Biostat 203B Homework 5

Due Mar 20 @ 11:59PM

AUTHOR
Sophia Luo, 106409469

```r
library(tidyverse)
```

```
── Attaching core tidyverse packages ─────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.4
── Conflicts ───────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```r
library(tidymodels)
```

```
── Attaching packages ────────────────────────────────────── tidymodels 1.3.0 ──
✔ broom        1.0.7     ✔ rsample      1.2.1
✔ dials        1.4.0     ✔ tune         1.3.0
✔ infer        1.0.7     ✔ workflows    1.2.0
✔ modeldata    1.4.0     ✔ workflowsets 1.1.0
✔ parsnip      1.3.1     ✔ yardstick    1.3.2
✔ recipes      1.2.0
── Conflicts ───────────────────────────────────────── tidymodels_conflicts() ──
✖ scales::discard() masks purrr::discard()
✖ dplyr::filter()   masks stats::filter()
✖ recipes::fixed()  masks stringr::fixed()
✖ dplyr::lag()      masks stats::lag()
✖ yardstick::spec() masks readr::spec()
✖ recipes::step()   masks stats::step()
```

```r
library(gtsummary)
library(ggplot2)
library(recipes) # Data Preprocessing
library(glmnet) # Logistic Regression
```

```
Loading required package: Matrix

Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':
    expand, pack, unpack
```

```
Loaded glmnet 4.1-8
```

```
library(caret) # Random Forest
```

```
Loading required package: lattice

Attaching package: 'caret'

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

    precision, recall, sensitivity, specificity

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

    lift
```

```
library(ranger) # Random Forest
library(xgboost) # XGBoost
```

```

Attaching package: 'xgboost'

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

    slice
```

```
library(stacks) # model stacking
library(broom) # extract model coefficients
```

Predicting ICU duration

Using the ICU cohort `mimiciv_icu_cohort.rds` you built in Homework 4, develop at least three machine learning approaches (logistic regression with enet regularization, random forest, boosting, SVM, MLP, etc) plus a model stacking approach for predicting whether a patient's ICU stay will be longer than 2 days. You should use the `los_long` variable as the outcome. You algorithms can use patient demographic information (gender, age at ICU `intime`, marital status, race), ICU admission information (first care unit), the last lab measurements before the ICU stay, and first vital measurements during ICU stay as features. You are welcome to use any feature engineering techniques you think are appropriate; but make sure to not use features that are not available at an ICU stay's `intime`. For instance, `last_careunit` cannot be used in your algorithms.

1. Data preprocessing and feature engineering.

```
# Load Data
cohort_data <- readRDS("../hw4/mimiciv_shiny/mimic_icu_cohort.rds")
```

```r
# Select Features & Target
mimiciv_icu_cohort <- cohort_data |>
  select(
    subject_id, hadm_id, stay_id,

    los_long, # Target variable (ICU stay > 2 days)

    # Demographics
    gender, age_intime, marital_status, race,

    # ICU admission
    first_careunit,

    # Lab measurements (Last before ICU)
    bicarbonate, chloride, creatinine, glucose, hematocrit,
    potassium, sodium, wbc,

    # Vital measurements (First in ICU)
    heart_rate, non_invasive_blood_pressure_diastolic,
    non_invasive_blood_pressure_systolic, respiratory_rate,
    temperature_fahrenheit
  ) |>
  # Convert categorical variables to factors
  mutate(
    gender = as.factor(gender),
    marital_status = as.factor(marital_status),
    race = as.factor(tolower(race))
  ) |>
  drop_na(los_long) |>
  print(width = Inf)
```

```
# A tibble: 94,444 × 22
   subject_id  hadm_id  stay_id los_long gender age_intime marital_status race
        <int>    <int>    <int> <fct>    <fct>       <int> <fct>          <fct>
 1   10000032 29079034 39553978 FALSE    F              52 WIDOWED        white
 2   10000690 25860671 37081114 TRUE     F              86 WIDOWED        white
 3   10000980 26913865 39765666 FALSE    F              76 MARRIED        black
 4   10001217 24597018 37067082 FALSE    F              55 MARRIED        white
 5   10001217 27703517 34592300 FALSE    F              55 MARRIED        white
 6   10001725 25563031 31205490 FALSE    F              46 MARRIED        white
 7   10001843 26133978 39698942 FALSE    M              76 SINGLE         white
 8   10001884 26184834 37510196 TRUE     F              77 MARRIED        black
 9   10002013 23581541 39060235 FALSE    F              57 SINGLE         other
10   10002114 27793700 34672098 TRUE     M              56 <NA>           other
   first_careunit                                   bicarbonate chloride
   <fct>                                                  <dbl>    <dbl>
 1 Medical Intensive Care Unit (MICU)                        25       95
 2 Medical Intensive Care Unit (MICU)                        26      100
 3 Medical Intensive Care Unit (MICU)                        21      109
 4 Surgical Intensive Care Unit (SICU)                      22      108
 5 Surgical Intensive Care Unit (SICU)                      30      104
 6 Medical/Surgical Intensive Care Unit (MICU/SICU)         NA       98
 7 Medical/Surgical Intensive Care Unit (MICU/SICU)         28       97
 8 Medical Intensive Care Unit (MICU)                       30       88
 9 Cardiac Vascular Intensive Care Unit (CVICU)             24      102
10 Other                                                    18       NA
   creatinine glucose hematocrit potassium sodium   wbc heart_rate
        <dbl>   <dbl>      <dbl>     <dbl>  <dbl> <dbl>      <dbl>
 1        0.7     102       41.1       6.7    126   6.9       91
 2        1        85       36.1       4.8    137   7.1       78
 3        2.3      89       27.3       3.9    144   5.3       76
 4        0.6     112       38.1       4.2    142  15.7       86
 5        0.5      87       37.4       4.1    142   5.4       79.3
 6       NA        NA       NA         4.1    139  NA         86
 7        1.3     131       31.4       3.9    138  10.4      124.
 8        1.1     141       39.7       4.5    130  12.2       49
 9        0.9     288       34.9       3.5    137   7.2       80
10        3.1      95       34.3       6.5    125  16.8      110.
   non_invasive_blood_pressure_diastolic non_invasive_blood_pressure_systolic
                                   <dbl>                                <dbl>
 1                                  48                                   84
 2                                  56.5                                106
 3                                 102                                  154
 4                                  90                                  151
 5                                  93.3                                156
 6                                  56                                   73
 7                                  78                                  110
 8                                  30.5                                174.
 9                                  62                                   98.5
10                                  80                                  112
   respiratory_rate temperature_fahrenheit
              <dbl>                  <dbl>
 1             24                     98.7
 2             24.3                   97.7
 3             23.5                   98
 4             18                     98.5
 5             14                     97.6
 6             19                     97.7
 7             16.5                   97.9
 8             13                     98.1
 9             14                     97.2
10             21                     97.9
# ℹ 94,434 more rows
```

```
# Check for missing values
mimiciv_icu_cohort |> tbl_summary(by = los_long)
```

