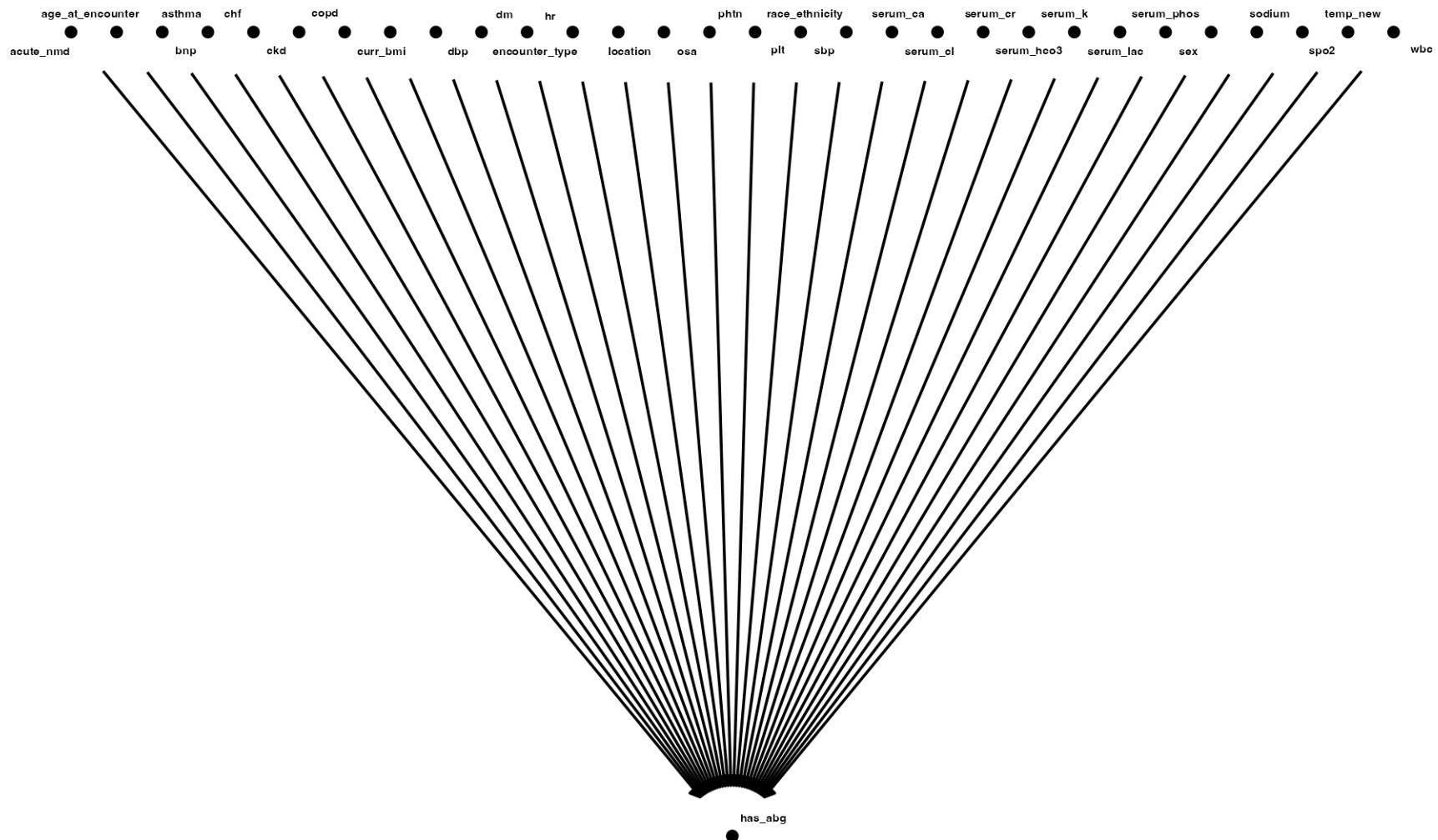
# Model diagrams: propensity models (GBM PS)
register_model_diagram("PS model: ABG test (GBM)", formula_abg, width = 10, height = 7)

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

PS model: ABG test (GBM)

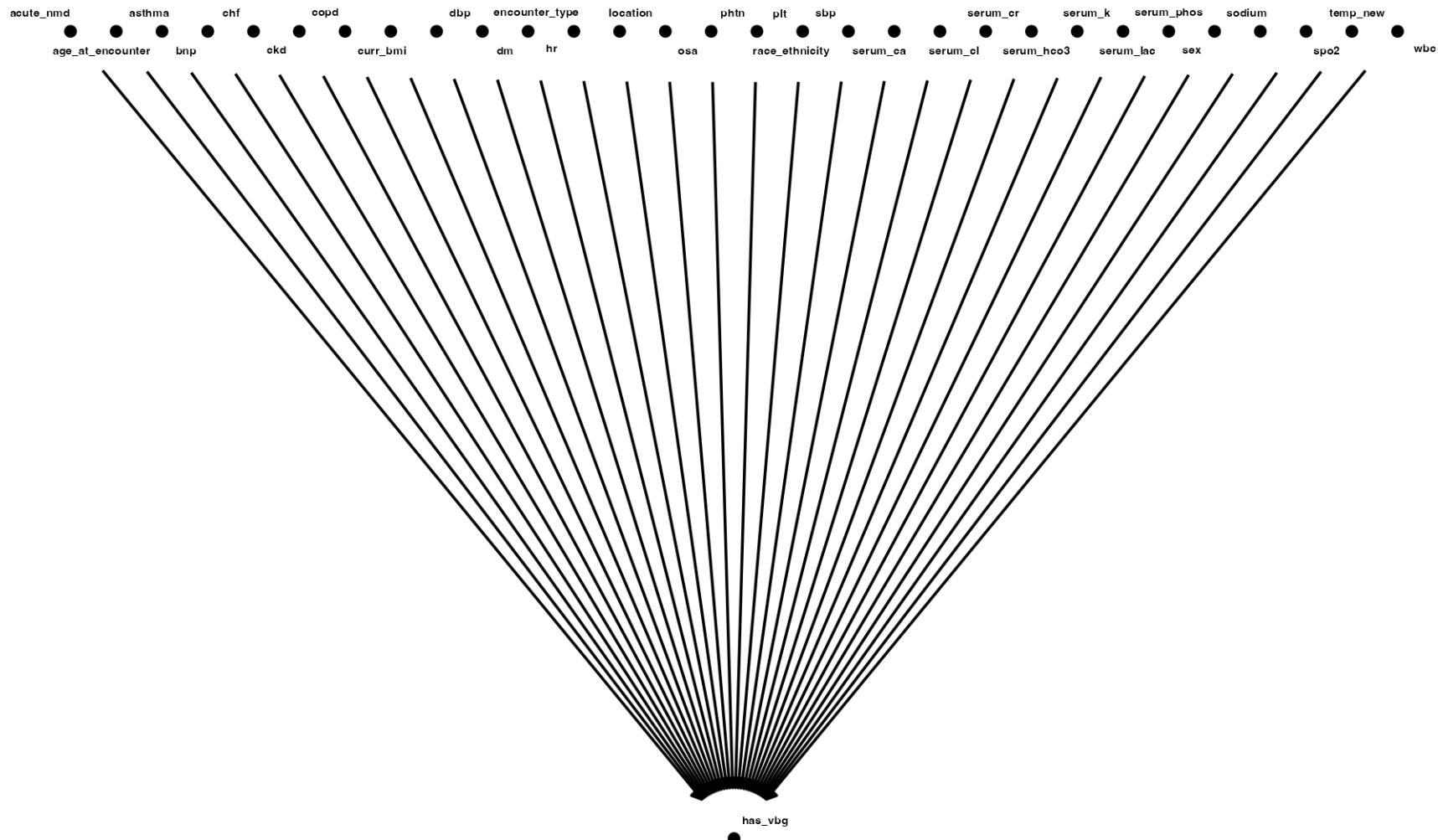


```
register_model_diagram("PS model: VBG test (GBM)", formula_vbg, width = 10, height = 7)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`

Ignoring unknown parameters: `colour`

PS model: VBG test (GBM)



Chunk propensity-config runtime: 1.63 s

Creating subset_data

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

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

table(subset_data$encounter_type)
```

```
2      3
171727 343559
```

```
dim(subset_data)
```

```
[1] 515286    546
```

Chunk sample-subset-data runtime: 5.66 s

Generating Codebook for the Full Dataset

Chunk codebook-export-full runtime: 92.94 s

New Variable - Death at 60 days

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

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

```

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

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

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

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

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

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

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

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

```

Chunk derive-death-60d runtime: 1.60 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

1.2 2) Baseline tables

1.2.1 2.1 Table 1A and 1B:

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

```

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

  ## derive ABG / VBG hypercapnia groups
  abg_group
  = case_when(
    has_abg == 0                  ~ "No ABG",
    has_abg == 1 & hypercap_on_abg == 0 ~ "ABG_NoHypercapnia",
    has_abg == 1 & hypercap_on_abg == 1 ~ "ABG_Hypercapnia",
    TRUE                          ~ "Missing"
  ),
  vbg_group = case_when(
    has_vbg == 0                  ~ "No VBG",
    has_vbg == 1 & hypercap_on_vbg == 0 ~ "VBG_NoHypercapnia",
    has_vbg == 1 & hypercap_on_vbg == 1 ~ "VBG_Hypercapnia",
    TRUE                          ~ "Missing"
  ),
  ## factorise groups with explicit NA/Missing level
  abg_group = factor(
    abg_group,
    levels = c("No ABG", "ABG_NoHypercapnia", "ABG_Hypercapnia", "Missing")
  ),
  vbg_group = factor(
    vbg_group,
    levels = c("No VBG", "VBG_NoHypercapnia", "VBG_Hypercapnia", "Missing")
  ),
  ## labelled covariates
  sex_label      = factor(sex, levels = c(0, 1), labels = c("Female", "Male")),
  race_ethnicity_label = factor(
    race_ethnicity,
    levels = c(0, 1, 2, 3, 4, 5, 6),
    labels = c("White", "Black or African American", "Hispanic"),

```

```

        "Asian", "American Indian", "Pacific Islander", "Unknown")
), location_label      = factor(
  location,
  levels = c(0, 1, 2, 3),
  labels = c("South", "Northeast", "Midwest", "West")
), encounter_type_label = factor(
  encounter_type,
  levels = c(2, 3),
  labels = c("Emergency", "Inpatient")
),
osa_label      = bin_lab(osa),
asthma_label   = bin_lab(asthma),
copd_label     = bin_lab(copd),
chf_label      = bin_lab(chf),
nmd_label      = bin_lab(nmd),
phtn_label     = bin_lab(phtn),
ckd_label      = bin_lab(ckd),
diabetes_label = bin_lab(dm)
)

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

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

```

```

    !!group_sym != "Missing") %>%          # drop "Missing" cohort
  droplevels() %>%                         # trim empty factor levels
  select(all_of(c(group_var, vars_baseline))) %>%
  gtsummary::tbl_summary(
    by    = !!group_sym,
    type = list(sex_label ~ "categorical"),
    statistic = list(
      gtsummary::all_continuous() ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
      gtsummary::all_categorical() ~ "{n} ({p}%)"
    ),
    digits  = list(gtsummary::all_continuous() ~ 1),
    missing  = "no"                           # no gtsummary missing column/row
  ) %>%
  gtsummary::modify_header(label = "**Variable**") %>%
  gtsummary::modify_caption(caption)
}

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

table1A

```

```
table1B
```

Chunk derive-table1-cohorts runtime: 7.31 s

Generating Word Doc for Table 1A & 1B

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

1.2.2 2.2 Table 1 (Overall ABG/VBG status)

Variable	No ABG N = 328,044¹	ABG_NoHypercapnia N = 129,429¹	ABG_Hypercapnia N = 5
Age (years)	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	62.1 ± 16.4; 0.0/57,813.0 missing (0.0%)
Current BMI kg/m2	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.8 ± 7.9; 33,496.0/57,813.0 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 (years)	59.4 ± 17.8; 0.0/365,623.0 missing (0.0%)	58.1 ± 17.8; 0.0/105,646.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 missing (0.0%)
Current BMI kg/m2	31.8 ± 8.5; 192,892.0/365,623.0 missing (52.8%)	28.7 ± 7.2; 69,615.0/105,646.0 missing (65.9%)	29.3 ± 7.9; 31,038.0/44,017.0 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 (%)

```

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

# ABG table with "Everyone" column first
tbl1_abg <- subset_data %>%
  select(all_of(vars_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: 2.89 s

1.2.3 2.3 Table 2 (Hypercapnia within cohorts)

```

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

# ABG cohort (has_abg == 1)
tbl2_abg <- subset_data %>%
  filter(has_abg == 1) %>%
  select(all_of(vars_abg), hyper_abg) %>%
  gtsummary::tbl_summary(
    by = hyper_abg,
    type = list(sex_label ~ "categorical"),

```

Table 1

Variable	Everyone¹	Did not get ABG N = 328,044¹	Did get ABG N = 18
Age (years)	59.2 ± 17.7; 0.0/515,286.0 missing (0.0%)	58.1 ± 18.1; 0.0/328,044.0 missing (0.0%)	61.2 ± 16.9; 0.0/187,242.0 mi
Current BMI kg/m2	31.1 ± 8.4; 293,545.0/515,286.0 missing (57.0%)	32.3 ± 8.7; 184,223.0/328,044.0 missing (56.2%)	29.0 ± 7.2; 109,322.0/187,242.0 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 (%)

```

statistic = list(
  gtsummary::all_continuous() ~ "{mean} ± {sd}; {N_miss}/{N_obs} missing ({p_miss}%)",
  gtsummary::all_categorical() ~ "{n} ({p}%)"
),
digits = list(gtsummary::all_continuous() ~ 1),
missing = "no"
) %>%
gtsummary::modify_header(
  label = "**Variable**",
  stat_1 = "**Got ABG & Hypercapnia**",
  stat_2 = "**Got ABG & No hypercapnia**"
)

# VBG cohort (has_vbg == 1)
tbl2_vbg <- subset_data %>%
  filter(has_vbg == 1) %>%
  select(all_of(vars_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: 3.46 s

Chunk tbl2-outcome-panel runtime: 1.22 s

1.2.4 2.4 Generating Word Docs for Table 1 and Table 2 (separate files)

Chunk export-table1-tbl2-word runtime: 0.77 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)

```

*Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`*

Table 1

Variable	Got ABG & Hypercapnia ¹	Got ABG & No hypercapnia ¹	Got VBG & Hypercapnia ¹
Age (years)	62.1 ± 16.4; 0.0/57,813.0 missing (0.0%)	60.8 ± 17.1; 0.0/129,429.0 missing (0.0%)	61.0 ± 16.7; 0.0/44,017.0 missing
Current BMI kg/m2	29.8 ± 7.9; 33,496.0/57,813.0 missing (57.9%)	28.6 ± 6.9; 75,826.0/129,429.0 missing (58.6%)	29.3 ± 7.9; 31,038.0/44,017.0 missing
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%)
Arterial PCO2	58.5 ± 20.4; 0.0/57,813.0 missing (0.0%)	35.5 ± 6.1; 0.0/129,429.0 missing (0.0%)	60.2 ± 12.6; 0.0/44,017.0 missing
VBG PCO2			

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.

```
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`
```

ABG binary: IMV ~ hypercapnia

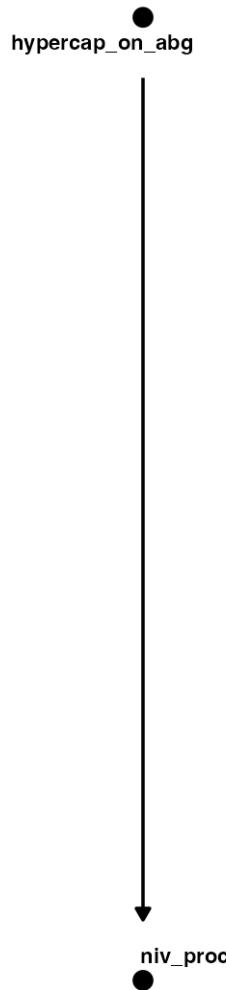
hypercap_on_abg



imv_proc

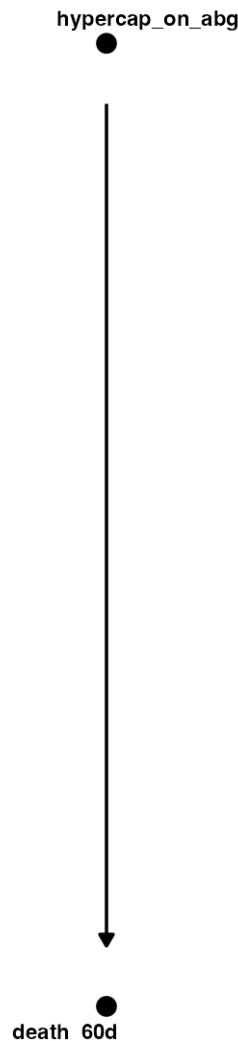
```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG binary: NIV ~ hypercapnia



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

ABG binary: Death60d ~ hypercapnia



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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: 21.21 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: 24.10 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)

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
 arrow_directed, : Ignoring unknown parameters: `colour`

```
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`
```

VBG binary: IMV ~ hypercapnia

hypercap_on_vbg
●



imv_proc
●

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG binary: NIV ~ hypercapnia

hypercap_on_vbg



niv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG binary: Death60d ~ hypercapnia

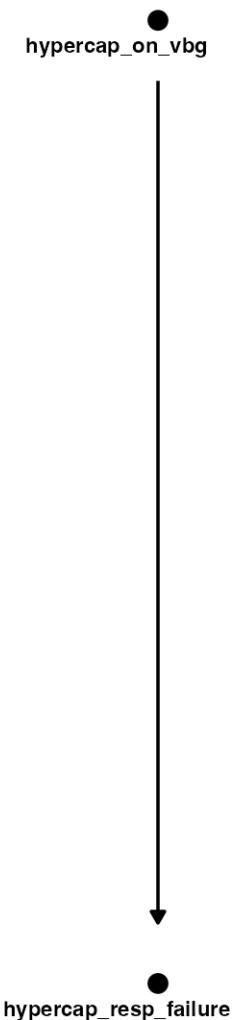
hypercap_on_vbg



death_60d

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG binary: HCRF ~ hypercapnia



```
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: 19.59 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,

```

	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)

```

estimate      = "{estimate}",
statistic     = "({conf.low}, {conf.high})",
coef_omit     = "(Intercept)",
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.

Chunk vbg-binary-or-table runtime: 22.97 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")

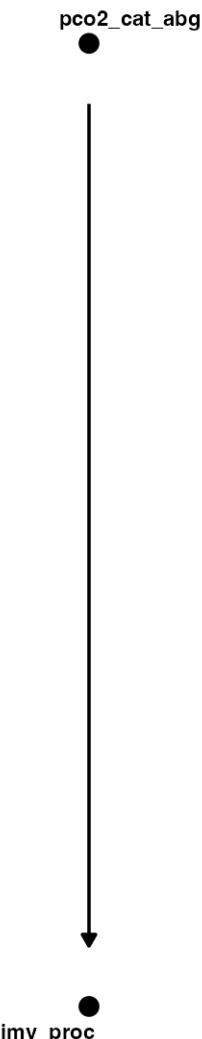
three_level_forms <- list(
  "ABG 3-level: IMV ~ CO2 category"      = imv_proc ~ pco2_cat_abg,
  "ABG 3-level: NIV ~ CO2 category"      = niv_proc ~ pco2_cat_abg,
  "ABG 3-level: Death60d ~ CO2 category" = death_60d ~ pco2_cat_abg,
  "ABG 3-level: HCRF ~ CO2 category"     = hypercap_resp_failure ~ pco2_cat_abg,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

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
 arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
 arrow_bidirected, : Ignoring unknown parameters: `colour`

ABG 3-level: IMV ~ CO2 category



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG 3-level: NIV ~ CO2 category

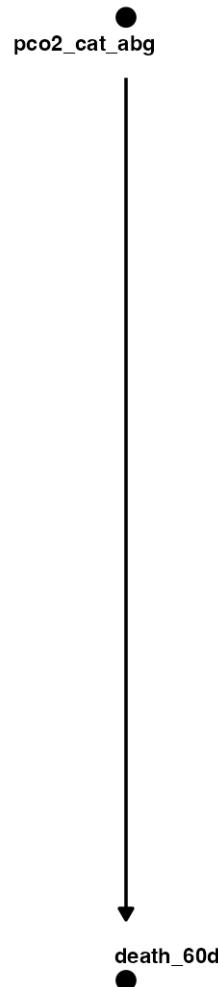
pco2_cat_abg



niv_proc

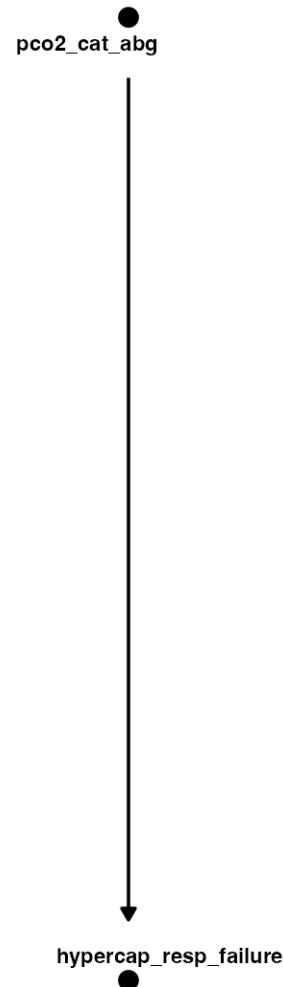
```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG 3-level: Death60d ~ CO2 category



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG 3-level: HCRF ~ CO2 category



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG 3-level: IMV ~ CO2 category

pco2_cat_vbg



imv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG 3-level: NIV ~ CO2 category

pco2_cat_vbg



niv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG 3-level: Death60d ~ CO2 category

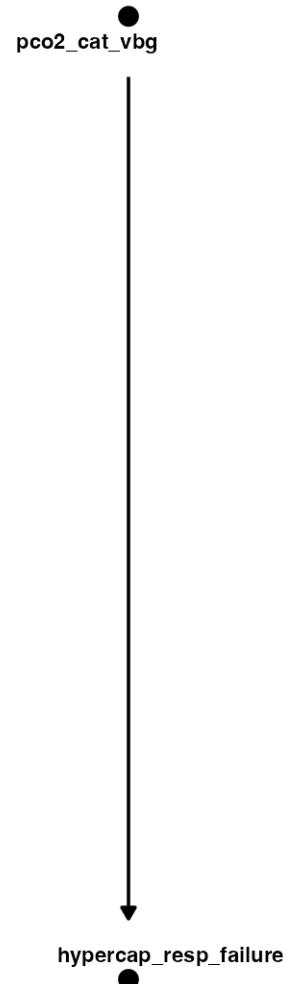
pco2_cat_vbg



death_60d

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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: 23.11 s

```

library(scales)

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

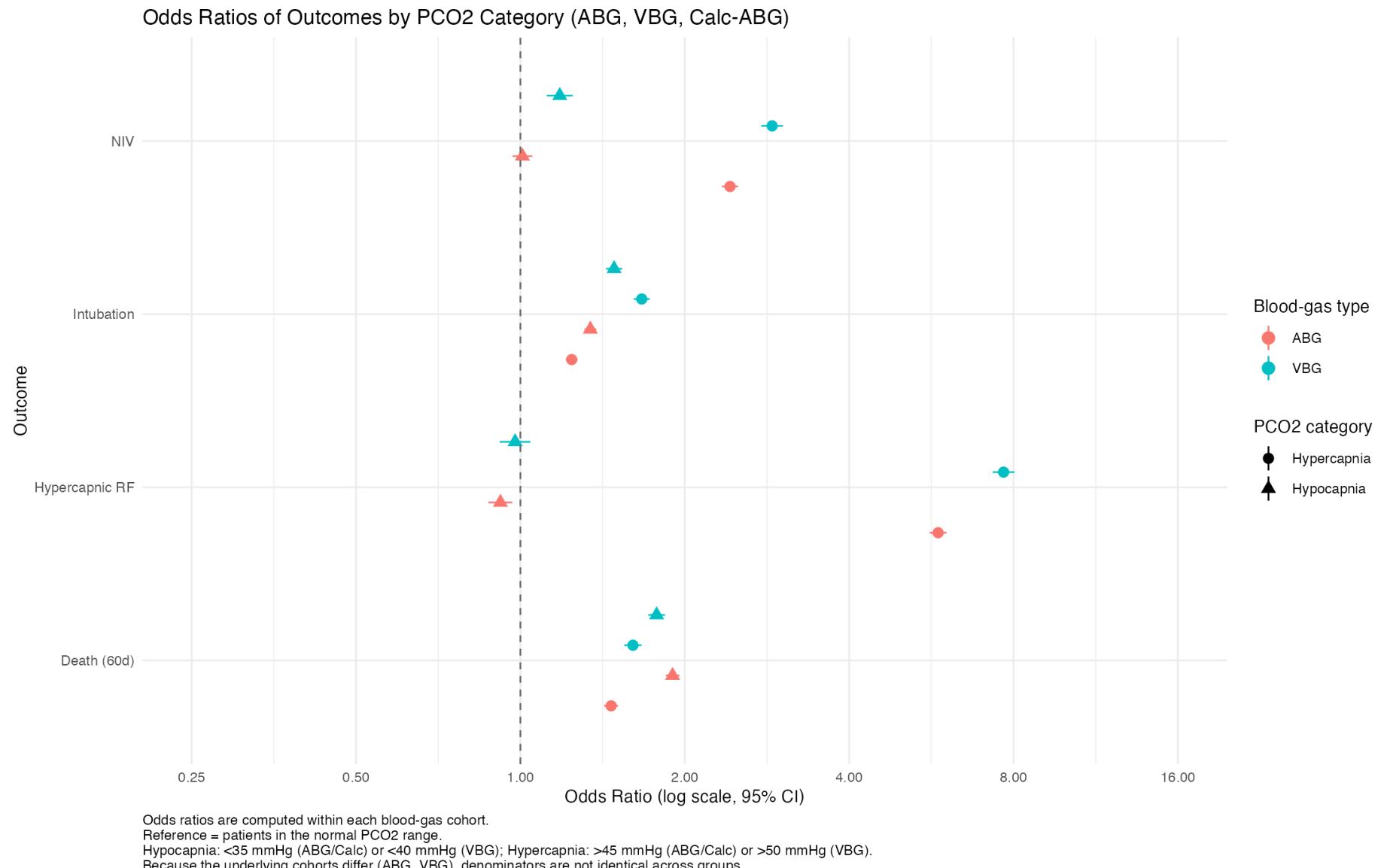
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +

```

```

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: 0.31 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: 0.03 s

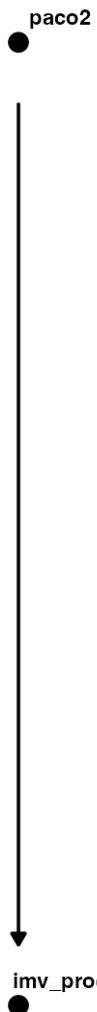
2.2.1 4.1 Unweighted, Restricted Cubic Spline Regression - ABG by PaCO2

```
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)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

ABG RCS: IMV ~ PaCO₂



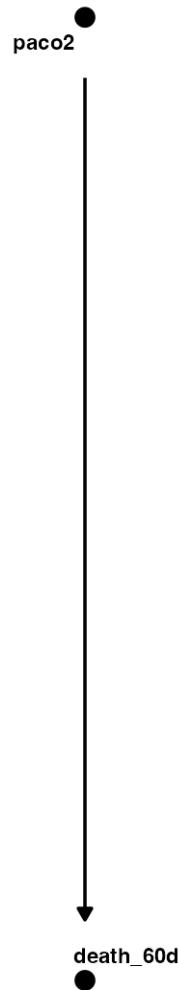
```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG RCS: NIV ~ PaCO₂



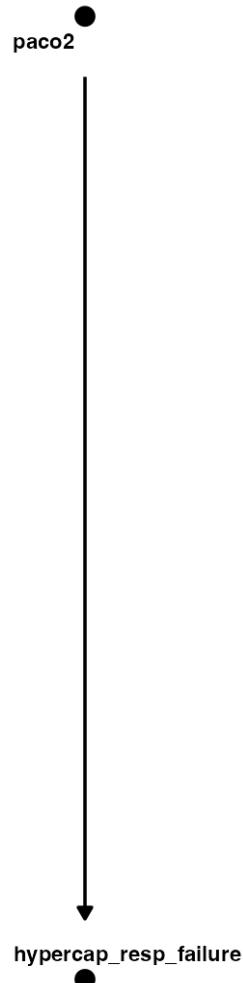
```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG RCS: Death60d ~ PaCO2



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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))

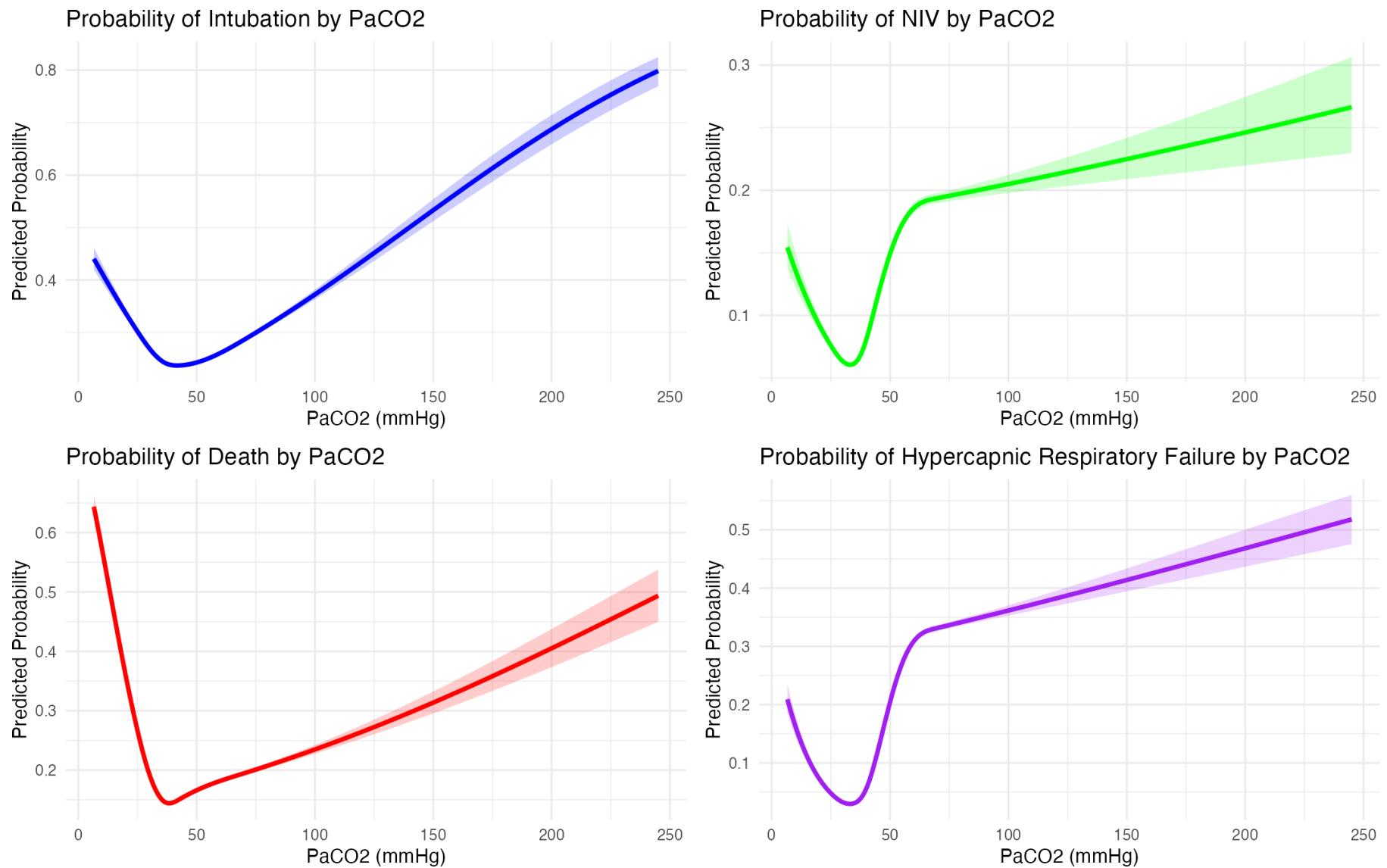
plot_death <- ggplot(pred_death, aes(x = paco2, y = yhat)) +
  geom_line(color = "red", linewidth = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "red", alpha = 0.2) +
  labs(title = "Probability of Death by PaCO2",
       x = "PaCO2 (mmHg)", y = "Predicted Probability") +
  theme_minimal()

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: 2.79 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")
```

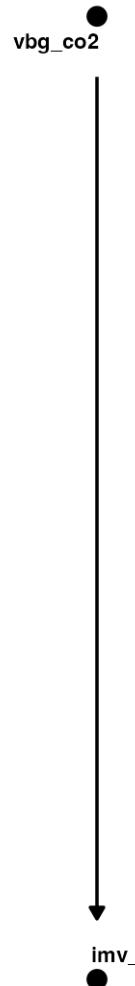
Chunk rcs-vbg-data-prep runtime: 0.03 s

```
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)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

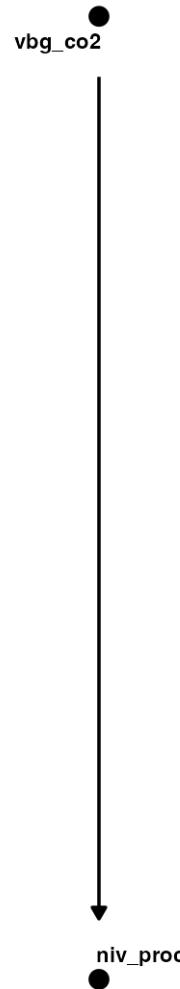
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

VBG RCS: IMV ~ CO2



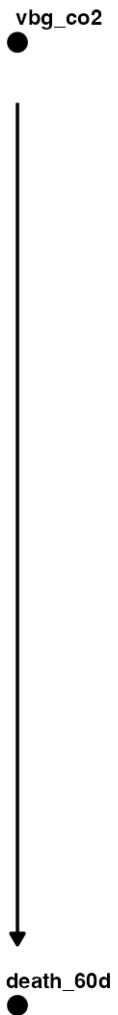
```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG RCS: NIV ~ CO2



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

VBG RCS: Death60d ~ CO2



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

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()

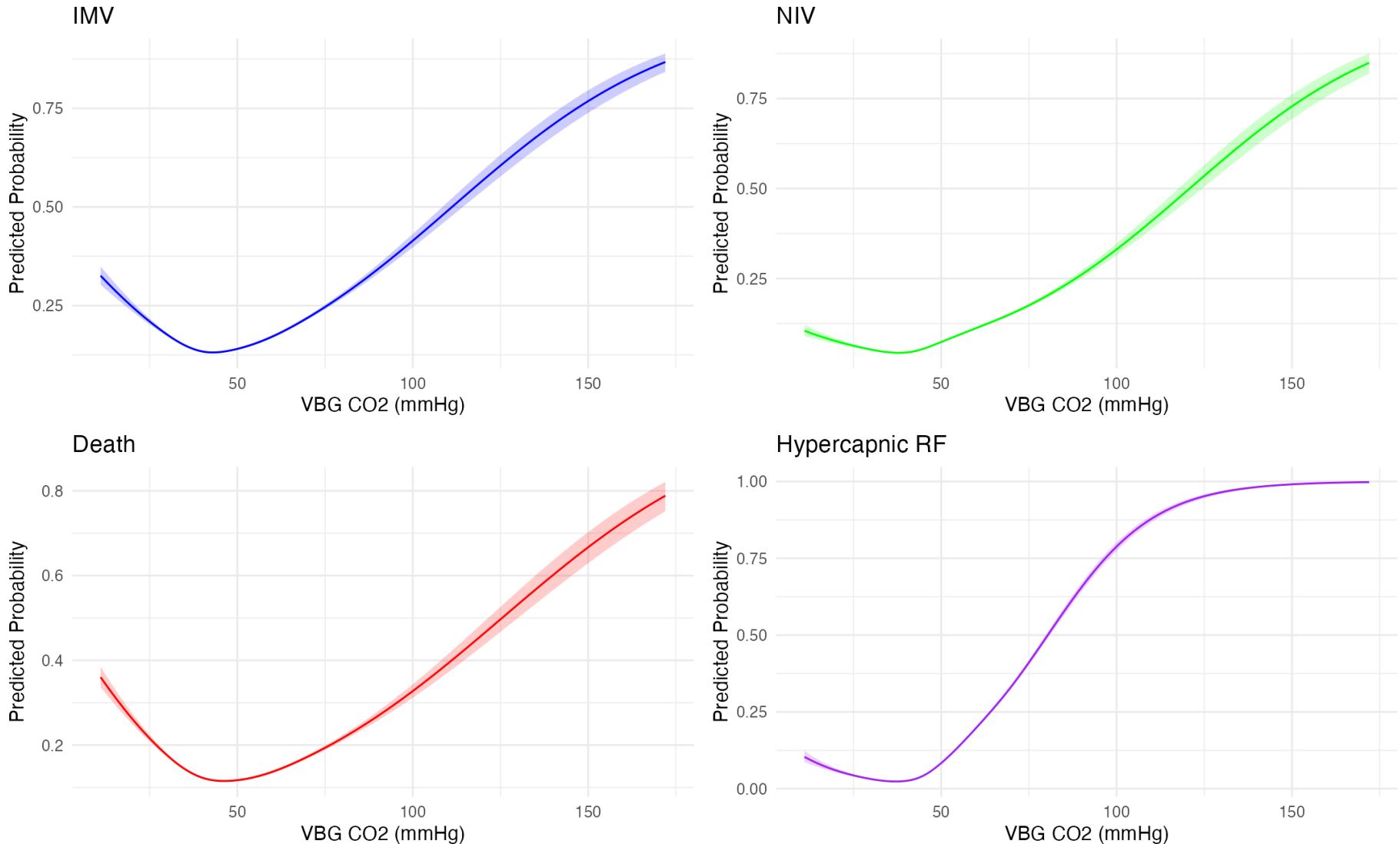
plot_hcrcf_vbg <- ggplot(pred_hcrcf_vbg, aes(x = vbg_co2, y = yhat)) +
  geom_line(color = "purple") +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.2) +
  labs(title = "Hypercapnic RF", x = "VBG CO2 (mmHg)", y = "Predicted Probability") +

```

```
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: 2.55 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 & stopping rule → faster fit, avoids over-fitting, lighter tails (which lead to extreme weights that are problematic).
2. `bal.tab()` documents balance; aim is to adjust spec until standard mean difference (SMD) < 0.1.
3. Weight stabilization (divide by mean) mitigates a few huge weights. I also winsorized, which is a way to avoid very extreme weights (ie you set <1st percentile to the 1st percentile value, and >99th percentile to 99th percentile).
4. Uses robust variance estimation (e.g. allows the variances to change by PaCO2) for IP-weighted GLM; works with splines via `rcs()`. This is a bit nuanced but I think good to change even though it adds complexity
5. Deterministic seed ensures result replication.

3.0.1 5.1 ABG IPW weighting and diagnostics

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

Chunk encode-encounter-type runtime: 0.09 s

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

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

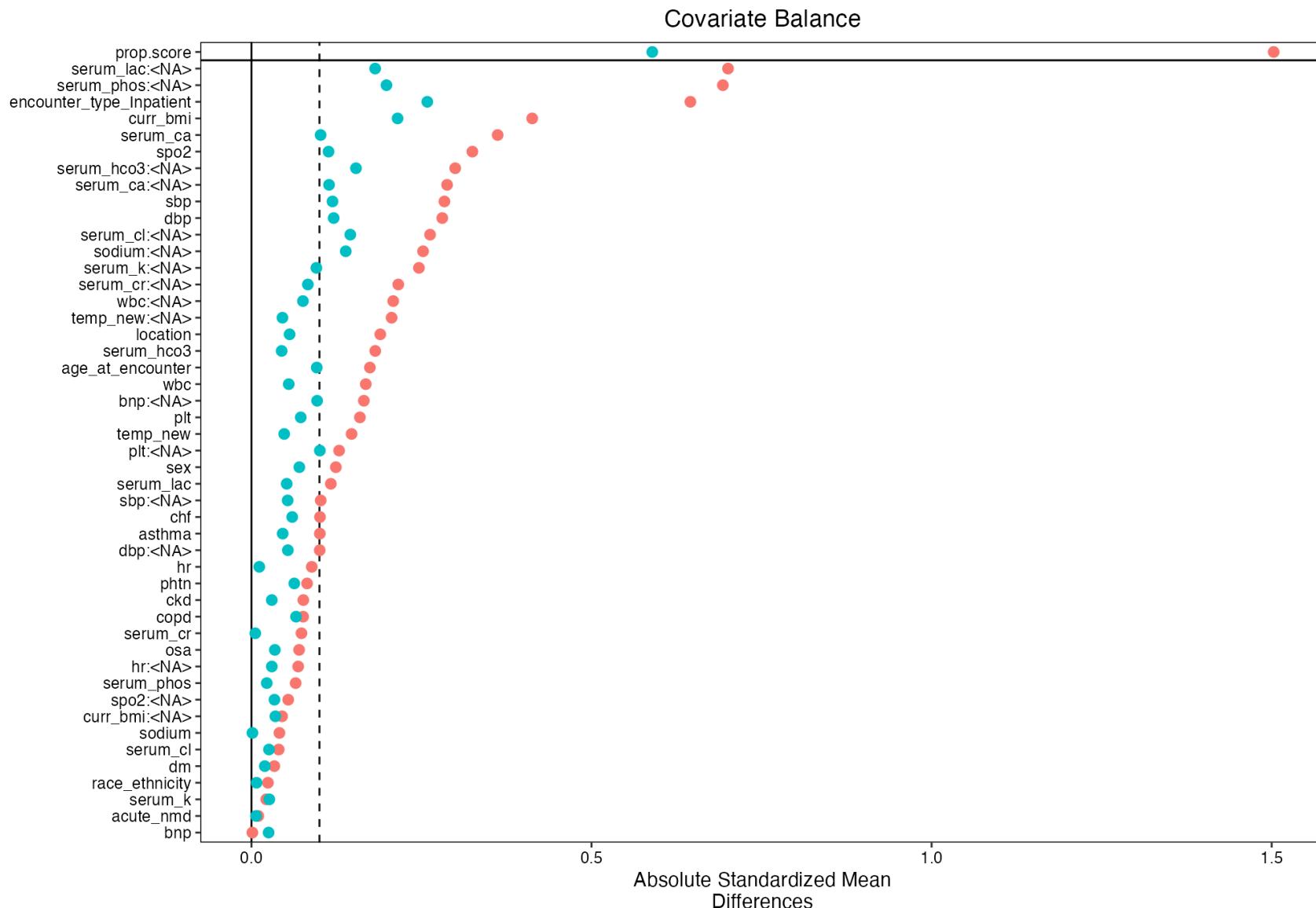
# overwrite inside the object and attach to data
weight_model$weights <- w
subset_data$w_abg    <- w

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

```

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

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



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

# 5. outcome models (examples)
```

```
ipw_abg_bin_forms <- list(  
  "ABG IPW: IMV ~ has_abg"      = imv_proc ~ has_abg,  
  "ABG IPW: NIV ~ has_abg"      = niv_proc ~ has_abg,  
  "ABG IPW: Death60d ~ has_abg" = death_60d ~ has_abg,  
  "ABG IPW: HCRF ~ has_abg"     = hypercap_resp_failure ~ has_abg  
)  
register_model_diagrams(ipw_abg_bin_forms)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

ABG IPW: IMV ~ has_abg

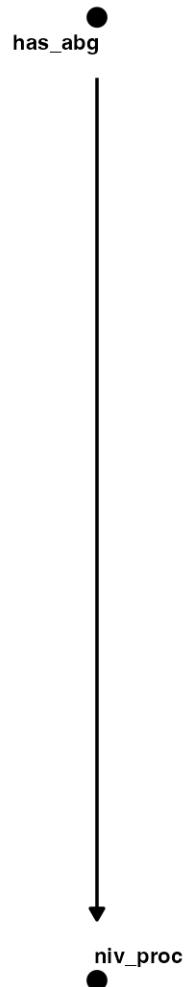
has_abg



imv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

ABG IPW: NIV ~ has_abg



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

ABG IPW: Death60d ~ has_abg

has_abg



death_60d

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG IPW: HCRF ~ has_abg

has_abg



hypercap_resp_failure

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

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

\$IMV
 OR.has_abg LCL UCL
 6.664119 6.497030 6.835505

\$NIV
 OR.has_abg LCL UCL
 1.978913 1.930275 2.028777

\$Death
 OR.has_abg LCL UCL
 2.034938 1.993179 2.077571

\$ICD
 OR.has_abg LCL UCL
 3.367251 3.270841 3.466503

Chunk ipw-abg-weighting runtime: 664.25 s

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

3.0.2 5.2 ABG IPW spline models

```
# set.seed(42) # reproducible GBM fit
#
# # 1. inverse-probability weights for receiving an ABG
#
```

```

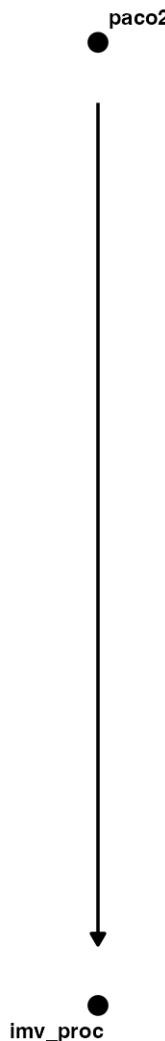
# # done in the last block, so not needed
#
# Model diagrams: IPW ABG spline models
ipw_abg_rcs_forms <- list(
  "ABG IPW RCS: IMV ~ PaCO2"      = imv_proc ~ rcs(paco2, 4),
  "ABG IPW RCS: NIV ~ PaCO2"      = niv_proc ~ rcs(paco2, 4),
  "ABG IPW RCS: Death60d ~ PaCO2" = death_60d ~ rcs(paco2, 4),
  "ABG IPW RCS: HCRF ~ PaCO2"     = hypercap_resp_failure ~ rcs(paco2, 4)
)
register_model_diagrams(ipw_abg_rcs_forms)

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

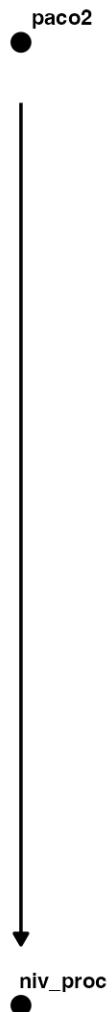
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

ABG IPW RCS: IMV ~ PaCO2



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG IPW RCS: NIV ~ PaCO₂



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

ABG IPW RCS: Death60d ~ PaCO2

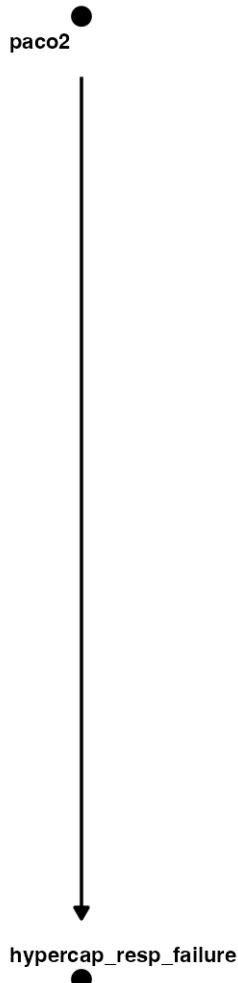
paco2



death_60d

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

ABG IPW RCS: HCRF ~ PaCO2