Characteristic	**TRUE** N = 46,337[1]	**FALSE** N = 48,107[1]
subject_id	15,021,968 (12,517,625, 17,521,224)	14,988,897 (12,506,011, 17,513,478)
hadm_id	25,011,290 (22,497,215, 27,470,855)	24,954,662 (22,465,369, 27,459,051)
stay_id	34,949,825 (32,473,497, 37,458,915)	35,045,664 (32,534,836, 37,518,493)
gender		
F	20,106 (43%)	21,471 (45%)
M	26,231 (57%)	26,636 (55%)
age_intime	67 (56, 77)	66 (54, 77)
marital_status		
DIVORCED	3,377 (8.0%)	3,555 (8.0%)
MARRIED	20,557 (49%)	21,344 (48%)
SINGLE	12,745 (30%)	14,039 (31%)
WIDOWED	5,319 (13%)	5,752 (13%)
Unknown	4,339	3,417
race		
asian	1,369 (3.0%)	1,516 (3.2%)
black	4,933 (11%)	5,452 (11%)
hispanic	1,687 (3.6%)	1,908 (4.0%)
other	8,036 (17%)	6,880 (14%)
white	30,312 (65%)	32,351 (67%)
first_careunit		

[1] Median (Q1, Q3); n (%)

Characteristic	TRUE N = 46,337[1]	FALSE N = 48,107[1]
Cardiac Vascular Intensive Care Unit (CVICU)	7,353 (16%)	7,416 (15%)
Medical Intensive Care Unit (MICU)	9,837 (21%)	10,862 (23%)
Medical/Surgical Intensive Care Unit (MICU/SICU)	6,667 (14%)	8,780 (18%)
Surgical Intensive Care Unit (SICU)	6,434 (14%)	6,574 (14%)
Other	16,046 (35%)	14,475 (30%)
bicarbonate	24.0 (21.0, 27.0)	24.0 (21.0, 27.0)
Unknown	6,272	5,277
chloride	102 (98, 105)	102 (98, 105)
Unknown	6,184	5,167
creatinine	1.00 (0.80, 1.60)	1.00 (0.80, 1.40)
Unknown	4,541	3,486
glucose	122 (100, 159)	118 (98, 154)
Unknown	6,340	5,314
hematocrit	35 (29, 40)	36 (30, 41)
Unknown	3,857	2,894
potassium	4.20 (3.90, 4.70)	4.20 (3.90, 4.60)
Unknown	6,200	5,187
sodium	138.0 (135.0, 141.0)	139.0 (136.0, 141.0)
Unknown	6,167	5,163
wbc	9.7 (7.0, 13.8)	9.0 (6.6, 12.6)
Unknown	3,906	2,944
heart_rate	87 (75, 102)	84 (73, 99)

[1] Median (Q1, Q3); n (%)

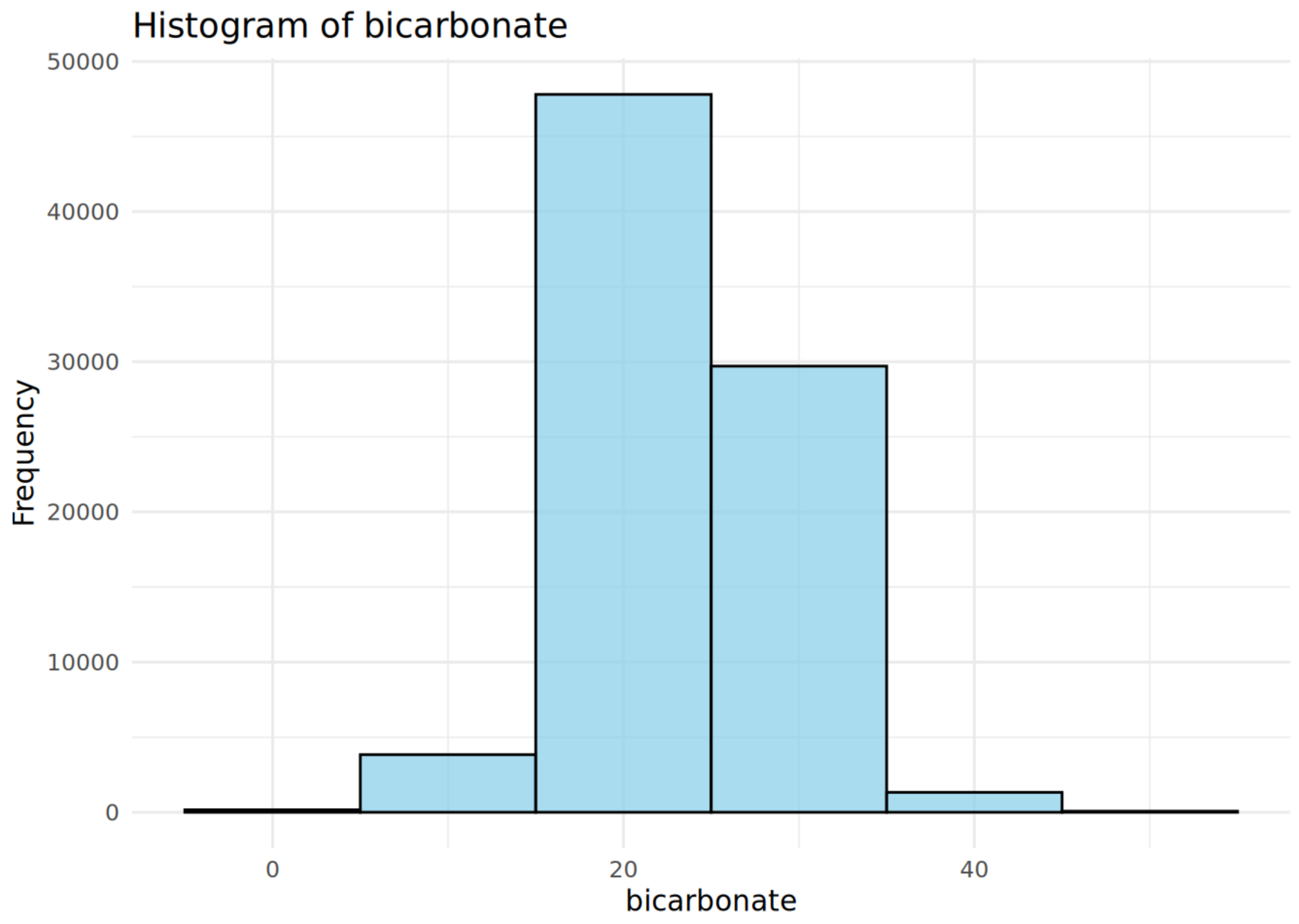
Characteristic	TRUE N = 46,337[1]	FALSE N = 48,107[1]
Unknown	1	84
non_invasive_blood_pressure_diastolic	67 (57, 79)	68 (58, 80)
Unknown	350	1,015
non_invasive_blood_pressure_systolic	120 (104, 137)	122 (107, 138)
Unknown	347	1,013
respiratory_rate	19.0 (16.0, 23.0)	18.0 (15.0, 22.0)
Unknown	14	181
temperature_fahrenheit	98.20 (97.70, 98.80)	98.10 (97.60, 98.60)
Unknown	230	1,386

[1] Median (Q1, Q3); n (%)

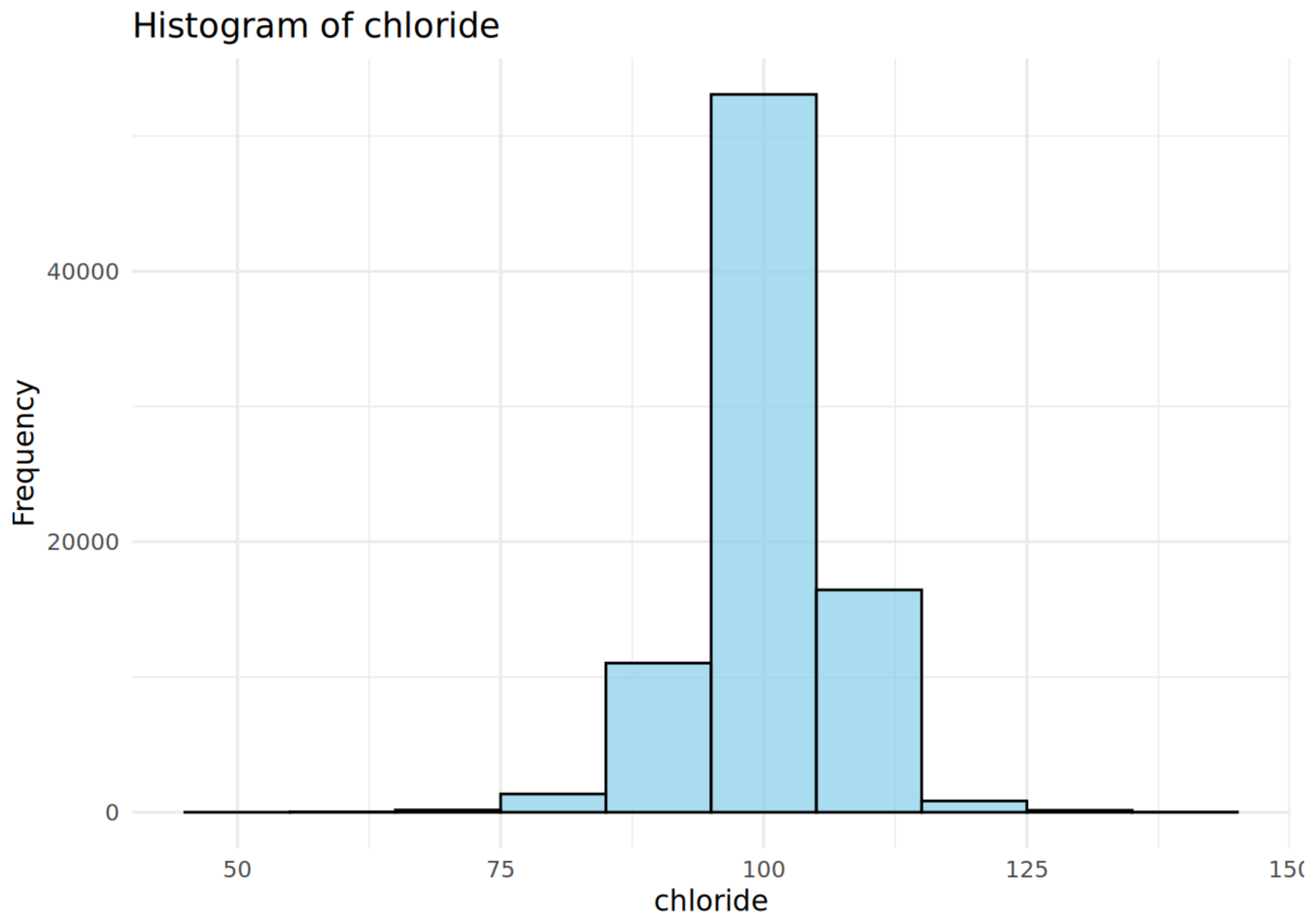
There are missing values in `marital_status`, lab measurements, and vital measurements. Missing values are visualized before deciding how to handle them.

```r
numeric_cols <- select(mimiciv_icu_cohort |>
                         select(-subject_id, -hadm_id, -stay_id, -los_long,
                                -gender, -age_intime, -race, -first_careunit),
                       where(is.numeric))  # Select numeric columns

# Loop through each numeric column and create a histogram
for (col in names(numeric_cols)) {
  p <- ggplot(mimiciv_icu_cohort,
              aes_string(x = col)) +
    geom_histogram(binwidth = 10, fill = "skyblue",
                   color = "black", alpha = 0.7) +
    labs(title = paste("Histogram of", col), x = col, y = "Frequency") +
    theme_minimal()
  print(p)
}
```