```
# 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)"))

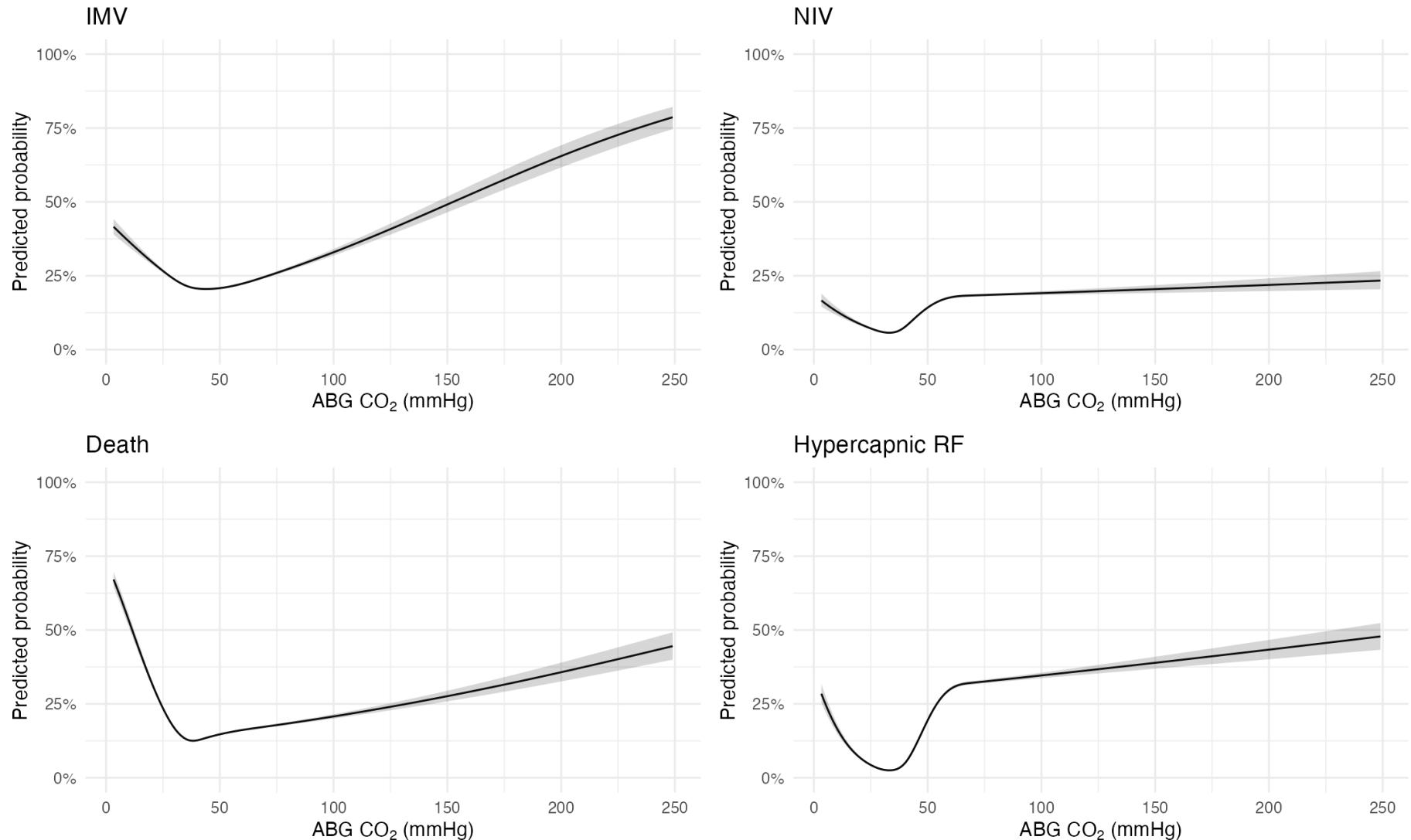
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

(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: 4.93 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)"))

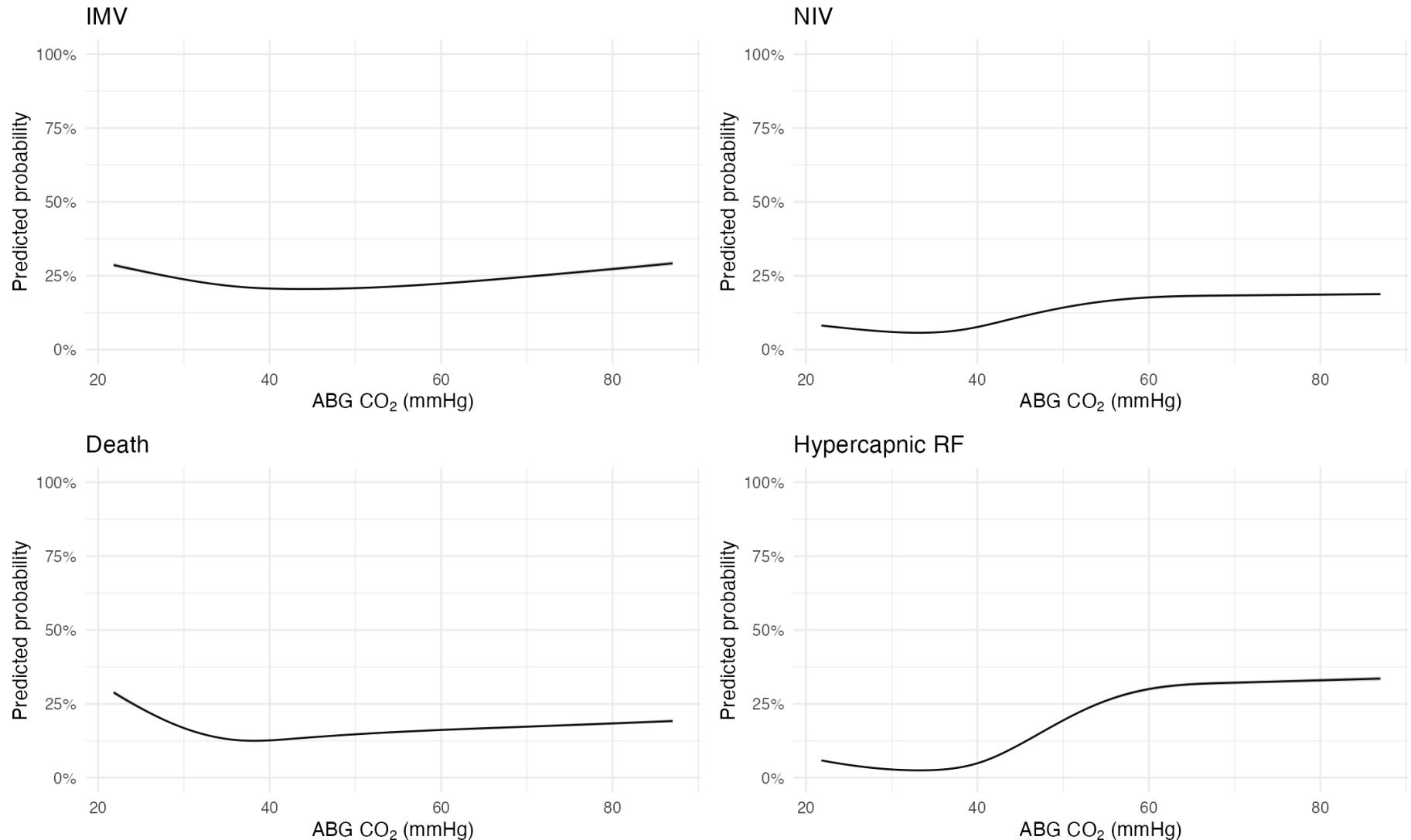
plt <- function(dat, title)
  ggplot(dat, aes(paco2, yhat)) +
    geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    scale_y_continuous(limits = c(0, 1), labels = percent_format(accuracy = 1)) +
    labs(title = title, x = xlab, y = "Predicted probability") +
    theme_minimal()

(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: 4.12 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)

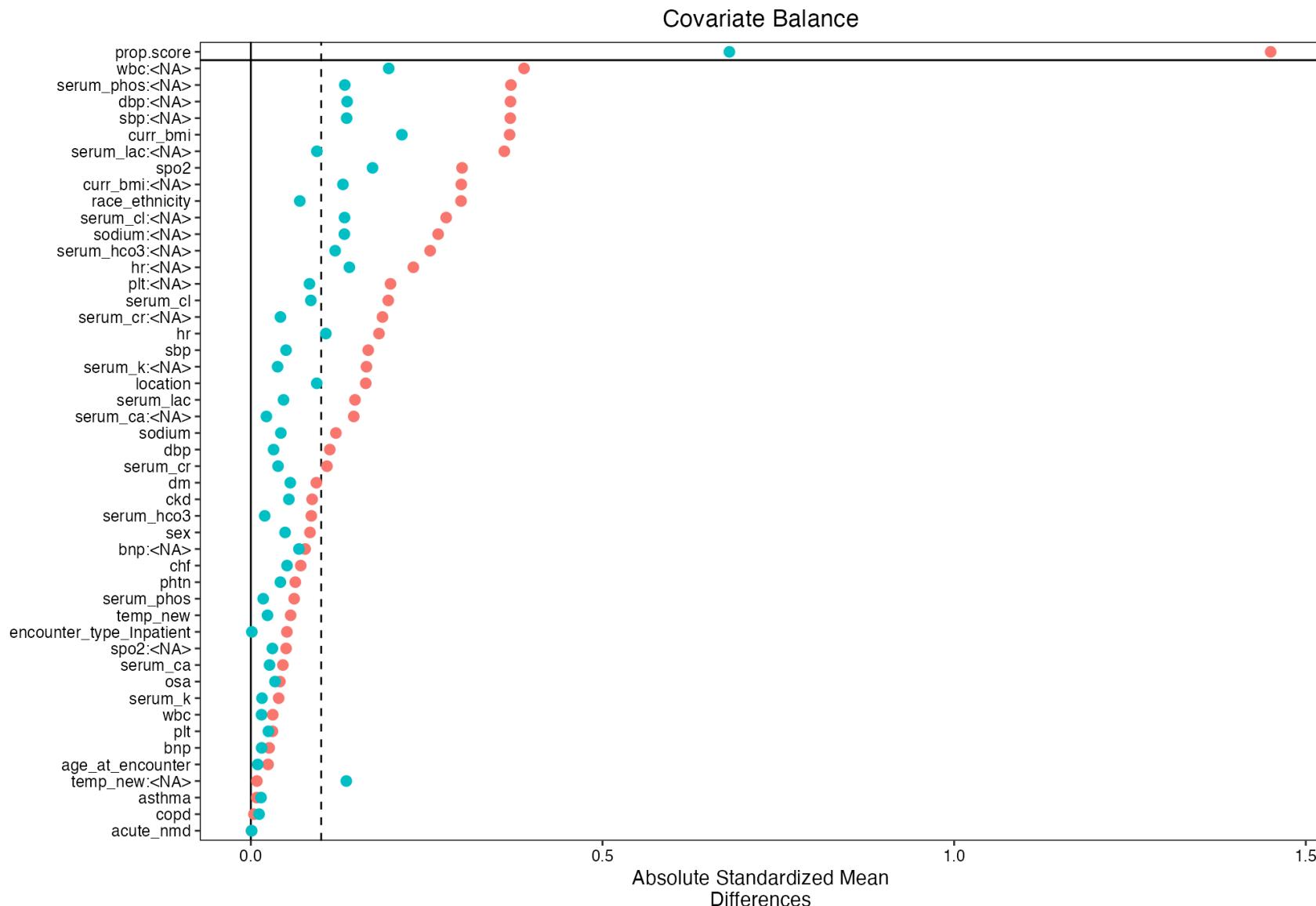
# 1. IPW for VBG -----
set.seed(42)
w_vbg <- do.call(
  weightit,
  c(
    list(
      formula_vbg,
      data      = subset_data,
      method    = "gbm",
      estimand  = "ATE",
      missing   = "ind",
      include.obj = TRUE
    ),
    gbm_params
  )
)

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

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

v_bal <- suppressWarnings(bal.tab(
```

```
w_vbg,  
un = TRUE,  
m.threshold = 0.1,  
binary = "std",  
s.d.denom = "pooled"  
))  
  
love.plot(  
  v_bal,  
  stats      = "m",           # standardized mean differences only  
  abs        = TRUE,  
  var.order   = "unadjusted",  
  sample.names = c("Raw", "IPW")  
)
```



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

```

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

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

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

# 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)

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

VBG IPW RCS: IMV ~ CO2

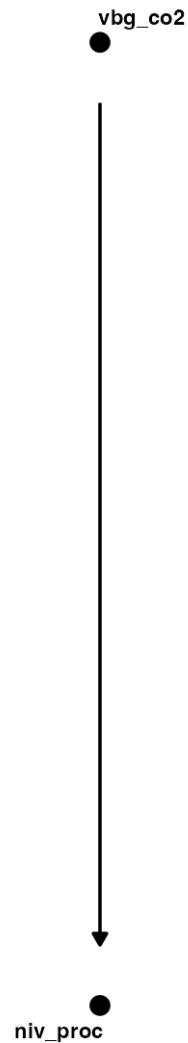
vbg_co2



imv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG IPW RCS: NIV ~ CO2



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG IPW RCS: Death60d ~ CO2

vbg_co2



death_60d

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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)

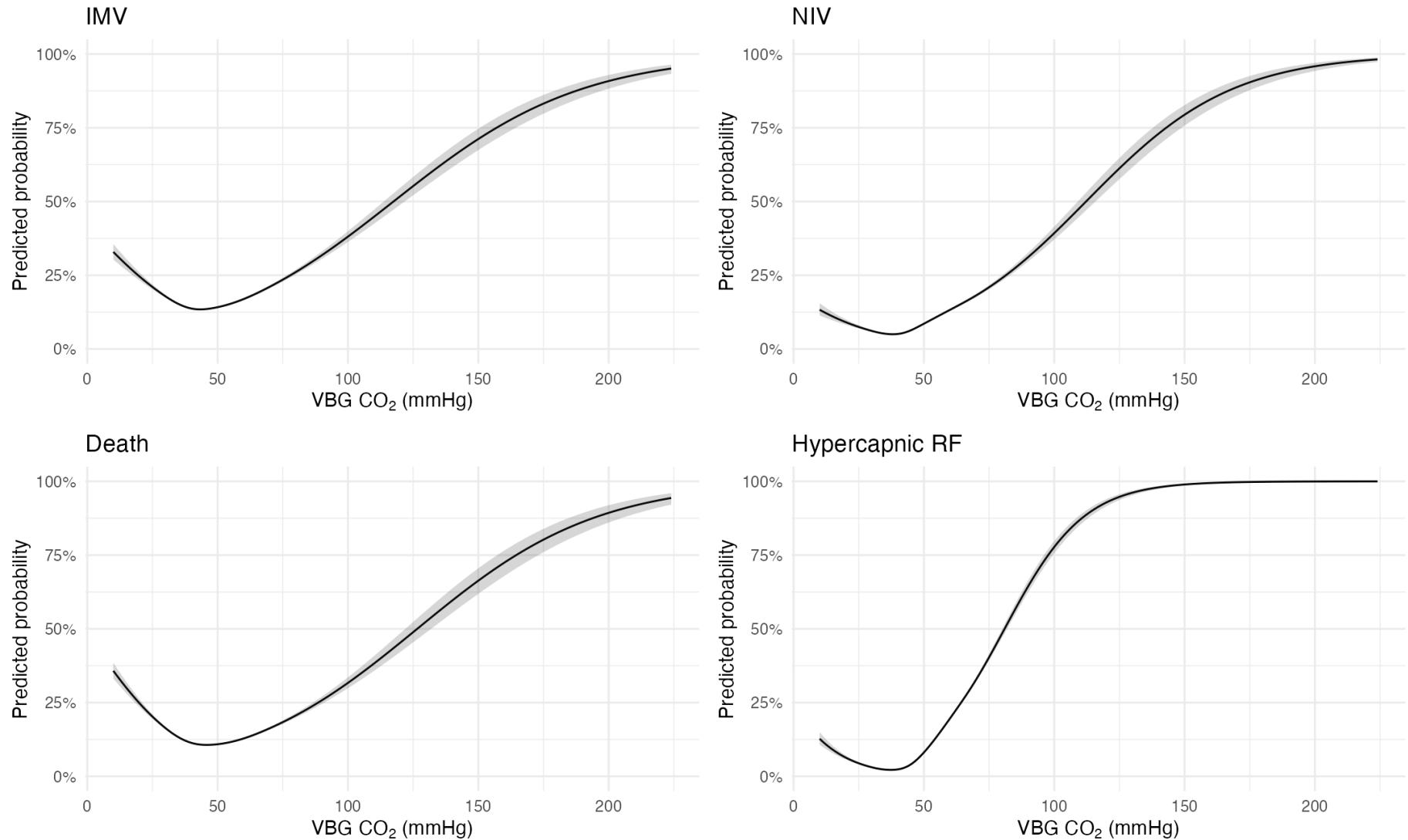
# 5. Plotting (gray scheme) -----
xlab <- expression(paste("VBG CO" [2], " (mmHg)"))

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: 648.18 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)
```

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`
```

```
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`
```

ABG IPW: IMV ~ hypercapnia

hypercap_on_abg



imv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG IPW: NIV ~ hypercapnia

hypercap_on_abg



niv_proc



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG IPW: Death60d ~ hypercapnia

hypercap_on_abg



death_60d

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

ABG IPW: HCRF ~ hypercapnia

hypercap_on_abg



hypercap_resp_failure

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG IPW: IMV ~ hypercapnia

hypercap_on_vbg



imv_proc

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

VBG IPW: NIV ~ hypercapnia

hypercap_on_vbg



niv_proc



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

VBG IPW: Death60d ~ hypercapnia

hypercap_on_vbg



death_60d

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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")
)

```

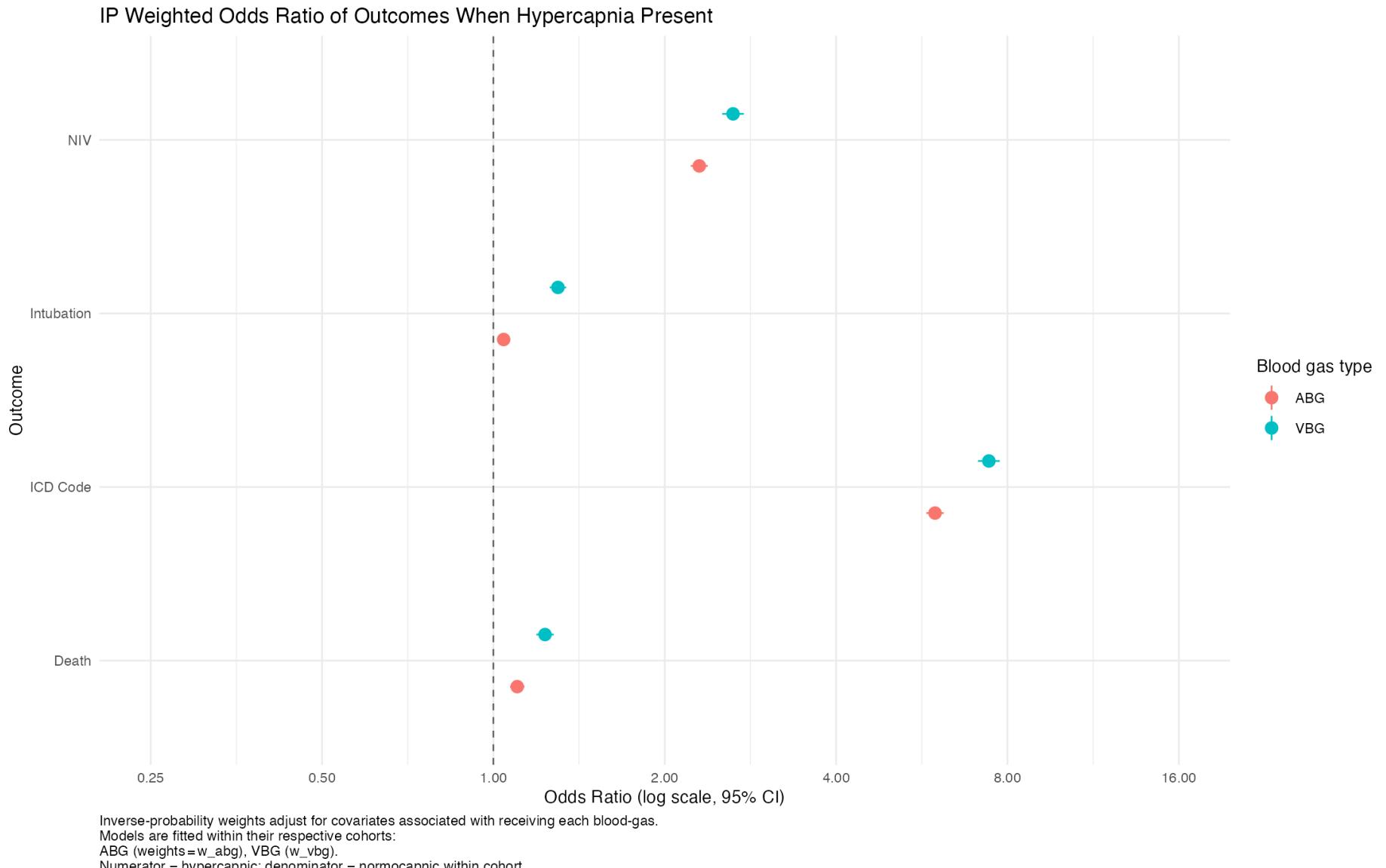
```

# 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))
```



Chunk ipw-binary-or-plot runtime: 15.13 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,

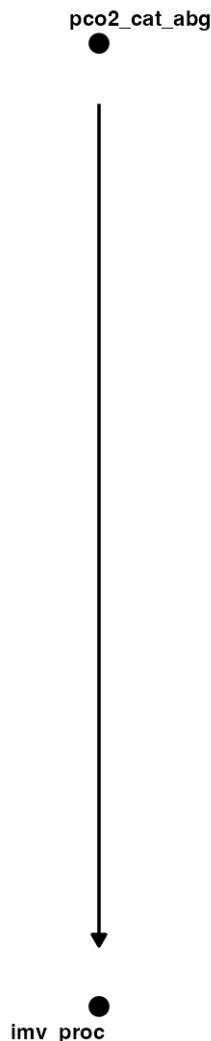
```

```
"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)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

ABG IPW 3-level: IMV ~ CO2 category



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

ABG IPW 3-level: NIV ~ CO2 category

pco2_cat_abg



niv_proc

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

ABG IPW 3-level: Death60d ~ CO2 category

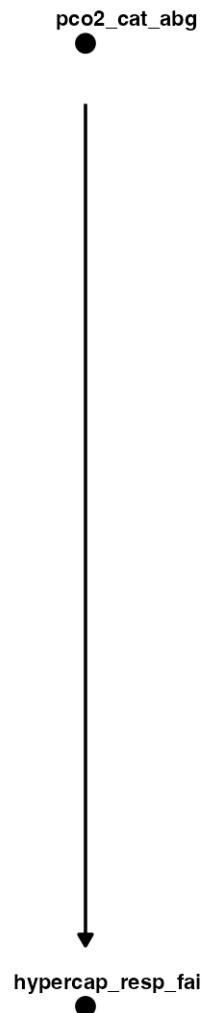
pco2_cat_abg



death_60d

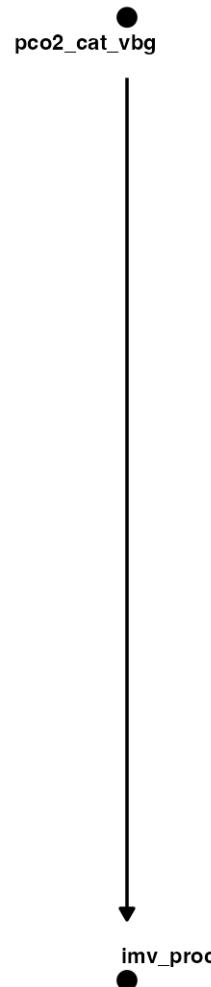
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

ABG IPW 3-level: HCRF ~ CO2 category



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`
```

VBG IPW 3-level: IMV ~ CO2 category



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

VBG IPW 3-level: NIV ~ CO2 category

pco2_cat_vbg



niv_proc

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

VBG IPW 3-level: Death60d ~ CO2 category

pco2_cat_vbg



death_60d

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

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"))

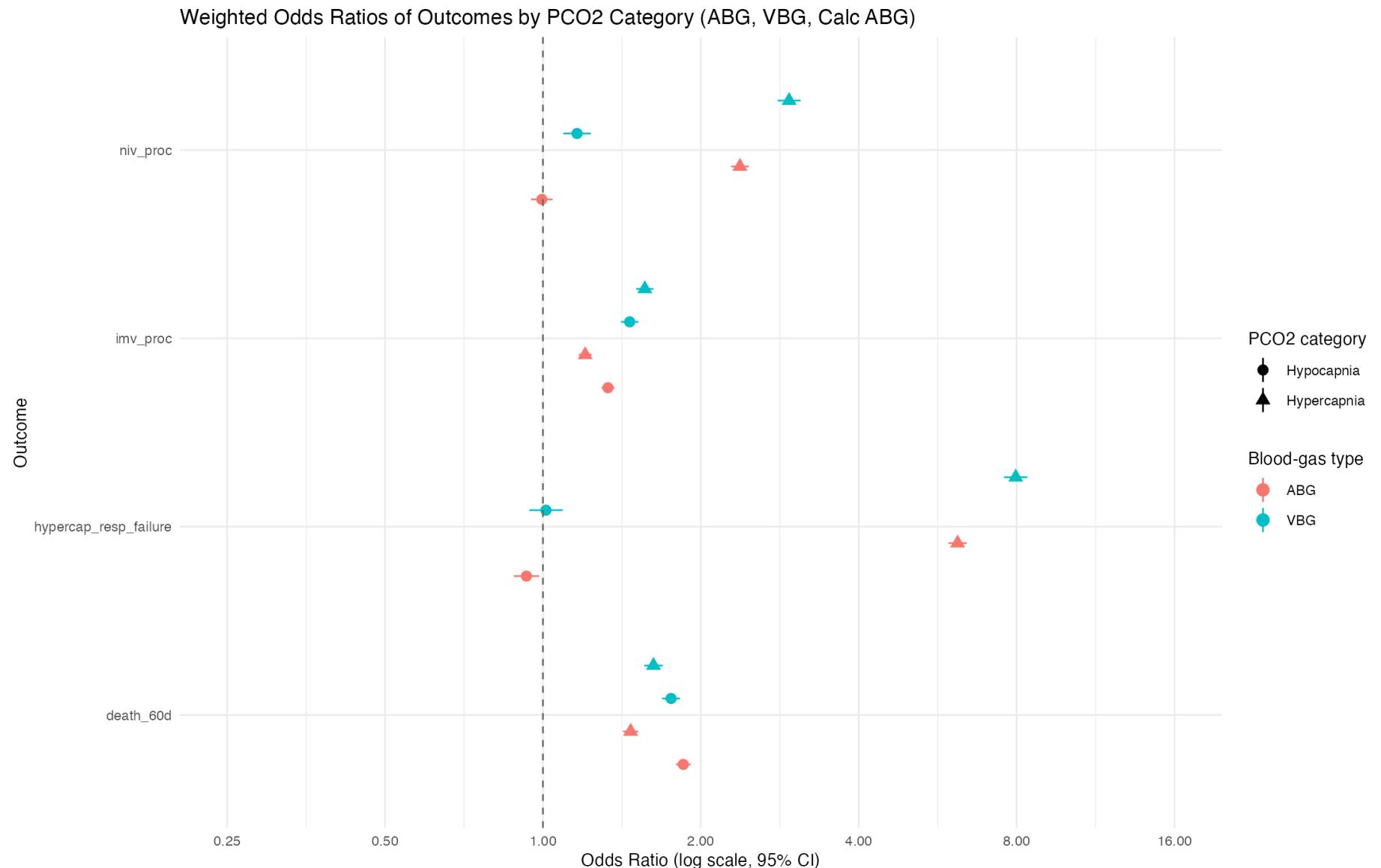
# 4. Plot weighted odds ratios
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure
  )
) +
  geom_pointrange(position = position_dodge(width = 0.7), size = 0.6) +
  geom_hline(yintercept = 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: 24.50 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"))

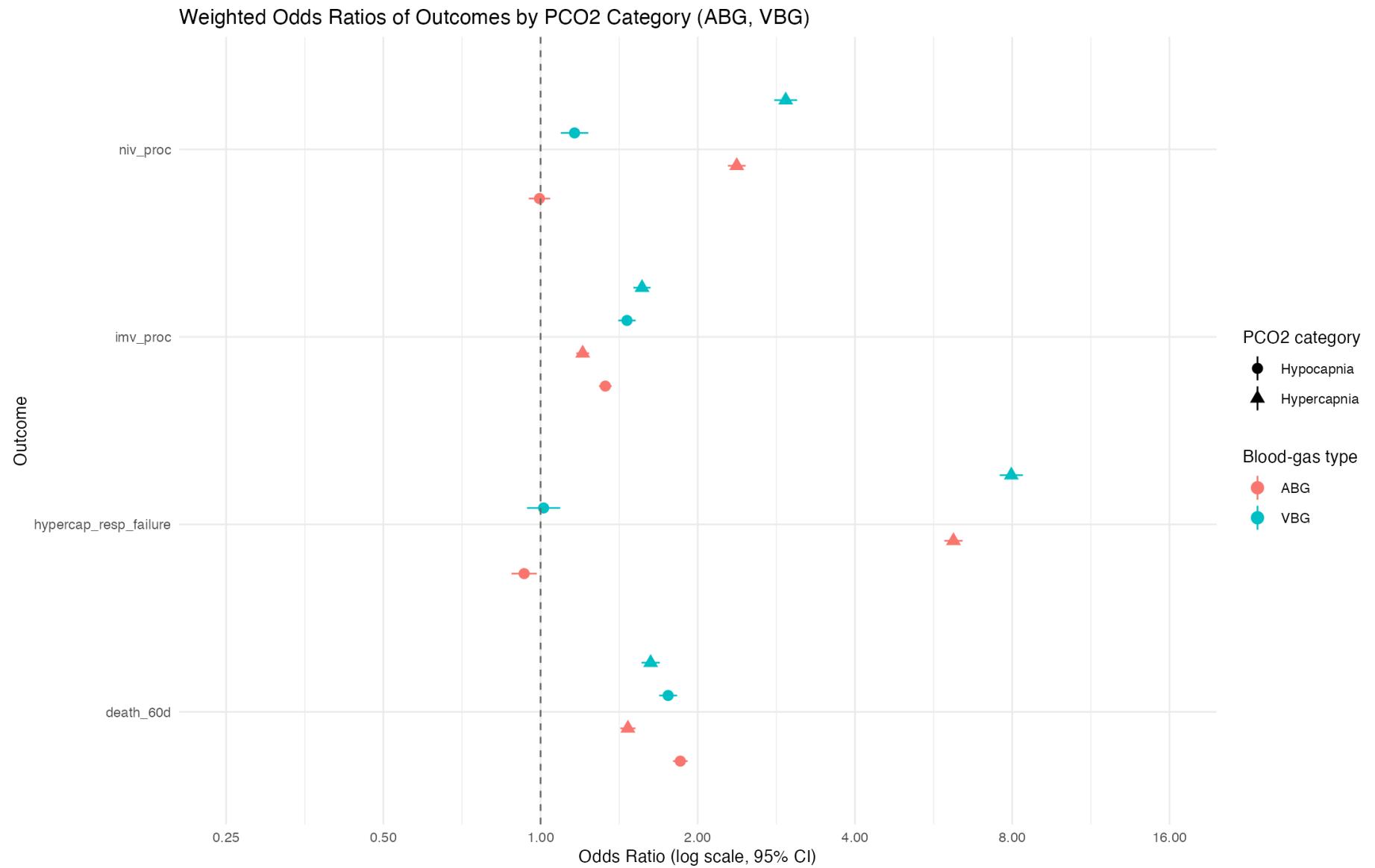
# 4. Plot weighted odds ratios
ggplot(
  combined_or_df,
  aes(
    x      = outcome,
    y      = estimate,
    ymin   = conf.low,
    ymax   = conf.high,
    color  = group,
    shape  = exposure

```

```

)
) +
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: 23.83 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")) w_abg else if (exists("weight_model")) weight_model else NULL
w_vbg_obj <- if (exists("w_vbg")) w_vbg else NULL

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

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

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

# Bind, clean, and factorize for plotting
```

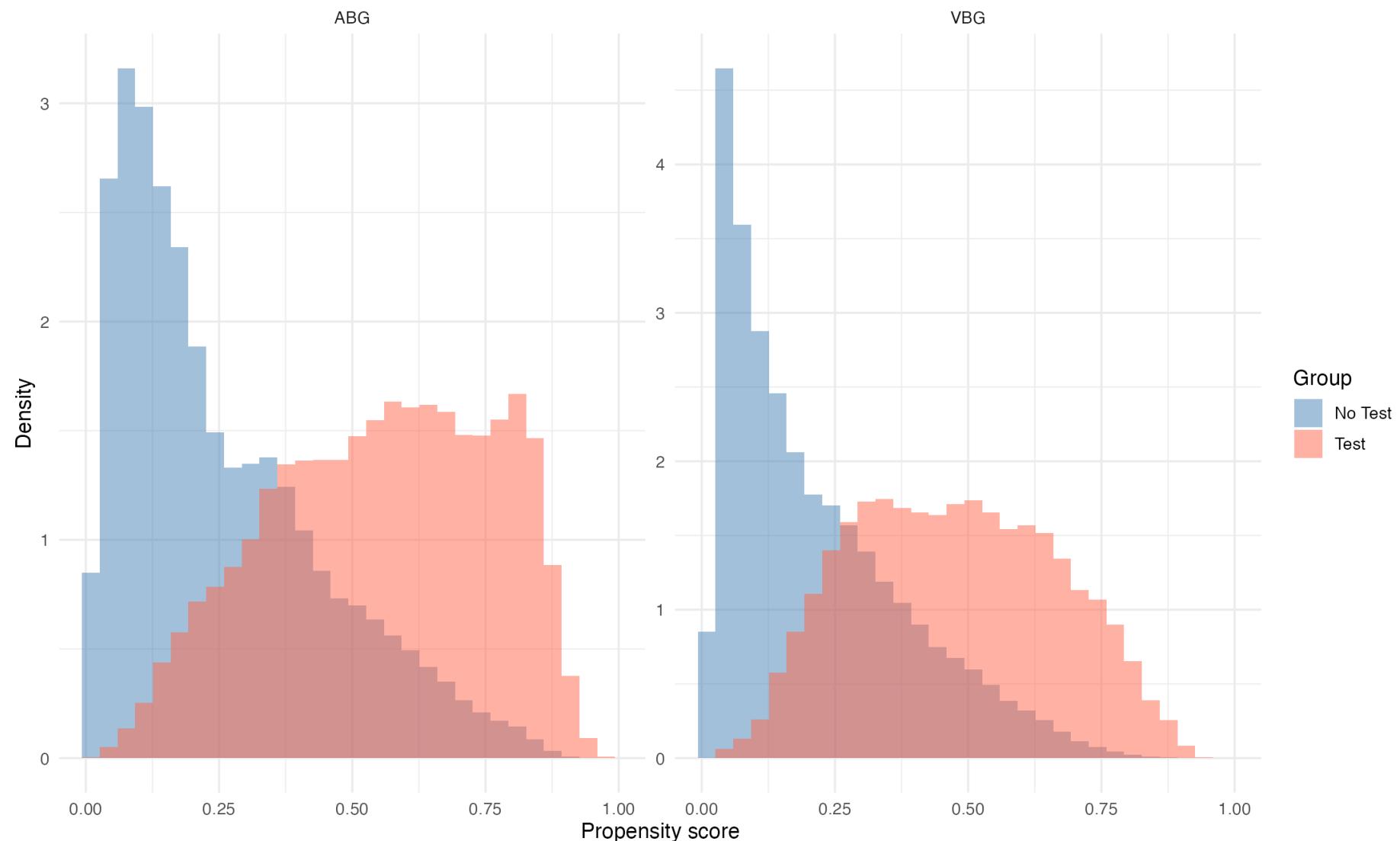
```

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

# Plot
ggplot(df_ps, aes(x = ps, fill = treat)) +
  geom_histogram(aes(y = 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: 0.77 s

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

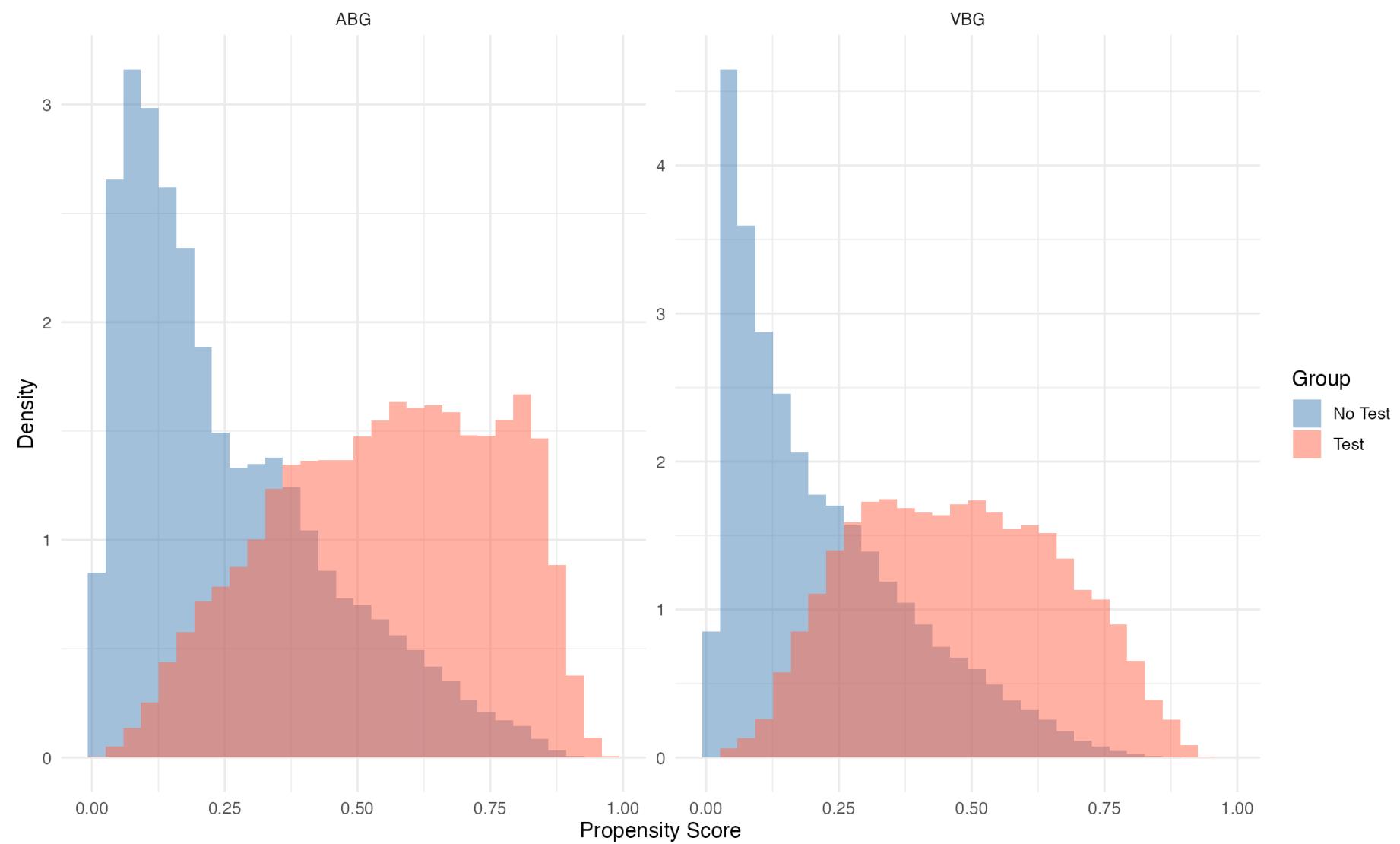
```

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

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: 0.61 s

4 Multiple Imputation Analysis

added 12/6/2025

4.1 7) Packages and reproducibility

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

Attaching package: 'mice'

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

filter

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

cbind, rbind

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

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

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

Attaching package: 'naniar'

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

prop_miss

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

Attaching package: 'skimr'

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

```
n_complete
```

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

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

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

Attaching package: 'future'

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

```
cluster
```

```

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

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: 2.13 s

4.1.1 Missing data / imputation summary (pre-imputation)

Chunk mi-missing-summary runtime: 0.09 s

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

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%
	140	0.0%

```

# --- 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)) +
    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 x %d cols)", R, length(cols_heat)))
print(p_heat)
}

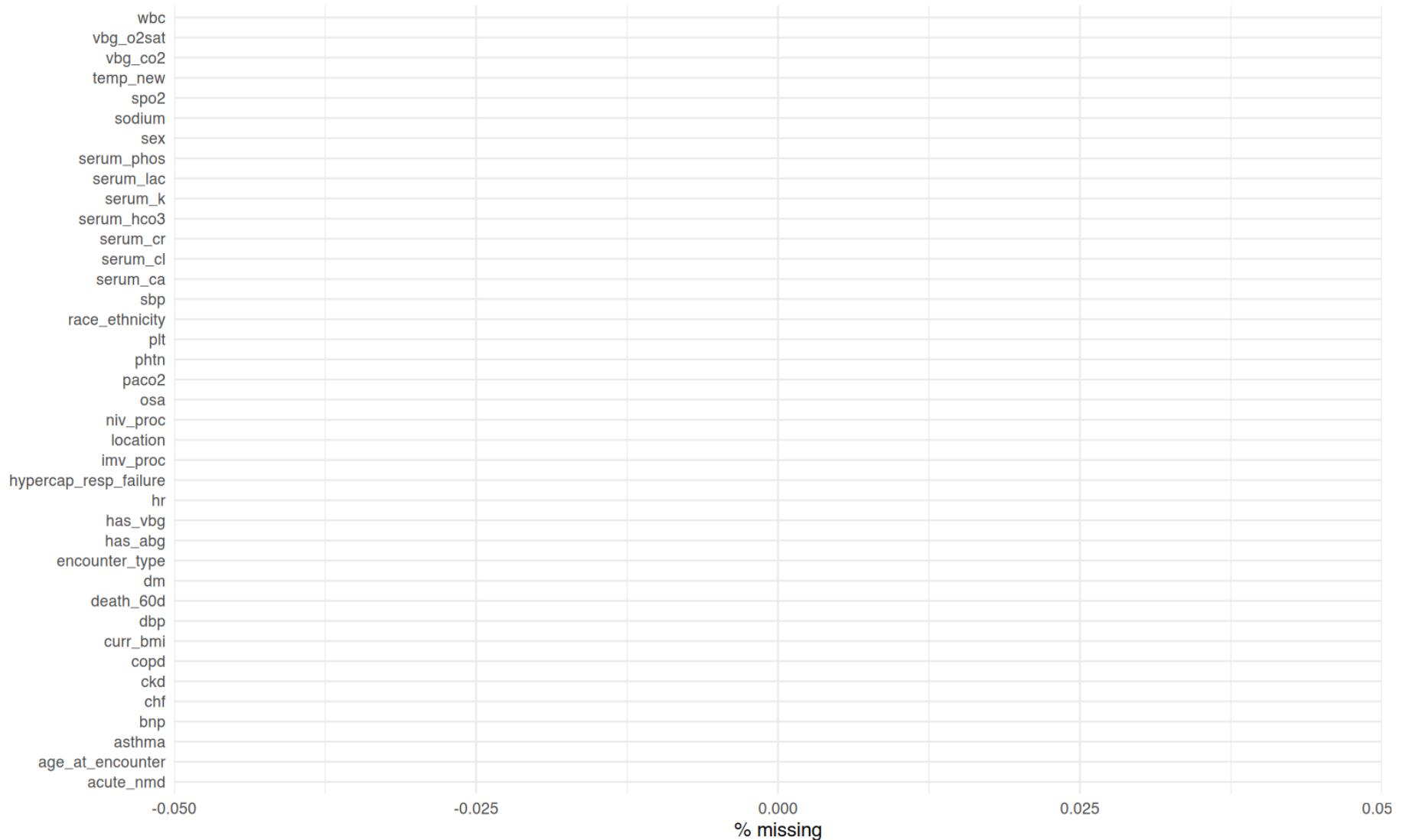
```

```

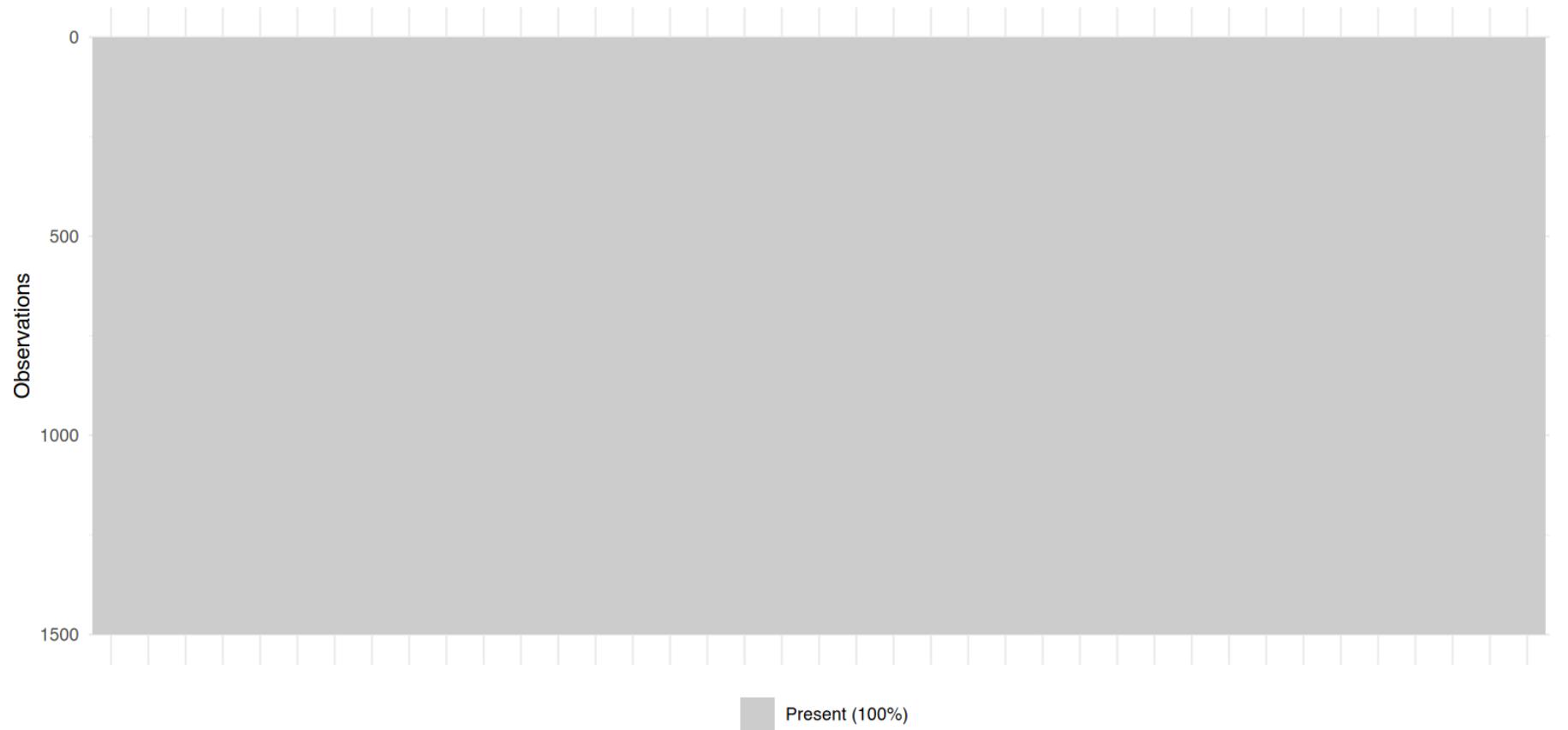
# A tibble: 39 x 3
  variable      n_miss pct_miss
  <chr>        <int>    <num>
1 age_at_encounter     0      0
2 sex                   0      0
3 race_ethnicity       0      0
4 curr_bmi              0      0
5 copd                  0      0
6 asthma                0      0
7 osa                   0      0
8 chf                   0      0
9 acute_nmd             0      0
10 phtn                 0      0
# i 29 more rows