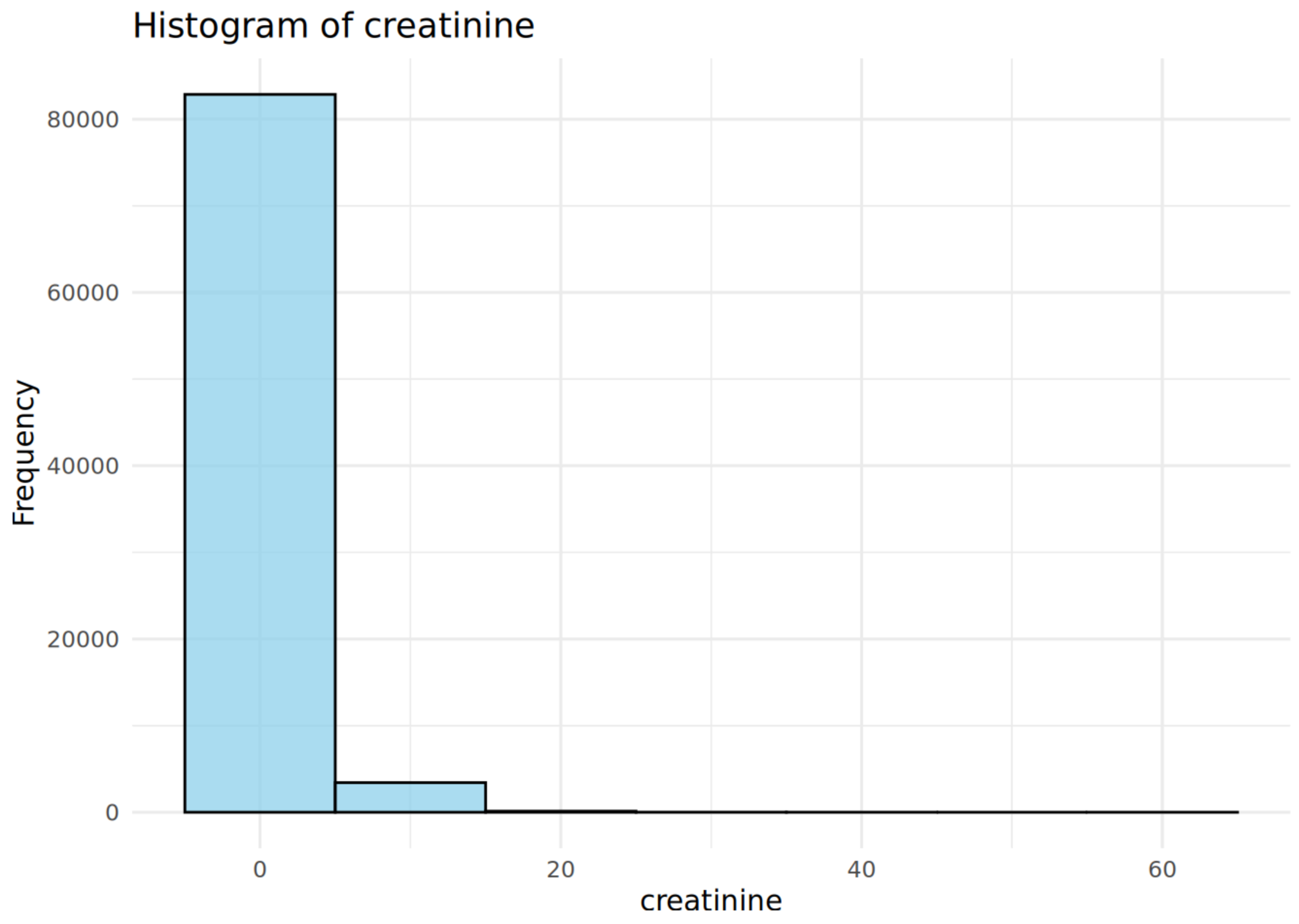
```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
ℹ Please use tidy evaluation idioms with `aes()`.
ℹ See also `vignette("ggplot2-in-packages")` for more information.

Warning: Removed 11549 rows containing non-finite outside the scale range
(`stat_bin()`).
```



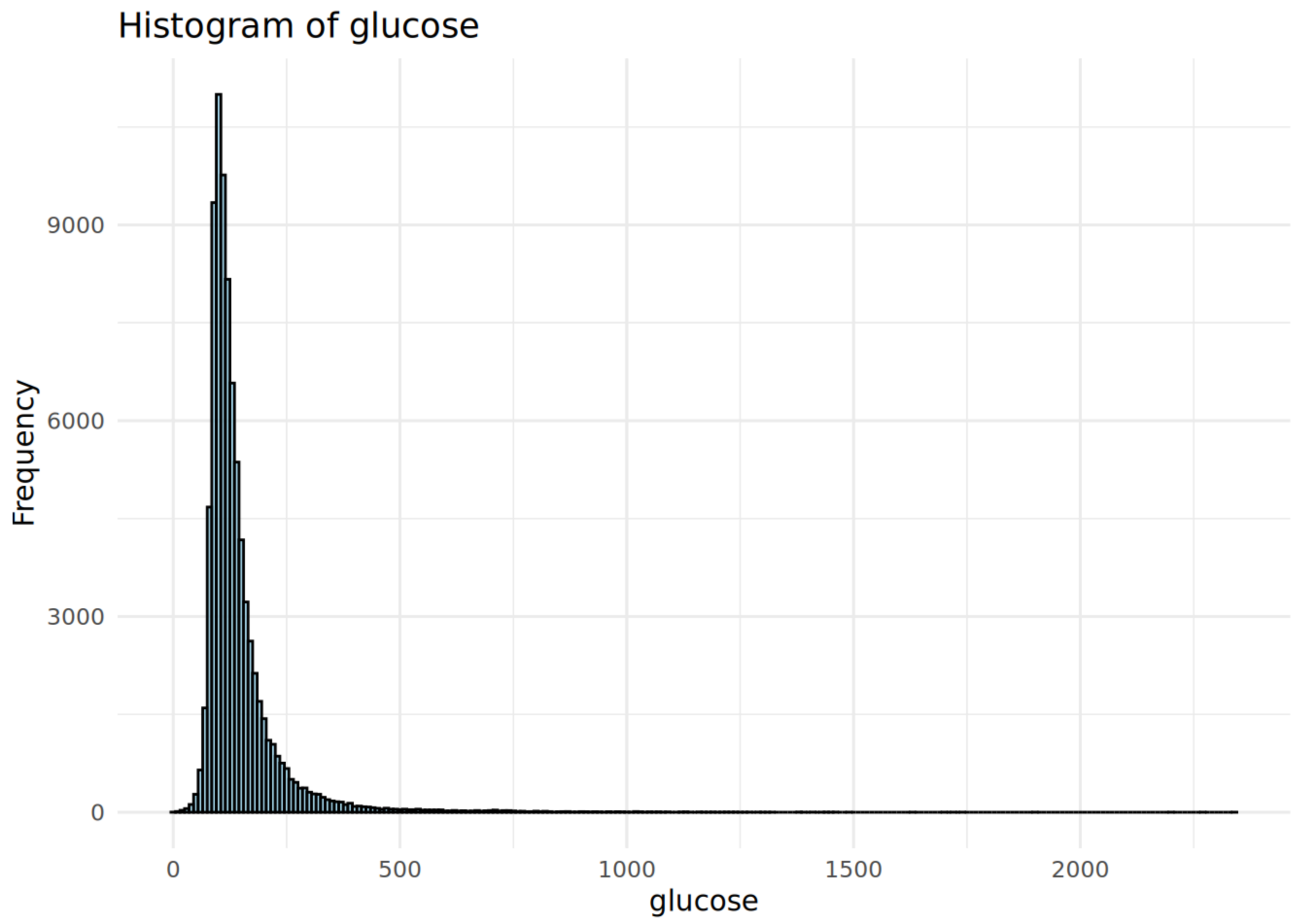
```
Warning: Removed 11351 rows containing non-finite outside the scale range
(`stat_bin()`).
```