```

Top missing variables (after MI)



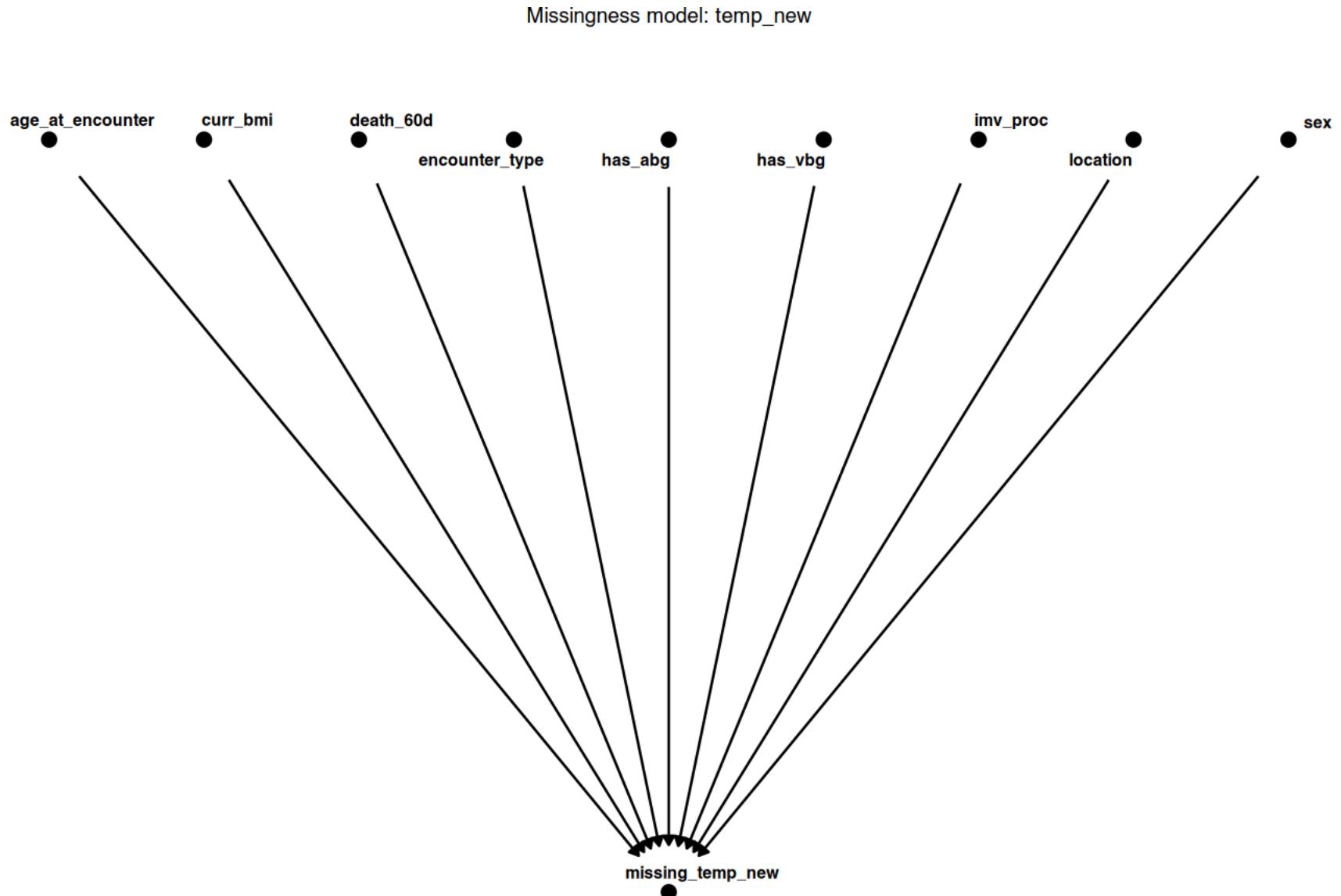
Missingness heatmap on imputed data (1500 rows x 39 cols)



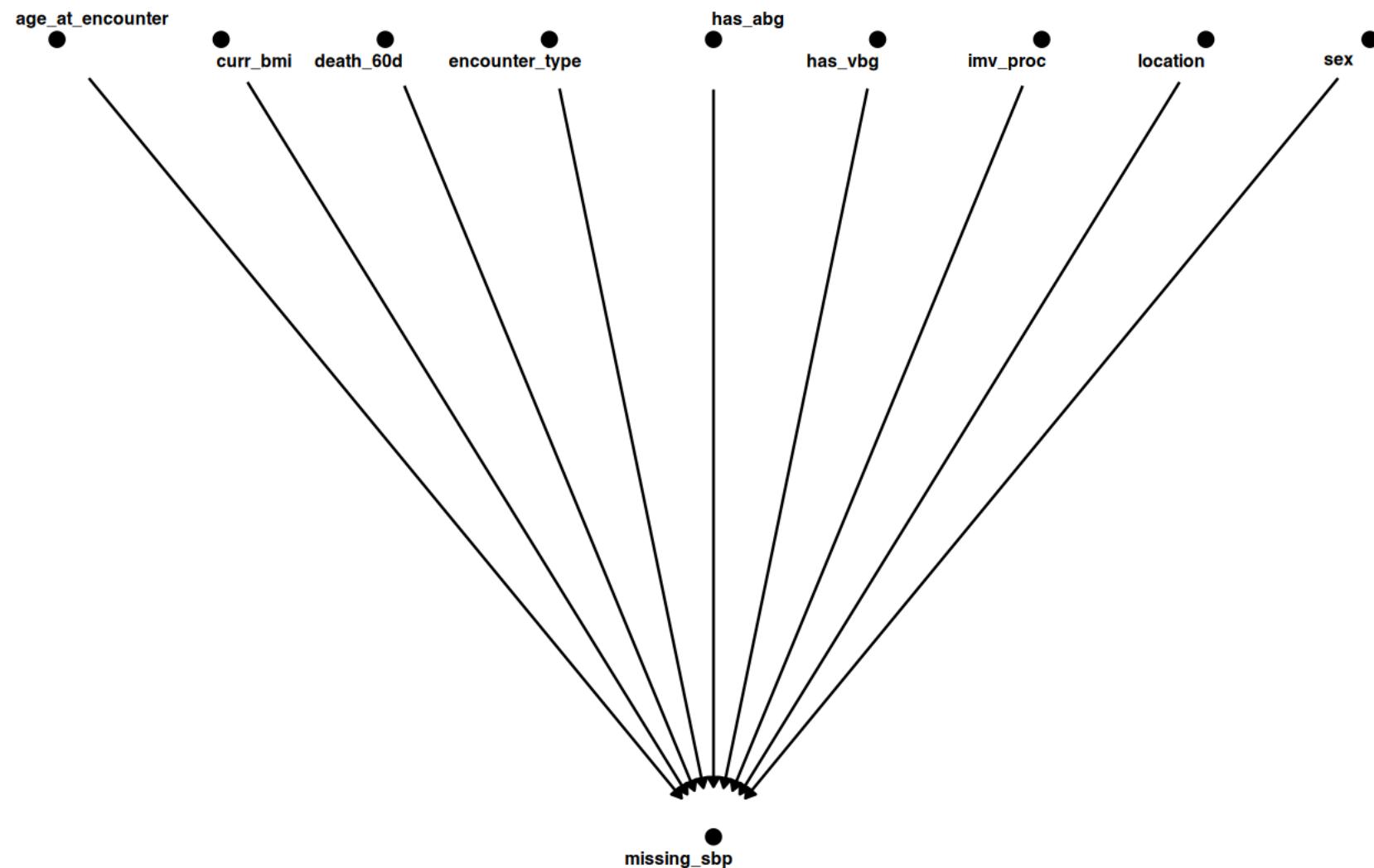
```
# 4) Optional, but often heavy: UpSet of co-missingness - skip by default.  
# If you really want it, do it for top 6-10 variables only:  
# naniar::gg_miss_upset(dat_imp[, head(top_vars, 8), drop = FALSE])
```

Chunk mi-missing-audit runtime: 1.65 s

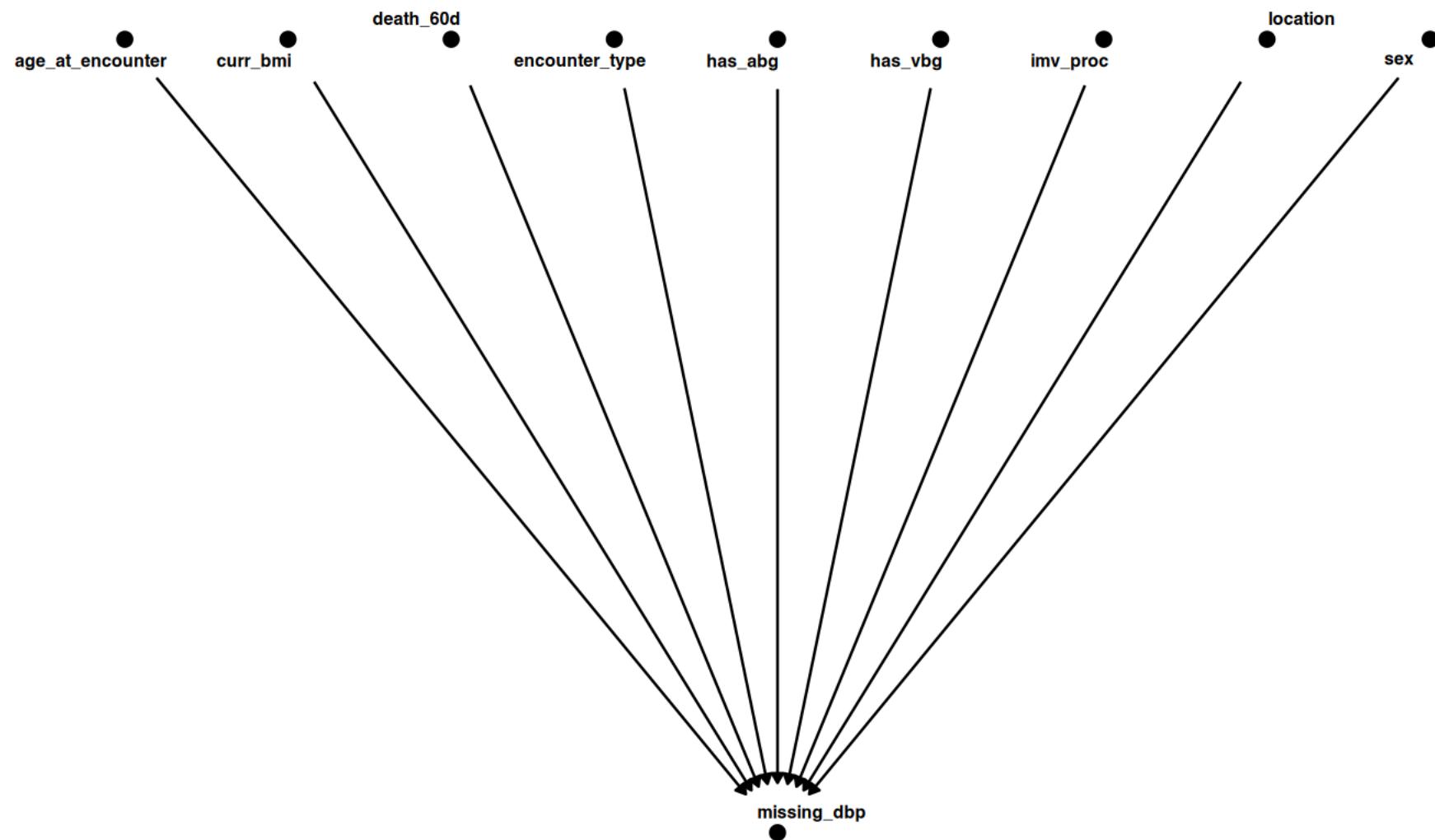
4.1.3 7.2 Missingness structure and drivers



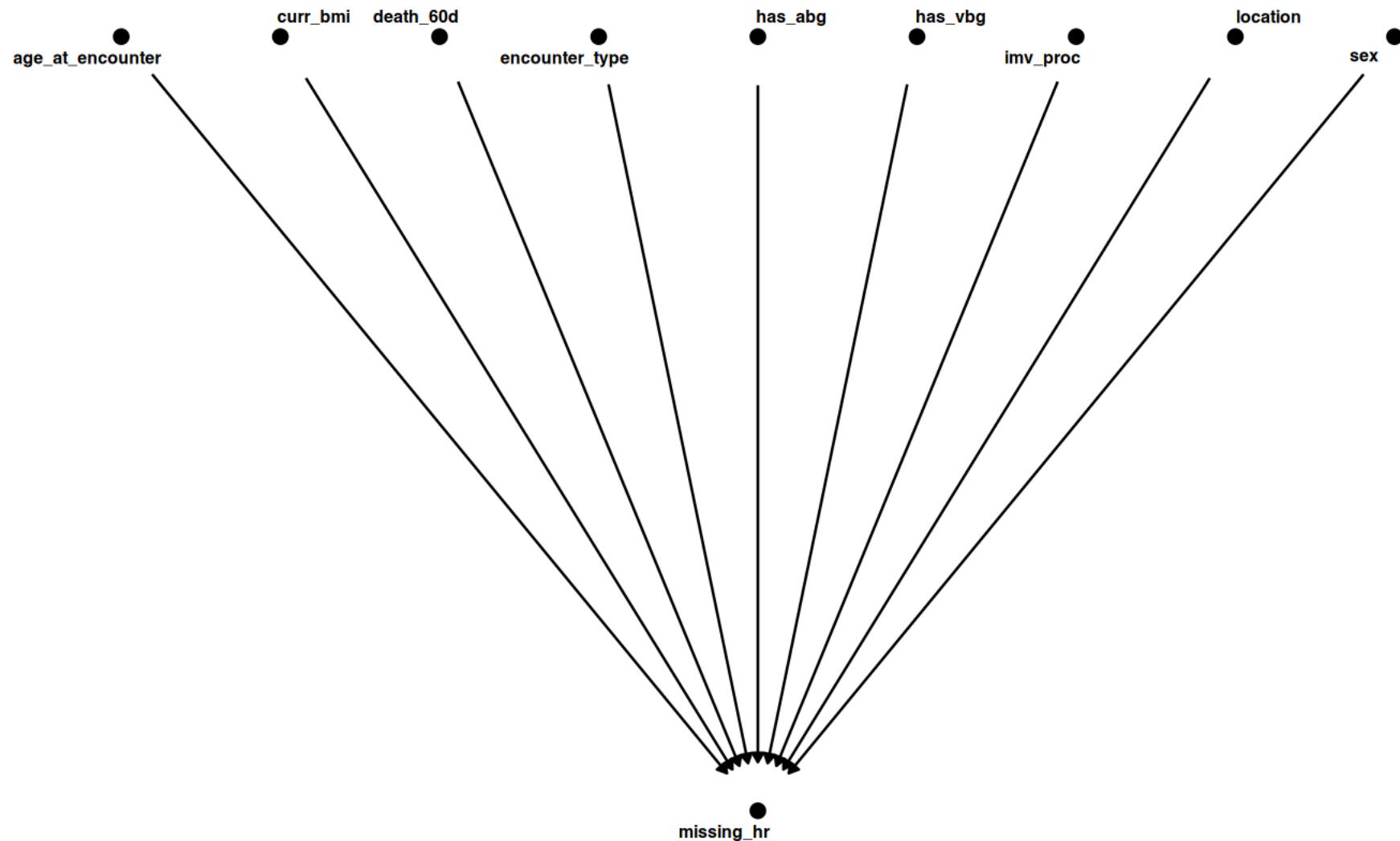
Missingness model: sbp



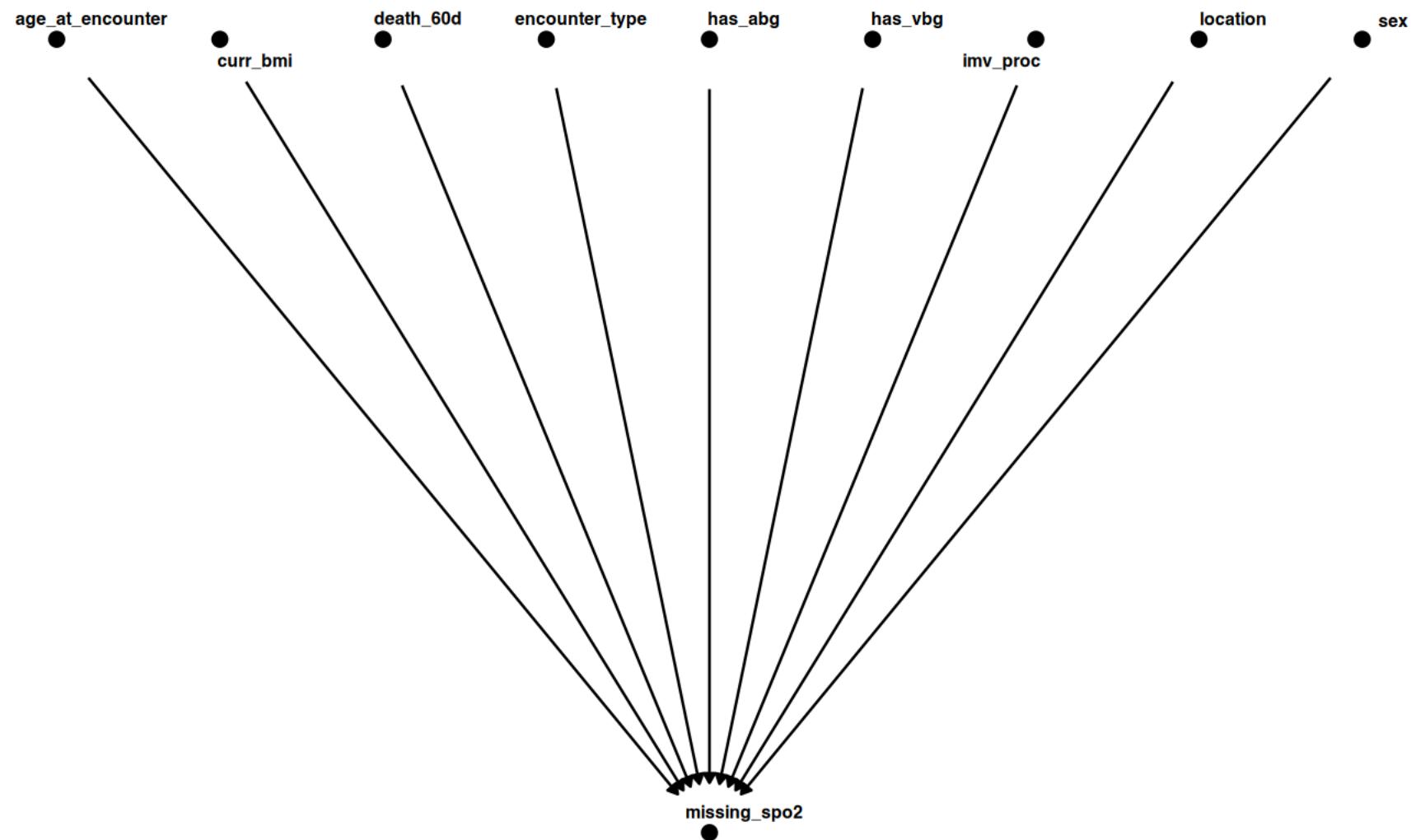
Missingness model: dbp



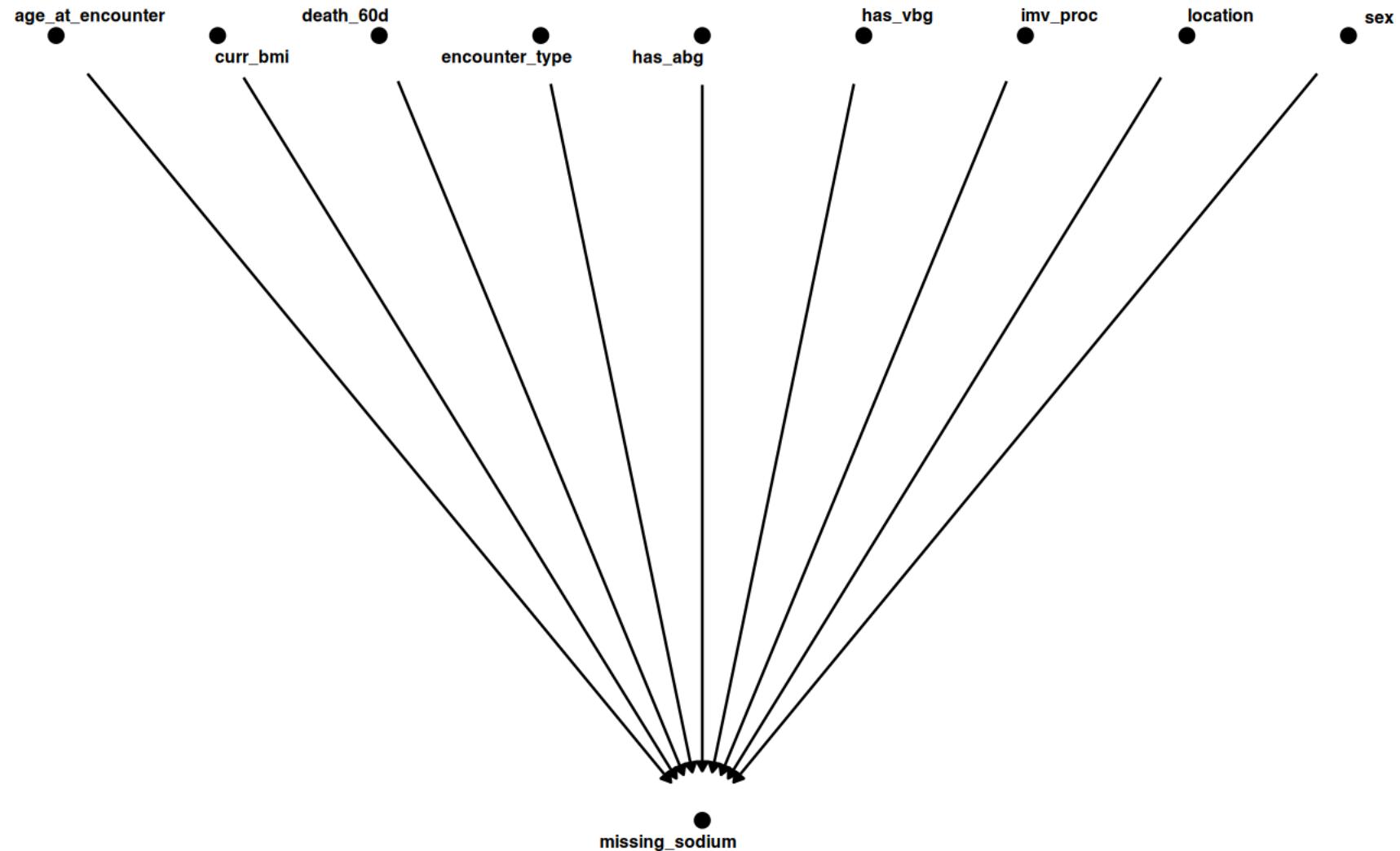
Missingness model: hr



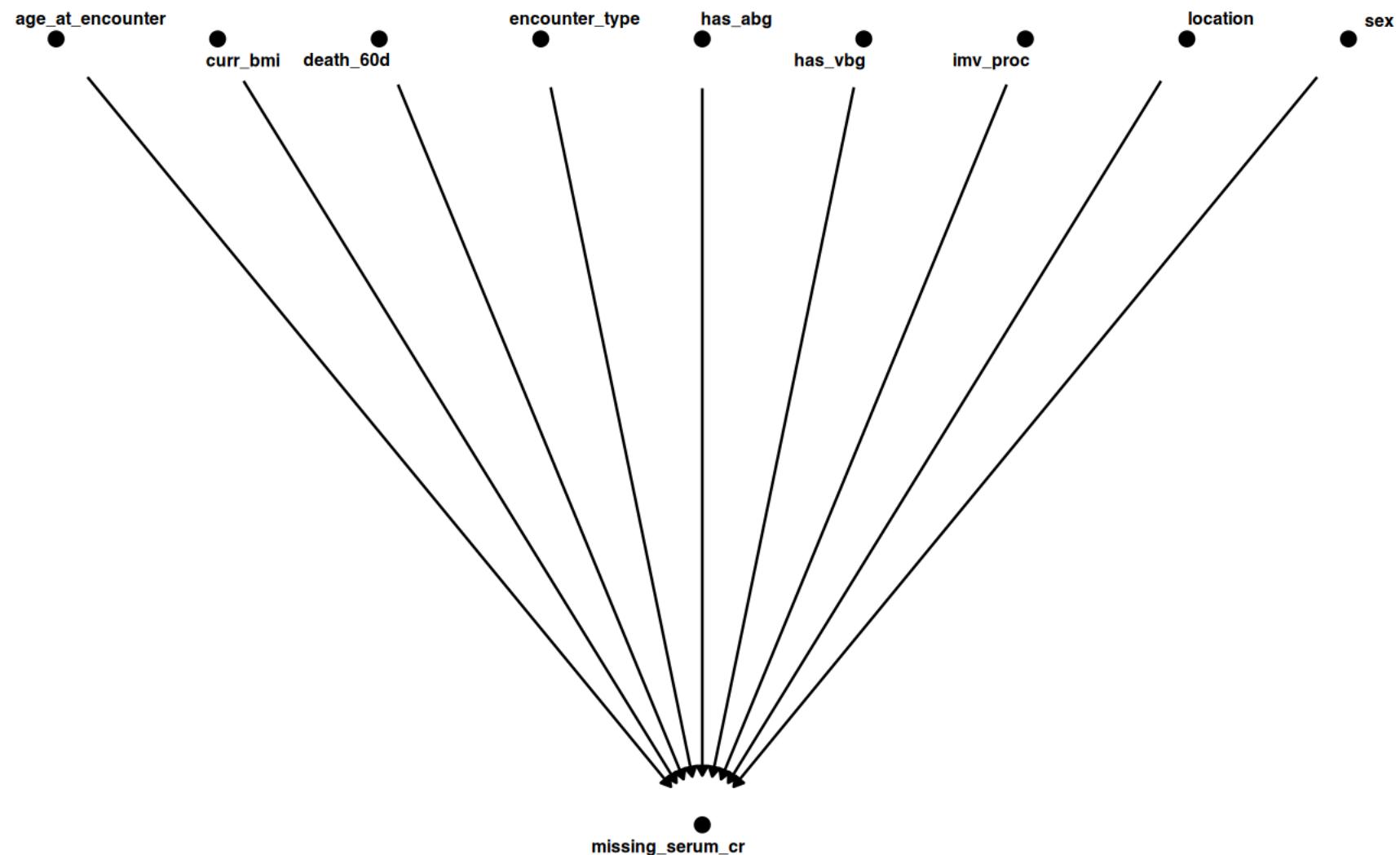
Missingness model: spo2



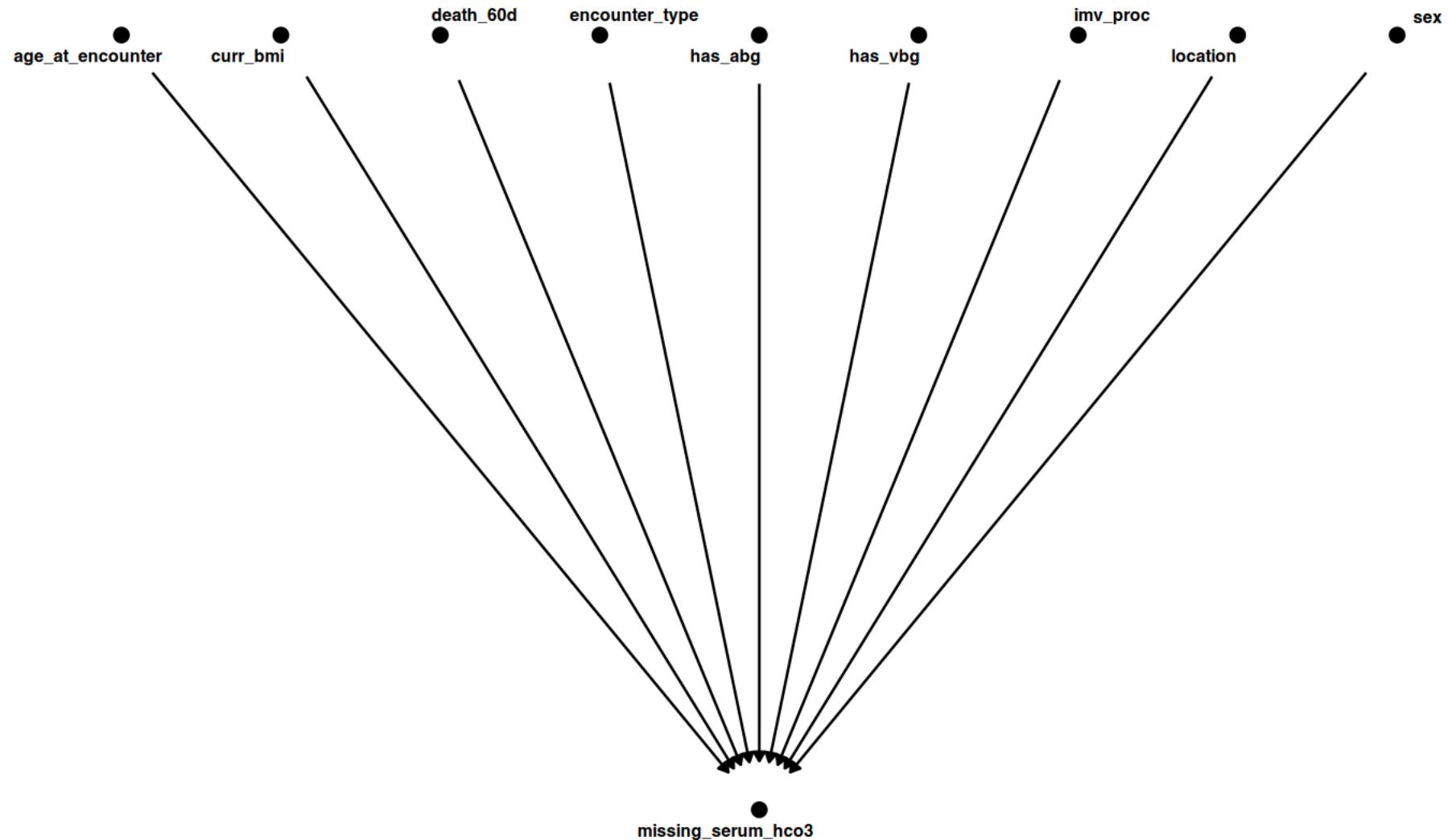
Missingness model: sodium



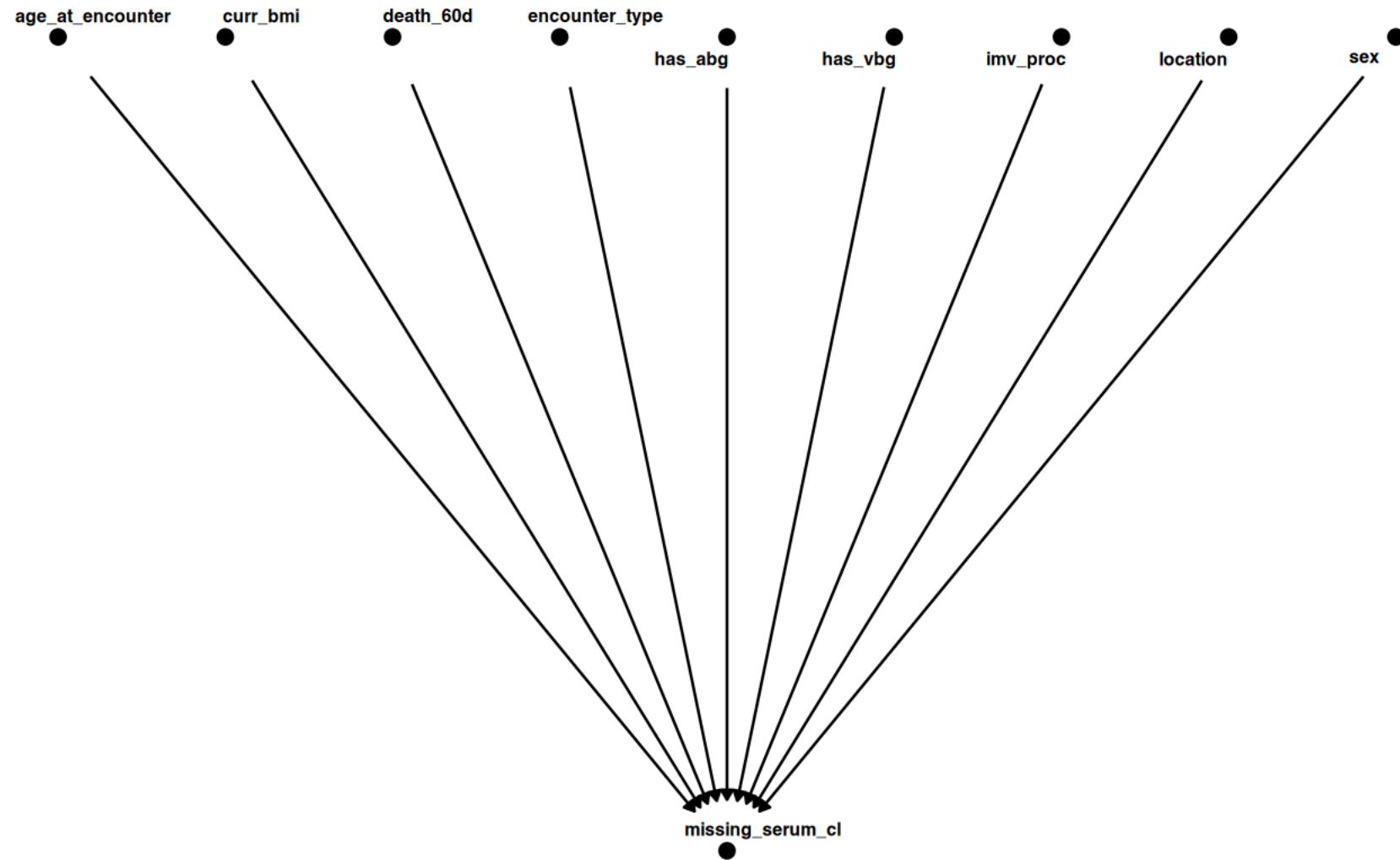
Missingness model: serum_cr



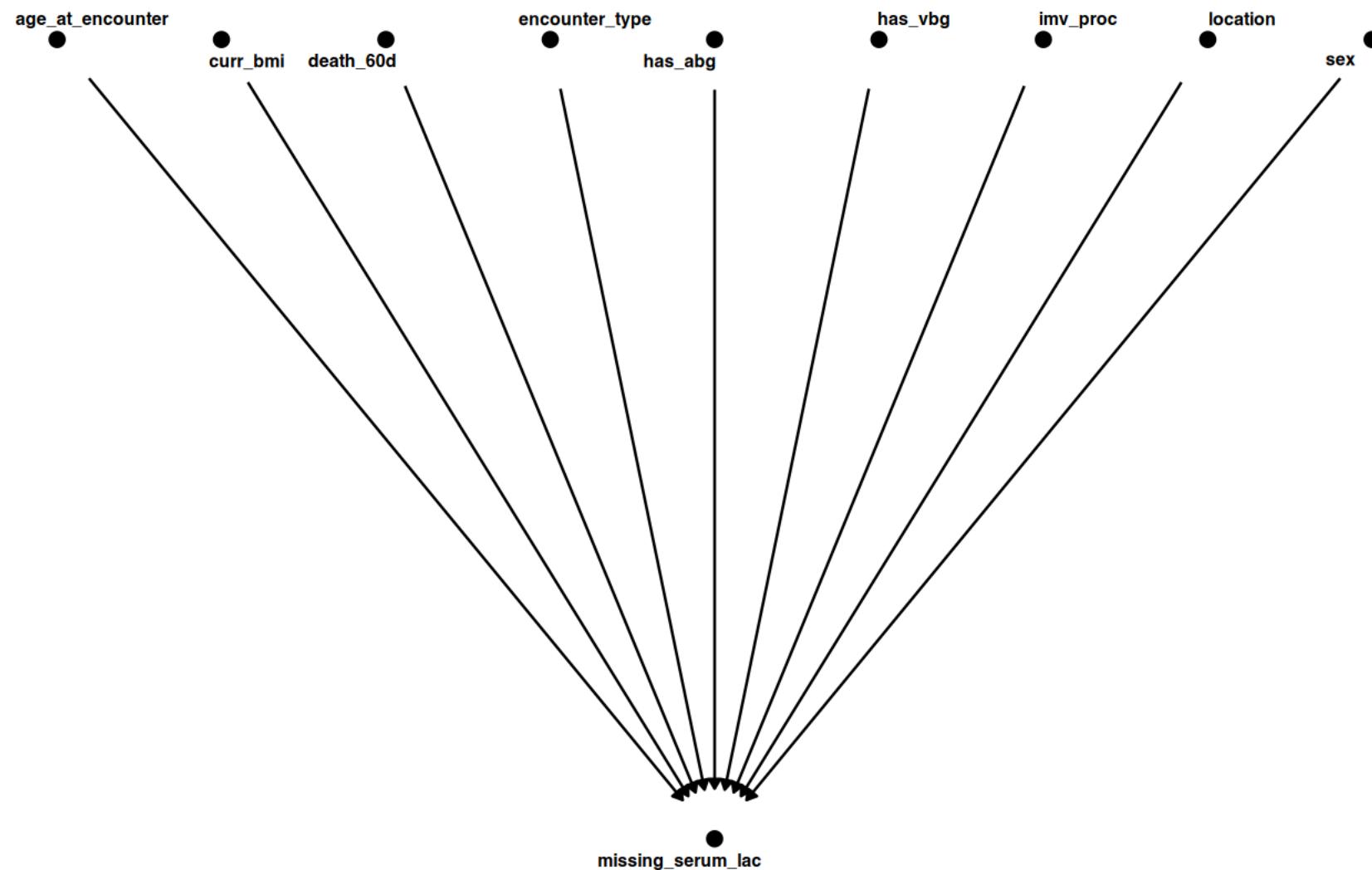
Missingness model: serum_hco3



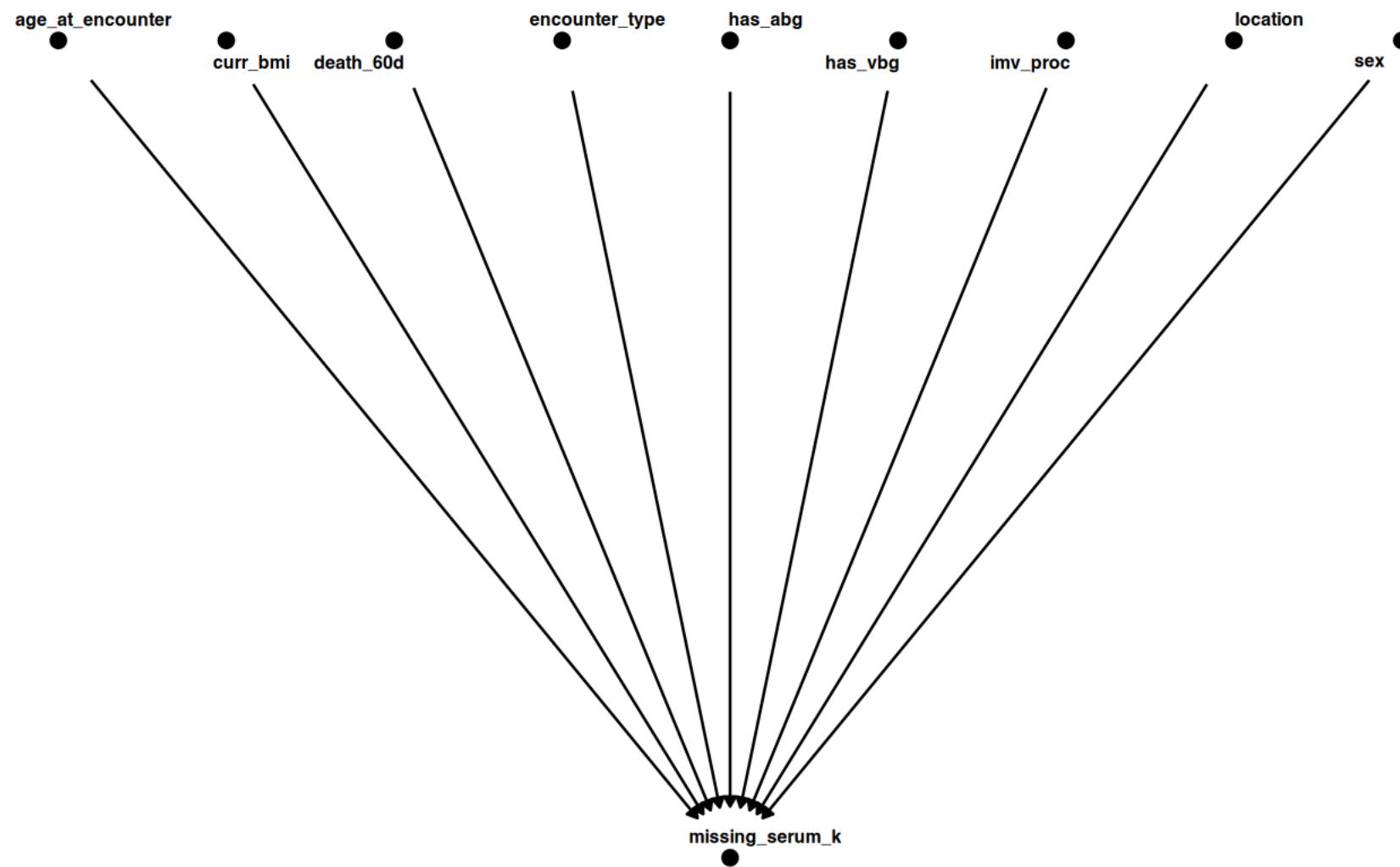
Missingness model: serum_cl



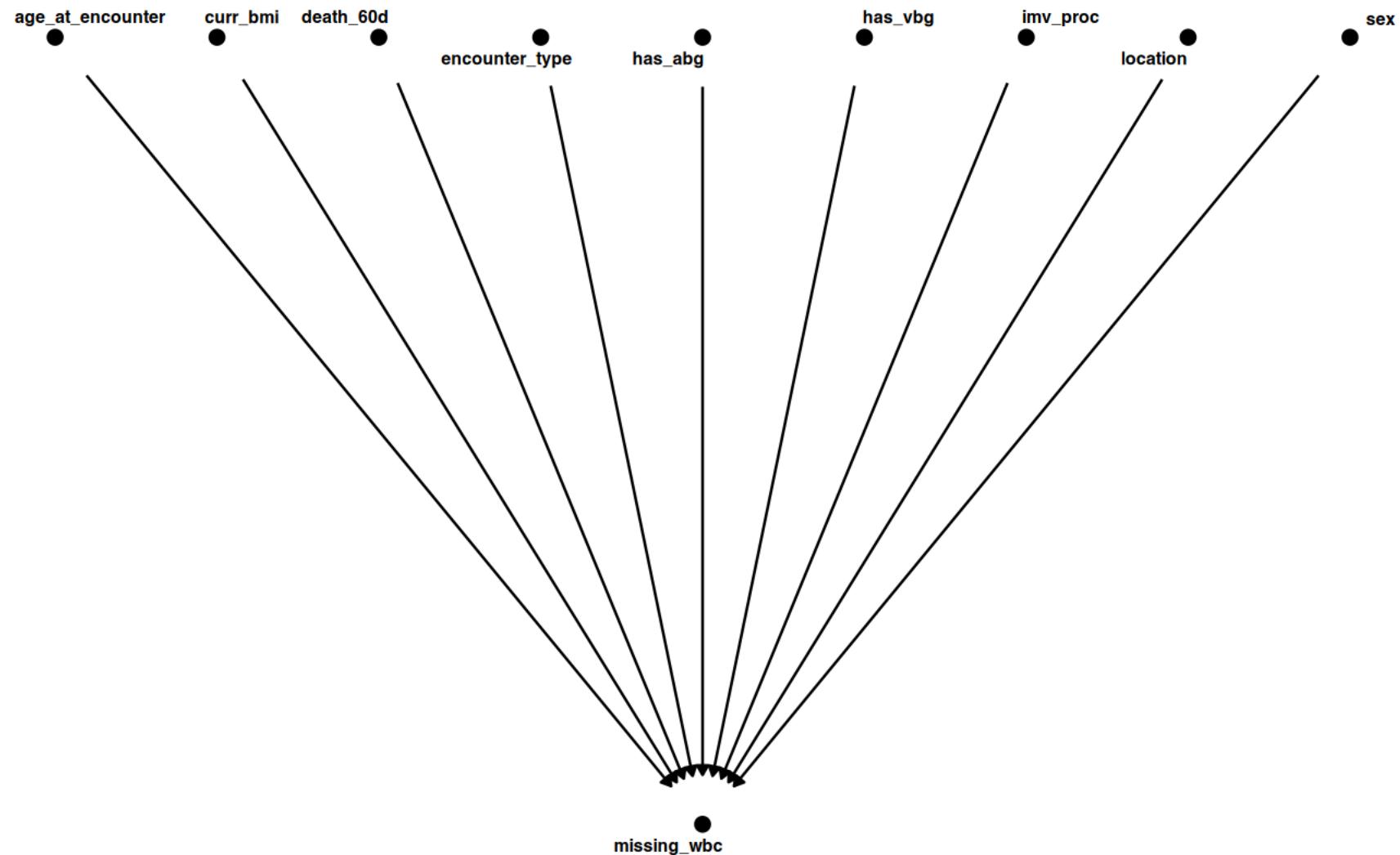
Missingness model: serum_lac



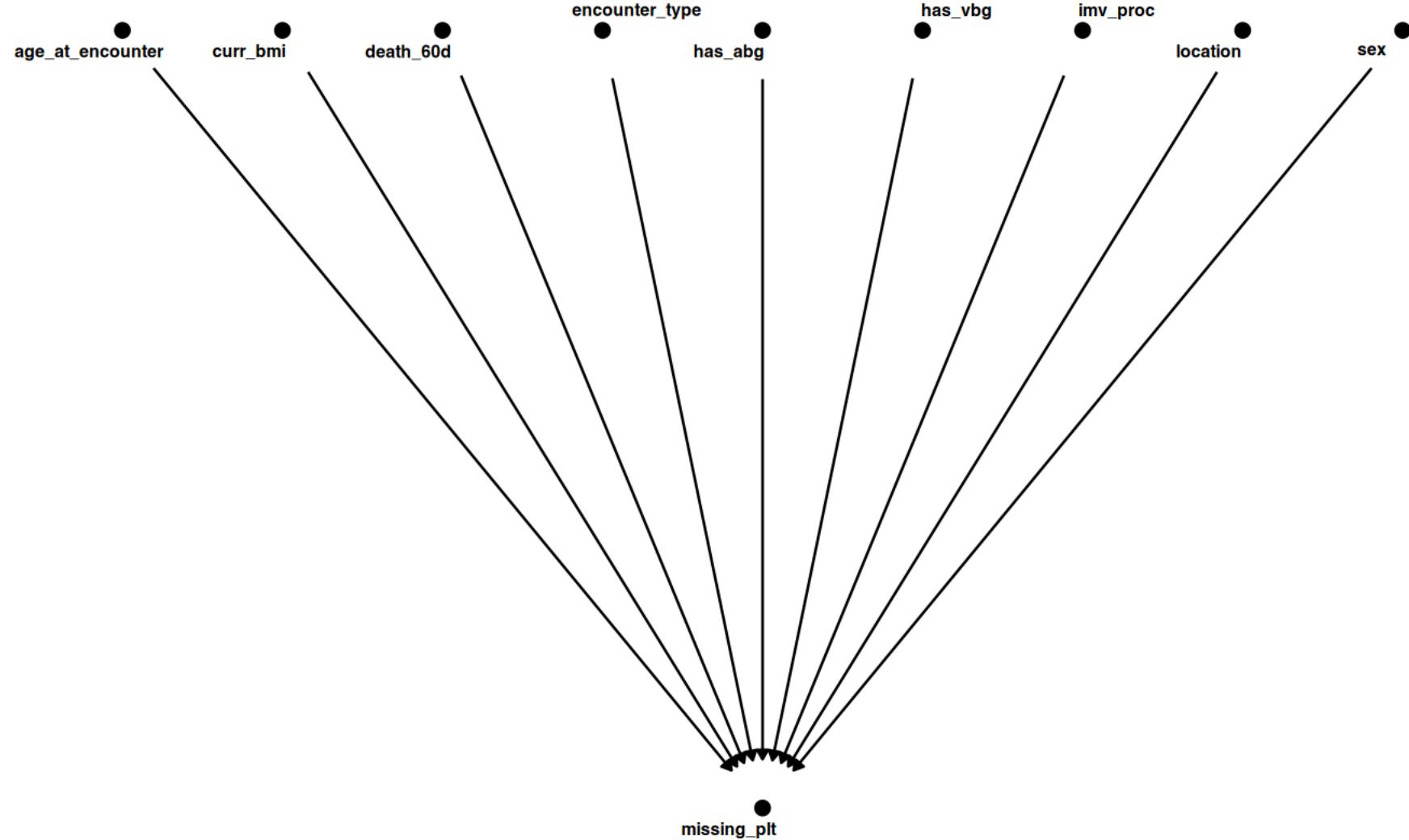
Missingness model: serum_k



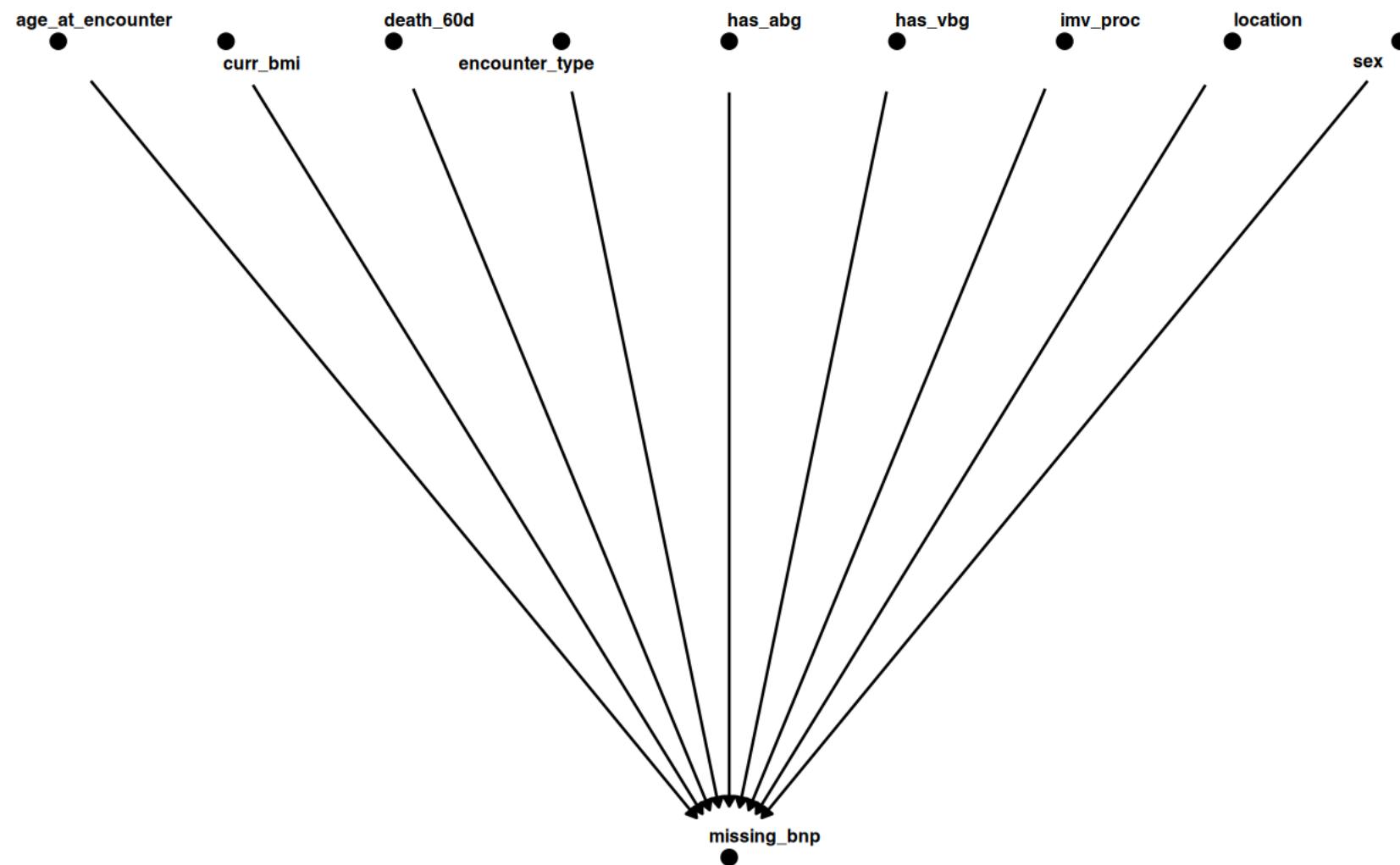
Missingness model: wbc



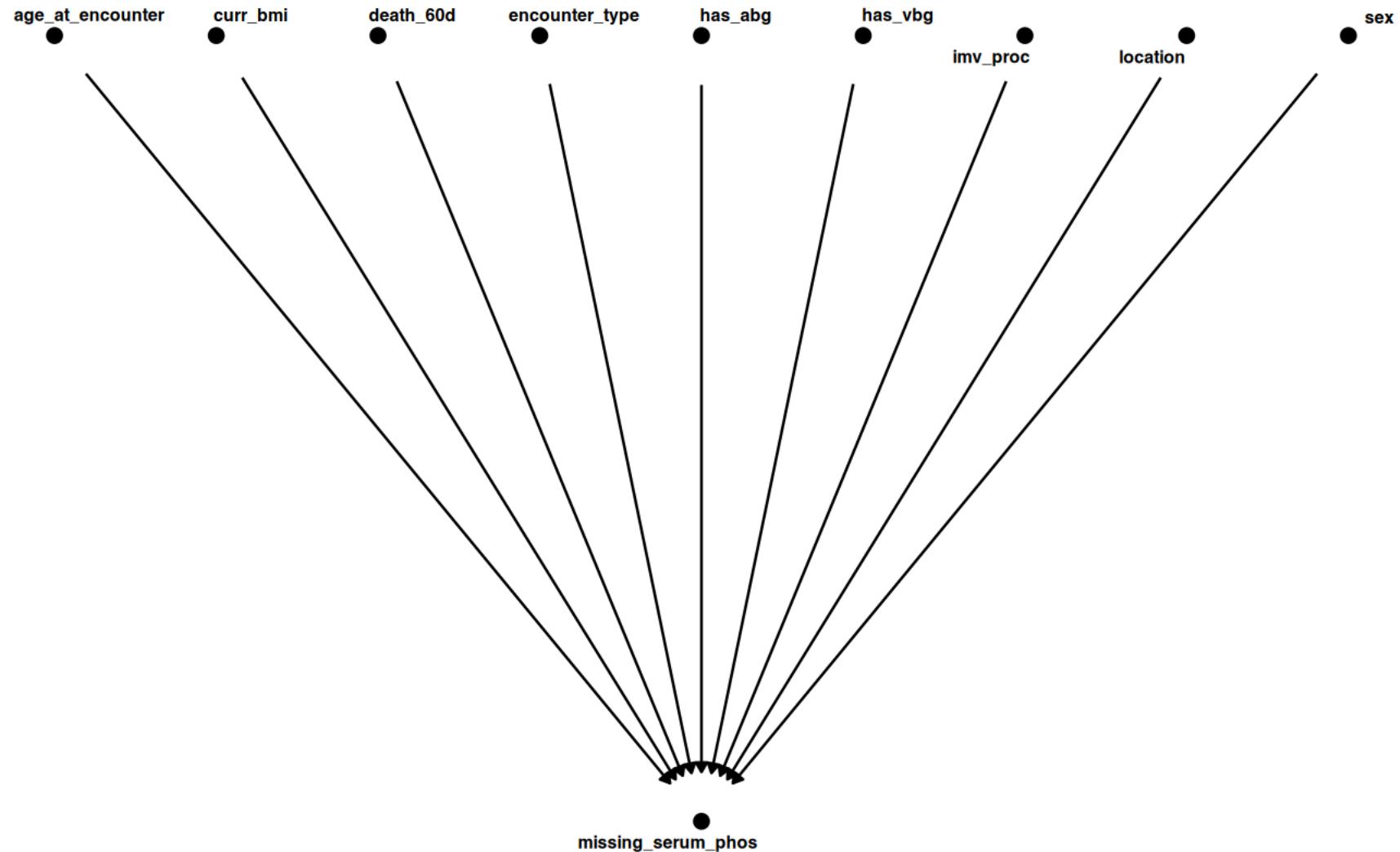
Missingness model: plt



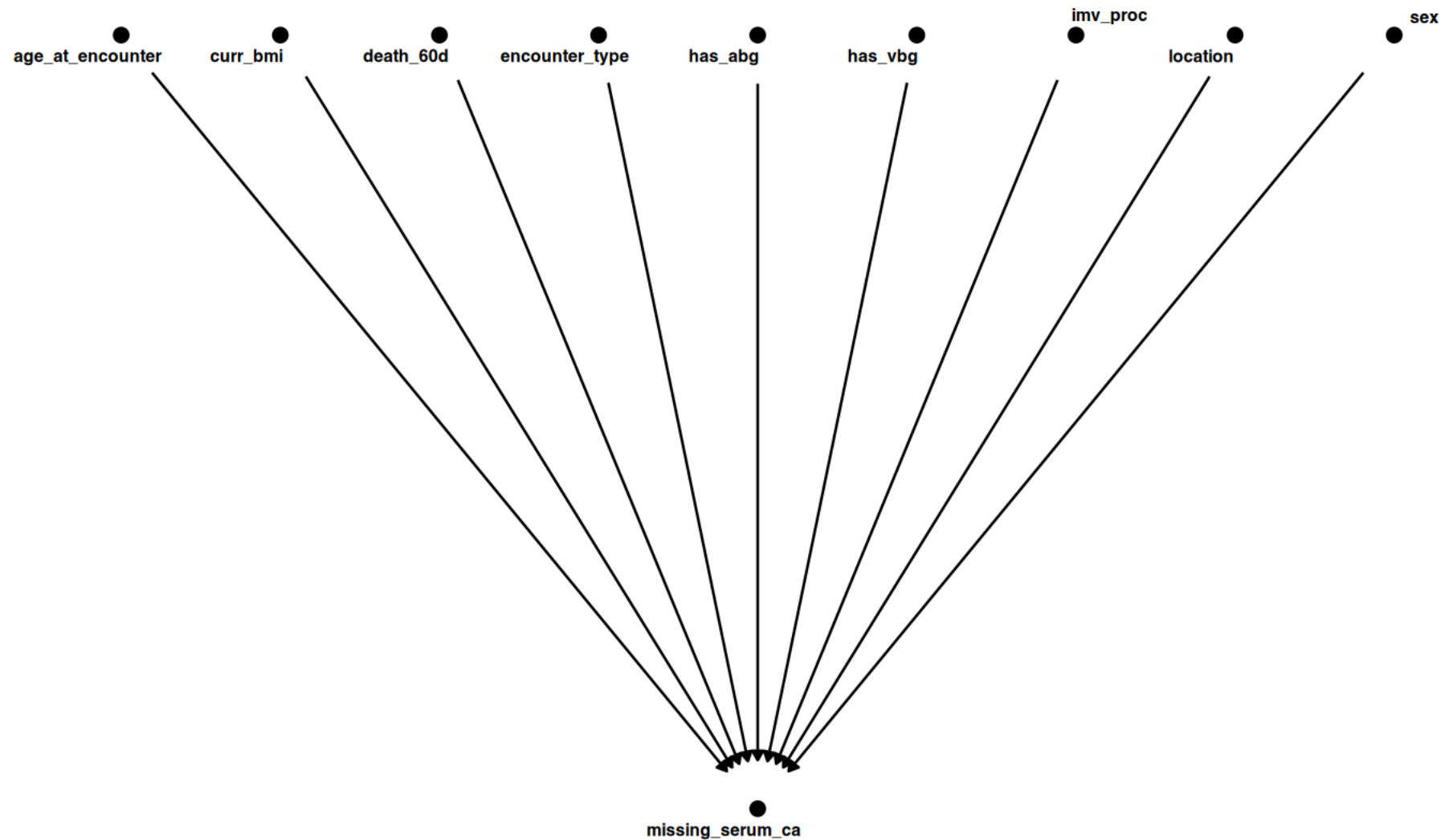
Missingness model: bnp



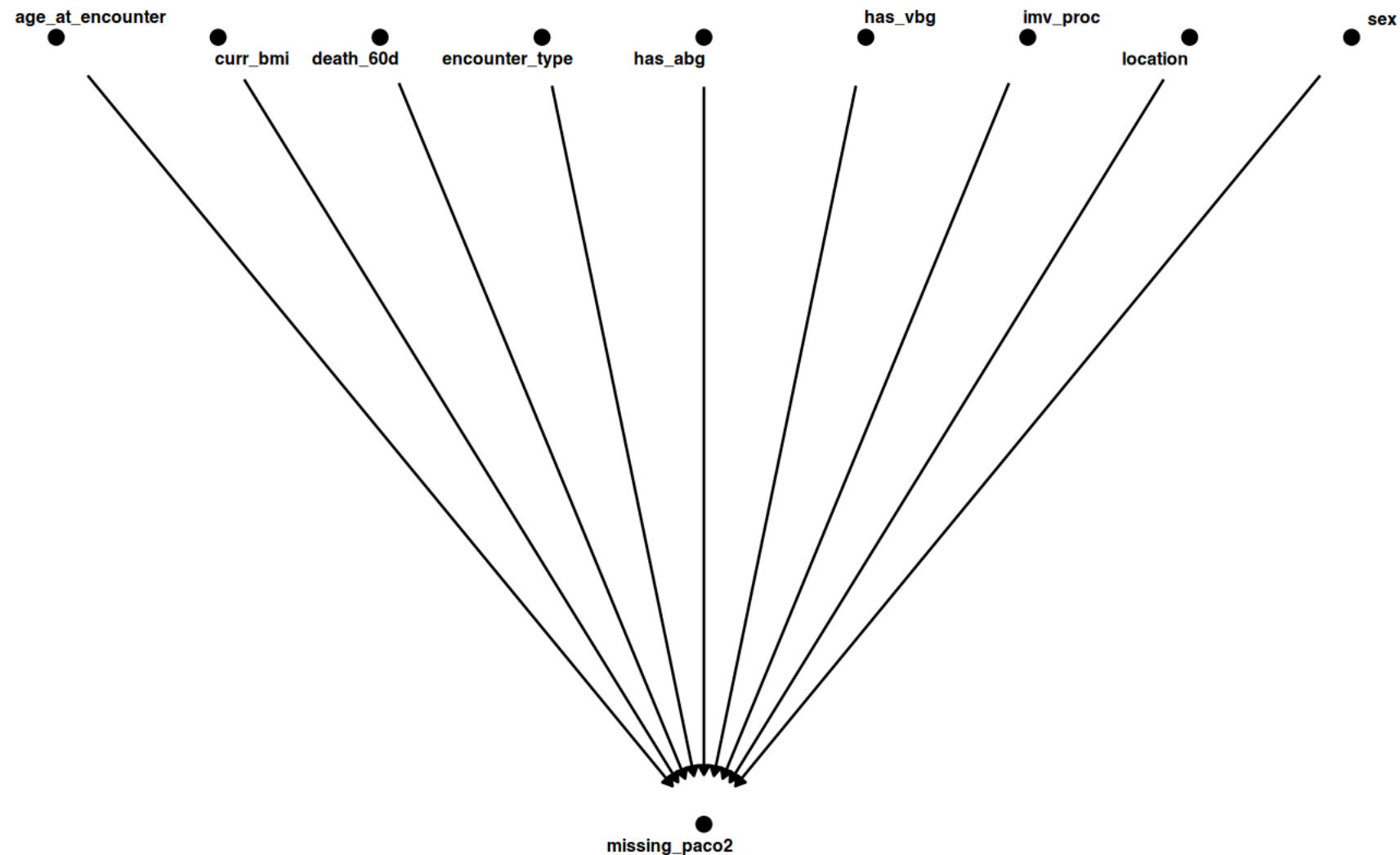
Missingness model: serum_phos



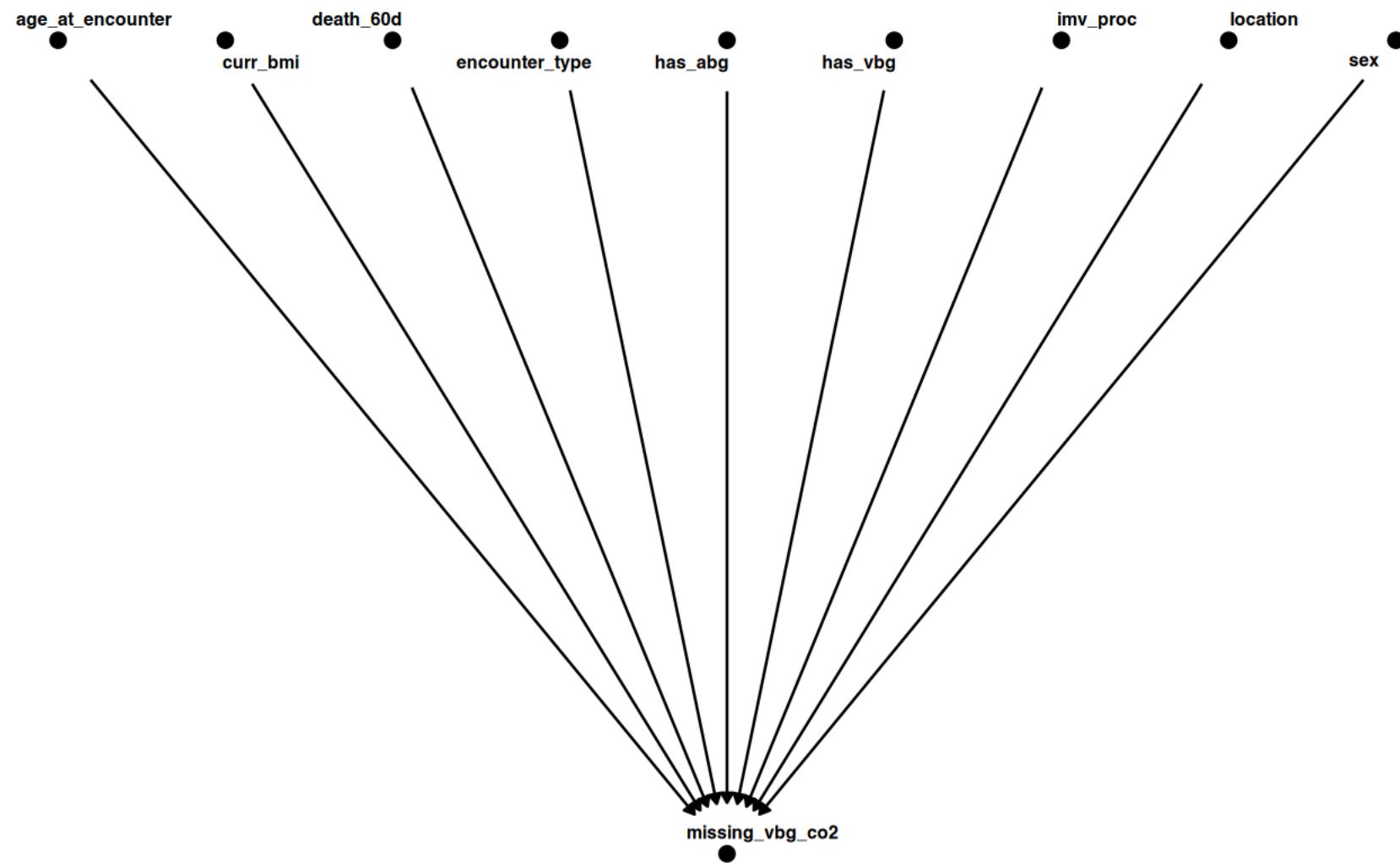
Missingness model: serum_ca



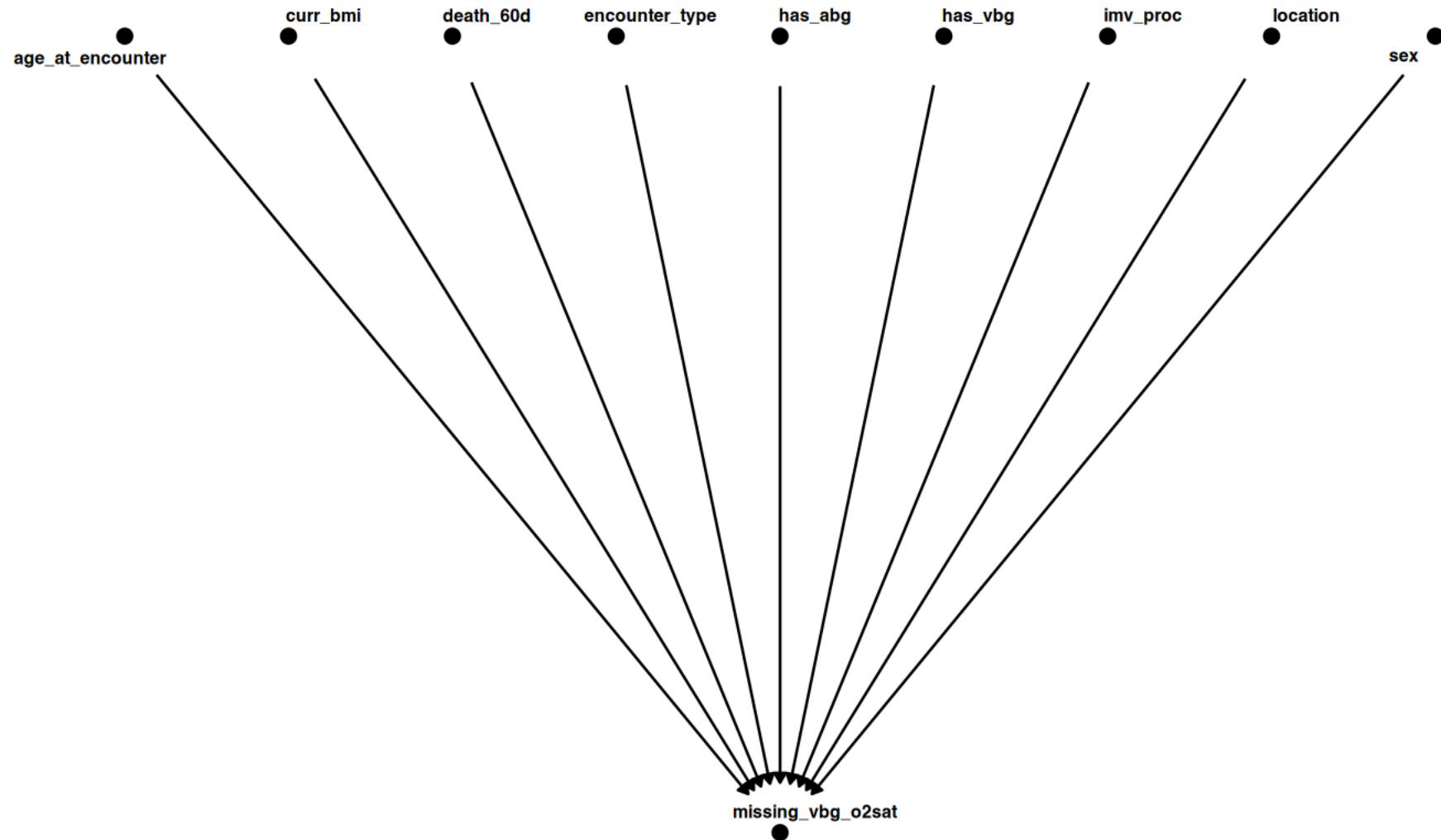
Missingness model: paco2



Missingness model: vbg_co2



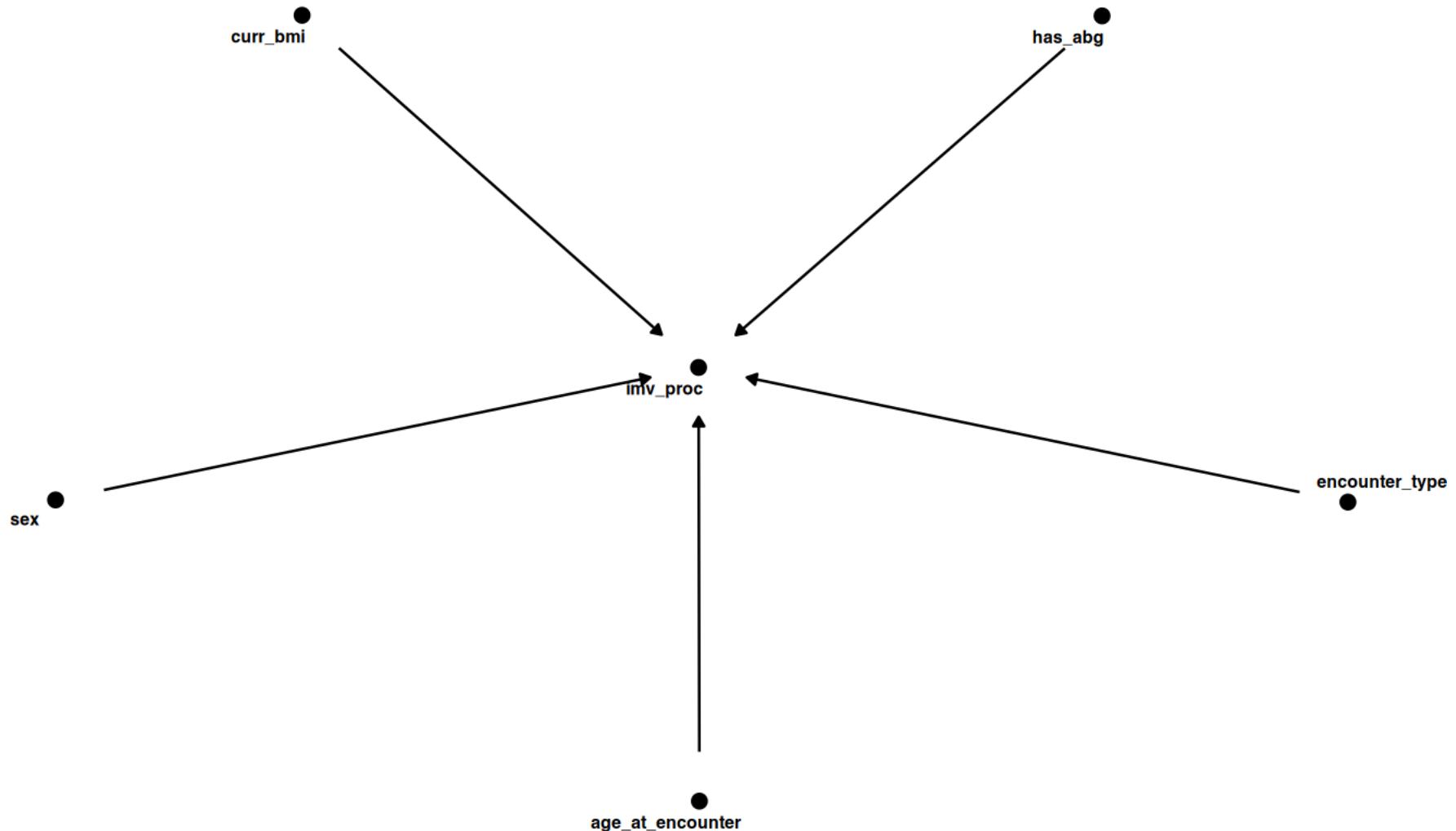
Missingness model: vbg_o2sat



Chunk mi-missing-structure runtime: 19.38 s

4.1.4 7.3 Monte Carlo error check after MI

MI MC error: IMV ~ ABG + covariates



Chunk mi-mcerror runtime: 8.91 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.

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

```
encounter_type class: factor
```

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

```
[1] Emergency Inpatient
```

```
Levels: Emergency Inpatient
```

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

```
# --- Helpers -----
```

```
# Map various encodings to strict 0/1 integer (for sex + 0/1 indicators)
```

```
to01 <- function(x) {
  if (is.logical(x)) return(as.integer(x))
  if (is.factor(x)) x <- as.character(x)
```

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

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

```
  # character encodings (case/space-insensitive)
```

```
  if (any(!is_num)) {
    s <- trimws(tolower(as.character(x[!is_num])))
```

```

    out[!is_num][s %in% c("0","no","false","female","f")] <- 0L
    out[!is_num][s %in% c("1","yes","true","male","m")] <- 1L
}
out
}

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

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

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

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

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

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

  # Inpatient synonyms: "inpatient", "inpt", "inpat", exact "ip"
  is_ip <- grepl("\binpatient\b", s_chr) |
    grepl("\binpt\b", s_chr) |
    grepl("\binpat\b", s_chr) |
    grepl("(^|[^a-z])ip([a-z]|$)", s_chr)

```

```

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

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

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

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

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

```

Emergency Inpatient
 171727 343559

```

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

```

```

    stop("normalize_encounter_type produced all NA; extend the synonym map to your raw values.")
}

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

```

Chunk mi-prep runtime: 2.88 s

Chunk type-invariants runtime: 0.25 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 (Option A: include PaCO2 for ABG + keep VBG CO2/O2Sat) -----
library(mice)
library(dplyr)

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

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

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

# Make binary comorbidities 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"))
})

# Force C02 variables to be numeric BEFORE we convert any leftover characters to factor
coerce_num <- function(x) suppressWarnings(as.numeric(as.character(x)))
for (nm in intersect(co2_vars, names(mi_df))) mi_df[[nm]] <- coerce_num(mi_df[[nm]])

# For MICE: convert any remaining characters → factors (after C02 numeric coercion)
mi_df <- dplyr::mutate(mi_df, across(where(is.character), ~ factor(.x)))

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

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

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

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

# predictor matrix; forbid treatments/outcomes as predictors
pred <- mice::quickpred(mi_df, mincor = 0.05, minpuc = 0.25)
pred[, intersect(colnames(pred), no_imp)] <- 0
pred[intersect(rownames(pred), no_imp), ] <- 0

```

```

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

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

# --- run MICE -----
set.seed(20251206)
imp <- mice::mice(
  data           = mi_df,
  m              = 5,
  maxit         = 10,
  predictorMatrix = pred,
  method         = meth,
  printFlag     = TRUE,
  seed           = 20251206
)

```

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


```

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

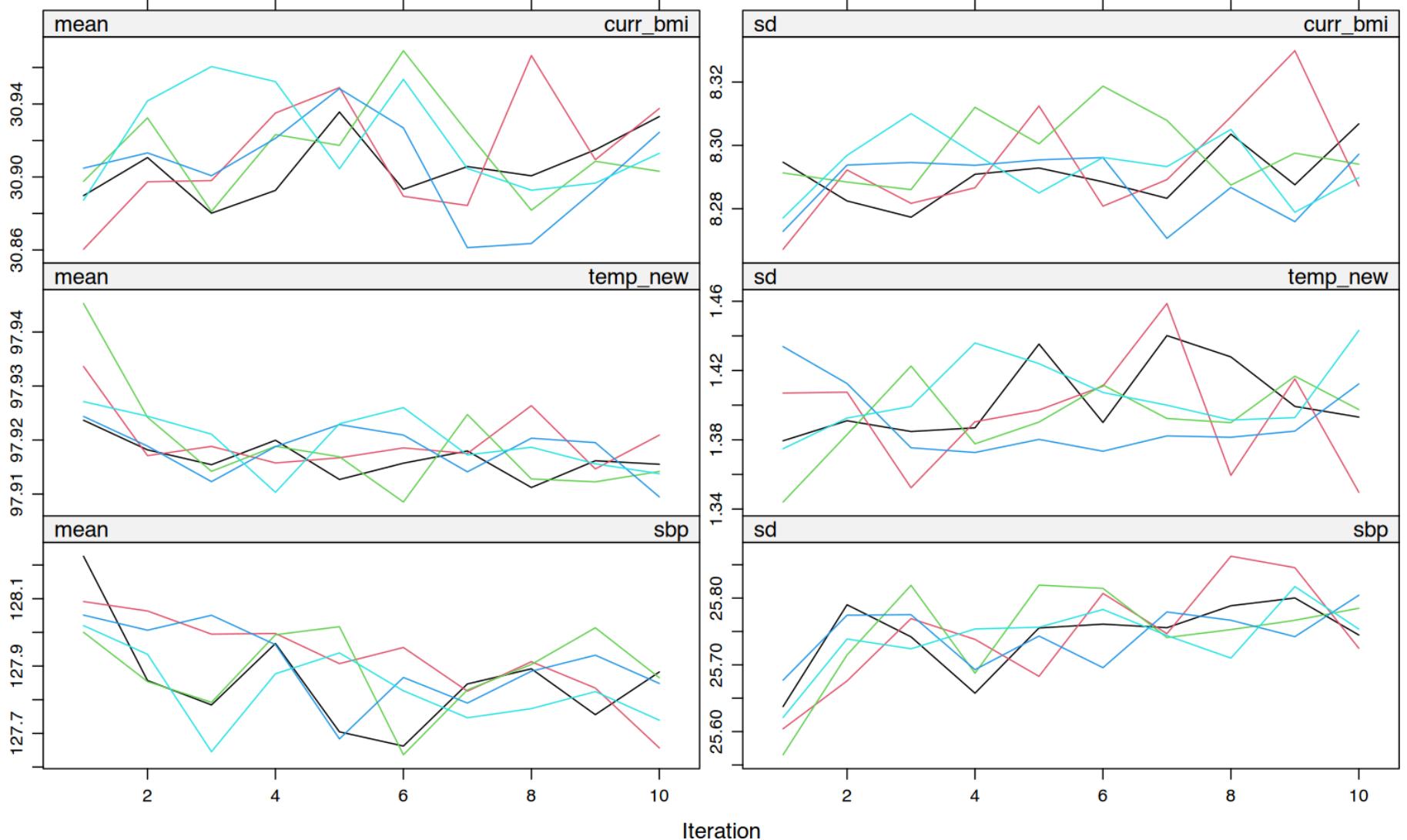
saveRDS(imp, file = mi_mids_file)

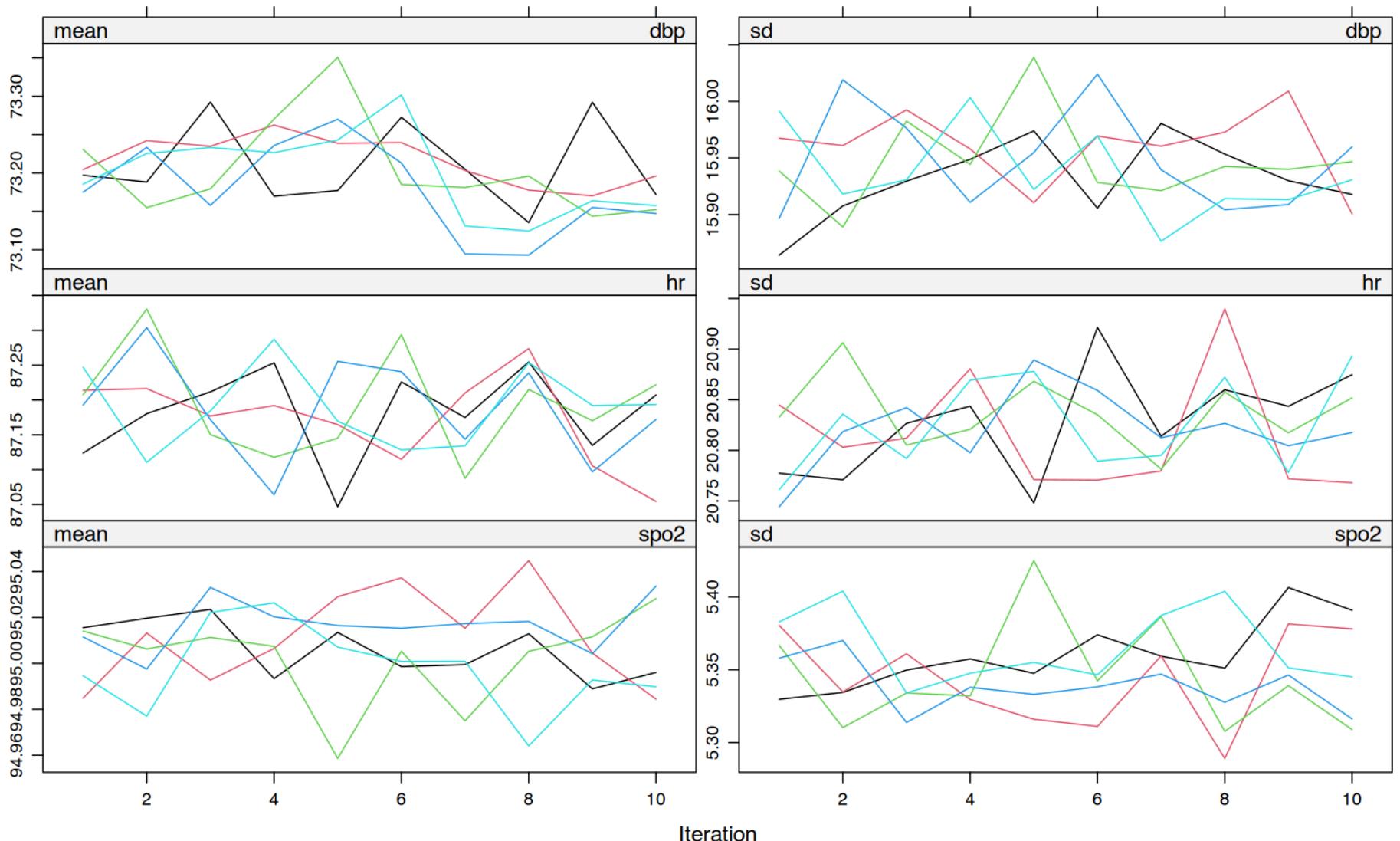
# quick sanity: these must exist and be numeric in completed data
imp_n <- imp$m
get_imp <- function(i, imp_obj = imp) mice::complete(imp_obj, action = i)
d1 <- get_imp(1)
stopifnot(all(c("paco2", "vbg_co2") %in% names(d1)))
stopifnot(is.numeric(d1$paco2), is.numeric(d1$vbg_co2))