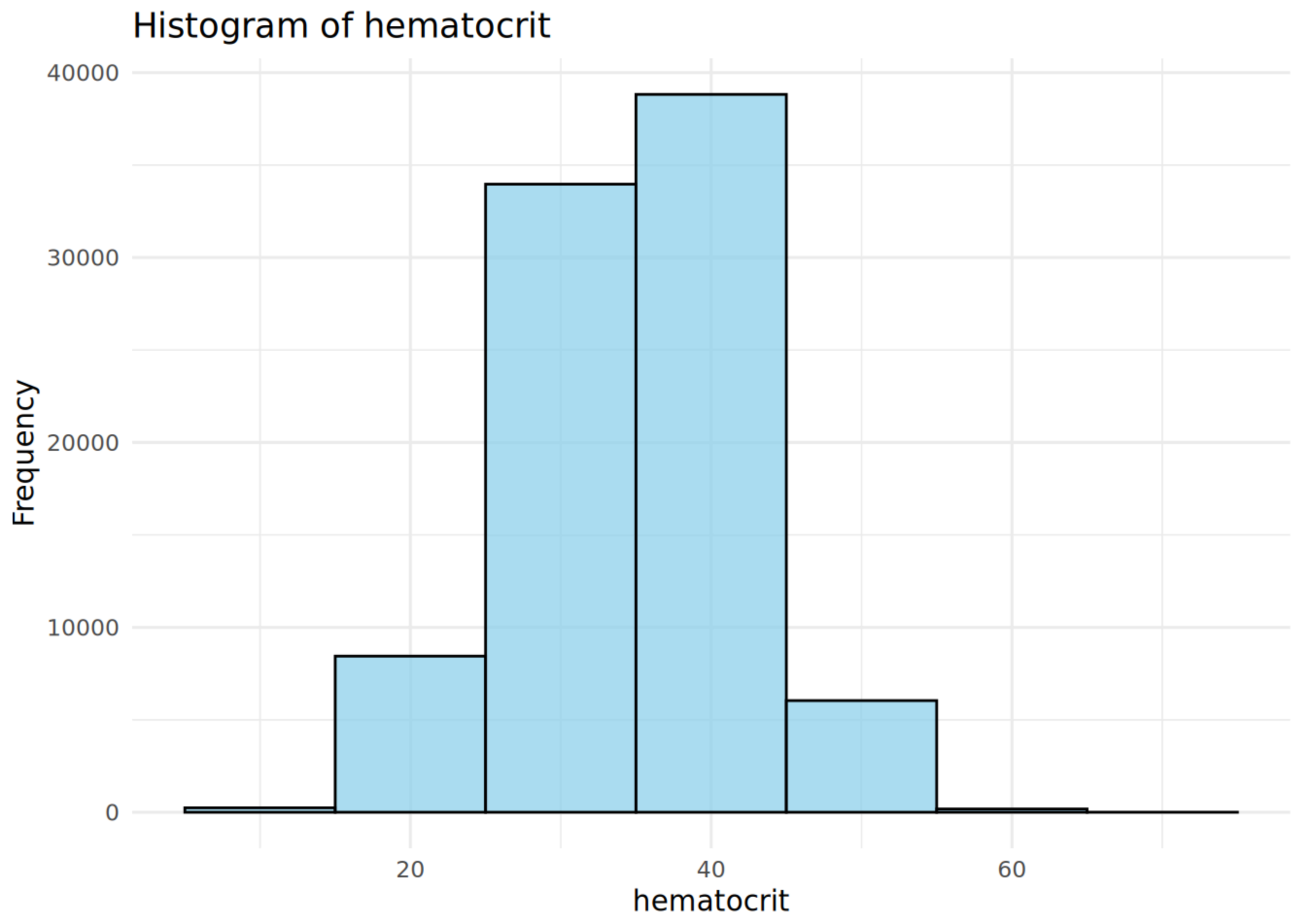
```
Warning: Removed 8027 rows containing non-finite outside the scale range
(`stat_bin()`).
```



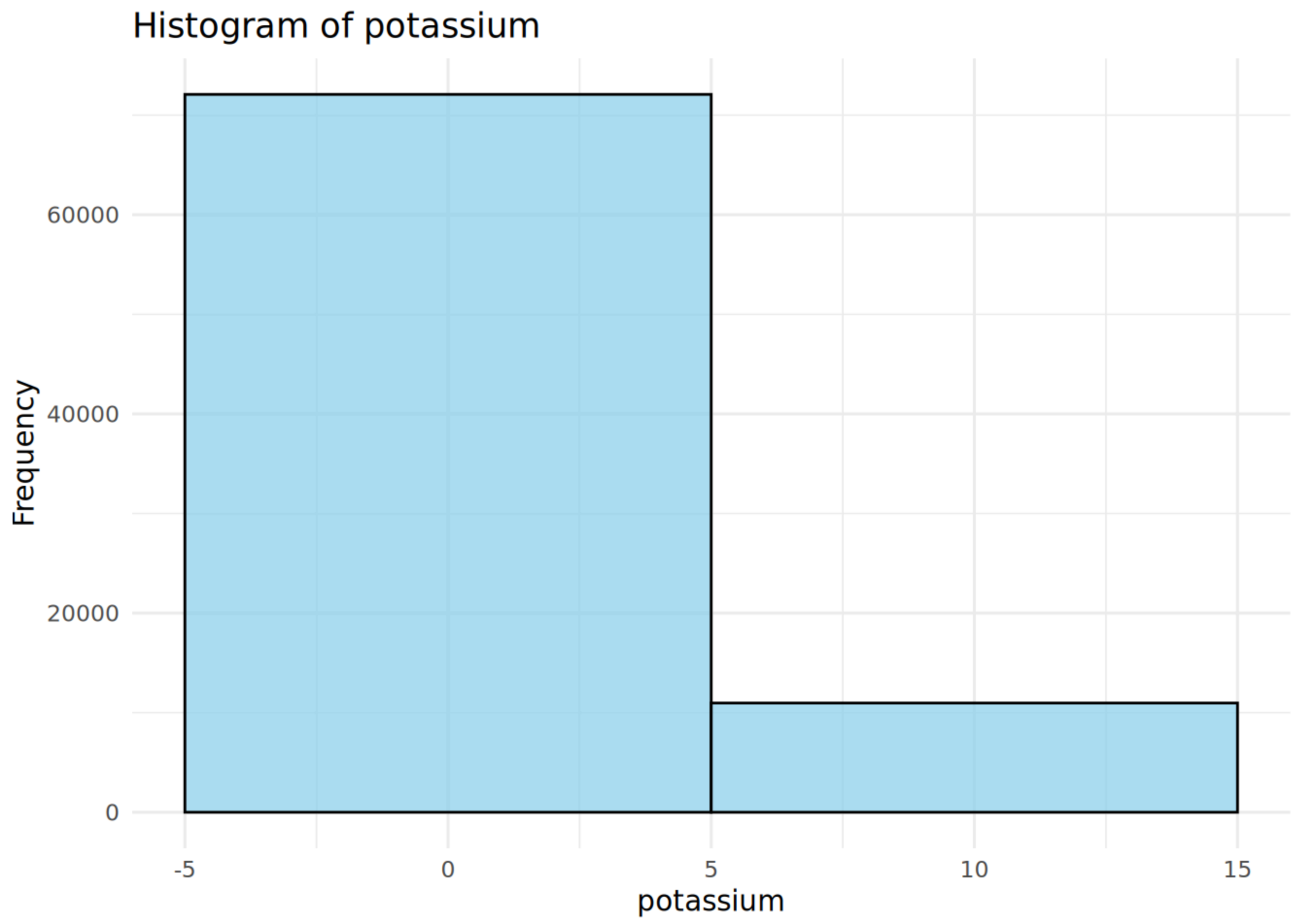
```
Warning: Removed 11654 rows containing non-finite outside the scale range
(`stat_bin()`).
```



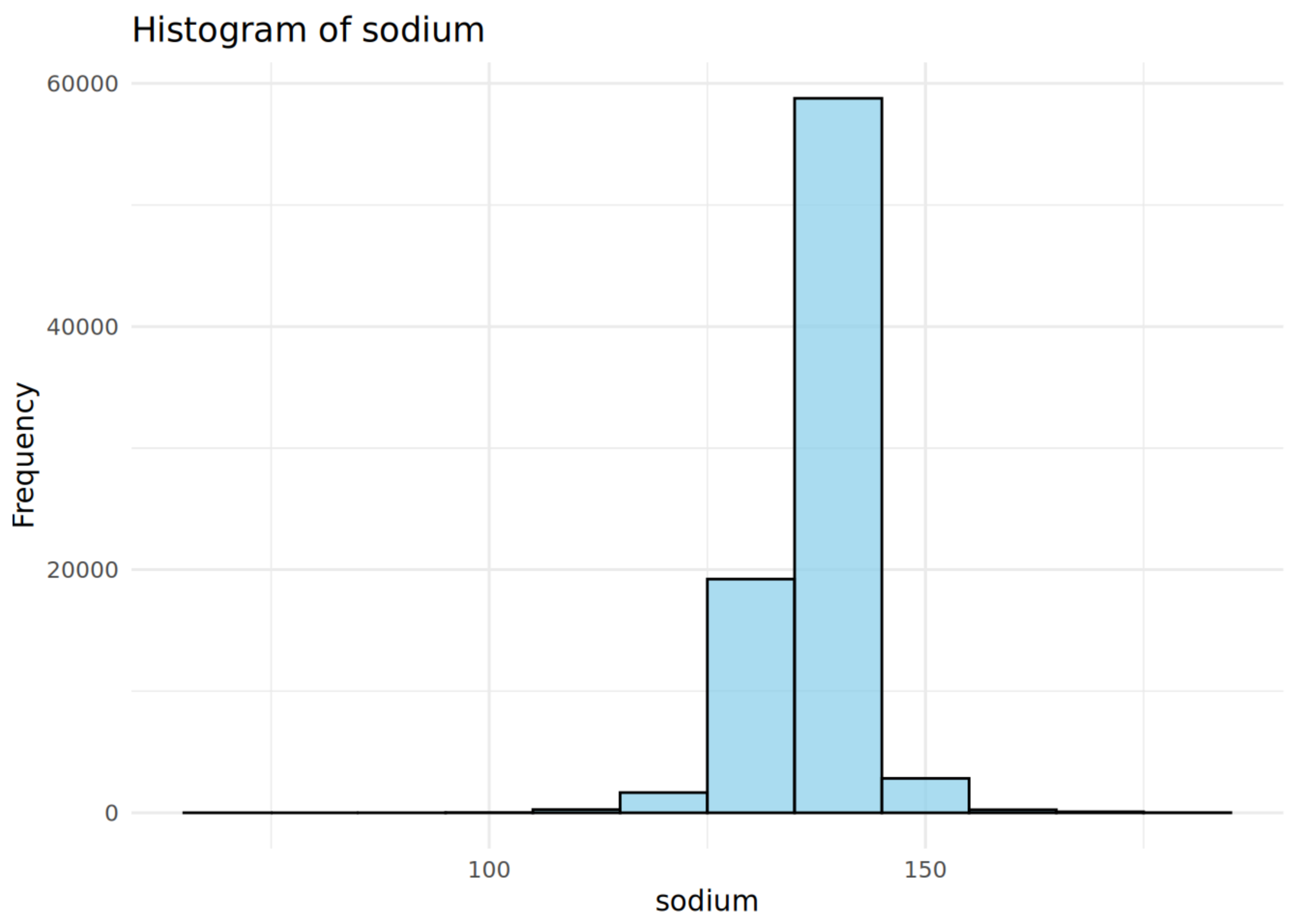
```
Warning: Removed 6751 rows containing non-finite outside the scale range
(`stat_bin()`).
```