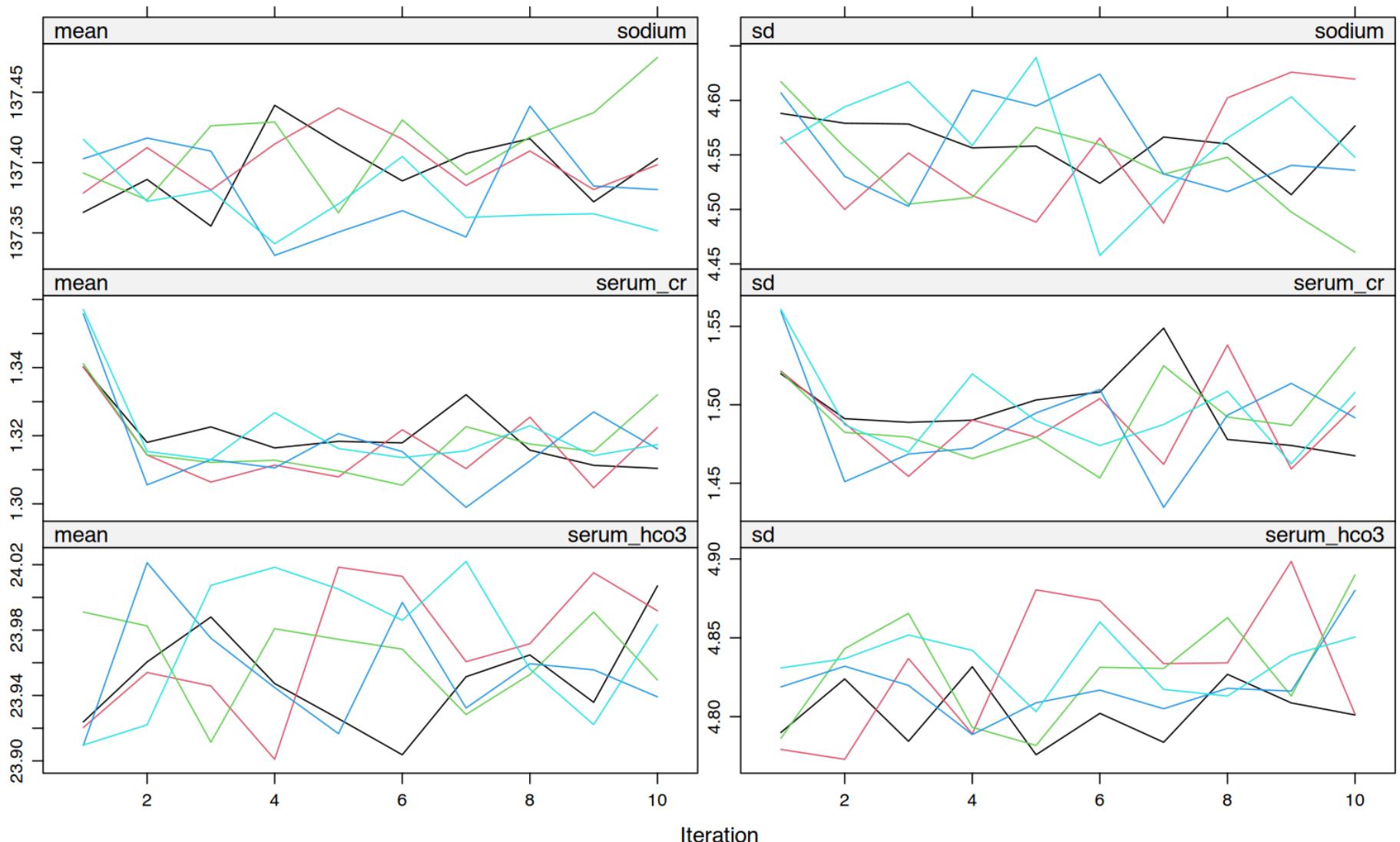
```

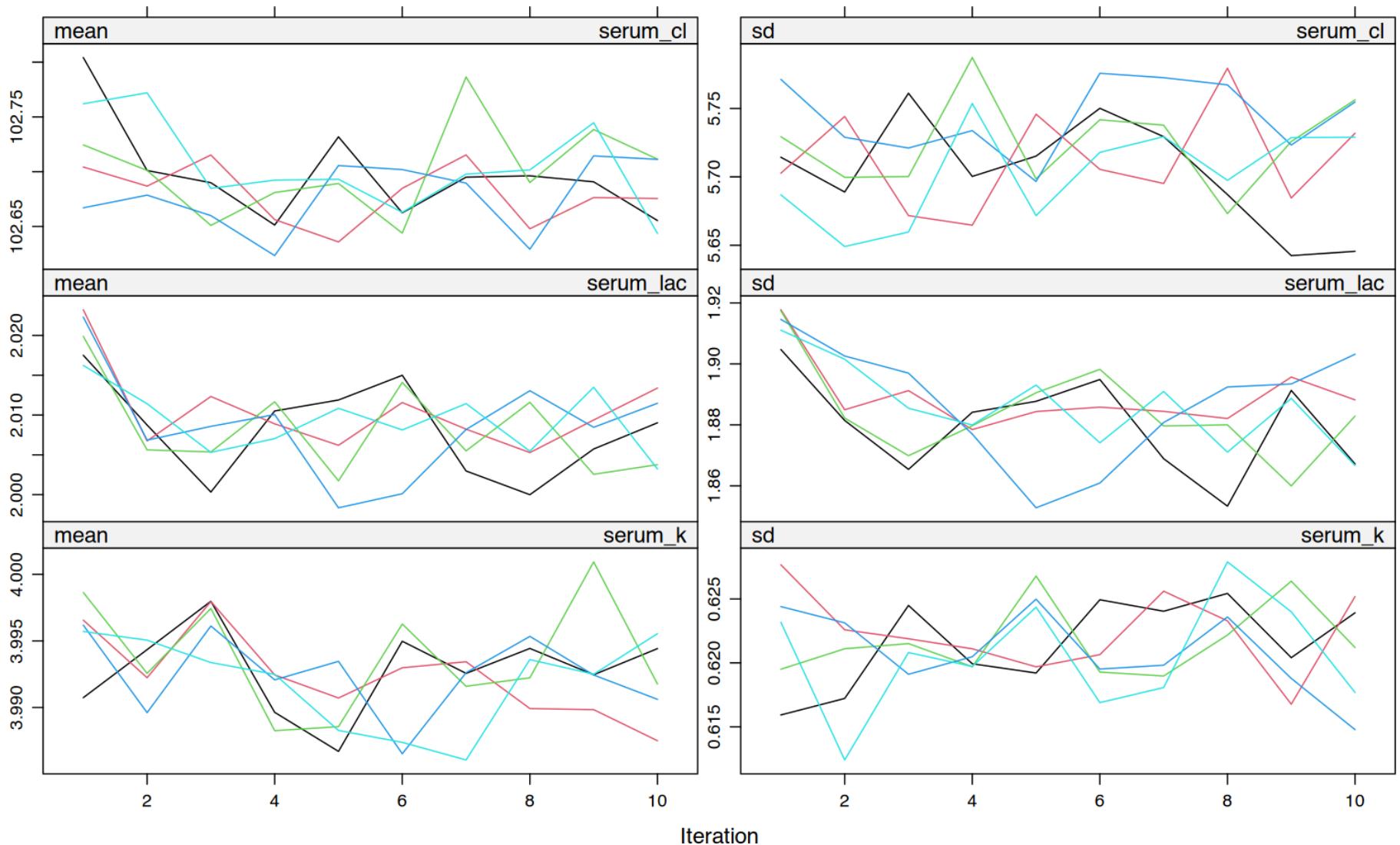
Chunk mi-exec runtime: 685.10 s

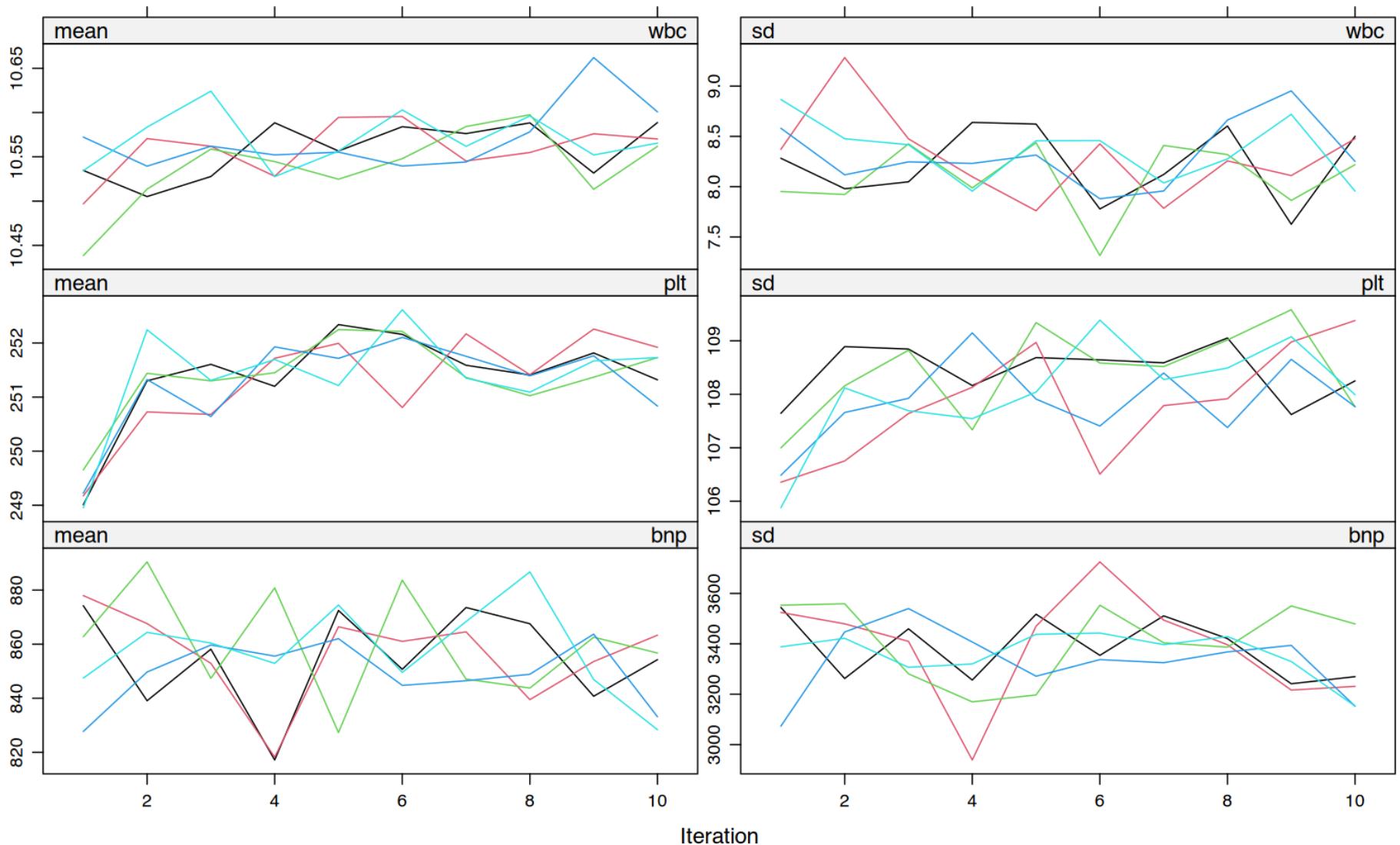
4.3.2 9.2 Convergence & plausibility checks

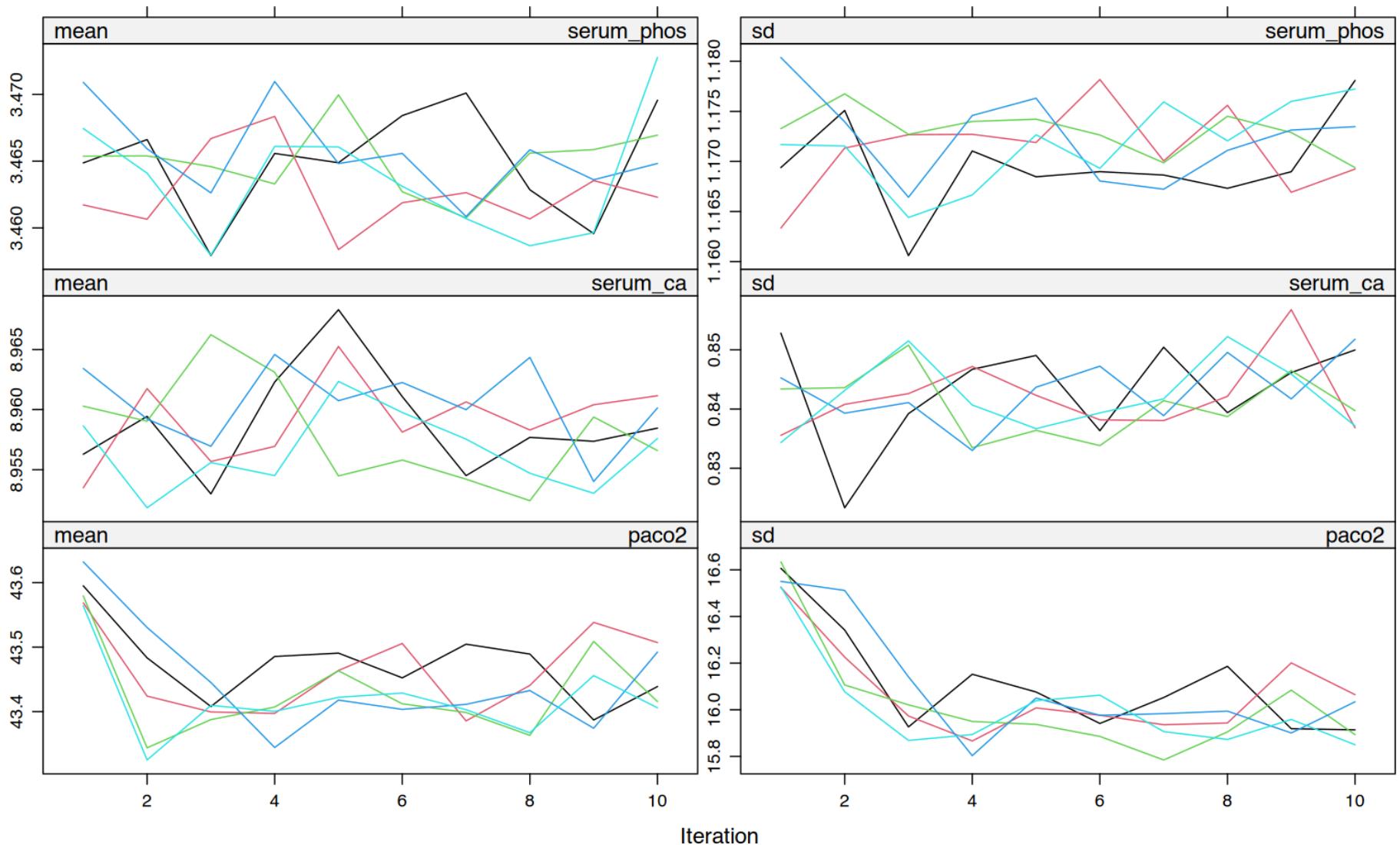


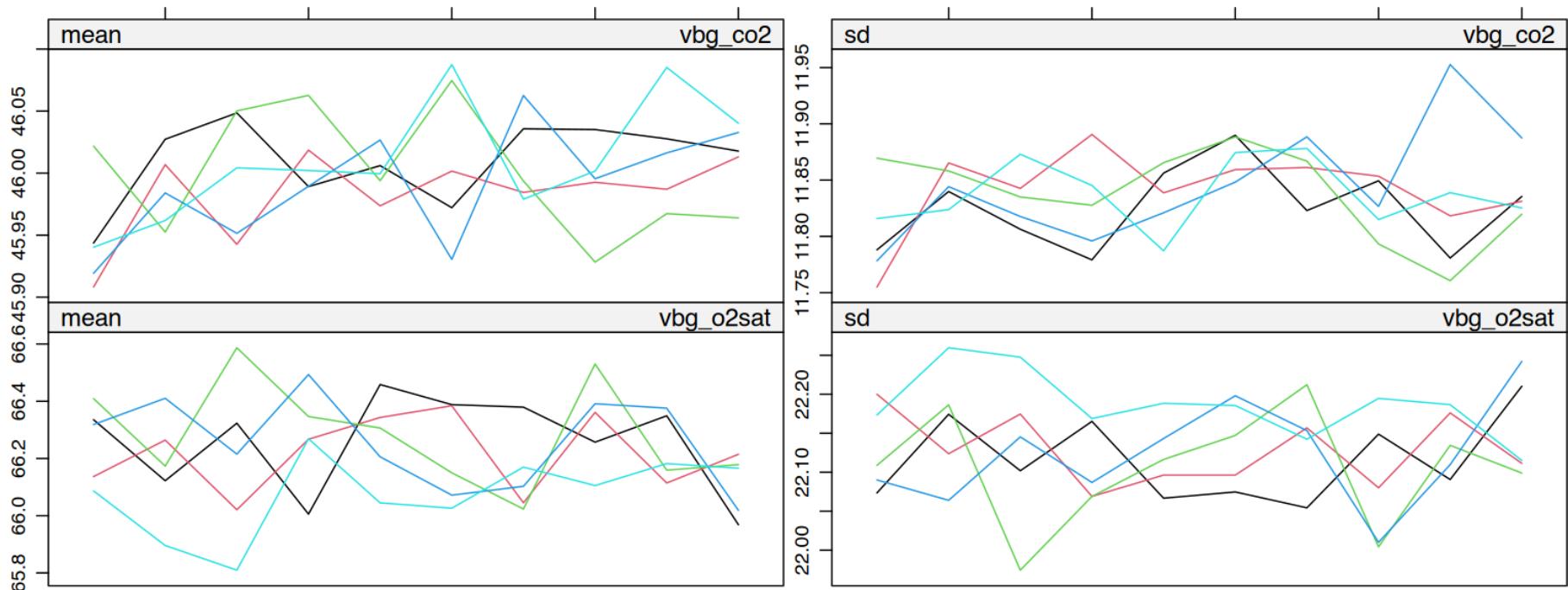




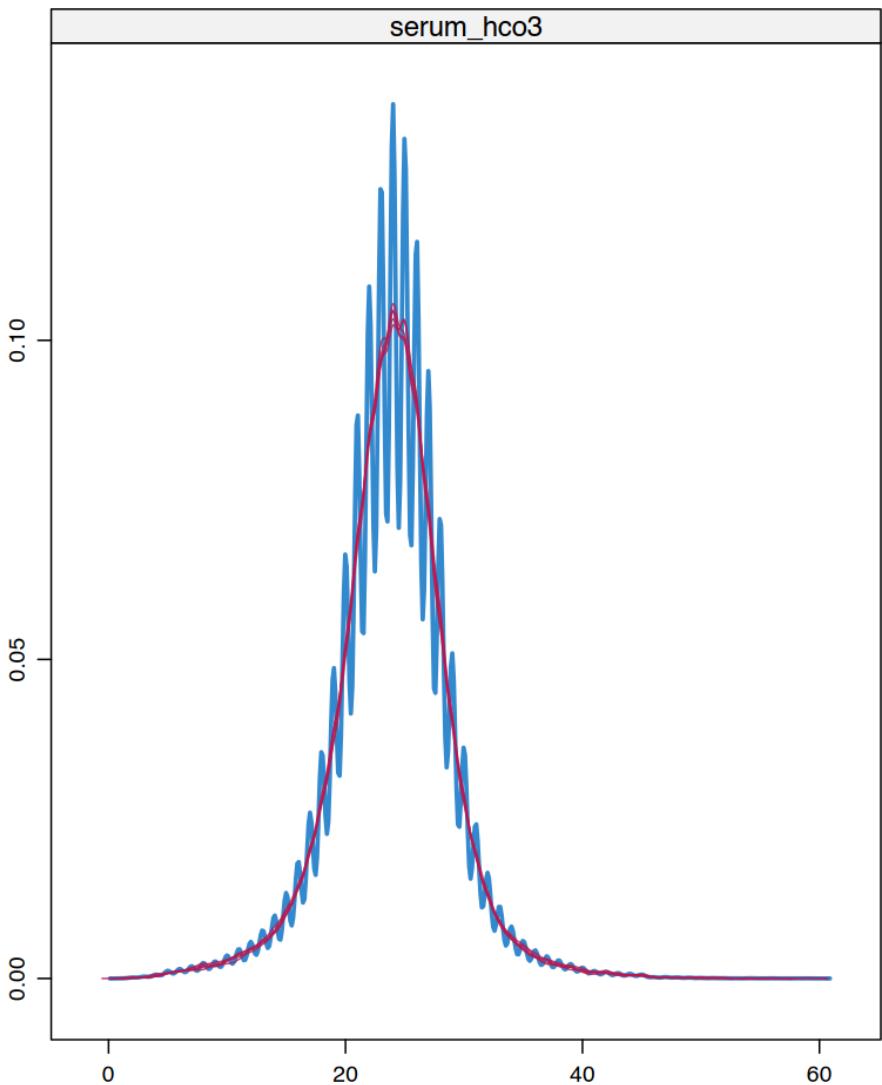
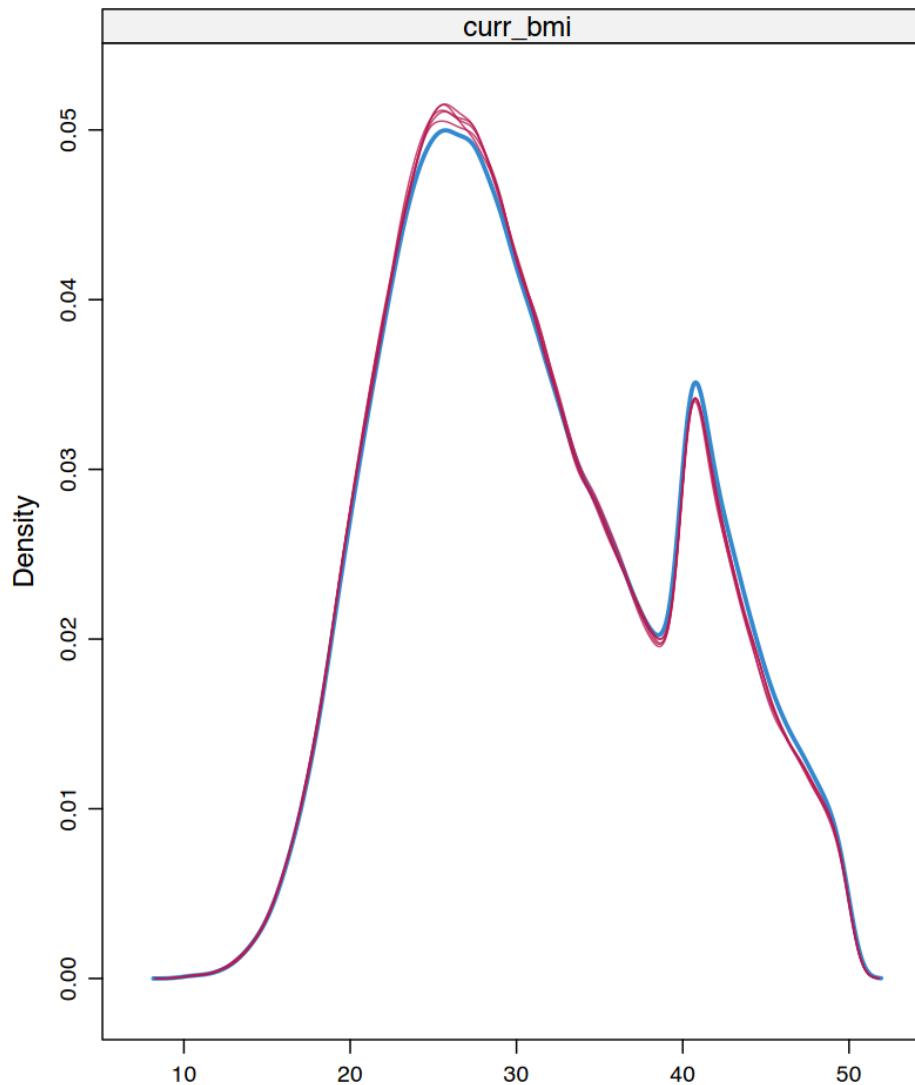


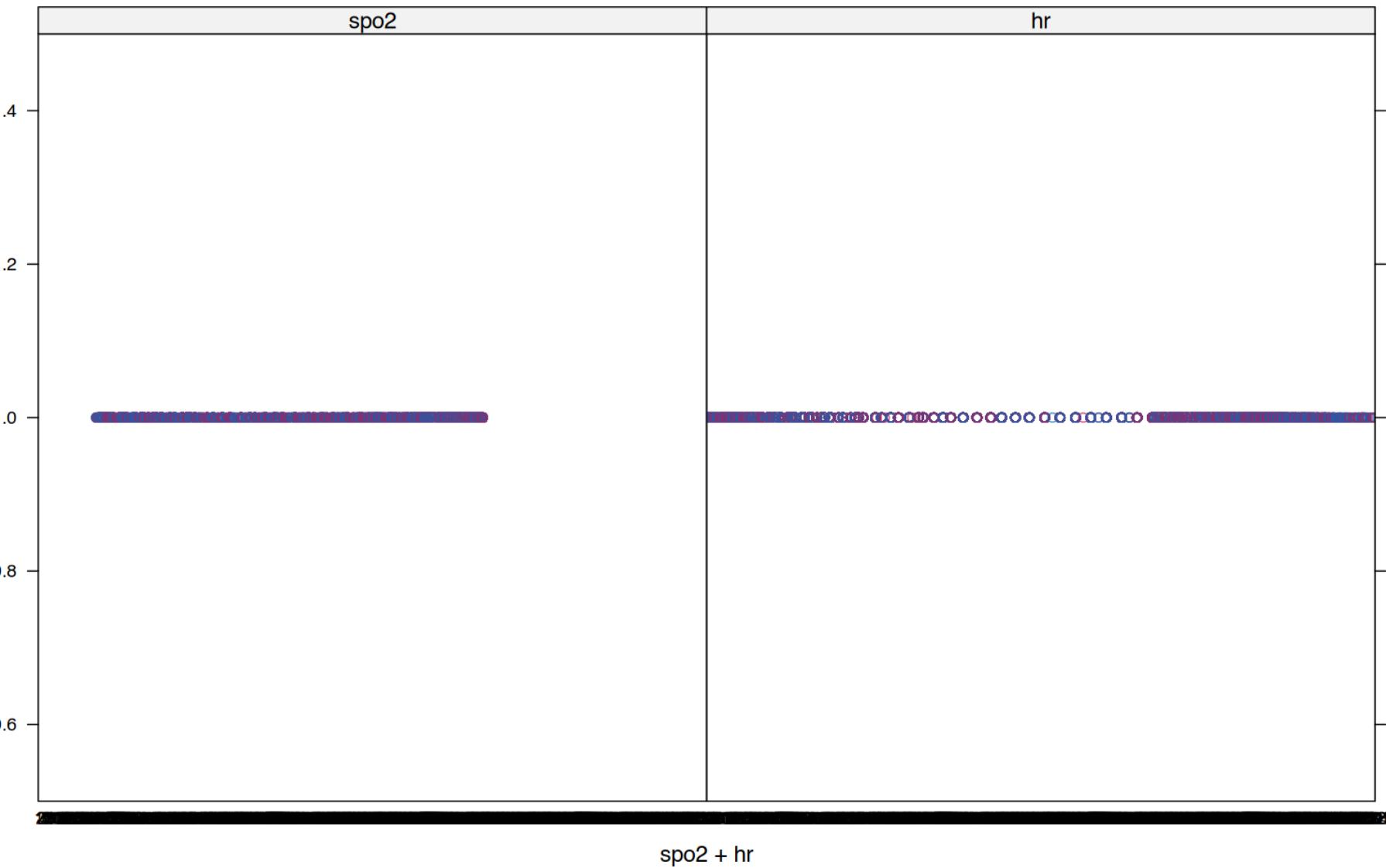






Iteration





Chunk mi-diagnostics runtime: 36.74 s

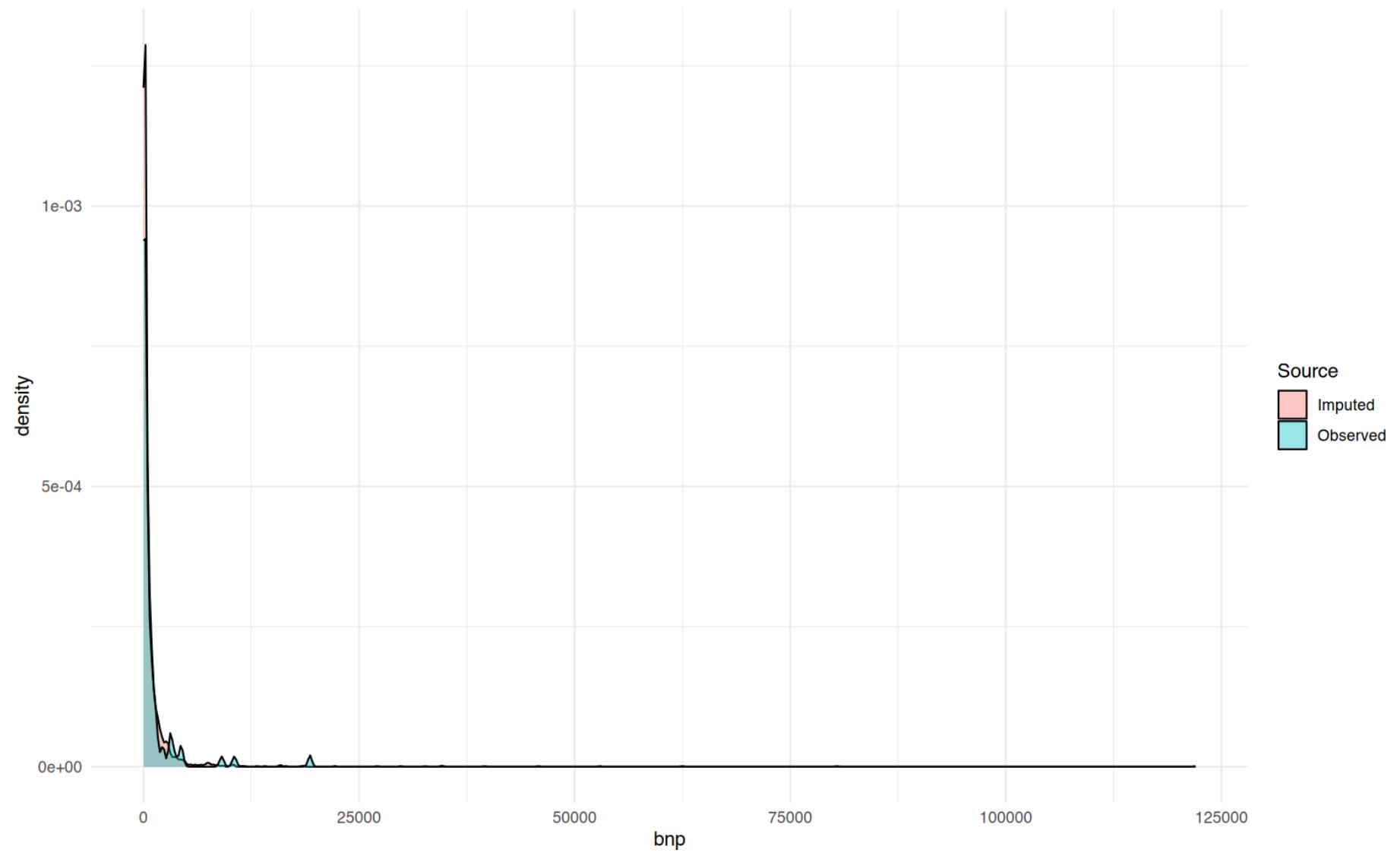
Chunk mi-missing-pattern runtime: 19.75 s

4.3.3 9.3 Observed vs imputed distributions (by strata)

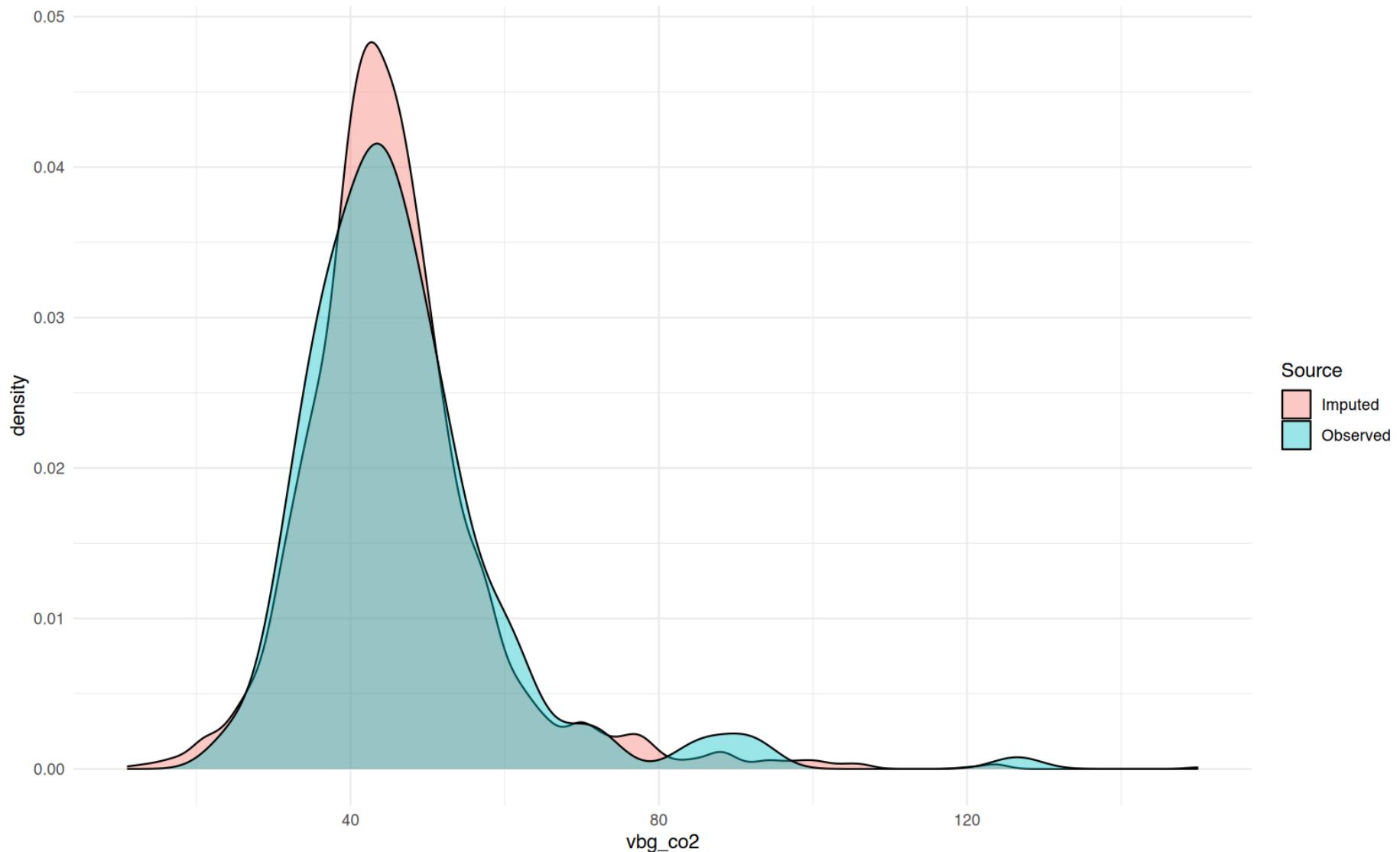
Observed vs imputed density: vbg_o2sat



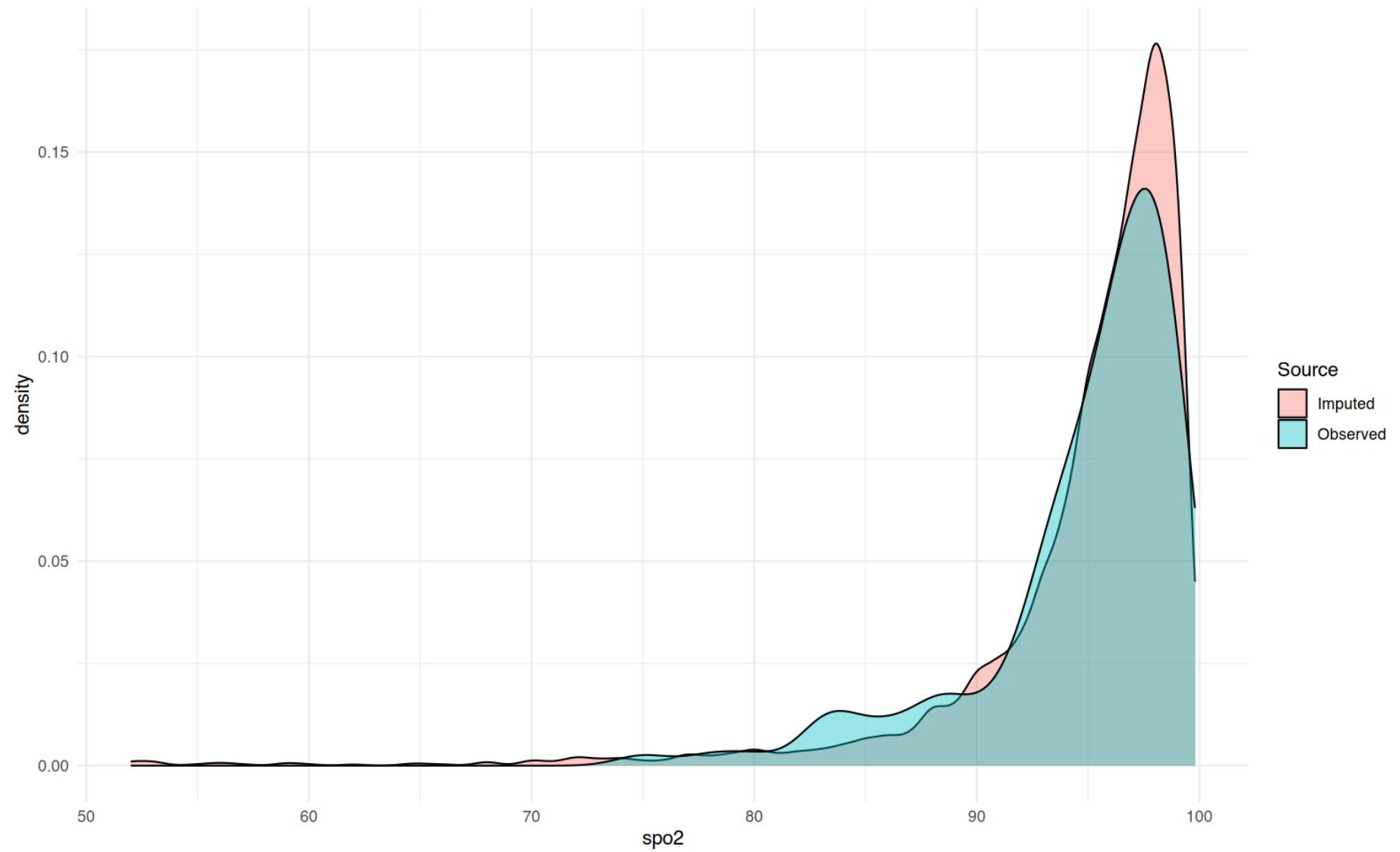
Observed vs imputed density: bnp



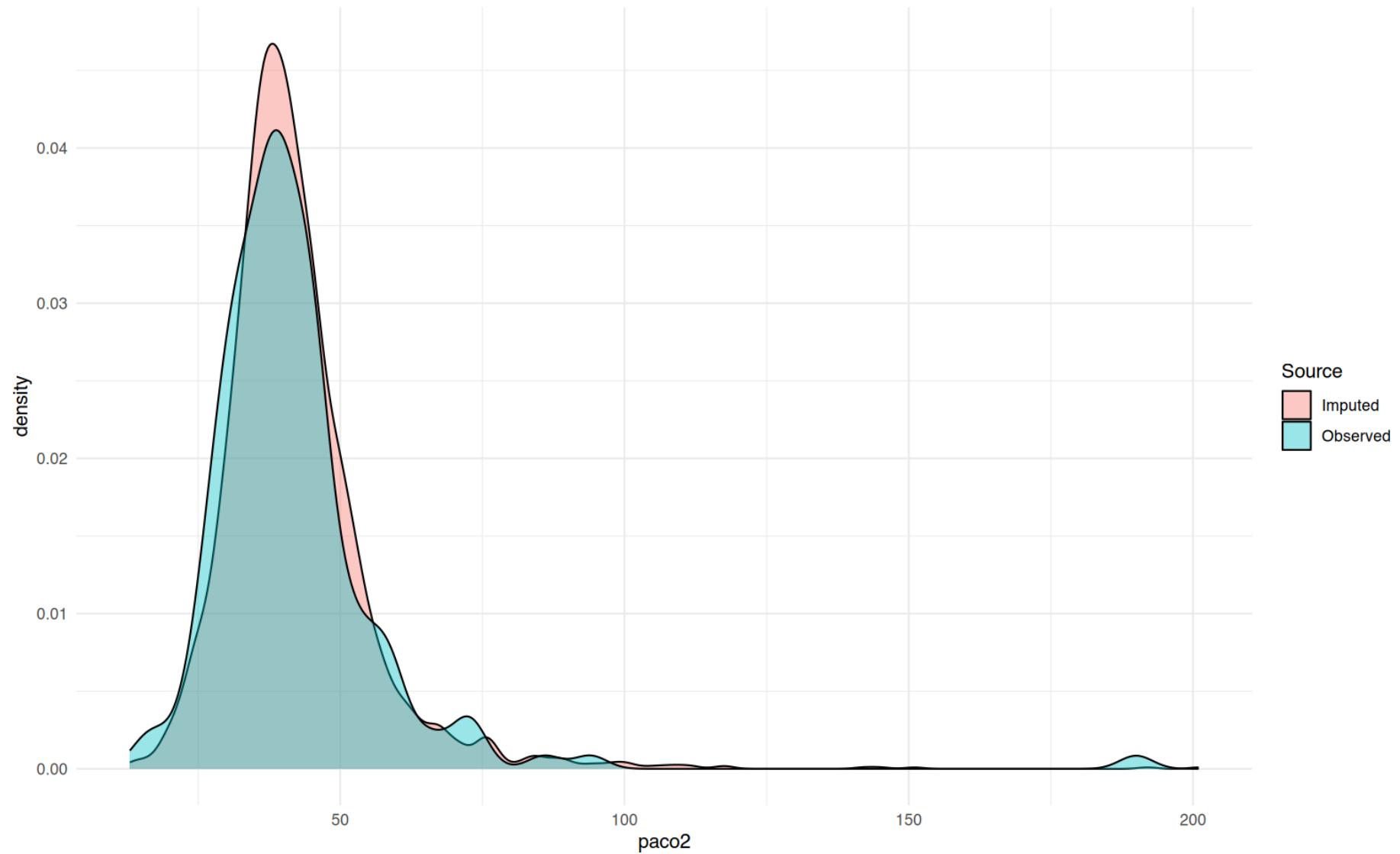
Observed vs imputed density: vbg_co2



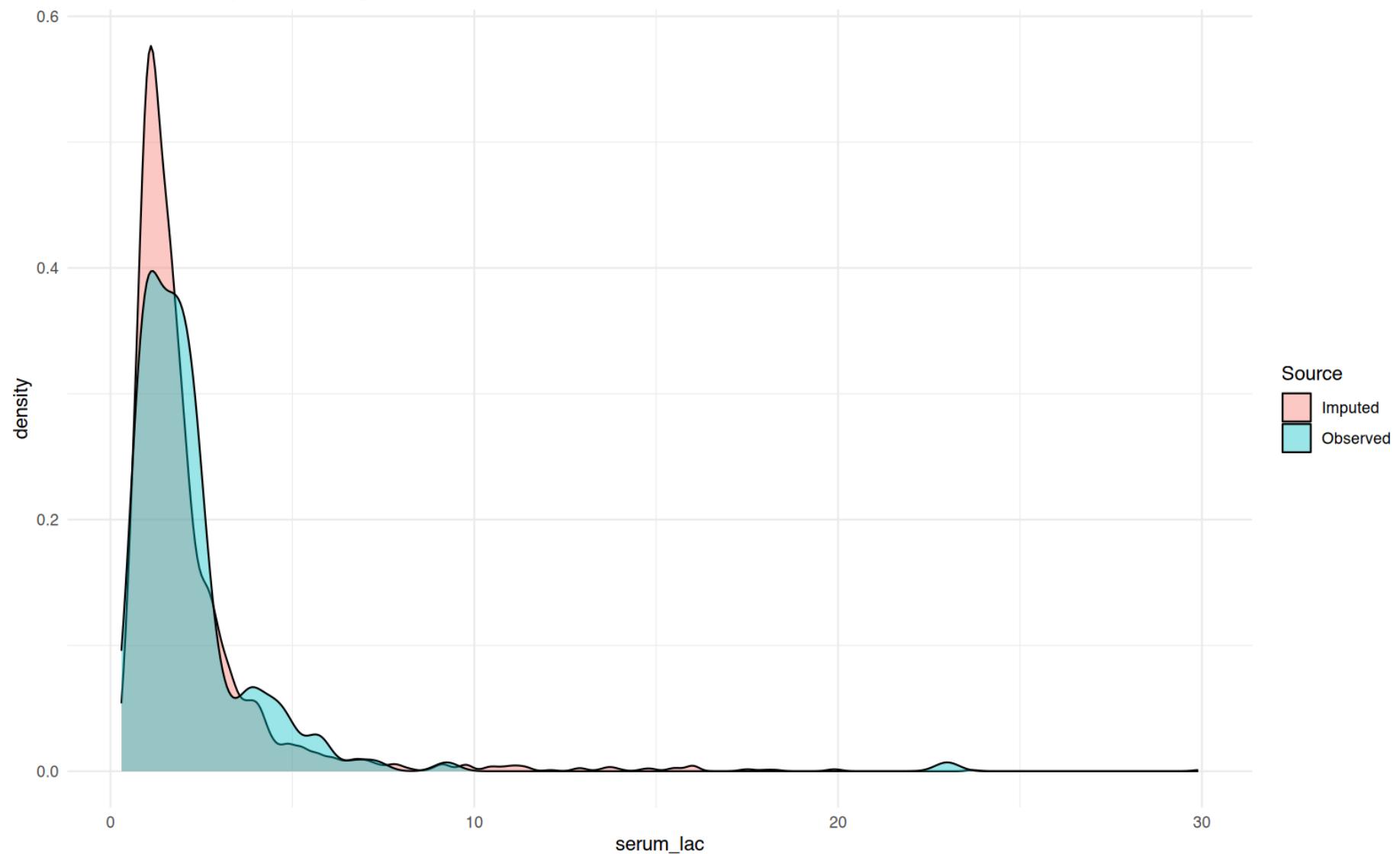
Observed vs imputed density: spo2



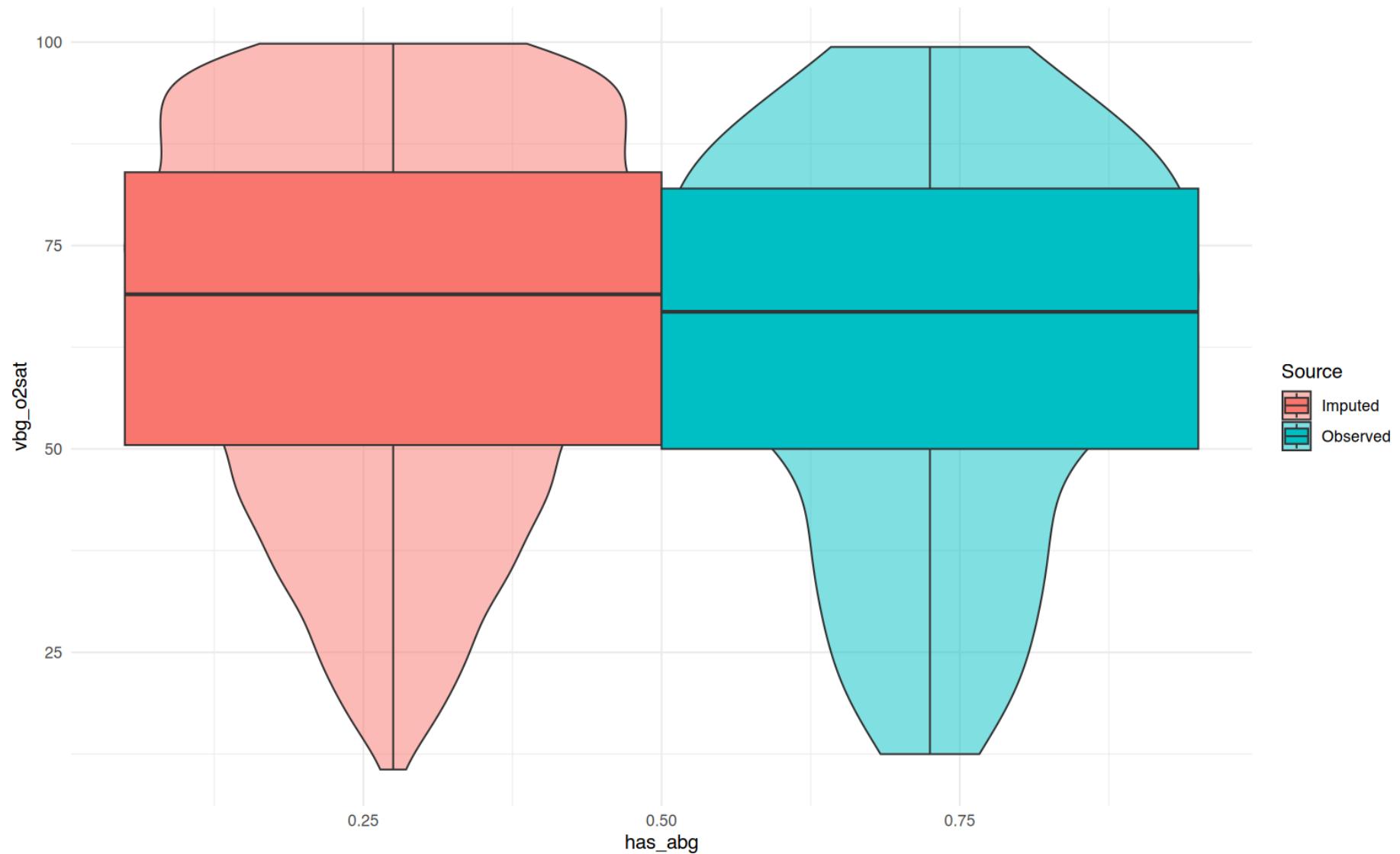
Observed vs imputed density: paco2



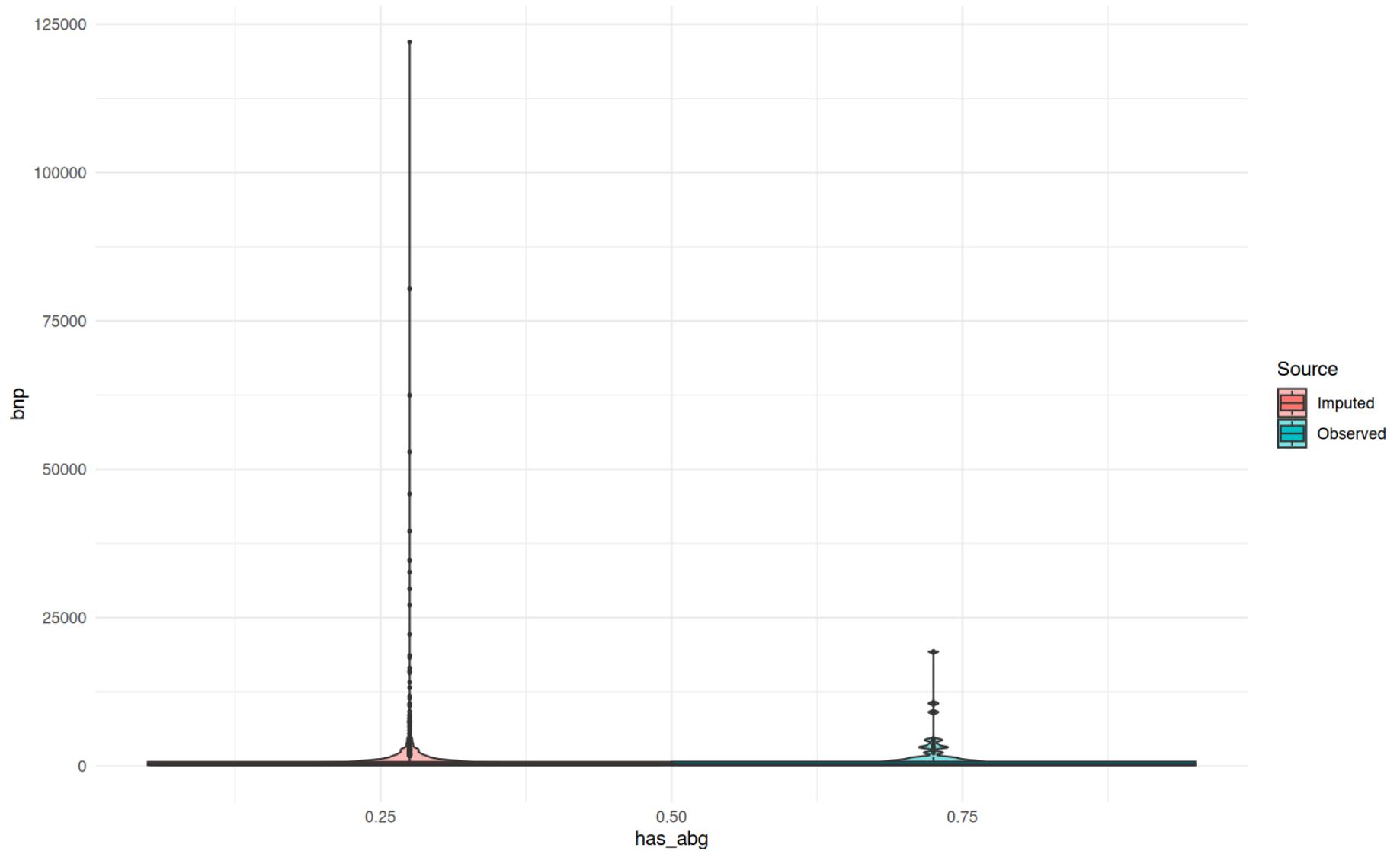
Observed vs imputed density: serum_lac



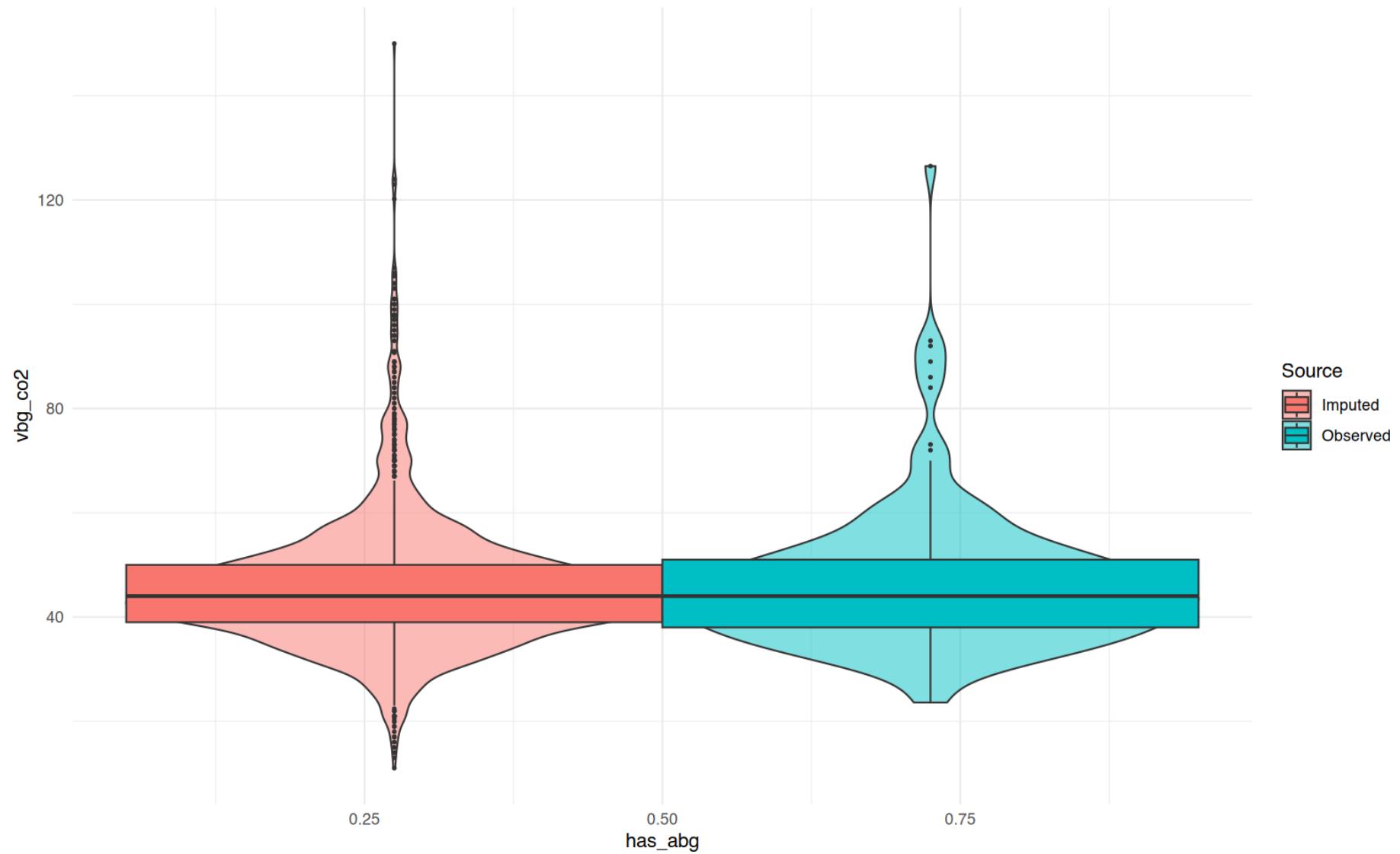
Observed vs imputed: vbg_o2sat by has_abg



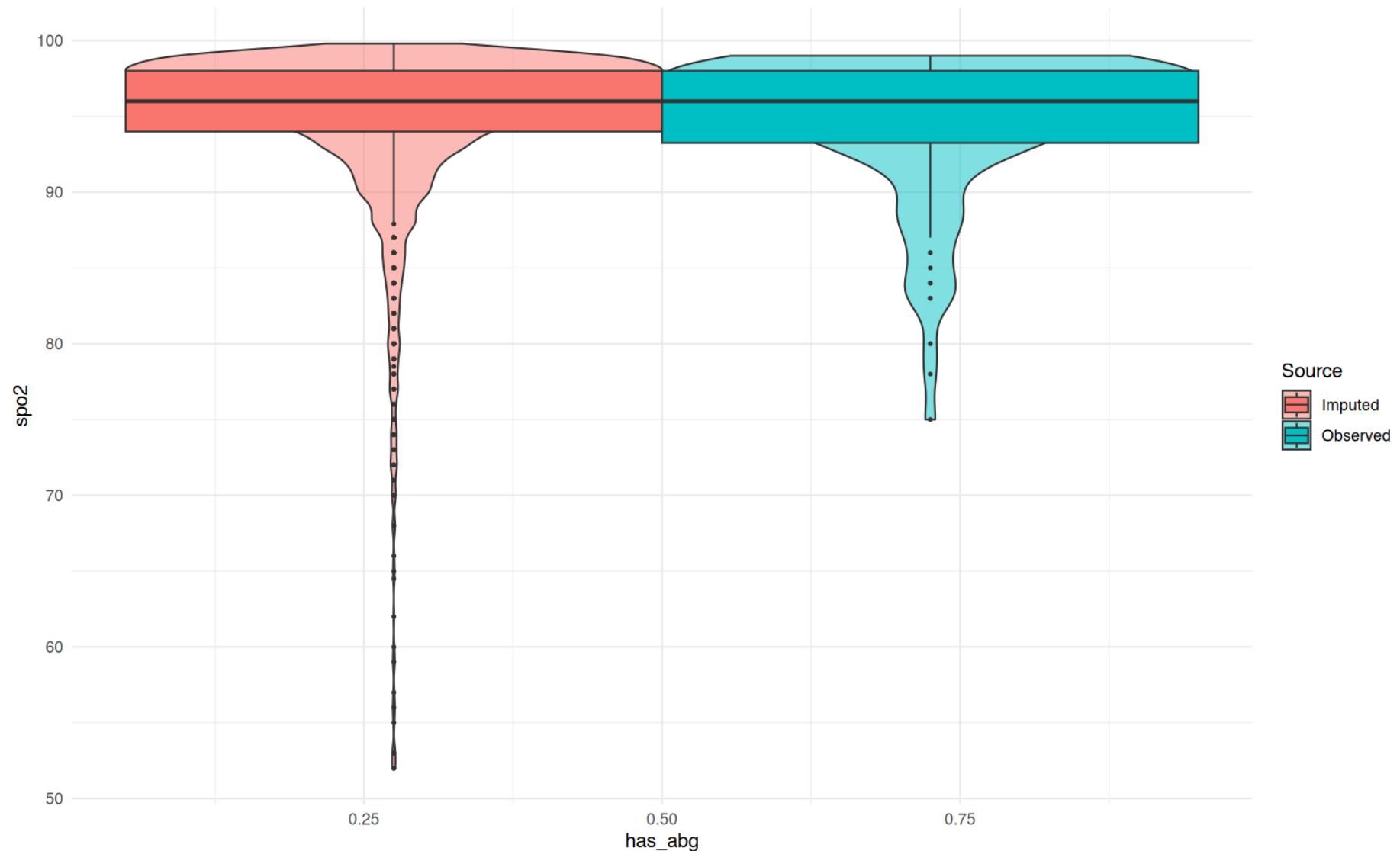
Observed vs imputed: bnp by has_abg



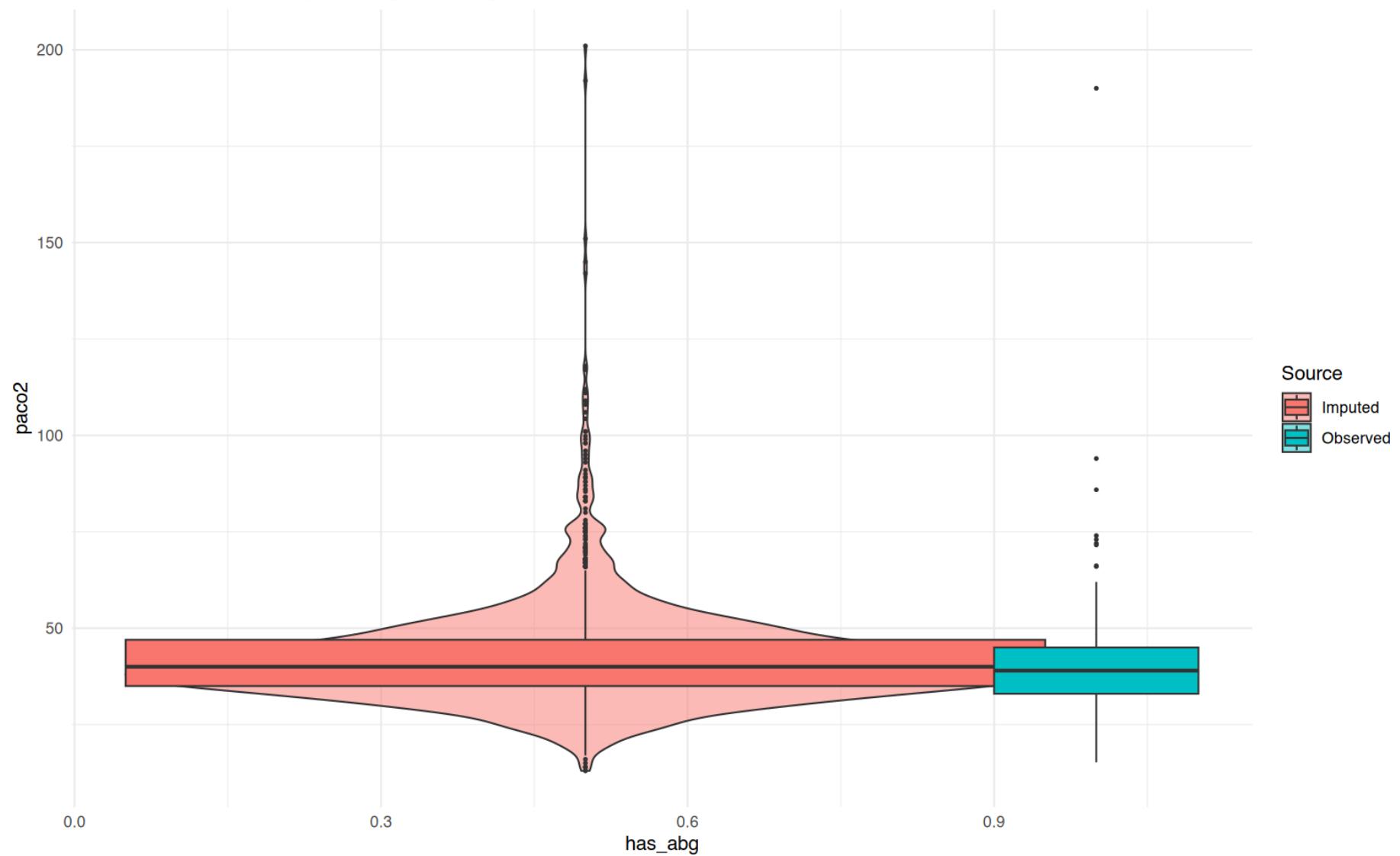
Observed vs imputed: vbg_co2 by has_abg



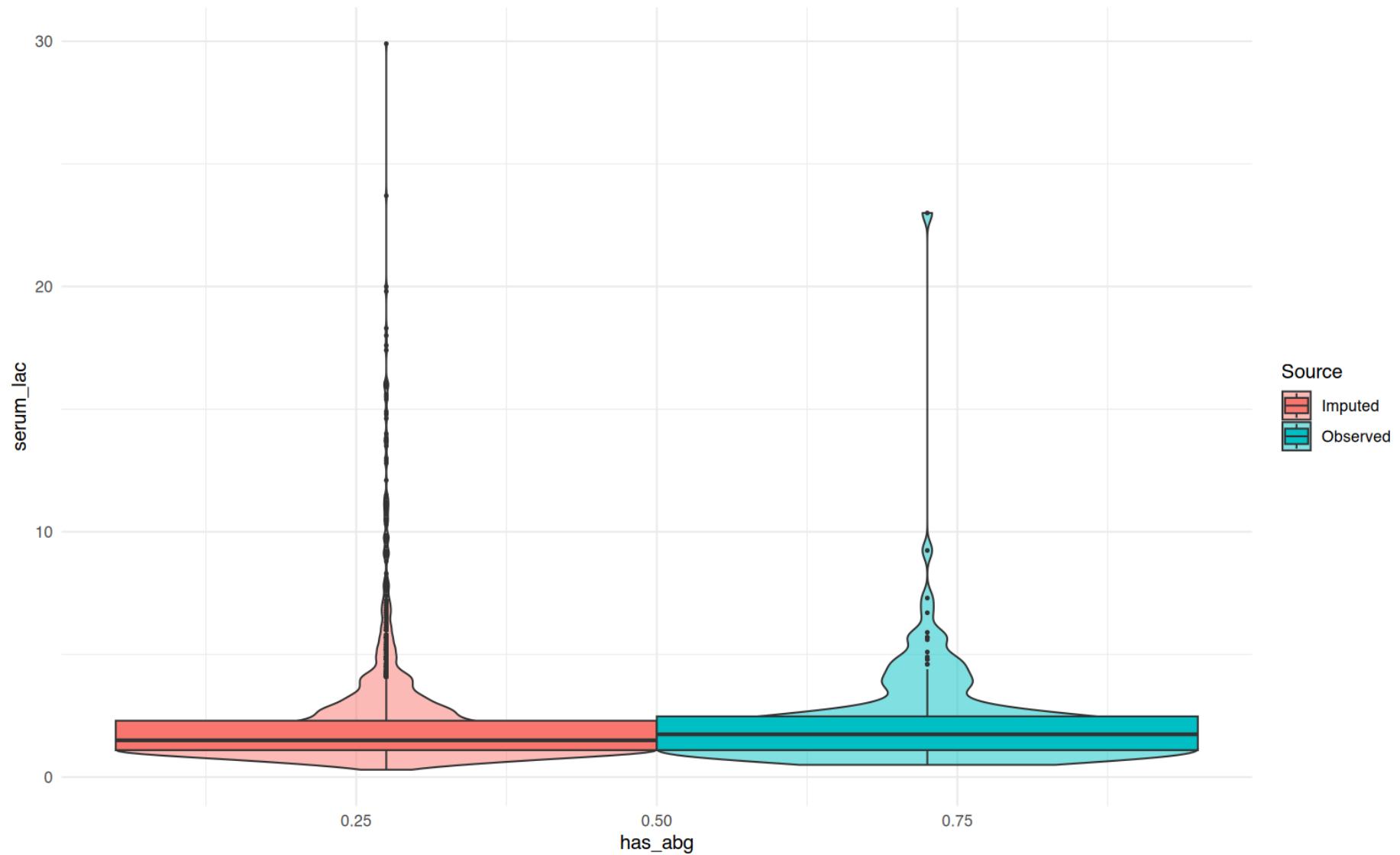
Observed vs imputed: spo2 by has_abg



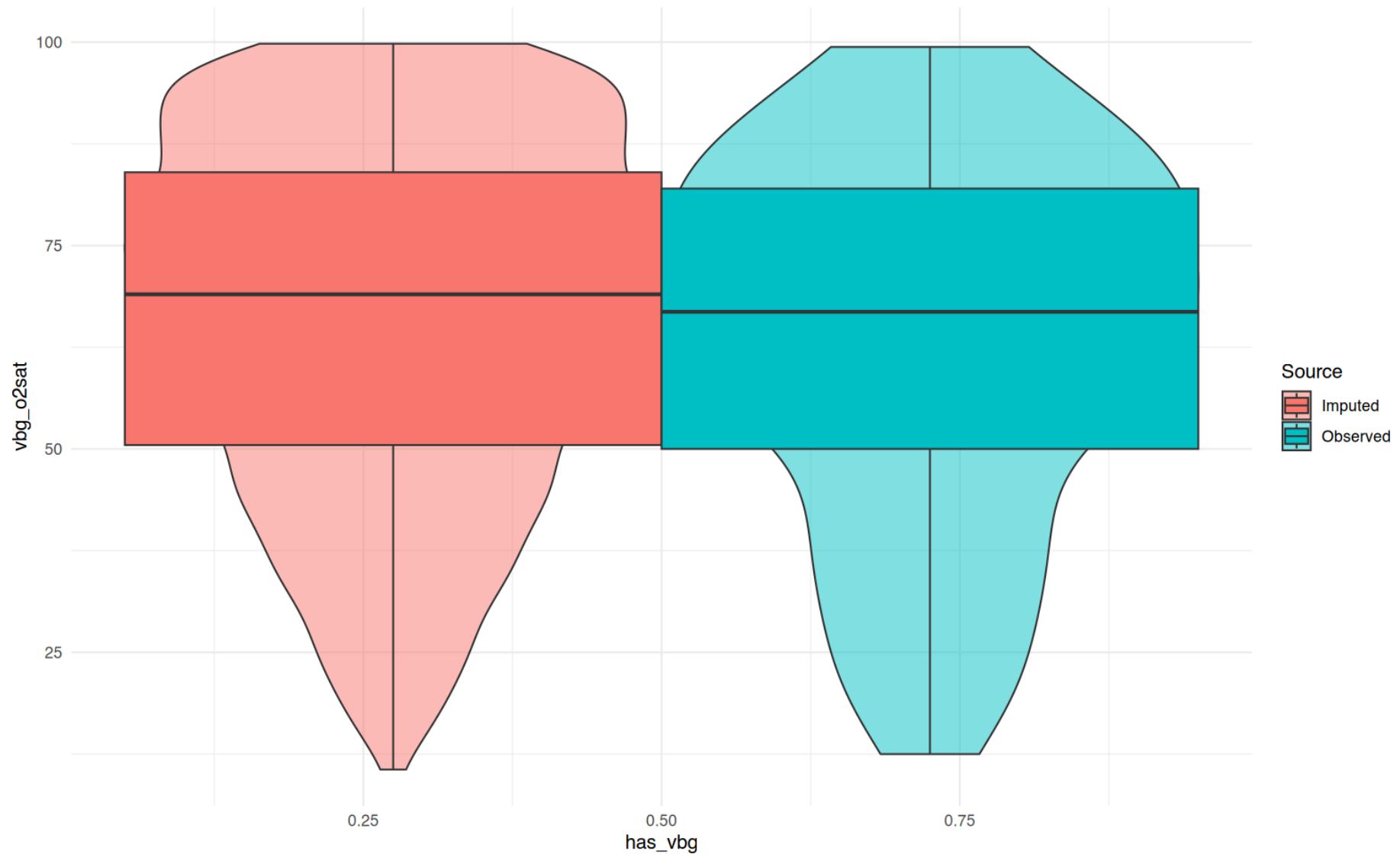
Observed vs imputed: paco2 by has_abg



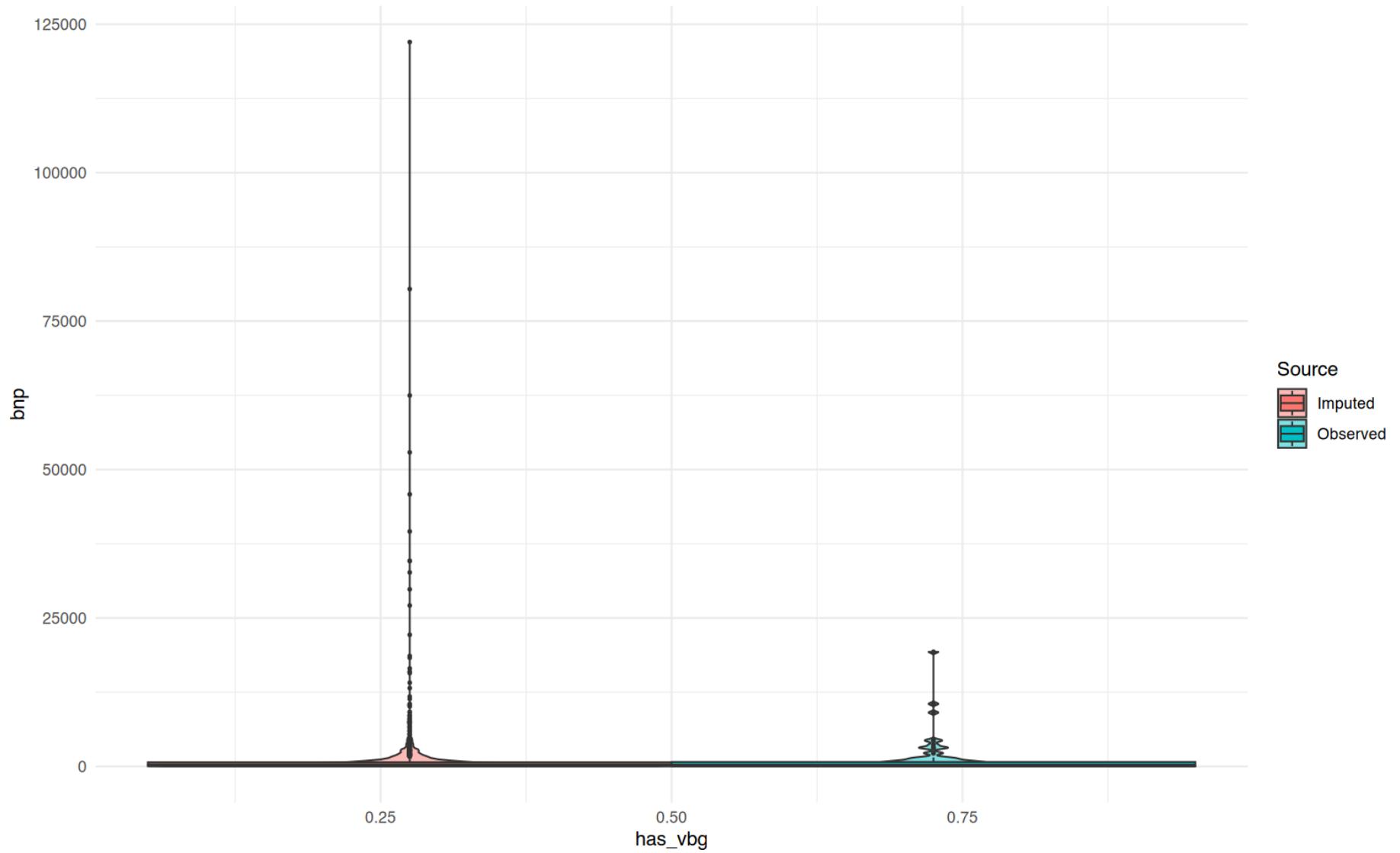
Observed vs imputed: serum_lac by has_abg



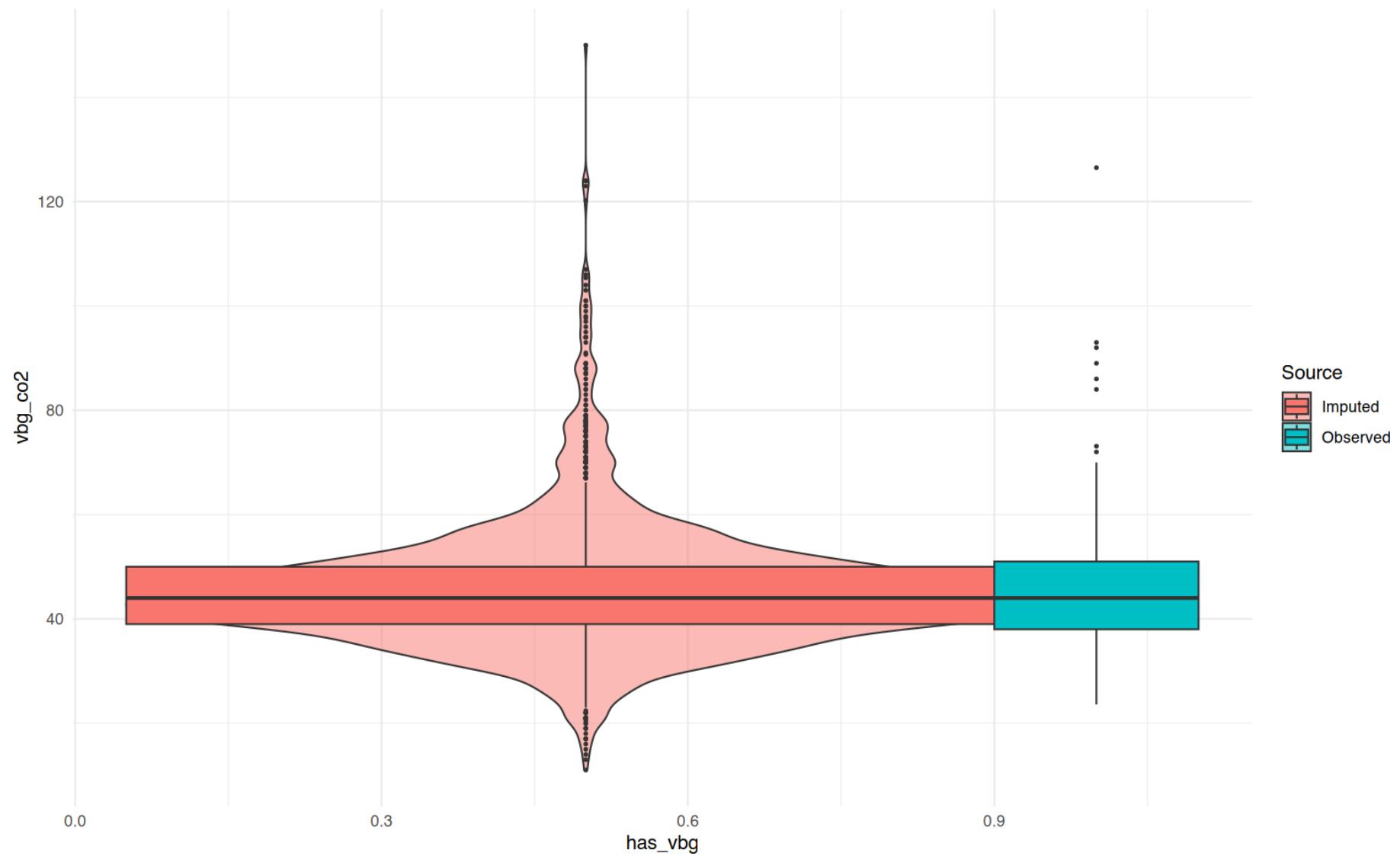
Observed vs imputed: vbg_o2sat by has_vbg



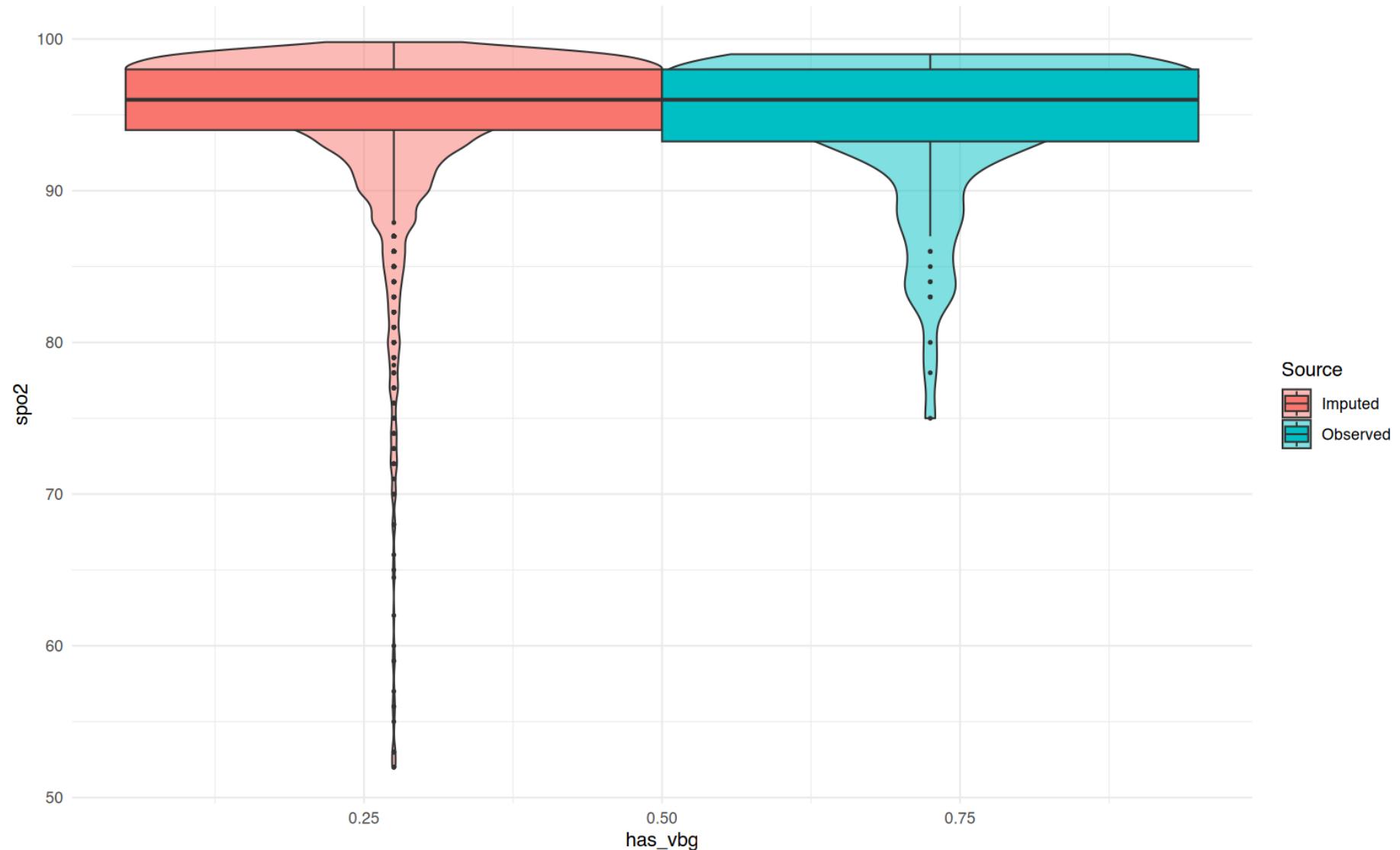
Observed vs imputed: bnp by has_vbg



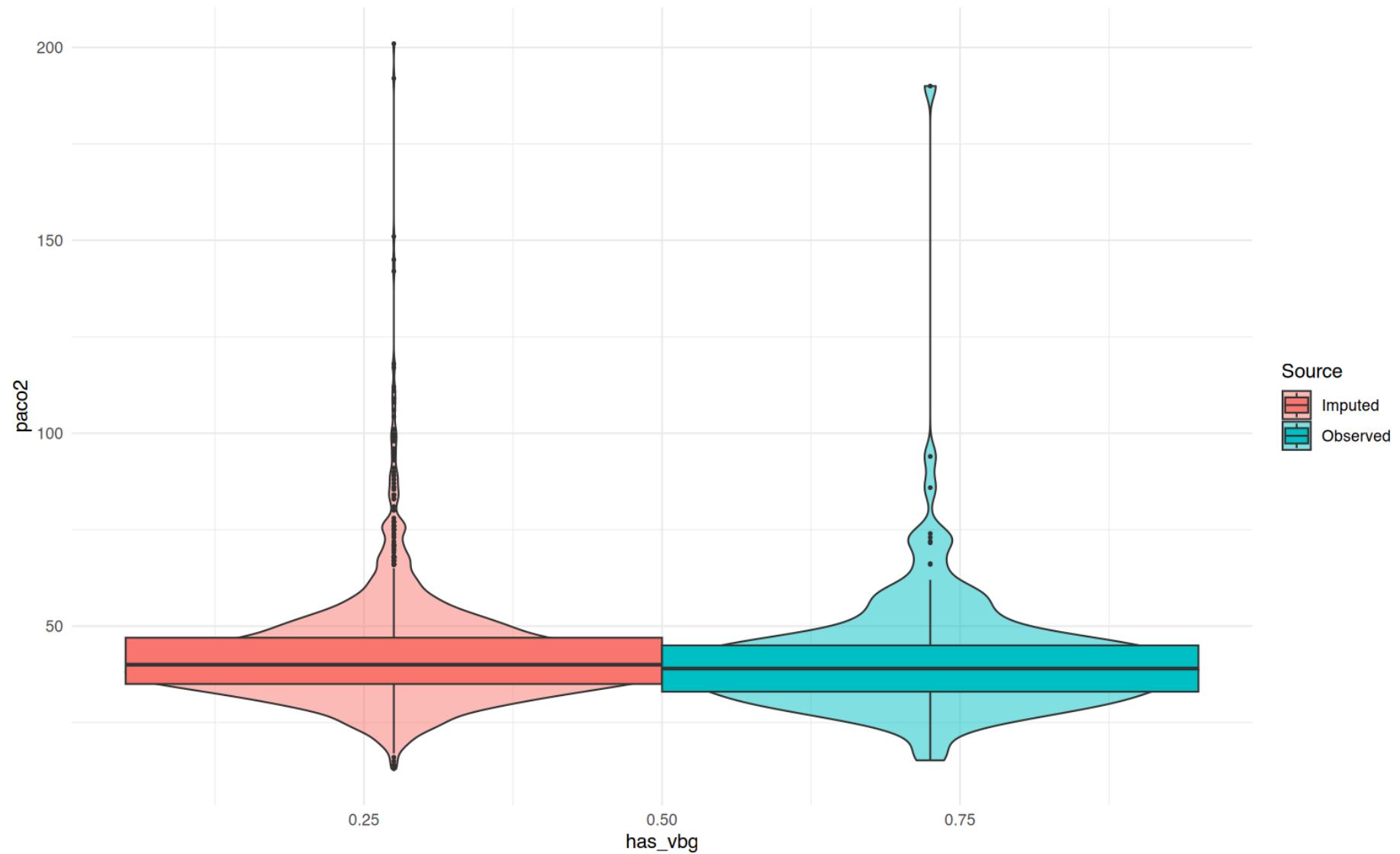
Observed vs imputed: vbg_co2 by has_vbg



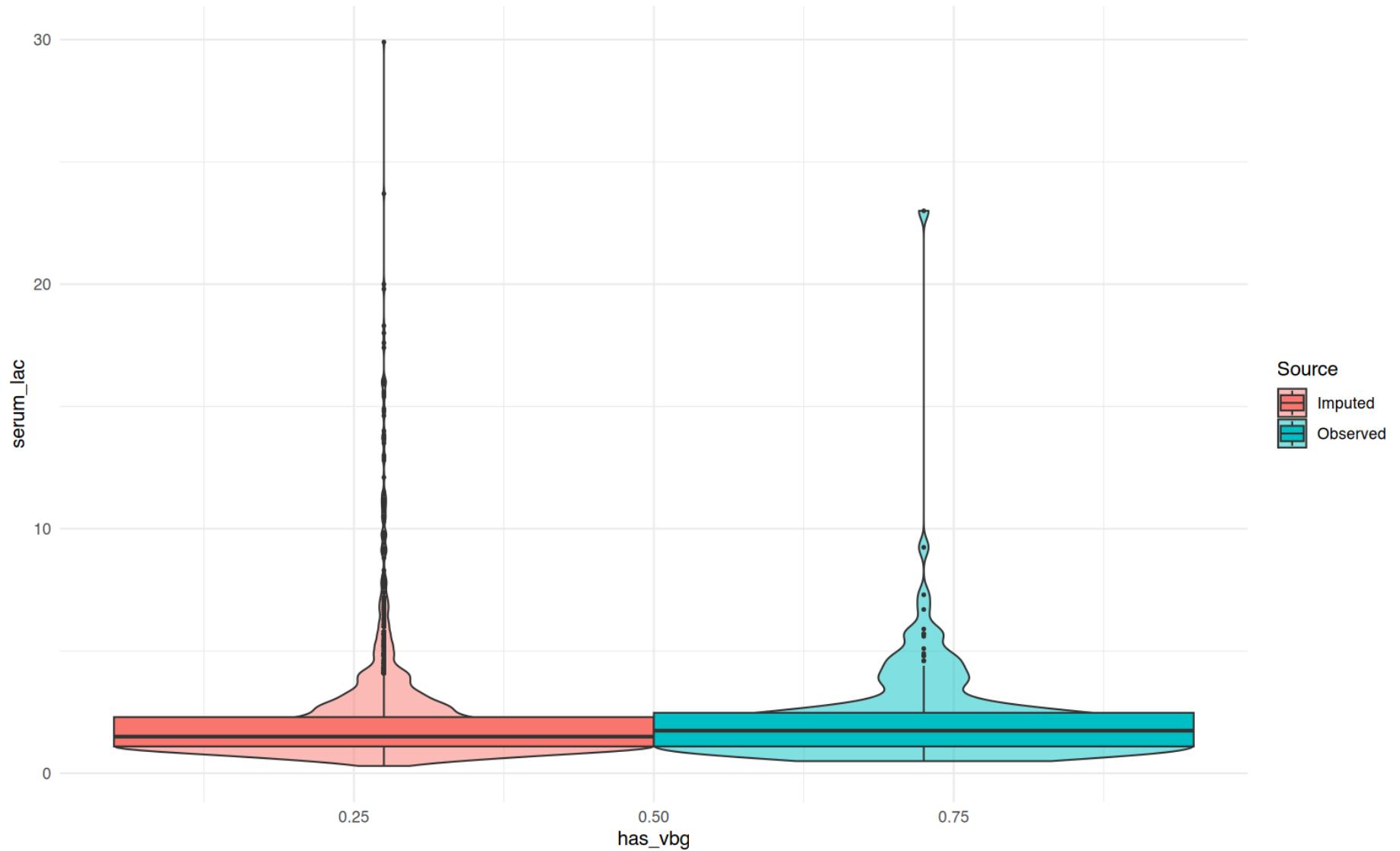
Observed vs imputed: spo2 by has_vbg



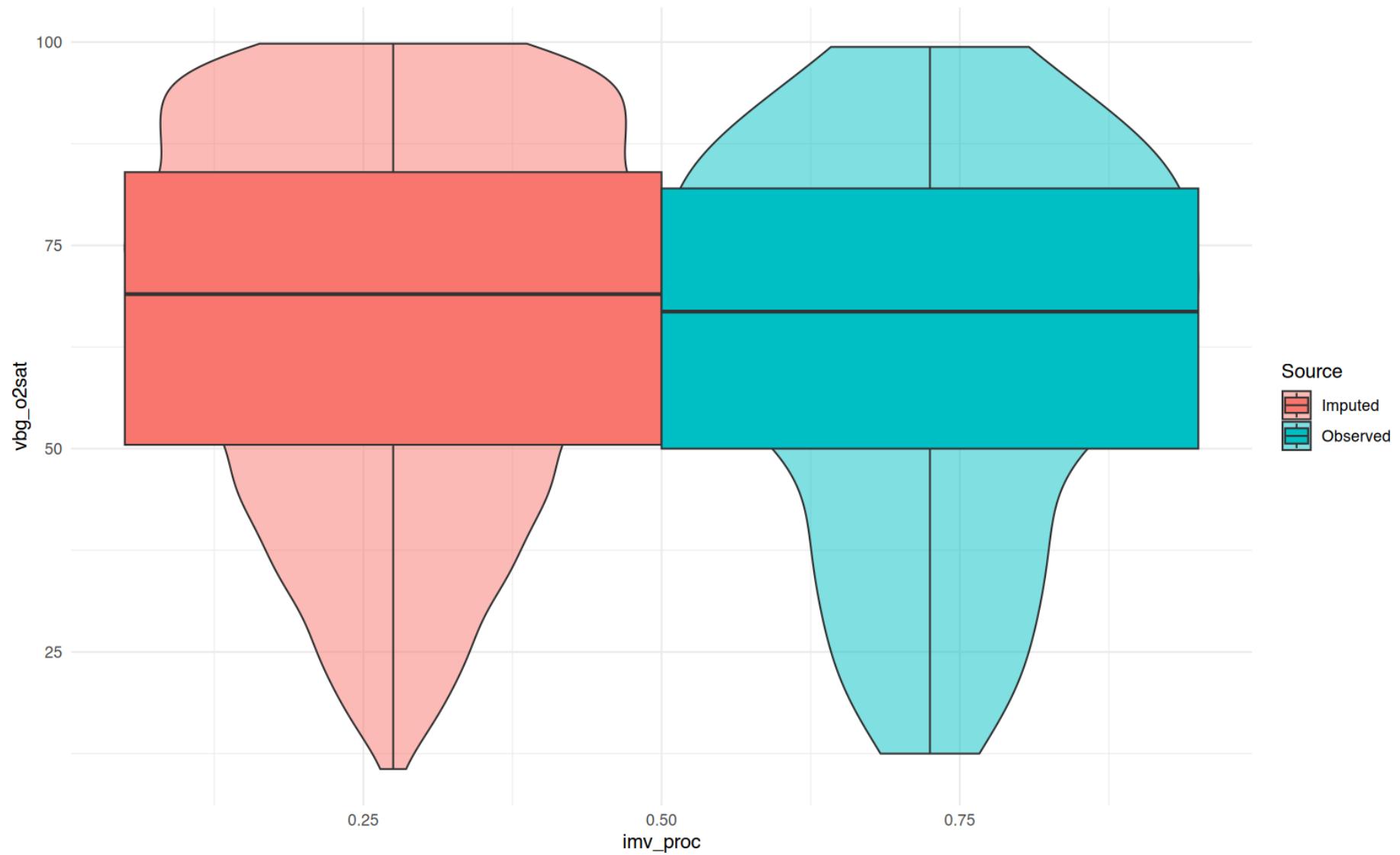
Observed vs imputed: paco2 by has_vbg



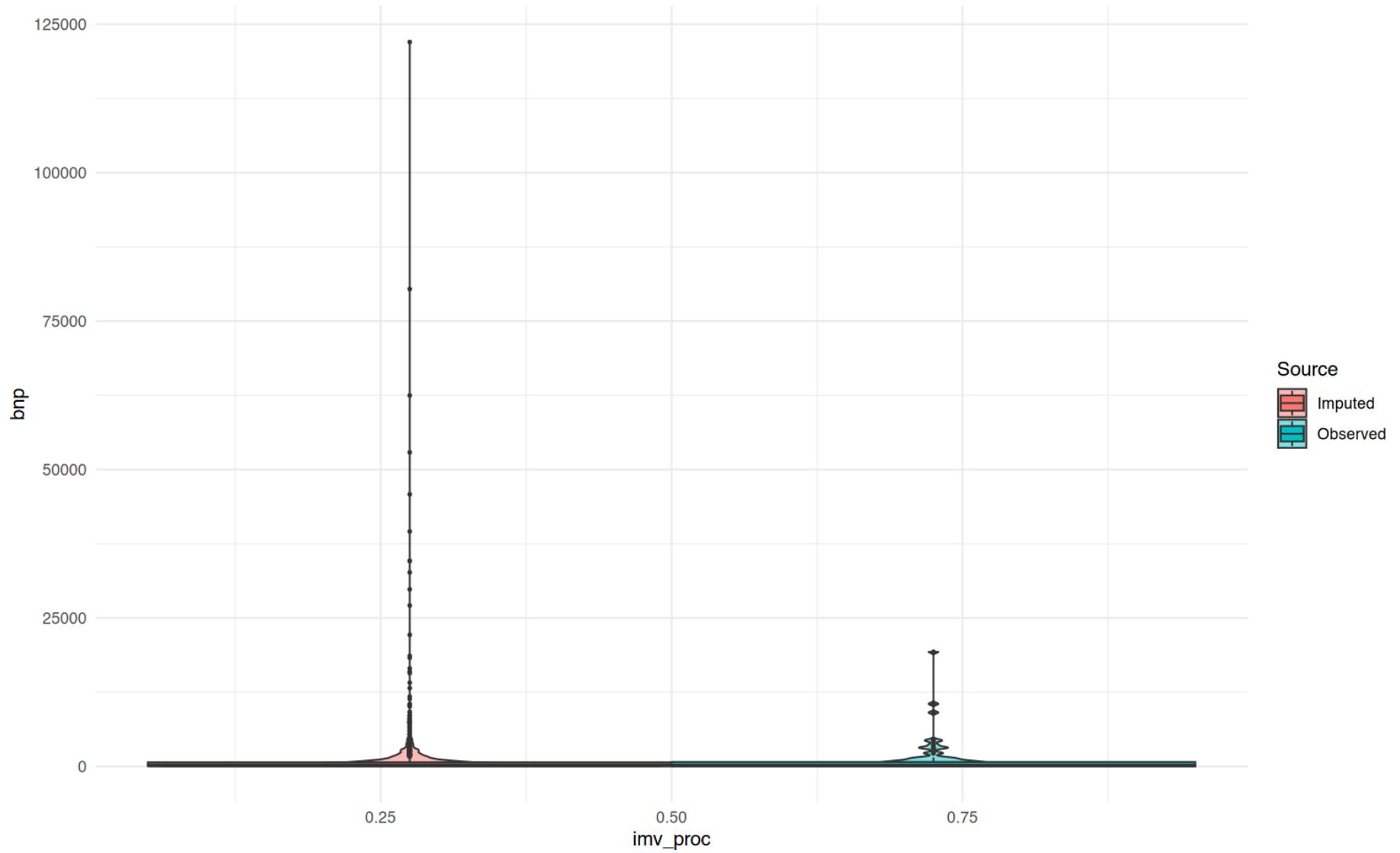
Observed vs imputed: serum_lac by has_vbg



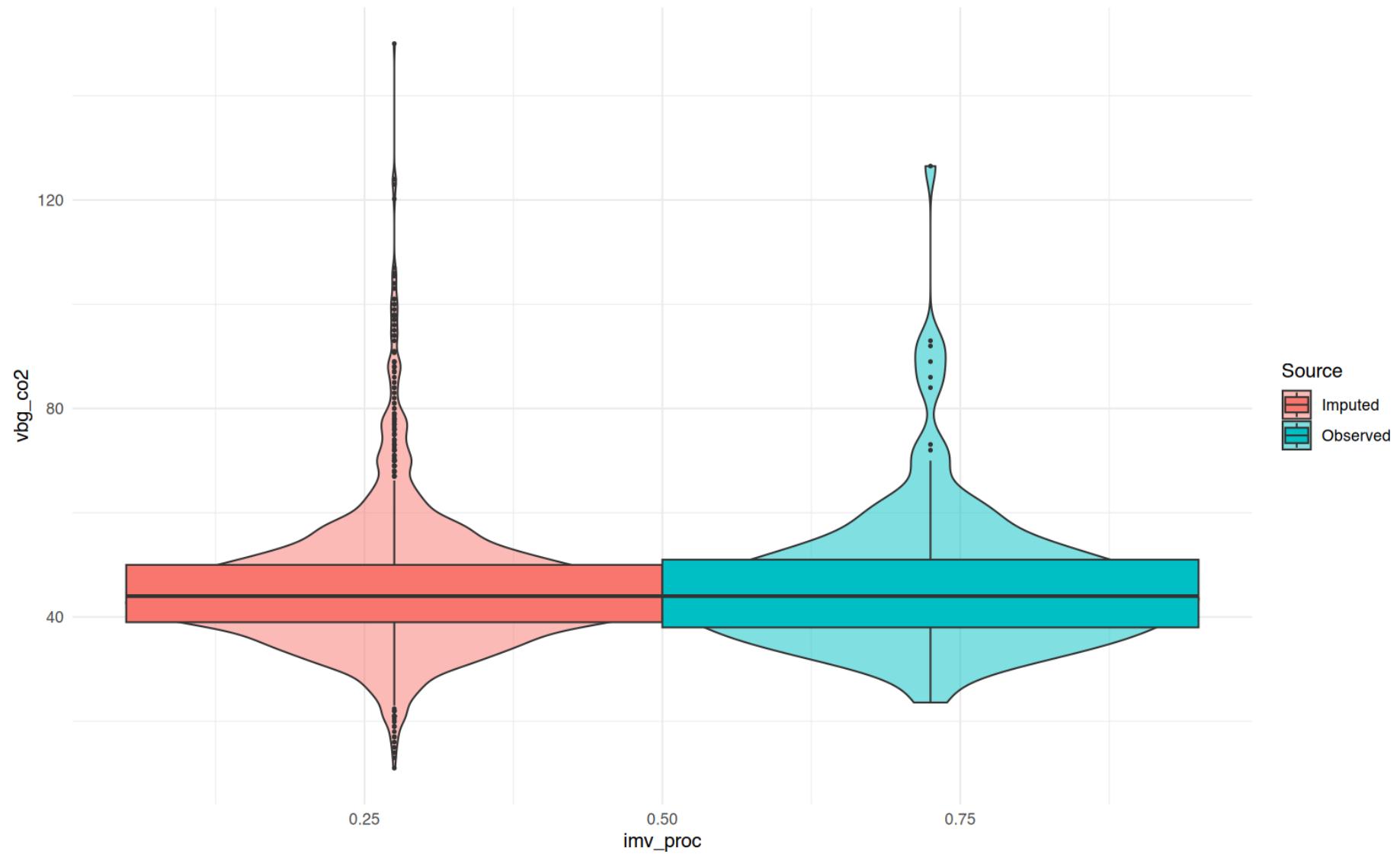
Observed vs imputed: vbg_o2sat by imv_proc



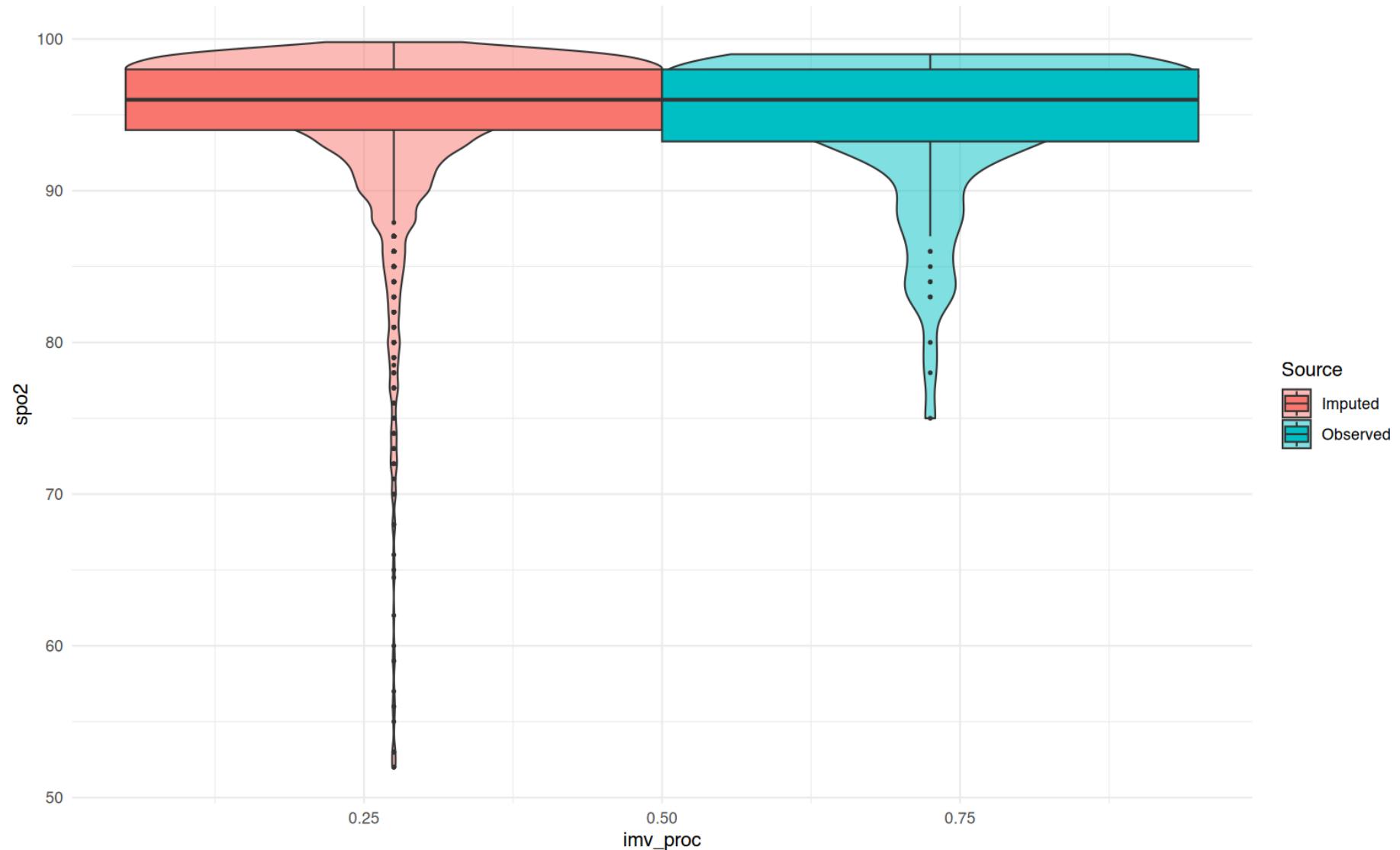
Observed vs imputed: bnp by imv_proc



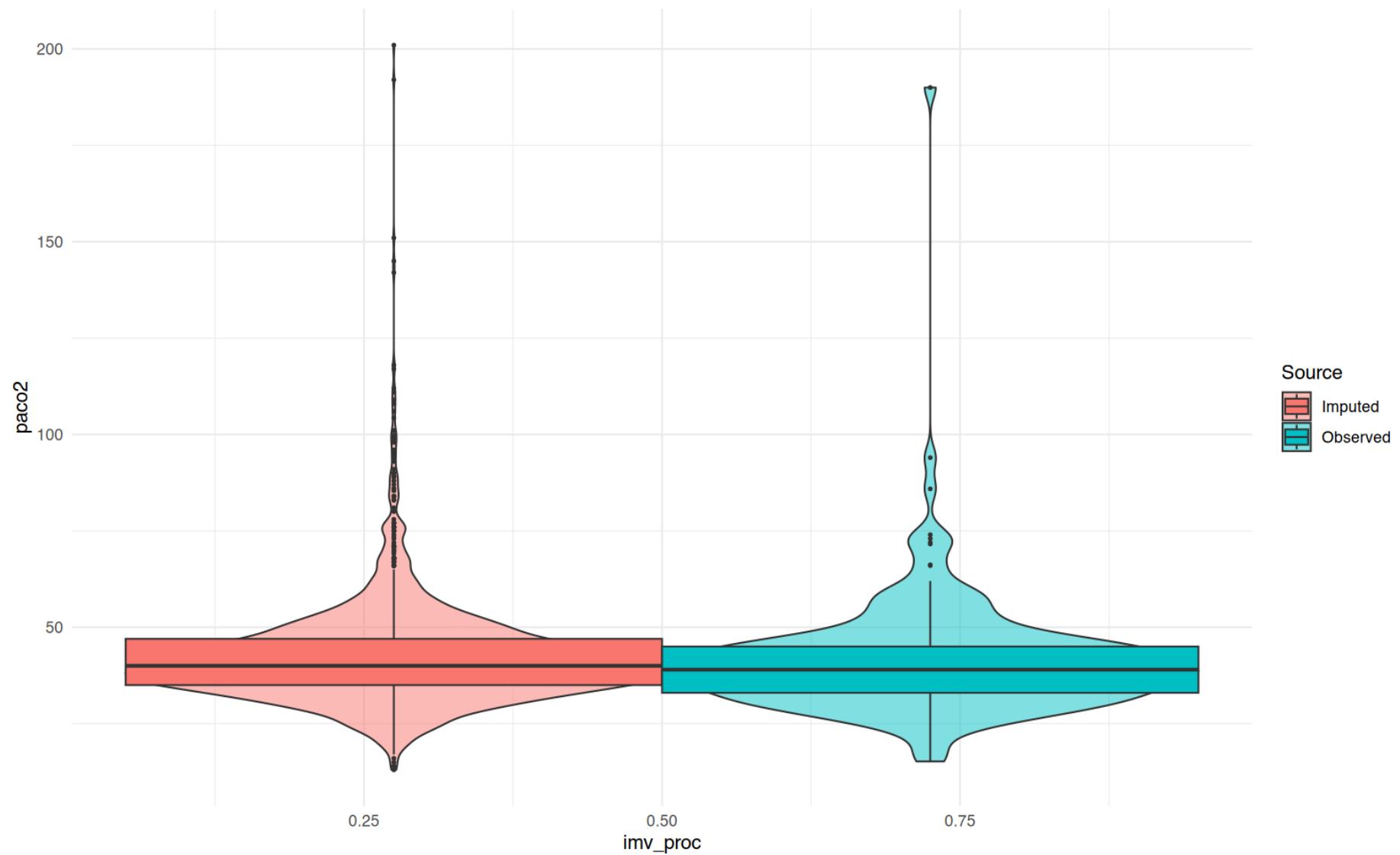
Observed vs imputed: vbg_co2 by imv_proc



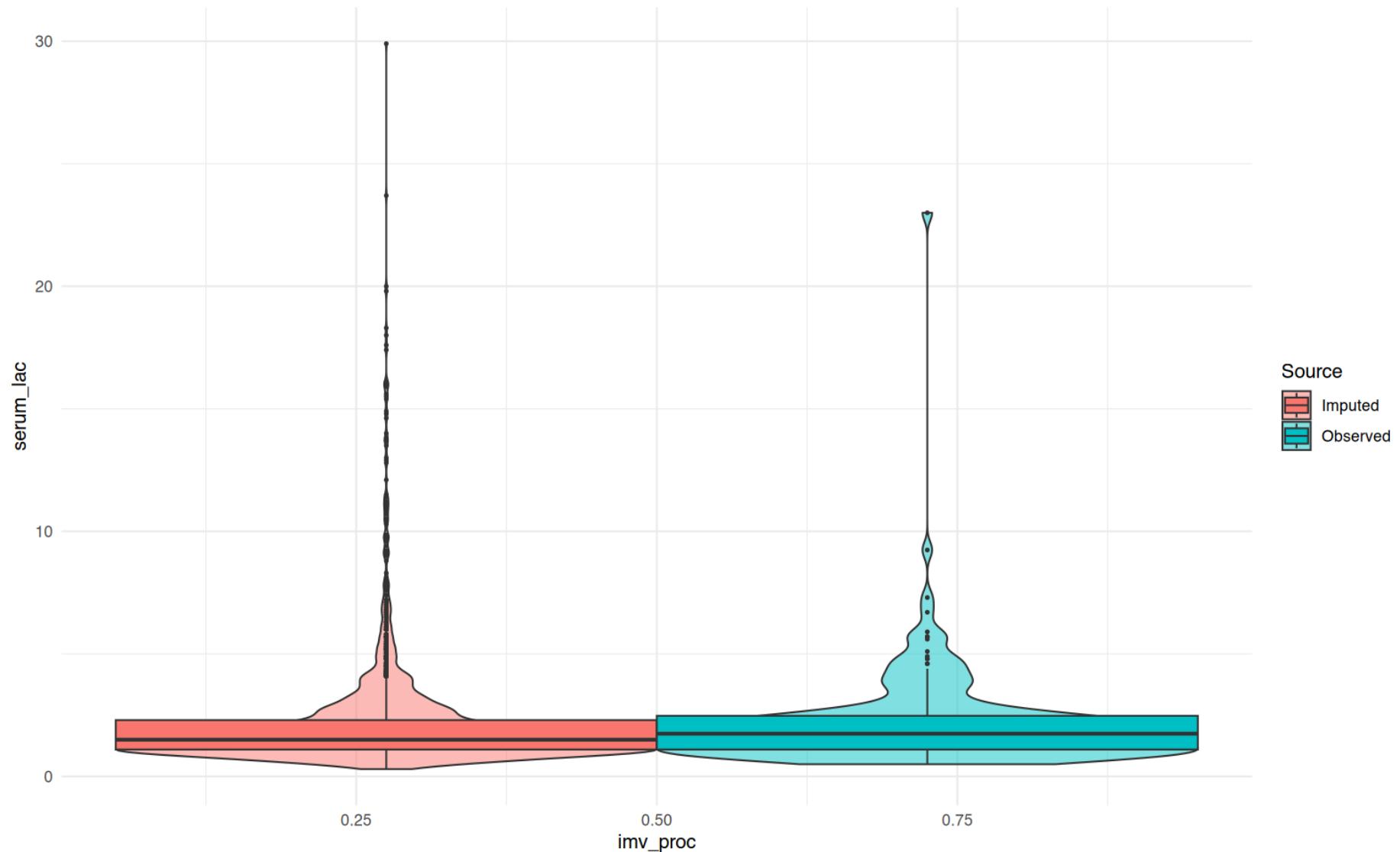
Observed vs imputed: spo2 by imv_proc



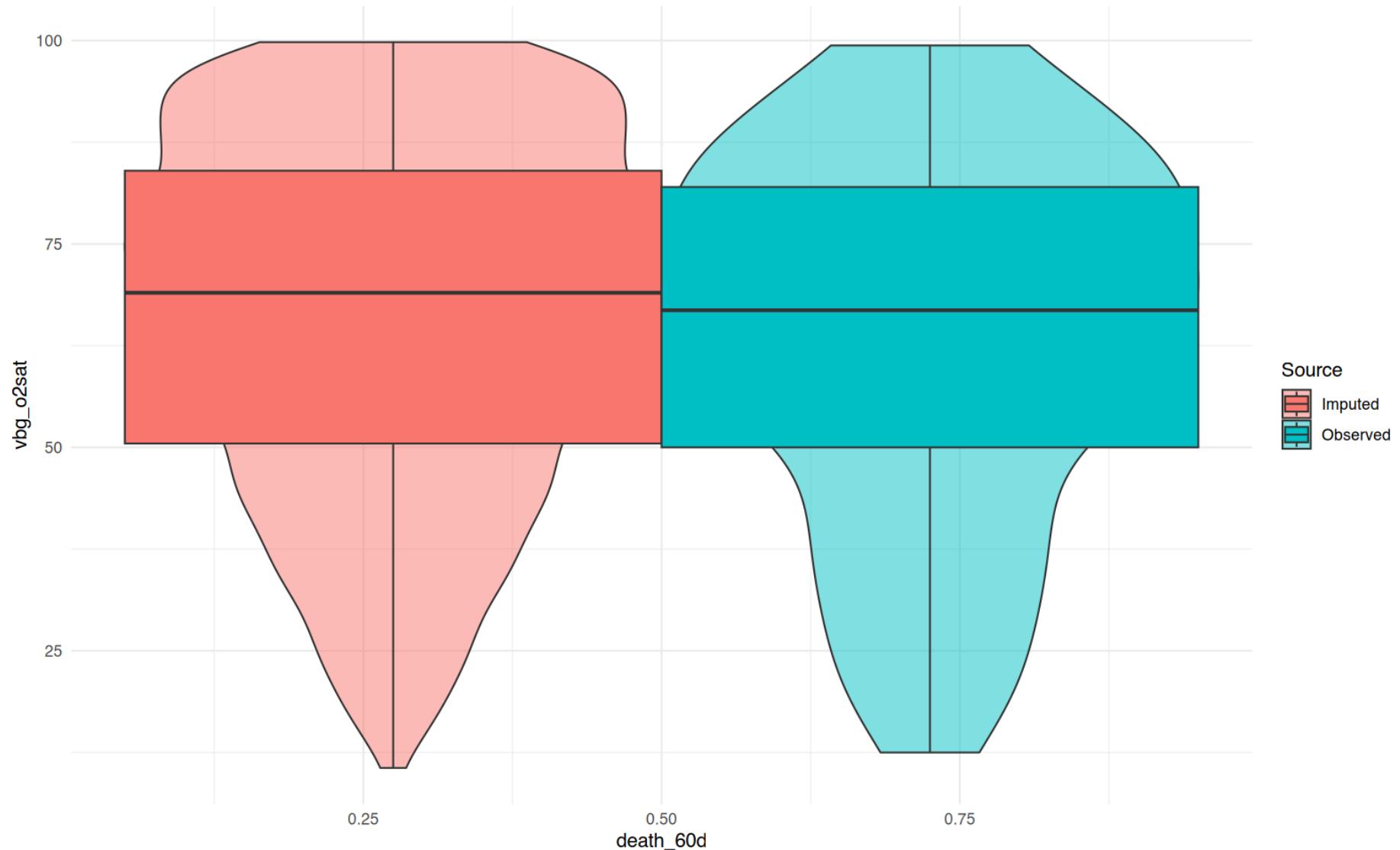
Observed vs imputed: paco2 by imv_proc



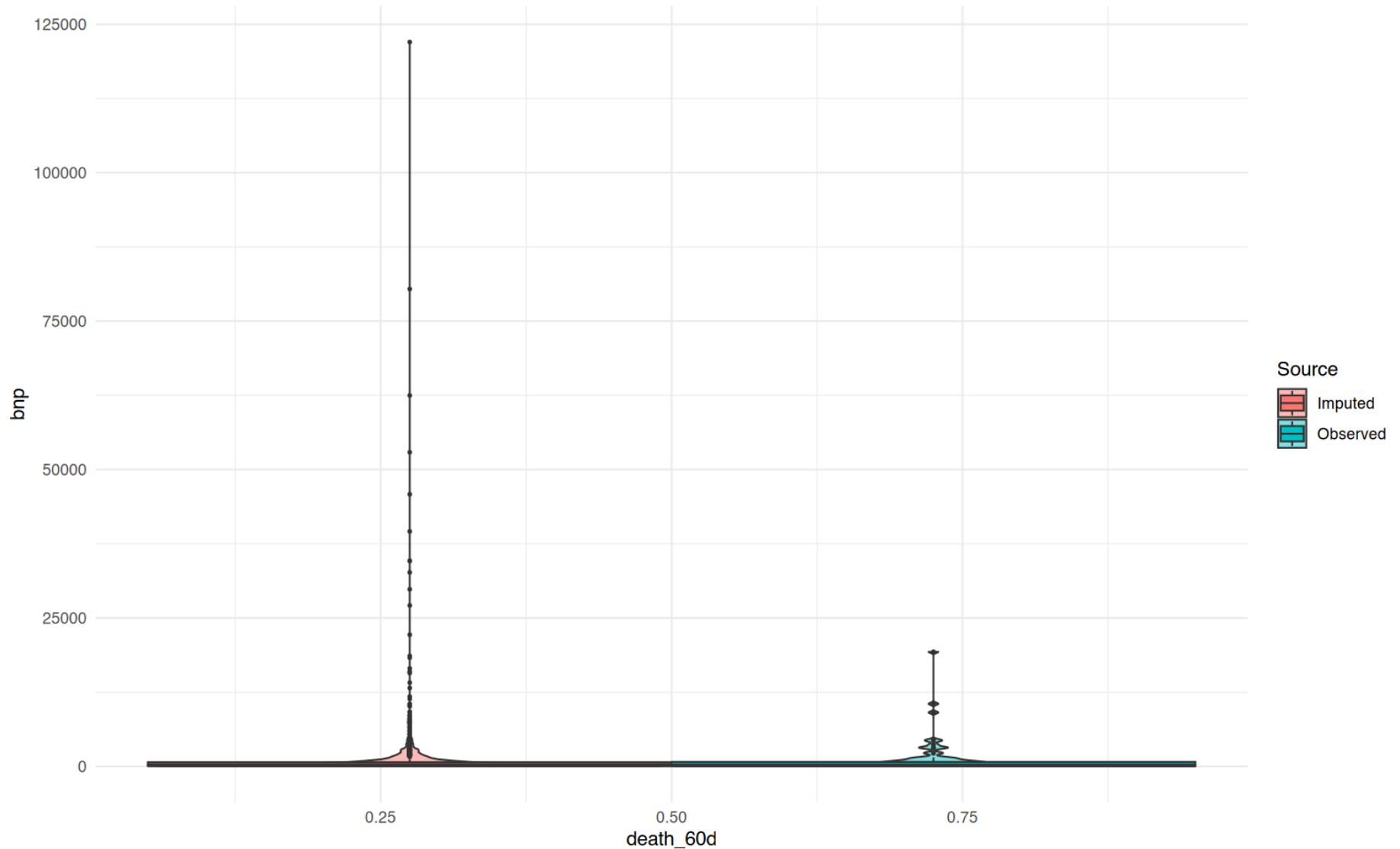
Observed vs imputed: serum_lac by imv_proc



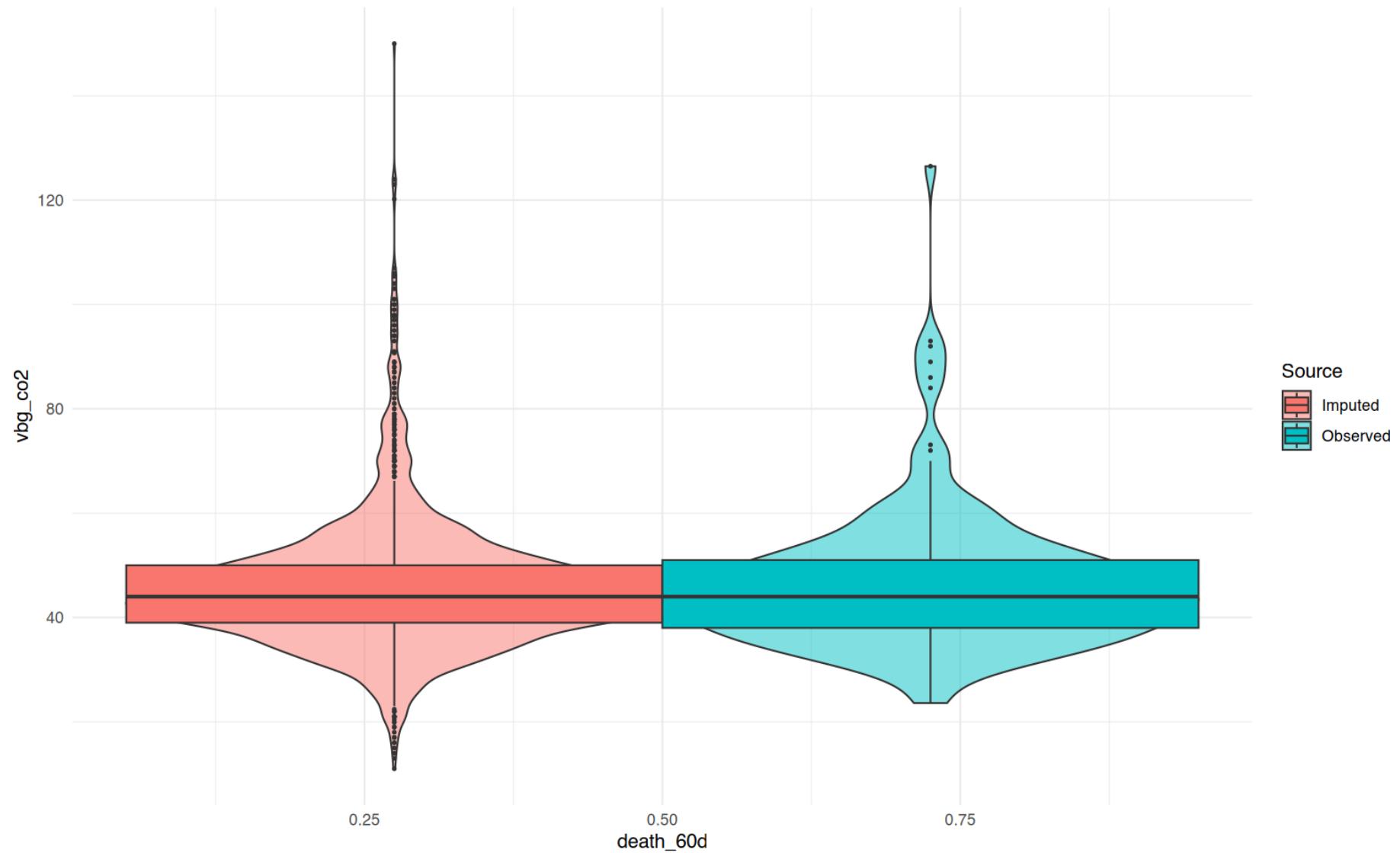
Observed vs imputed: vbg_o2sat by death_60d



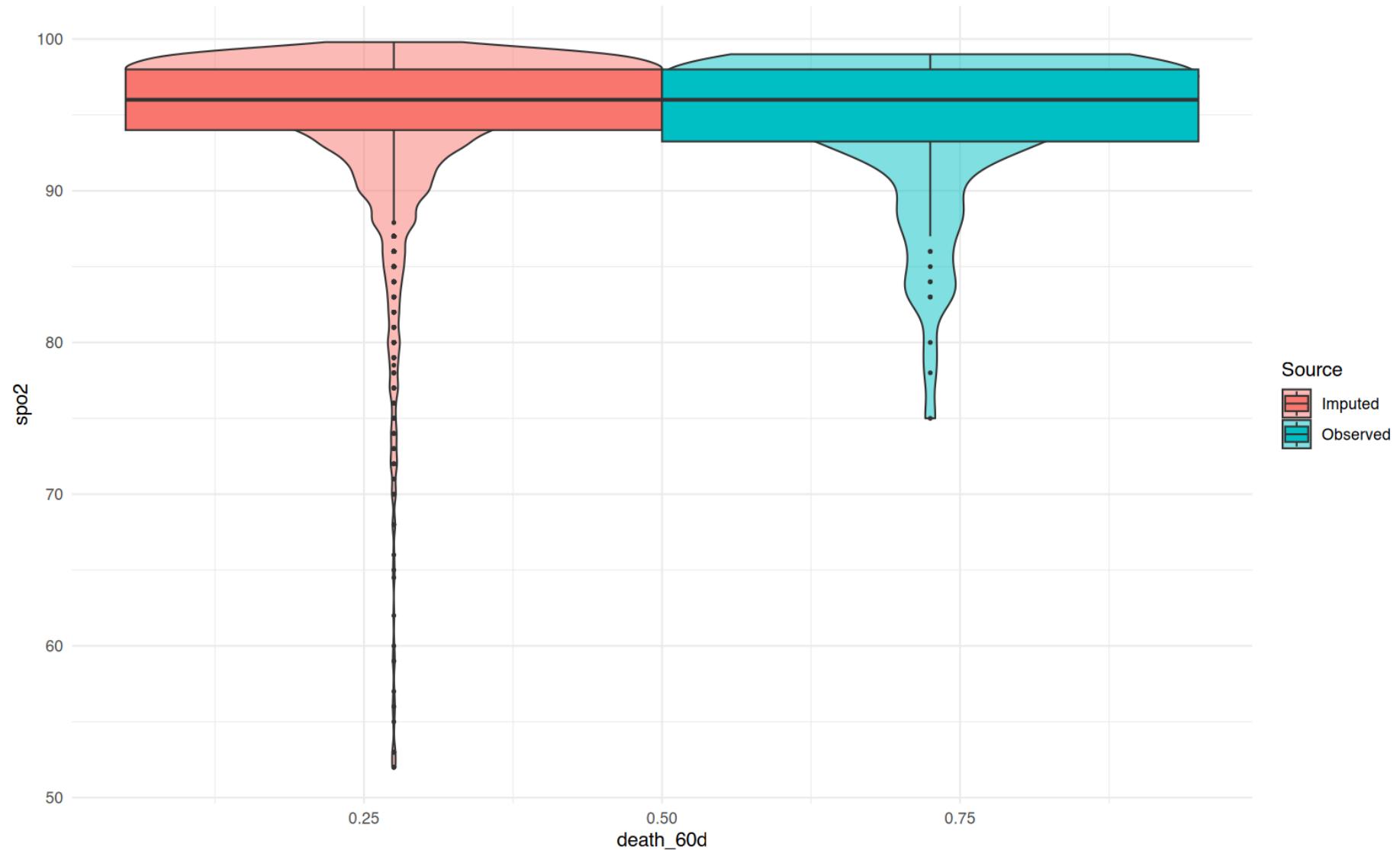
Observed vs imputed: bnp by death_60d



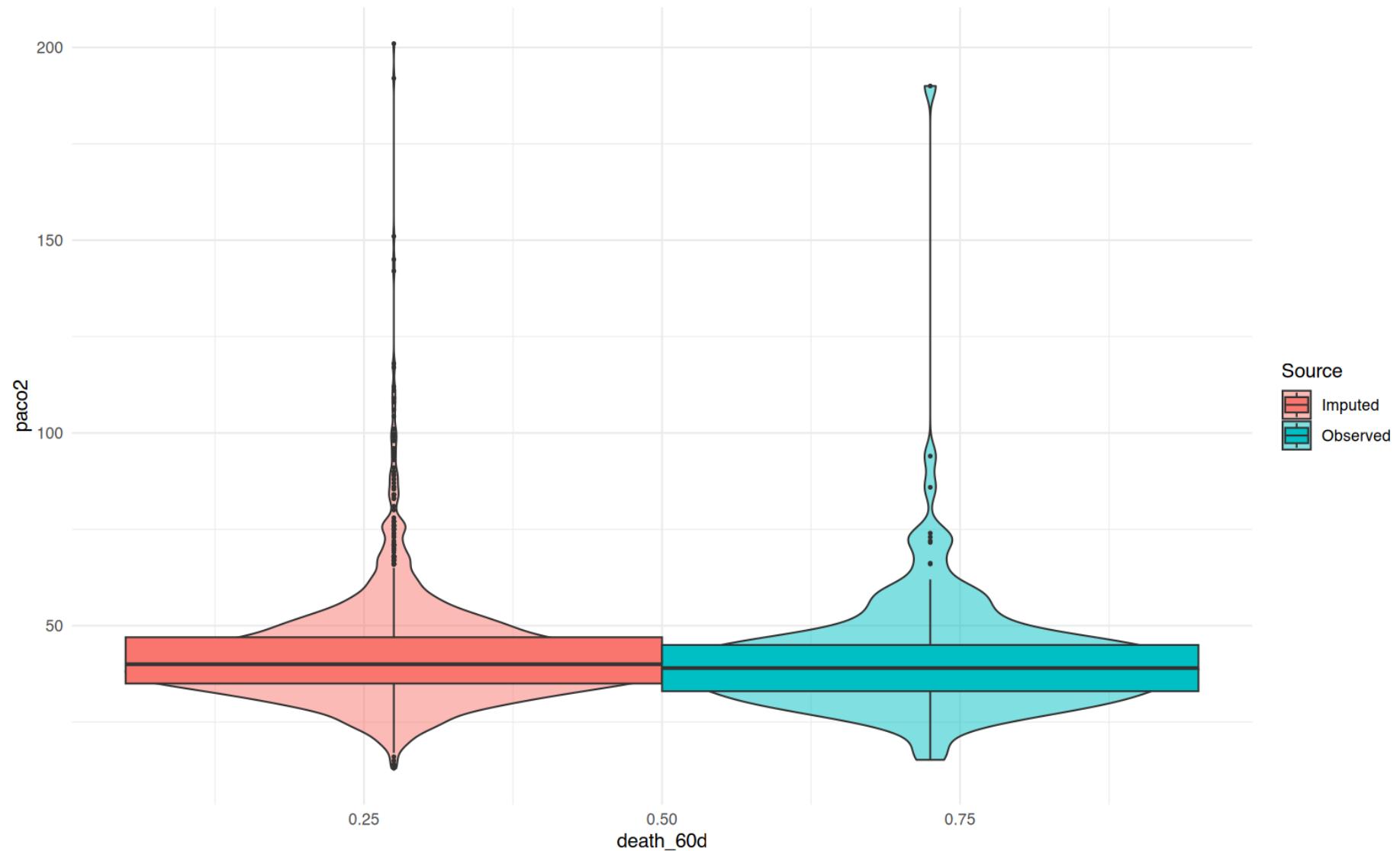
Observed vs imputed: vbg_co2 by death_60d



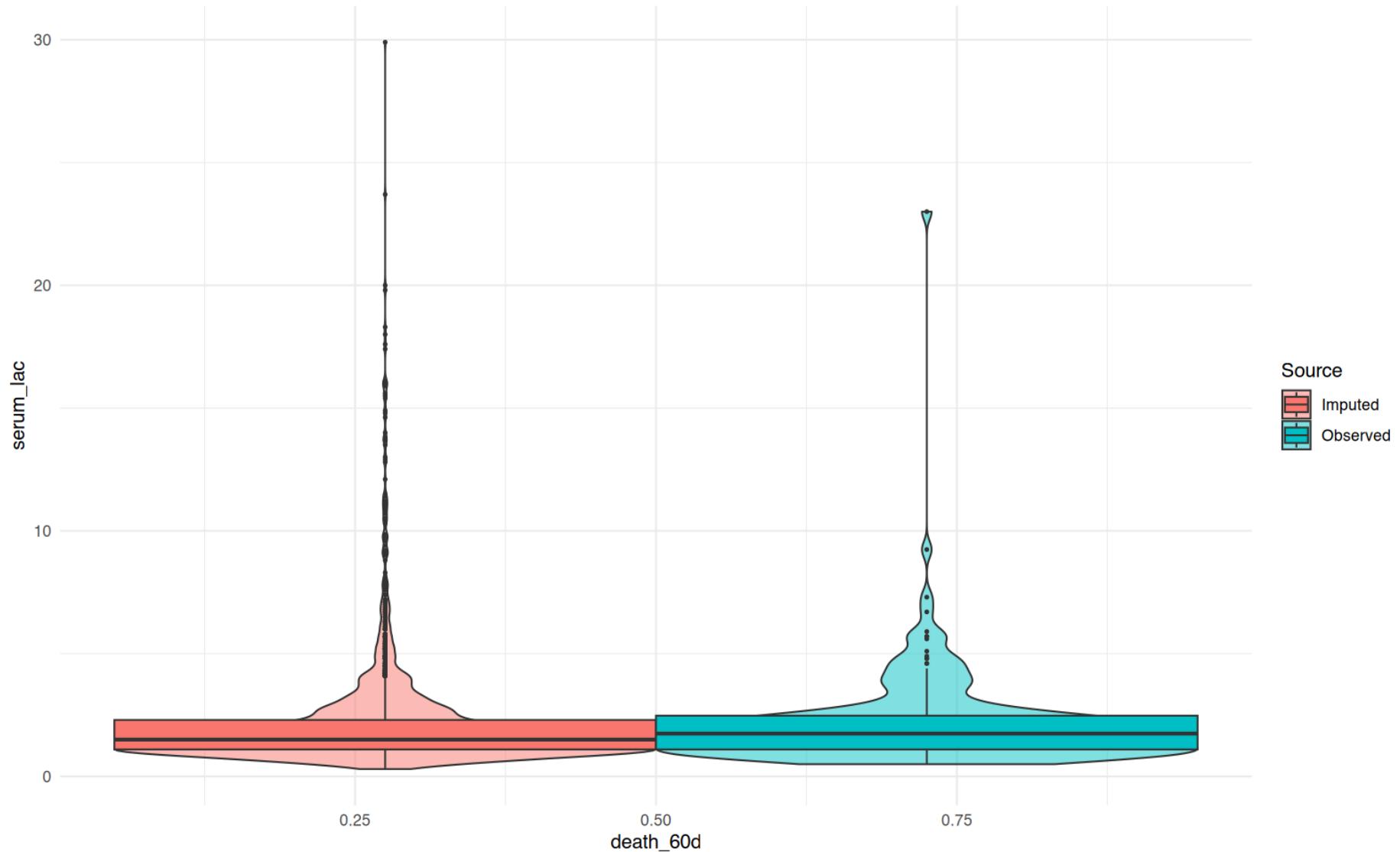
Observed vs imputed: spo2 by death_60d



Observed vs imputed: paco2 by death_60d



Observed vs imputed: serum_lac by death_60d



Chunk mi-imputation-diagnostics runtime: 12.83 s

4.4 10) Refit propensity models within each imputation

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

4.4.1 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) mice::complete(imp_obj, action = i)

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

with_progress({
```

```

p <- progressor(along = seq_len(imp_n))

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

W_abg_list <- 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
)
saveRDS(W_abg_list, mi_w_abg_file)