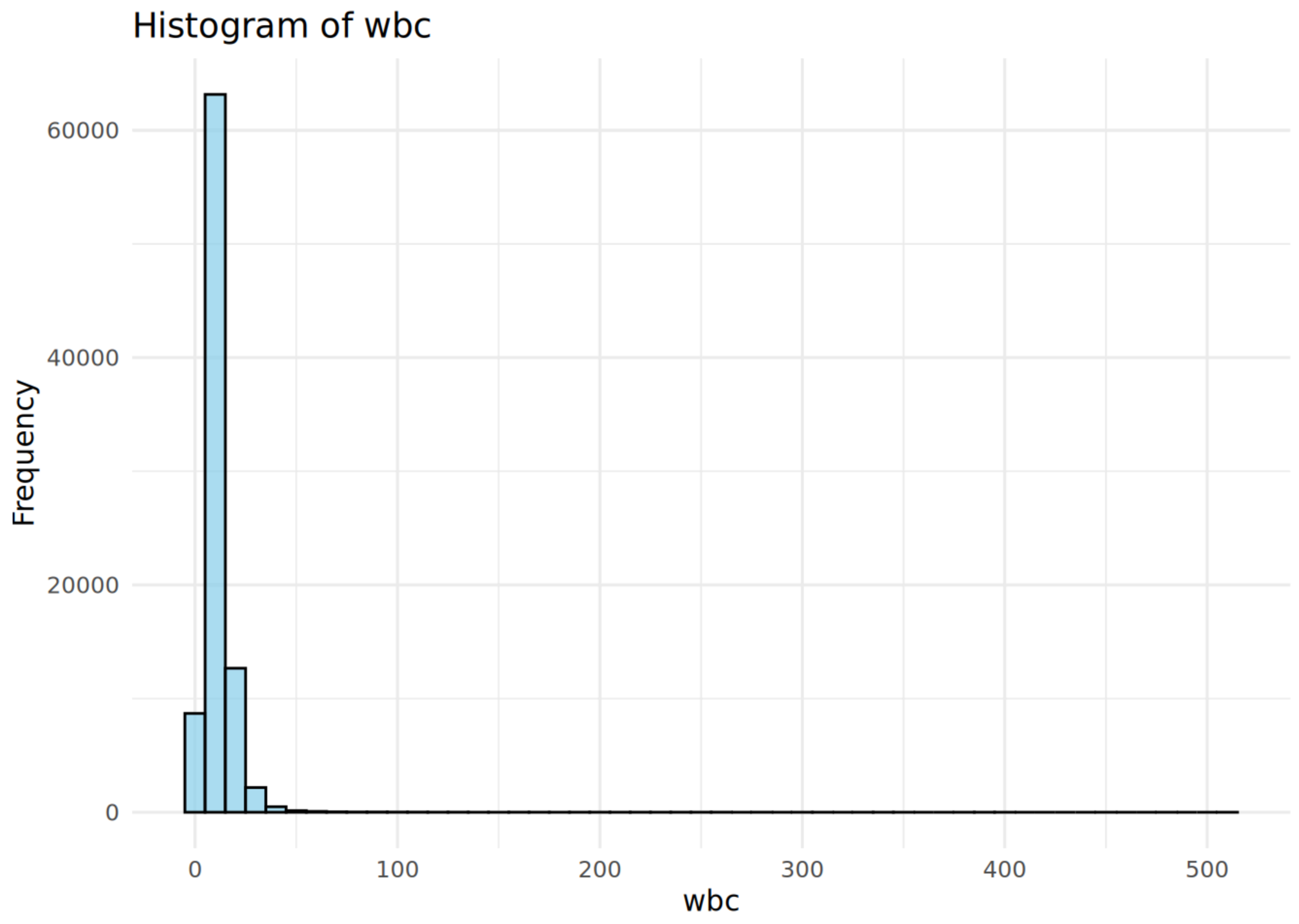
```
Warning: Removed 11387 rows containing non-finite outside the scale range
(`stat_bin()`).
```