```

Chunk mi-propensity-abg runtime: 111.77 s

	n	min	p99.99%	max	ess
[1,]	15429	0.575	3.376	3.377	12684.00
[2,]	15429	0.575	3.439	3.441	12616.44
[3,]	15429	0.575	3.376	3.379	12674.97
[4,]	15429	0.576	3.276	3.278	12710.69
[5,]	15429	0.575	3.277	3.277	12738.67

Chunk mi-weight-diagnostics-abg runtime: 0.12 s

4.4.2 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) mice::complete(imp_obj, action = i)

# 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: 0.21 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]
}))

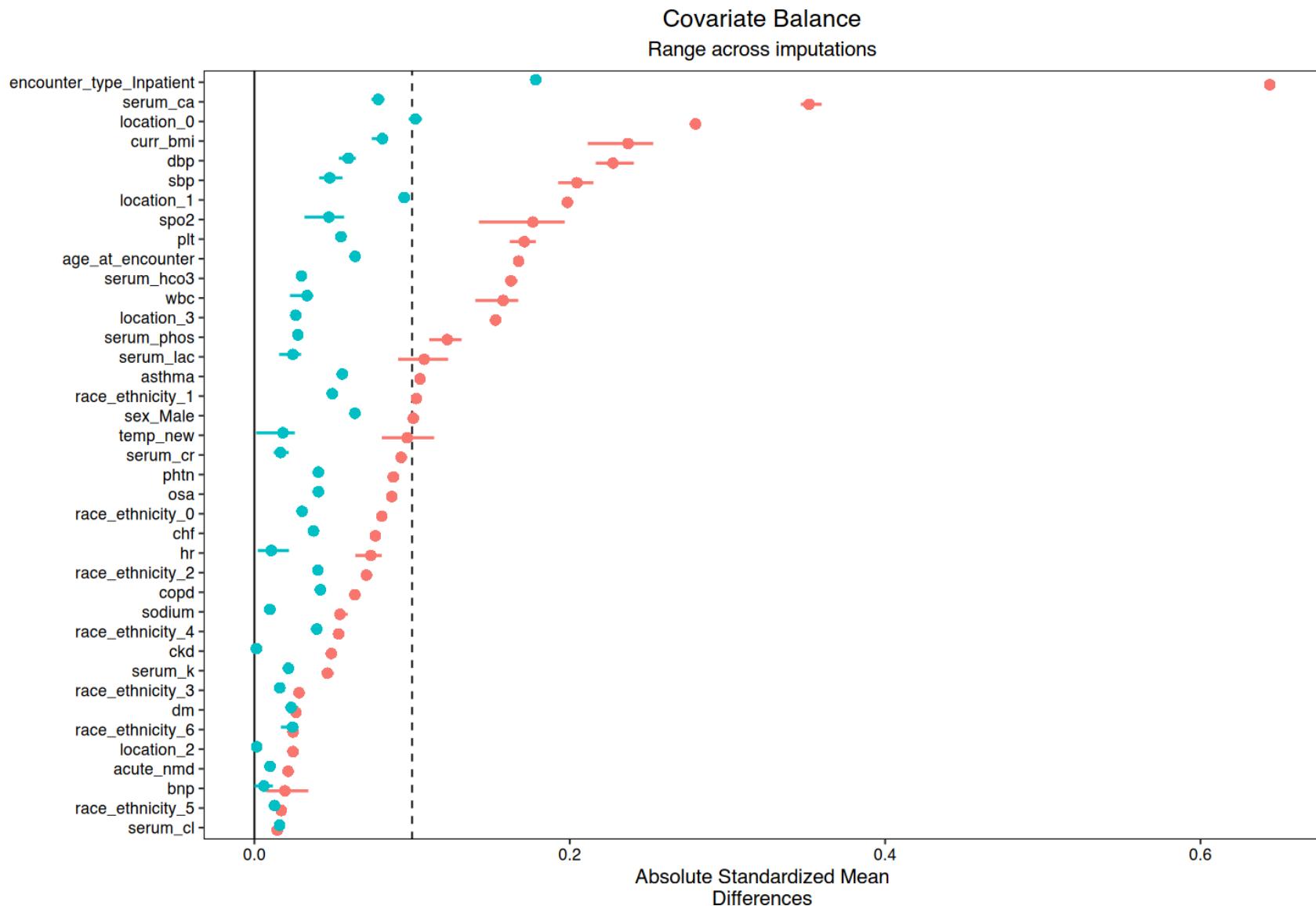
# Long data for cobalt with weights and imputation id
make_long_for_cobalt <- function(imp_n, get_imp, W_list, treat_var, covars) {
  stopifnot(length(W_list) == imp_n)
  do.call(rbind, lapply(seq_len(imp_n), function(i) {
    di <- get_imp(i)[, c(treat_var, covars), drop = FALSE]
    di$.imp <- i
 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

# 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: 1.78 s

4.4.3 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) mice::complete(imp_obj, action = i)

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

with_progress({
  p <- progressor(along = seq_len(imp_n))
  options(future.globals.maxSize = +Inf)
  if (utils::object.size(imp$data) > 4e8) future::plan(sequential)

  W_vbg_list <- 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
)
})

saveRDS(W_vbg_list, mi_w_vbg_file)

```

Chunk mi-propensity-vbg runtime: 90.55 s

	n	min	p99.99%	max	ess
[1,]	15429	0.584	3.791	3.791	10850.22
[2,]	15429	0.582	3.964	3.964	10735.87
[3,]	15429	0.581	3.906	3.907	10732.43
[4,]	15429	0.582	3.911	3.911	10796.66
[5,]	15429	0.583	3.898	3.900	10816.61

Chunk mi-weight-diagnostics-vbg runtime: 0.01 s

4.4.4 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) mice::complete(imp_obj, action = i)
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]
  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)
  }
  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)

```

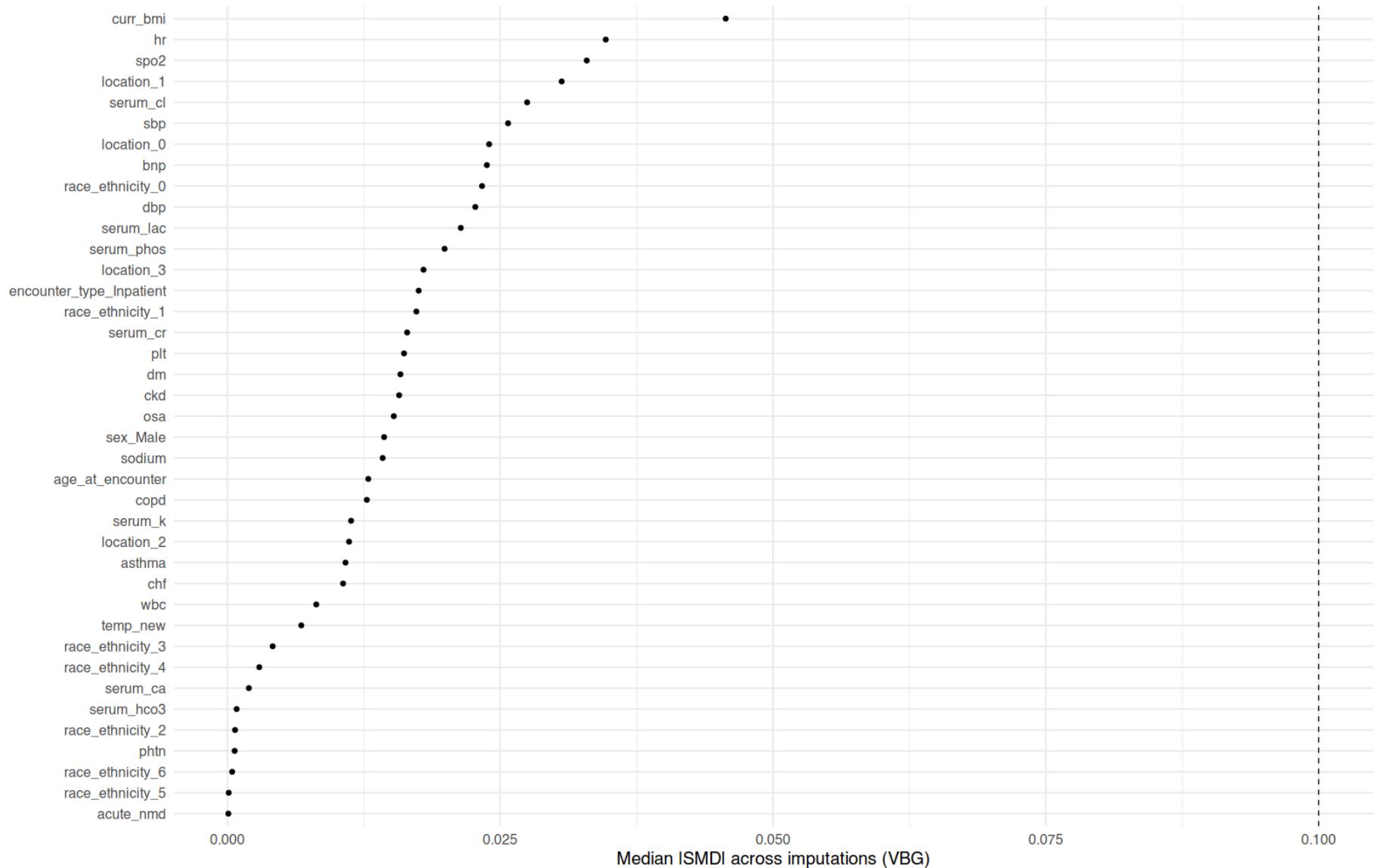
```

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: 1.18 s

4.5 11) Weighted outcome models within each imputation + pooling

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

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$w <- as.numeric(weights)
  keep <- is.finite(data$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())
cv  <- coef_var_from_svy(fit, treat_name)
list(coef = cv$b, vcov = cv$V)
}

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

```

Chunk mi-pool-helpers runtime: 0.00 s

4.5.2 11.2 ABG: outcomes = IMV, NIV, Death(60d), Hypercapnic RF

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

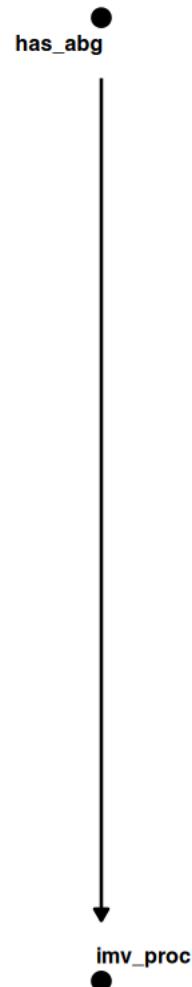
# Inputs assumed present: imp, W_abg_list
stopifnot(exists("imp"), exists("W_abg_list"))
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) mice::complete(imp_obj, action = i)
stopifnot(length(W_abg_list) == imp_n)

outcomes_abg <- list(
  imv_proc = imv_proc ~ has_abg,
  niv_proc = niv_proc ~ has_abg,
  death    = death_60d ~ has_abg,
  hcrf     = hypercap_resp_failure ~ has_abg
)
purrr::iwalk(outcomes_abg, function(fml, nm) {
  register_model_diagram(paste("MI IPW ABG:", nm, "~ has_abg"), fml)
})
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

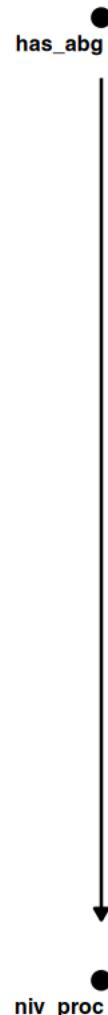
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

MI IPW ABG: imv_proc ~ has_abg



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW ABG: niv_proc ~ has_abg



```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

MI IPW ABG: death ~ has_abg

has_abg



death_60d

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

MI IPW ABG: hcrf ~ has_abg

has_abg



hypercap_resp_failure

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

```

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

```

outcome	term	logOR	SE	OR	LCL	UCL	m_used	m_total
imv_proc	has_abg	2.0426845	0.0720502	7.711282	6.695703	8.880900	5	5
niv_proc	has_abg	0.5831149	0.0707555	1.791610	1.559607	2.058126	5	5
death	has_abg	0.8948883	0.0594409	2.447062	2.177950	2.749427	5	5
hcrf	has_abg	1.0788030	0.0809181	2.941157	2.509801	3.446649	5	5

Chunk mi-abg-outcomes runtime: 3.45 s

4.5.3 11.3 Repeat for VBG

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

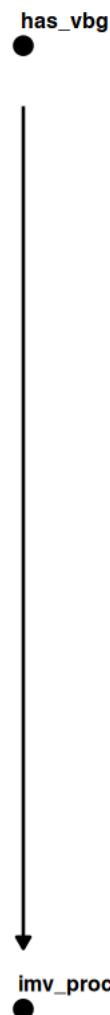
# Inputs assumed present: imp, W_vbg_list
stopifnot(exists("imp"), exists("W_vbg_list"))
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) mice::complete(imp_obj, action = i)
stopifnot(length(W_vbg_list) == imp_n)

outcomes_vbg <- list(
  imv_proc = imv_proc ~ has_vbg,
  niv_proc = niv_proc ~ has_vbg,
  death    = death_60d ~ has_vbg,
  hcrf     = hypercap_resp_failure ~ has_vbg
)
purrr::iwalk(outcomes_vbg, function(fml, nm) {
  register_model_diagram(paste("MI IPW VBG:", nm, "~ has_vbg"), fml)
})
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

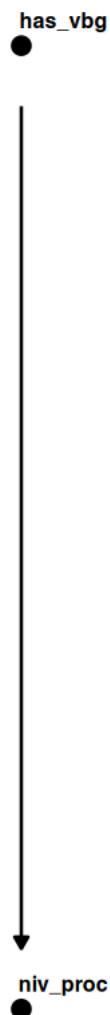
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

MI IPW VBG: imv_proc ~ has_vbg



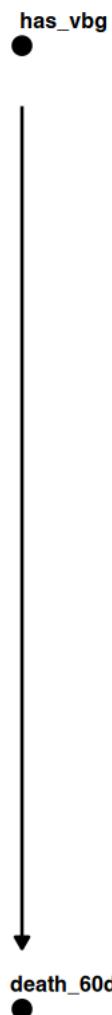
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW VBG: niv_proc ~ has_vbg



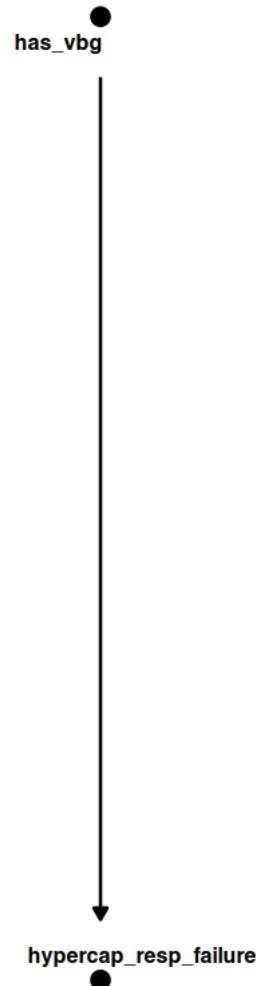
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW VBG: death ~ has_vbg



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW VBG: hcrf ~ has_vbg



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

```

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

```

outcome	term	logOR	SE	OR	LCL	UCL	m_used	m_total
imv_proc	has_vbg	0.5197363	0.0575220	1.681584	1.502294	1.882272	5	5
niv_proc	has_vbg	0.2952344	0.0758074	1.343441	1.157951	1.558645	5	5
death	has_vbg	0.2599468	0.0597722	1.296861	1.153491	1.458050	5	5
hcrf	has_vbg	0.7453523	0.0763203	2.107184	1.814418	2.447189	5	5

Chunk mi-vbg-outcomes runtime: 3.11 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)
```

Attaching package: 'fastshap'

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

explain

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

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

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

```

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

```

```

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

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

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

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

```

Chunk unnamed-chunk-2 runtime: 0.02 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 = 0.15,
  nsim      = 8,
  top_k     = 20,
  seed       = 123
)

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

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

# Subsample rows for SHAP speed
set.seed(shap_cfg$seed)

```

```

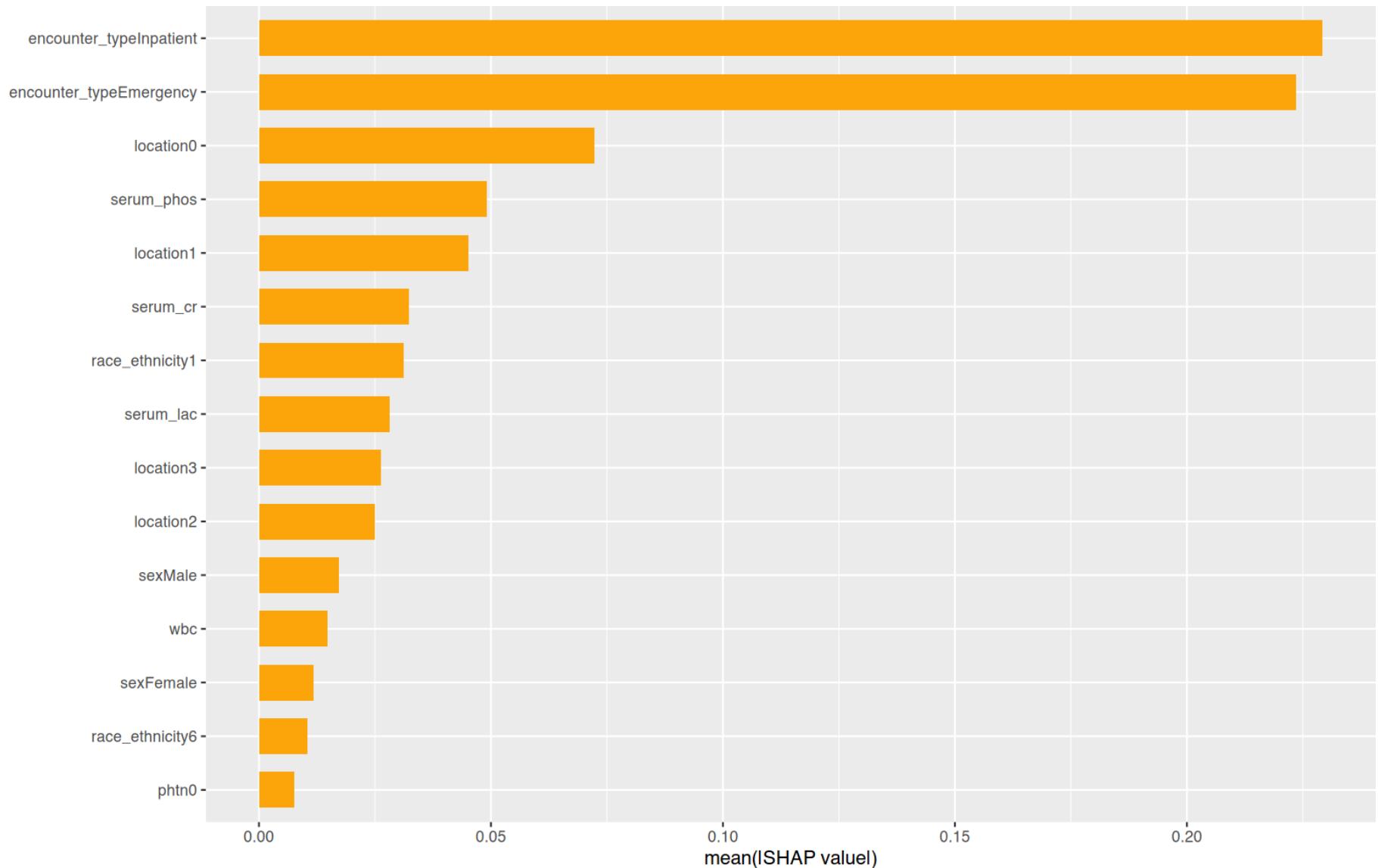
ix_abg <- sample.int(nrow(bk_abg$X), max(50L, floor(shap_cfg$frac_rows * nrow(bk_abg$X))))
X_abg  <- bk_abg$X[ix_abg, , drop = FALSE]

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

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

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

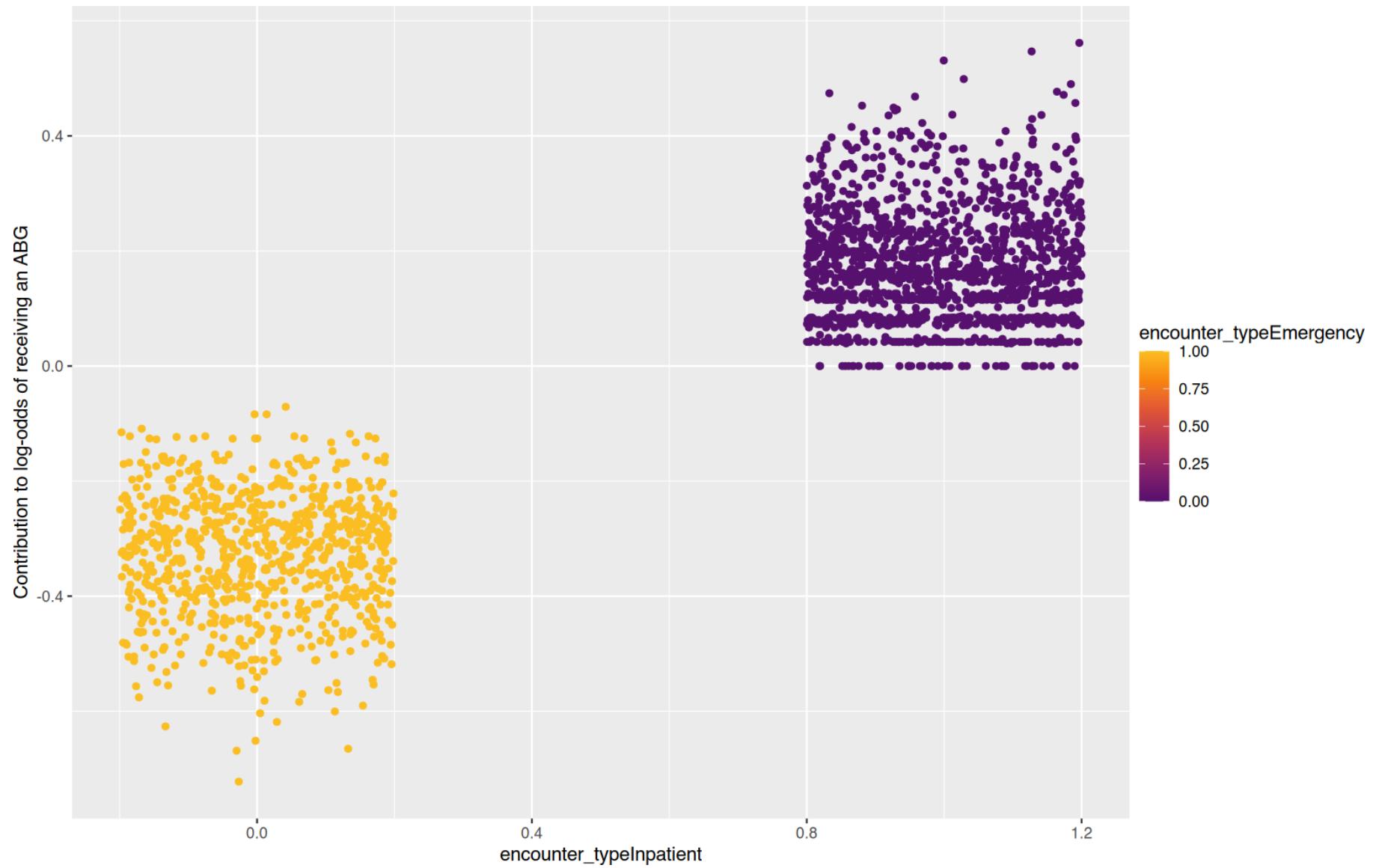
```



```
# 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: 28.62 s

```
# Repeat for VBG
```

```

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

bk_vbg <- make_shap_backend_any(W1_vbg)

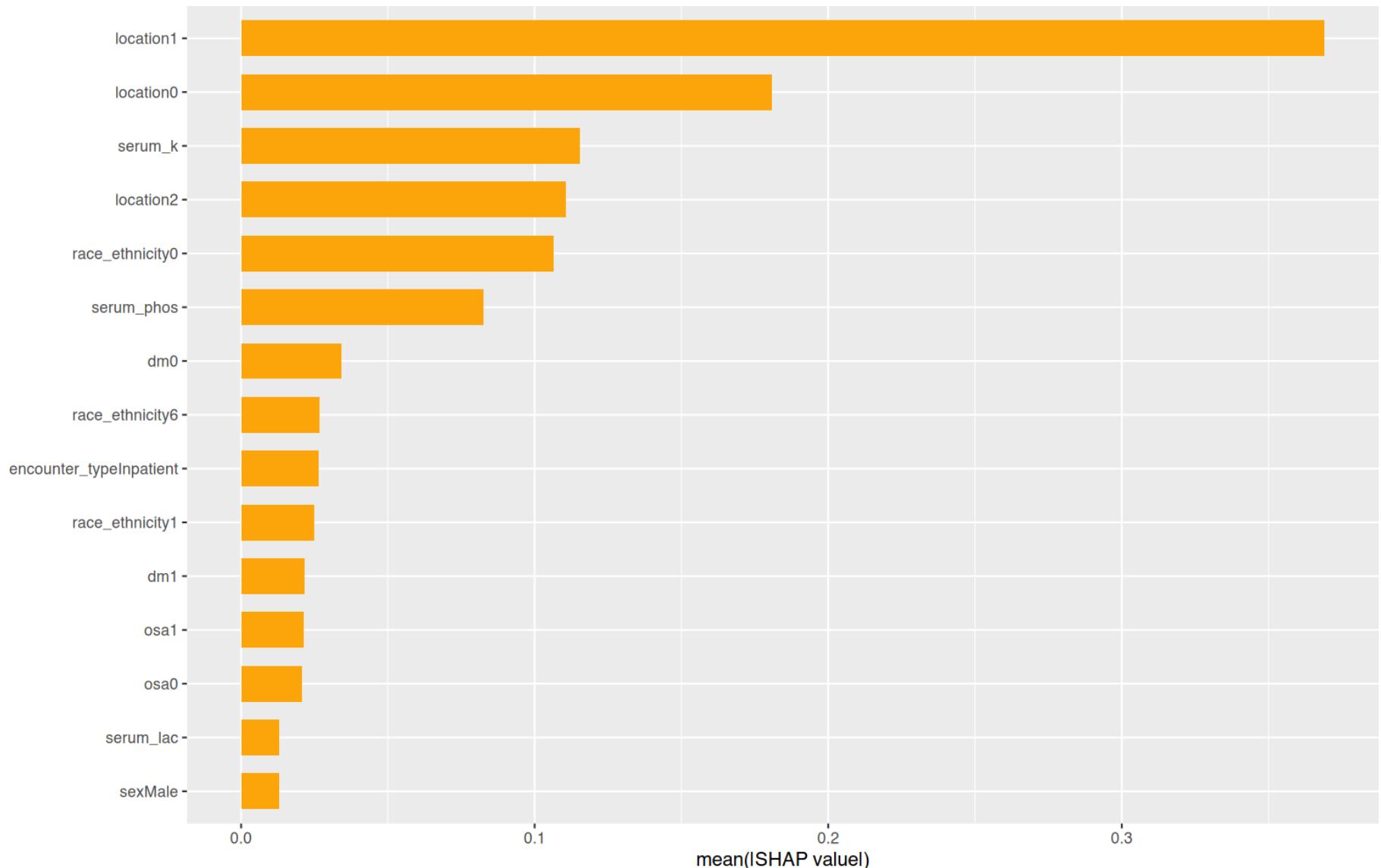
set.seed(shap_cfg$seed)
ix_vbg <- sample.int(nrow(bk_vbg$X), max(50L, floor(shap_cfg$frac_rows * nrow(bk_vbg$X))))
X_vbg <- bk_vbg$X[ix_vbg, , drop = FALSE]

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

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

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

```

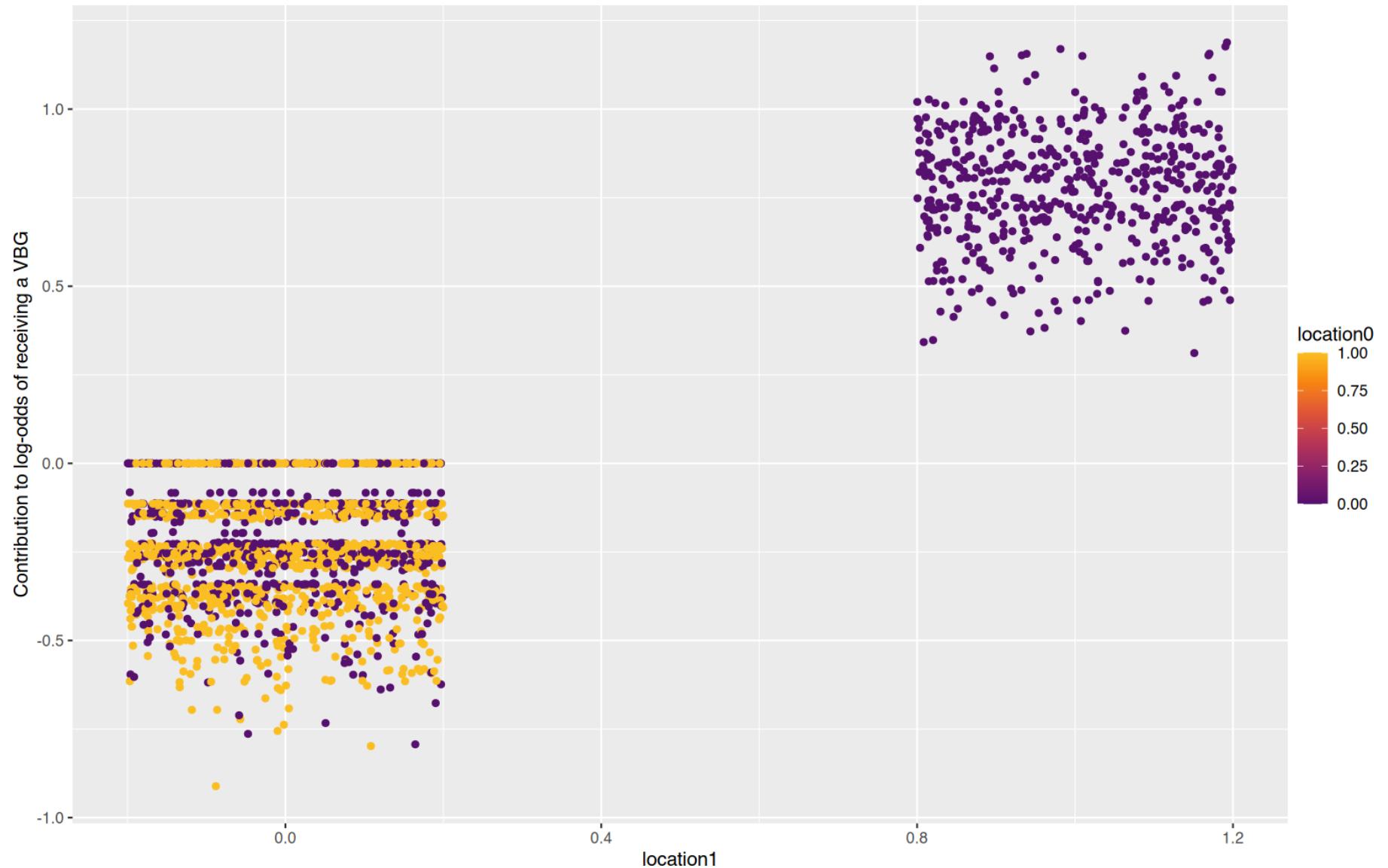


```

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))
```



Chunk unnamed-chunk-3 runtime: 30.01 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())
  )

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

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

```

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

# 3-level CO2 category maker; can use fixed clinical cutpoints or data-driven
# --- CO2 category helper (3-level) -----
make_co2_cat <- function(x,
                        fixed_breaks = NULL,
                        labels = c("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_threellevel runtime: 0.01 s

Chunk mi-co2-cat-checks runtime: 0.17 s

Chunk pool-spline-helpers runtime: 0.00 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.

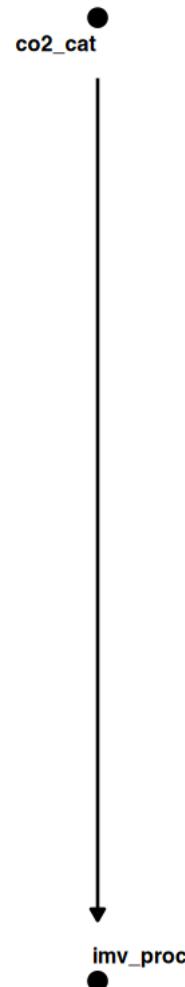
if (!exists("imp")) imp <- readRDS(mi_mids_file)
imp_n <- imp$m
if (!exists("get_imp")) get_imp <- function(i, imp_obj = imp) mice::complete(imp_obj, action = i)

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)
```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

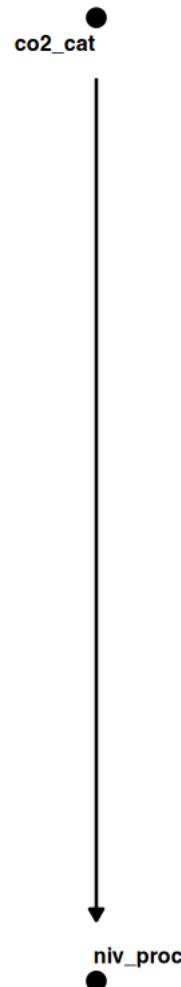
Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

MI IPW ABG 3-level: IMV ~ CO2 category



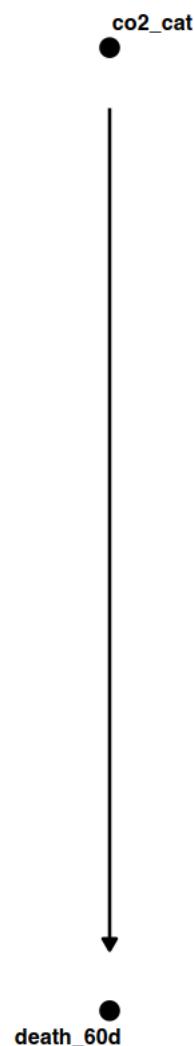
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW ABG 3-level: NIV ~ CO2 category



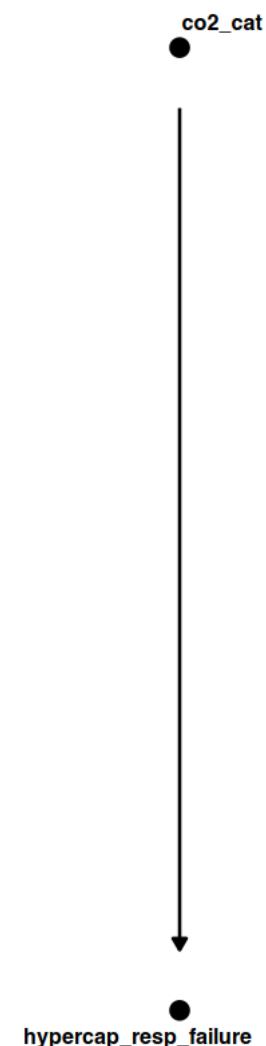
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW ABG 3-level: Death60d ~ CO2 category



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW ABG 3-level: HCRF ~ 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: 1.63 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) mice::complete(imp_obj, action = i)

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)

```

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow =
arrow_directed, : Ignoring unknown parameters: `colour`

Warning in geom_dag_edges_arc(mapping, data = data_bidirected, arrow =
arrow_bidirected, : Ignoring unknown parameters: `colour`

MI IPW VBG 3-level: IMV ~ CO2 category

co2_cat



imv_proc

Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW VBG 3-level: NIV ~ CO2 category



Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`
Ignoring unknown parameters: `colour`

MI IPW VBG 3-level: Death60d ~ CO2 category

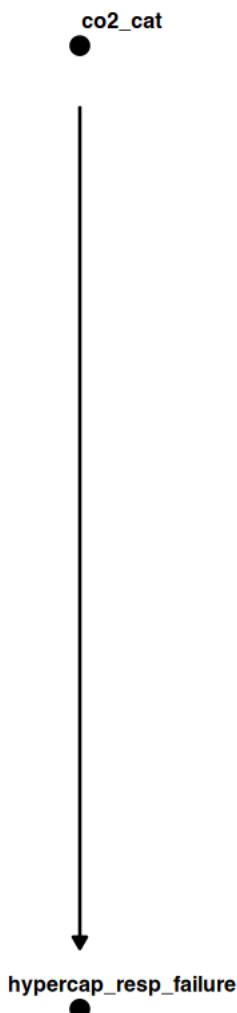
co2_cat



death_60d

```
Warning in geom_dag_edges_link(mapping, data = data_directed, arrow = arrow_directed, : Ignoring unknown parameters: `colour`  
Ignoring unknown parameters: `colour`
```

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: 1.42 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) mice::complete(imp_obj, action = i)

# 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

5

```

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

```

	term	logOR	SE	OR	LCL	UCL
1	co2_catHypocapnia	0.20900300	0.07982726	1.232449	1.0539464	1.441183
2	co2_catHypercapnia	0.09617711	0.08150294	1.100954	0.9384097	1.291653

Chunk unnamed-chunk-6 runtime: 0.28 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) mice::complete(imp_obj, action = i)
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)

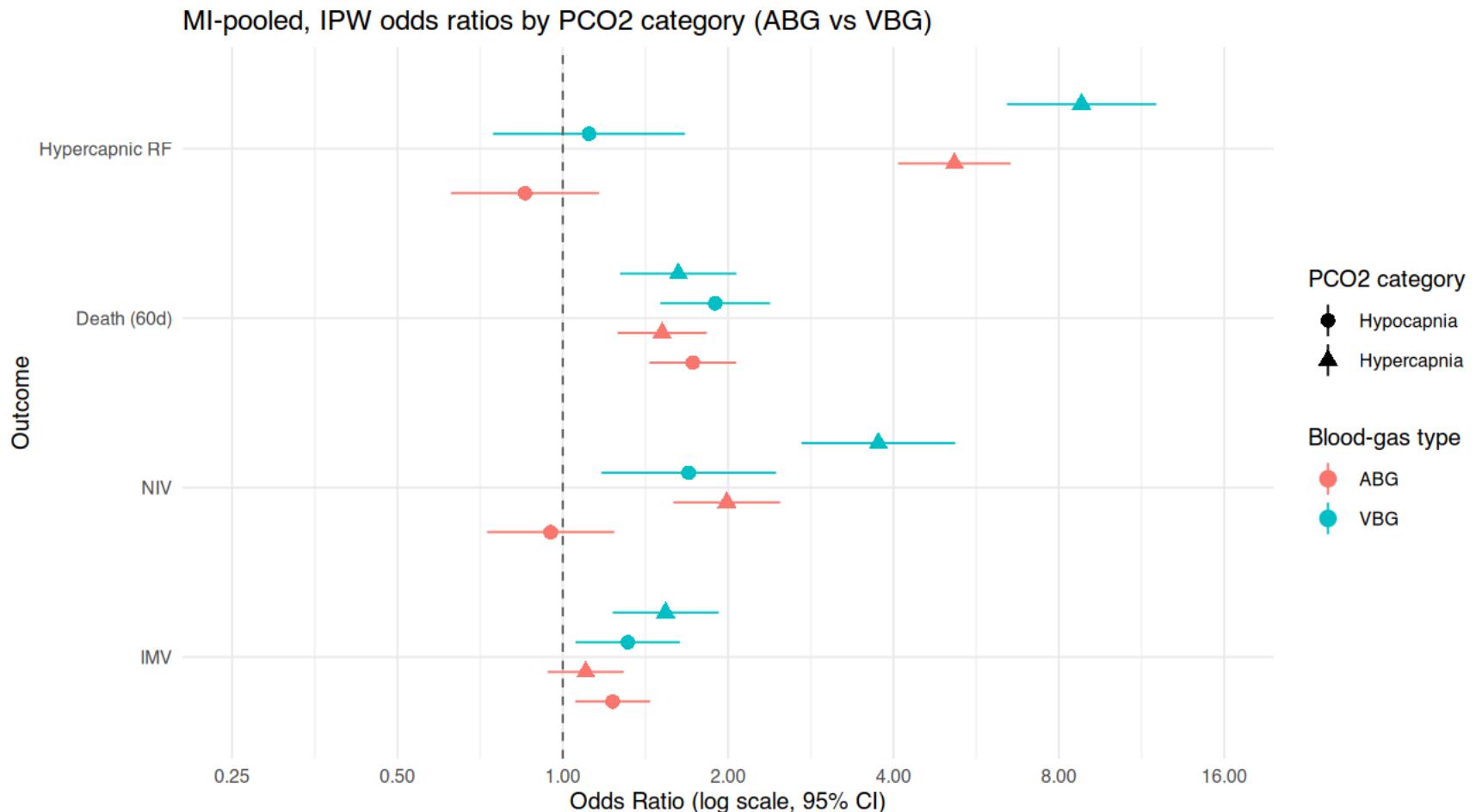
# decode "co2_cat<level>" → exposure
combined <- combined |>
  dplyr::mutate(exposure = gsub("^co2_cat", "", term),
                exposure = factor(exposure, levels = c("Eucapnia", "Hypocapnia", "Hypercapnia")),
                outcome  = factor(outcome,
                                  levels = c("imv_proc", "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: 1.27 s

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

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	0.2090030	0.0798273	1.2324487	1.0539464	1.441183	ABG
imv_proc	co2_catHypercapnia	0.0961771	0.0815029	1.1009540	0.9384097	1.291653	ABG
niv_proc	co2_catHypocapnia	-0.0507950	0.1359344	0.9504735	0.7281659	1.240651	ABG
niv_proc	co2_catHypercapnia	0.6874050	0.1143564	1.9885485	1.5892568	2.488160	ABG
death_60d	co2_catHypocapnia	0.5451362	0.0928174	1.7248433	1.4379438	2.068985	ABG
death_60d	co2_catHypercapnia	0.4166159	0.0952319	1.5168198	1.2585514	1.828088	ABG
hypercap_resp_failure	co2_catHypocapnia	-0.1580041	0.1585031	0.8538462	0.6258339	1.164931	ABG
hypercap_resp_failure	co2_catHypercapnia	1.6414145	0.1202252	5.1624667	4.0786790	6.534239	ABG

Chunk unnamed-chunk-7 runtime: 0.58 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	0.2721406	0.1116347	1.312771	1.0547845	1.633859	VBG
imv_proc	co2_catHypercapnia	0.4310273	0.1133604	1.538838	1.2322489	1.921706	VBG
niv_proc	co2_catHypocapnia	0.5276364	0.1869173	1.694921	1.1750122	2.444876	VBG

outcome	term	logOR	SE	OR	LCL	UCL	group
niv_proc	co2_catHypercapnia	1.3229096	0.1642450	3.754329	2.7209726	5.180128	VBG
death_60d	co2_catHypocapnia	0.6387422	0.1173918	1.894097	1.5047913	2.384120	VBG
death_60d	co2_catHypercapnia	0.4840969	0.1243921	1.622709	1.2716157	2.070739	VBG
hypercap_resp_failure	co2_catHypocapnia	0.1098726	0.2051862	1.116136	0.7465503	1.668688	VBG
hypercap_resp_failure	co2_catHypercapnia	2.1746904	0.1596515	8.799461	6.4351390	12.032453	VBG

Chunk unnamed-chunk-8 runtime: 0.52 s

4.9.3 15.3 Visualization

```

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

mi_ipw_rcs_forms <- list(
  "MI IPW RCS: IMV ~ CO2 (ABG)"      = imv_proc ~ splines::ns(co2, 4),
  "MI IPW RCS: NIV ~ CO2 (ABG)"      = niv_proc ~ splines::ns(co2, 4),
  "MI IPW RCS: Death60d ~ CO2 (ABG)" = death_60d ~ splines::ns(co2, 4),
  "MI IPW RCS: HCRF ~ CO2 (ABG)"     = hypercap_resp_failure ~ splines::ns(co2, 4),
  "MI IPW RCS: IMV ~ CO2 (VBG)"      = imv_proc ~ splines::ns(co2, 4),
  "MI IPW RCS: NIV ~ CO2 (VBG)"      = niv_proc ~ splines::ns(co2, 4),
  "MI IPW RCS: Death60d ~ CO2 (VBG)" = death_60d ~ splines::ns(co2, 4),
  "MI IPW RCS: HCRF ~ CO2 (VBG)"     = hypercap_resp_failure ~ splines::ns(co2, 4)
)
register_model_diagrams(mi_ipw_rcs_forms)

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

```

```

stopifnot(length(W_abg_list) == imp_n,
          length(W_vbg_list) == imp_n)

# CO2 support sanity: ensure trimmed ranges exist for both groups
rng_abg <- quantile(
  unlist(lapply(seq_len(imp_n), function(i) {
    d <- get_imp(i)
    as.numeric(d$paco2[d$has_abg == 1])
  })),
  c(.02, .98), na.rm = TRUE
)
rng_vbg <- quantile(
  unlist(lapply(seq_len(imp_n), function(i) {
    d <- get_imp(i)
    as.numeric(d$vbg_co2[d$has_vbg == 1])
  })),
  c(.02, .98), na.rm = TRUE
)
stopifnot(rng_abg[1] < rng_abg[2], rng_vbg[1] < rng_vbg[2])

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

# Pool predicted curves on the link scale, then transform with logistic

```

```

pool_rcs_curve <- function(group = c("ABG", "VBG"), outcome,
                           df = 4, grid_n = 220, trim = c(0.02, 0.98)) {
  group <- match.arg(group)

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

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

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

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

```

```

d2$co2 <- d2$vbg_co2
w  <- W_vbg_list[[i]]$weights[g]
}

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

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

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

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

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

if (m == 1L) {
  etaBar <- as.numeric(ETA)
  Tvar    <- as.numeric(VAR)
} else {
  etaBar <- rowMeans(ETA)
  Wbar    <- rowMeans(VAR)
}

```

```

      B      <- apply(ETA, 1, stats::var)
      Tvar   <- Wbar + (1 + 1/m) * B
    }
    seBar <- sqrt(pmax(Tvar, 0))

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

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

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

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

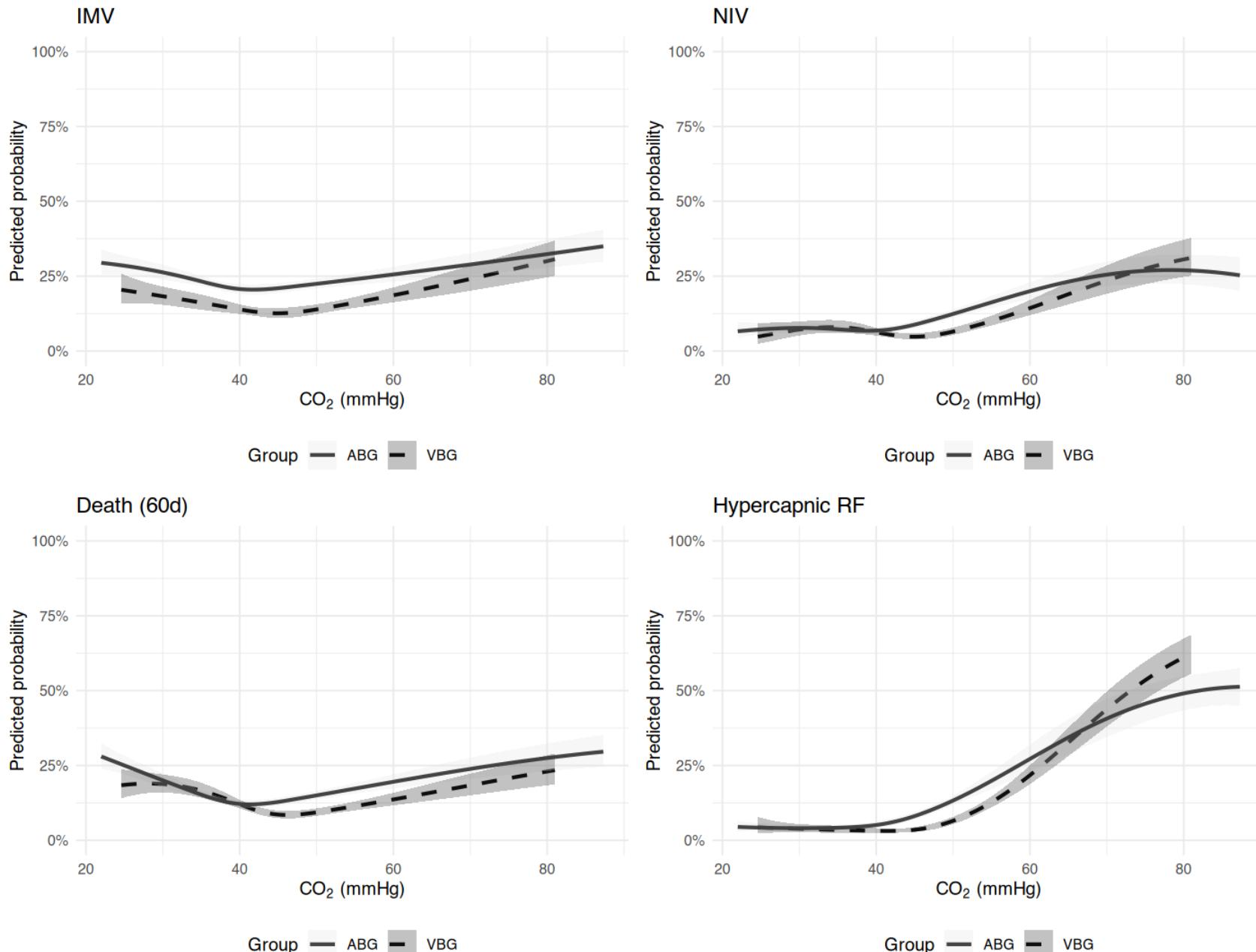
```

```
    fill = "Group", linetype = "Group") +
  theme_minimal(base_size = 10) +
  theme(legend.position = "bottom")
}

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

```

MI-pooled, propensity-weighted predicted probability: ABG vs VBG CO₂ (restricted cubic splines, 2–98% range)



Chunk ipw-rcs-overlay-mi-abg-vbg runtime: 2.74 s

4.10 16) Save, export, and session info

```
saveRDS(list(abg = abg_results, vbg = vbg_results), mi_pooled_file)
```

Chunk mi-save-exports runtime: 0.00 s

```
sessionInfo()
```

```
R version 4.5.0 (2025-04-11)
Platform: aarch64-apple-darwin20
Running under: macOS 26.2
```

```
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] shapviz_0.10.2    fastshap_0.1.1    rlang_1.1.6
[4] progressr_0.18.0   future.apply_1.20.0 future_1.67.0
[7] mitools_2.4       skimr_2.2.1      visdat_0.6.0
[10] naniar_1.1.0     miceadds_3.18-36  mice_3.18.0
[13] tidyverse_1.3.1   ggdag_0.2.13    dagitty_0.3-4
```

```
[16] fs_1.6.6           here_1.0.2          sensitivitymw_2.1
[19] lubridate_1.9.4    tibble_3.3.0         survey_4.4-8
[22] survival_3.8-3    Matrix_1.7-4        rms_8.0-0
[25] Hmisc_5.2-3       patchwork_1.3.2    officer_0.7.0
[28] modelsummary_2.5.0 scales_1.4.0        labelled_2.15.0
[31] haven_2.5.5       gt_1.1.0           ggplot2_4.0.0
[34] gbm_2.2.2          flextable_0.9.10   dplyr_1.1.4
[37] codebookkr_0.1.8   cobalt_4.6.1       broom_1.0.11
[40] WeightIt_1.5.0    purrr_1.2.0       gtsummary_2.4.0
[43] kableExtra_1.4.0
```

loaded via a namespace (and not attached):

```
[1] polspline_1.1.25      polyclip_1.10-7      datawizard_1.2.0
[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      minqa_1.2.8
[19] DBI_1.2.3            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] ggrepel_0.9.6         listenv_0.9.1     cards_0.7.0
[31] MatrixModels_0.5-4    performance_0.15.1 parallelly_1.45.1
[34] svglite_2.2.1         commonmark_2.0.0   codetools_0.2-20
[37] xml2_1.4.0            ggforce_0.5.0      shape_1.4.6.1
[40] tidyselect_1.2.1      farver_2.1.2      lme4_1.1-38
[43] viridis_0.6.5          effectsize_1.0.1  base64enc_0.1-3
[46] jsonlite_2.0.0        mitml_0.4-5       tidygraph_1.3.1
[49] Formula_1.2-5         iterators_1.0.14 emmeans_1.11.2-8
[52] systemfonts_1.2.3     foreach_1.5.2     tools_4.5.0
[55] ragg_1.5.0             Rcpp_1.1.0        glue_1.8.0
[58] pan_1.9                gridExtra_2.3     xfun_0.53
[61] chk_0.10.0             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           digest_0.6.37    timechange_0.3.0
[70] R6_2.6.1               estimability_1.5.1 textshaping_1.0.3
```

```
[73] colorspace_2.1-2      markdown_2.0          utf8_1.2.6
[76] generics_0.1.4        fontLiberation_0.1.0   data.table_1.17.8
[79] graphlayouts_1.2.2    htmlwidgets_1.6.4     parameters_0.28.2
[82] pkgconfig_2.0.3       gtable_0.3.6         lmtest_0.9-40
[85] S7_0.2.0              htmtools_0.5.8.1    fontBitstreamVera_0.1.1
[88] reformulas_0.4.2     knitr_1.50          rstudioapi_0.17.1
[91] uuid_1.2-1            nloptr_2.2.1         coda_0.19-4.1
[94] checkmate_2.3.3      nlme_3.1-168        curl_7.0.0
[97] repr_1.1.7             cachem_1.1.0        zoo_1.8-14
[100] stringr_1.6.0        foreign_0.8-90     pillar_1.11.1
[103] vctrs_0.6.5          jomo_2.7-6          xtable_1.8-4
[106] cluster_2.1.8.1      htmlTable_2.4.3    evaluate_1.0.5
[109] tinytex_0.57          magick_2.9.0        mvtnorm_1.3-3
[112] cli_3.6.5             compiler_4.5.0     crayon_1.5.3
[115] labeling_0.4.3       forcats_1.0.1       stringi_1.8.7
[118] viridisLite_0.4.2     tables_0.9.31      glmnet_4.1-10
[121] bayestestR_0.17.0     V8_7.0.0          quantreg_6.1
[124] fontquiver_0.2.1      hms_1.1.4          rbibutils_2.4
[127] igraph_2.2.1          memoise_2.0.1      xgboost_1.7.11.1
```

Chunk mi-session runtime: 0.06 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	87.0	death_60d	86.3
1	vbg_o2sat	80.6	death_60d	86.3
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	82.9	death_60d	81.8
1	bnp	72.4	death_60d	81.8
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	72.0	death_60d	71.0
1	vbg_co2	61.9	death_60d	71.0
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	70.3	death_60d	70.6
1	spo2	73.3	death_60d	70.6
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	66.5	death_60d	63.7
1	paco2	39.2	death_60d	63.7
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	62.3	death_60d	59.8
1	serum_lac	38.3	death_60d	59.8
0	serum_lac	71.8	has_abg	59.8
1	serum_lac	38.9	has_abg	59.8
0	serum_lac	64.9	282 has_vbg	59.8

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	location	2.30	2.28	2.32	0
temp_new	has_abg	0.46	0.45	0.47	0
sbp	encounter_typeInpatient	0.05	0.05	0.05	0
sbp	location	109.41	100.50	119.11	0
dbp	encounter_typeInpatient	0.05	0.05	0.05	0
dbp	location	53.71	50.39	57.26	0
hr	has_abg	0.50	0.49	0.51	0
hr	has_vbg	2.89	2.82	2.95	0
spo2	encounter_typeInpatient	0.26	0.25	0.26	0
spo2	imv_proc	0.48	0.46	0.50	0
sodium	age_at_encounter	0.97	0.97	0.97	0
sodium	has_abg	0.14	0.12	0.15	0
sodium	has_vbg	0.08	0.07	0.09	0
serum_cr	age_at_encounter	0.97	0.97	0.97	0
serum_cr	has_abg	0.22	0.21	0.23	0
serum_cr	has_vbg	0.20	0.19	0.22	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	0.15	0.18	0
serum_cl	age_at_encounter	0.97	0.97	0.97	0
serum_cl	has_abg	0.16	0.15	0.18	0
serum_cl	has_vbg	0.10	0.09	0.11	0
serum_lac	location	1.44	1.43	1.45	0
serum_lac	has_abg	0.24	0.23	0.24	0
serum_lac	has_vbg	0.47	0.46	0.48	0
serum_k	age_at_encounter	0.97	0.97	0.97	0
serum_k	has_abg	0.15	0.14	0.16	0
serum_k	has_vbg	0.12	0.11	0.13	0
wbc	age_at_encounter	0.98	0.98	0.98	0
wbc	location	1.37	1.36	1.39	0
plt	age_at_encounter	0.98	0.97	0.98	0
plt	encounter_typeInpatient	0.43	0.41	0.44	0
plt	has_abg	0.25	0.24	0.27	0
plt	has_vbg	0.18	0.17	0.19	0
bnp	age_at_encounter	0.97	0.97	0.97	0
bnp	location	1.25	1.24	1.27	0
serum_phos	encounter_typeInpatient	0.13	0.13	0.14	0
serum_phos	location	0.82	0.81	0.83	0
serum_phos	has_abg	0.36	0.35	0.37	0
serum_phos	has_vbg	0.38	0.37	0.39	0
serum_phos	imv_proc	0.39	0.37	0.41	0
serum_ca	age_at_encounter	0.97	0.97	0.97	0
serum_ca	encounter_typeInpatient	0.42	0.40	0.43	0
serum_ca	has_abg	0.19	0.18	0.21	0
vbg_o2sat	encounter_typeInpatient	0.25	0.24	0.26	0

Monte Carlo error vs SE (should be much smaller)

Term	Estimate	SE	MC error	MC error / SE	2.5%	97.5%
(Intercept)	-3.673	0.044	—	—	-3.766911229	-3.579206530
has_abg	2.187	0.012	—	—	2.163401734	2.209664808
age_at_encounter	-0.007	0.000	—	—	-0.007430933	-0.006299448
curr_bmi	-0.010	0.001	—	—	-0.012910708	-0.008050564
sexMale	0.289	0.010	—	—	0.269858965	0.308176817
encounter_typeInpatient	1.126	0.015	—	—	1.096026962	1.156402828

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	1.23 (1.05, 1.44)	1.10 (0.94, 1.29)
ABG	NIV	0.95 (0.73, 1.24)	1.99 (1.59, 2.49)
ABG	Death (60d)	1.72 (1.44, 2.07)	1.52 (1.26, 1.83)
ABG	Hypercapnic RF	0.85 (0.63, 1.16)	5.16 (4.08, 6.53)
VBG	IMV	1.31 (1.05, 1.63)	1.54 (1.23, 1.92)
VBG	NIV	1.69 (1.18, 2.44)	3.75 (2.72, 5.18)
VBG	Death (60d)	1.89 (1.50, 2.38)	1.62 (1.27, 2.07)
VBG	Hypercapnic RF	1.12 (0.75, 1.67)	8.80 (6.44, 12.03)

Weighted survey GLMs; weights = MI-specific GBM IPW; m = 5 imputations (seed 20251206); reference = Eucapnia.