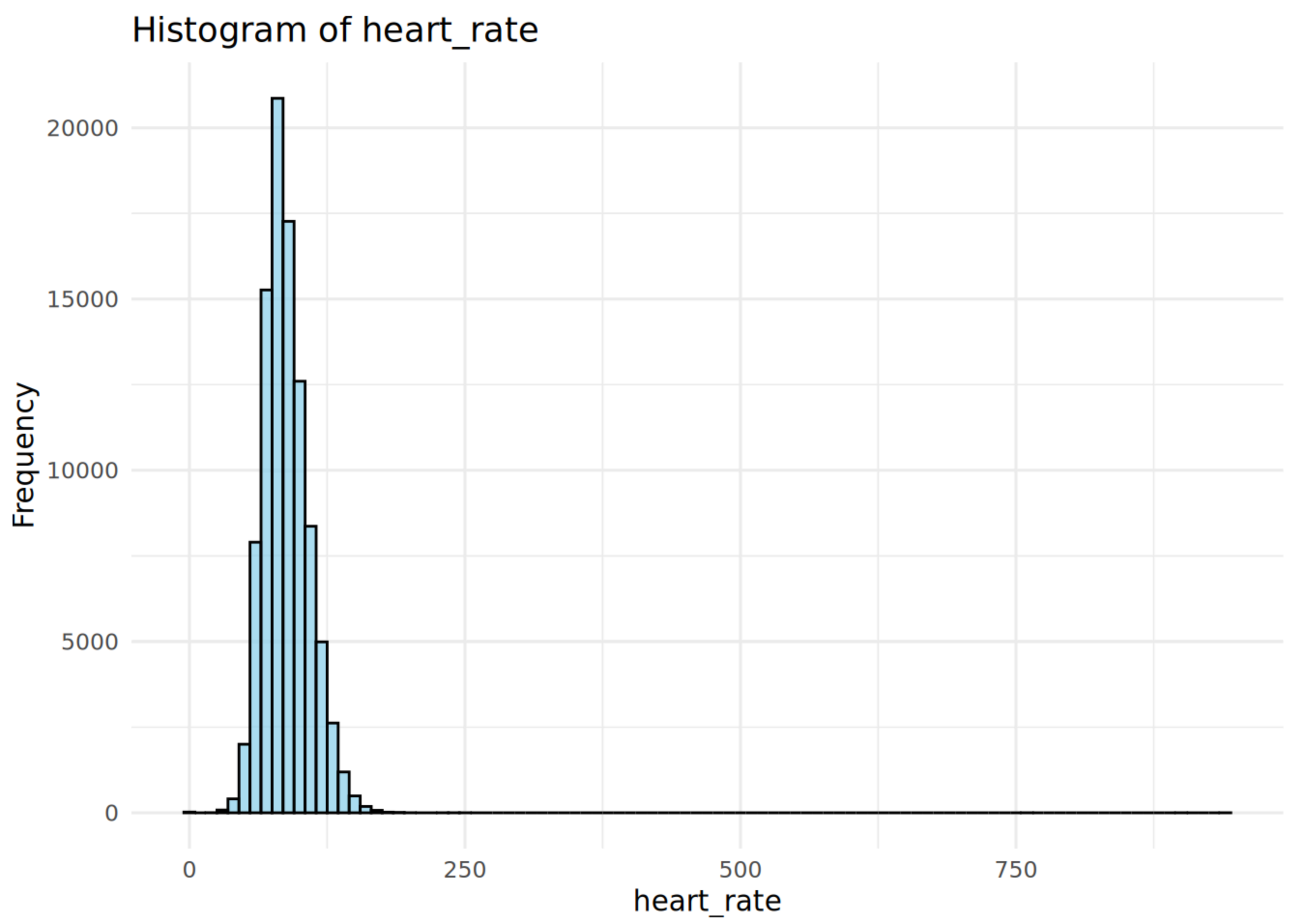
```
Warning: Removed 11330 rows containing non-finite outside the scale range
(`stat_bin()`).
```



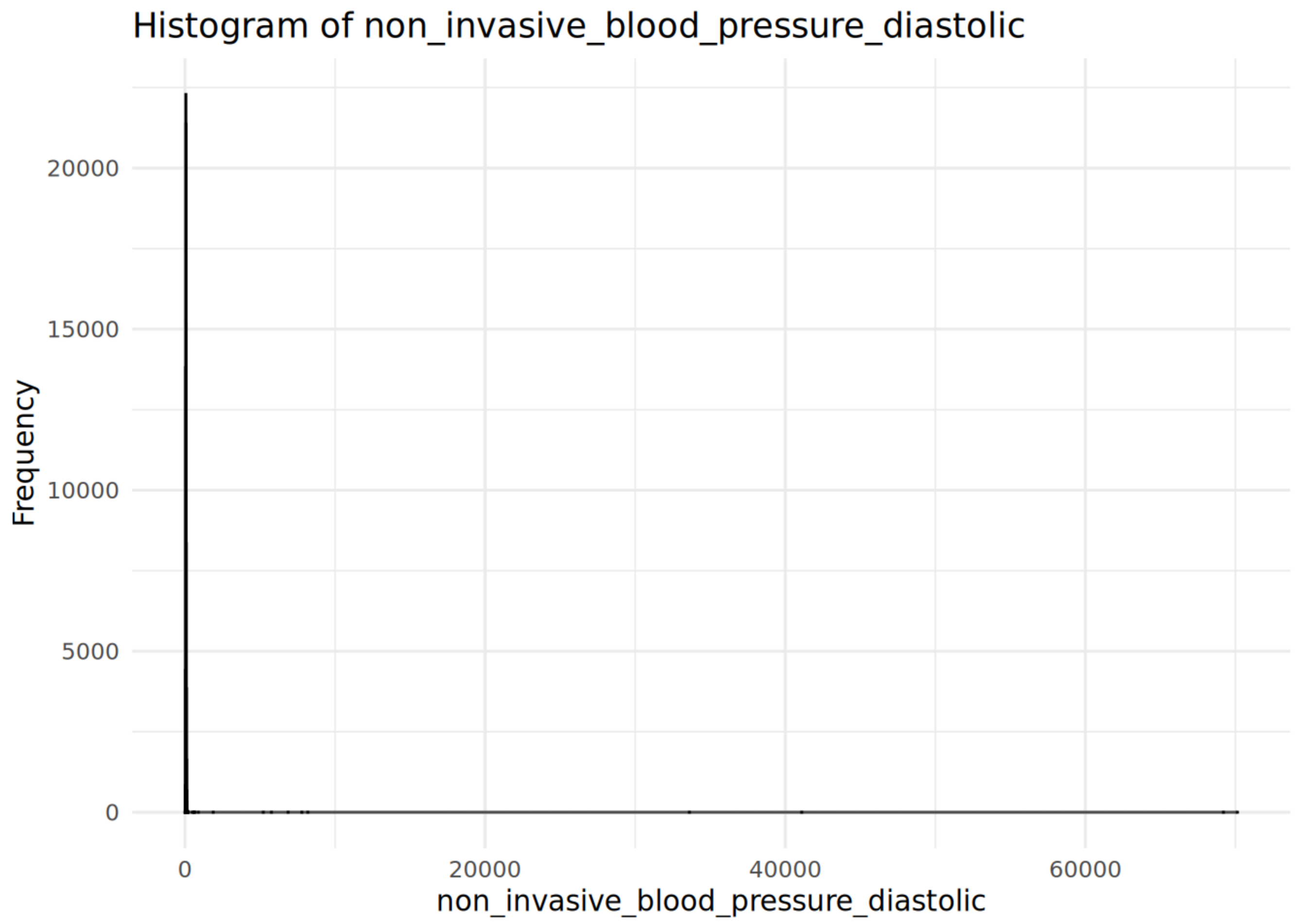
```
Warning: Removed 6850 rows containing non-finite outside the scale range
(`stat_bin()`).
```



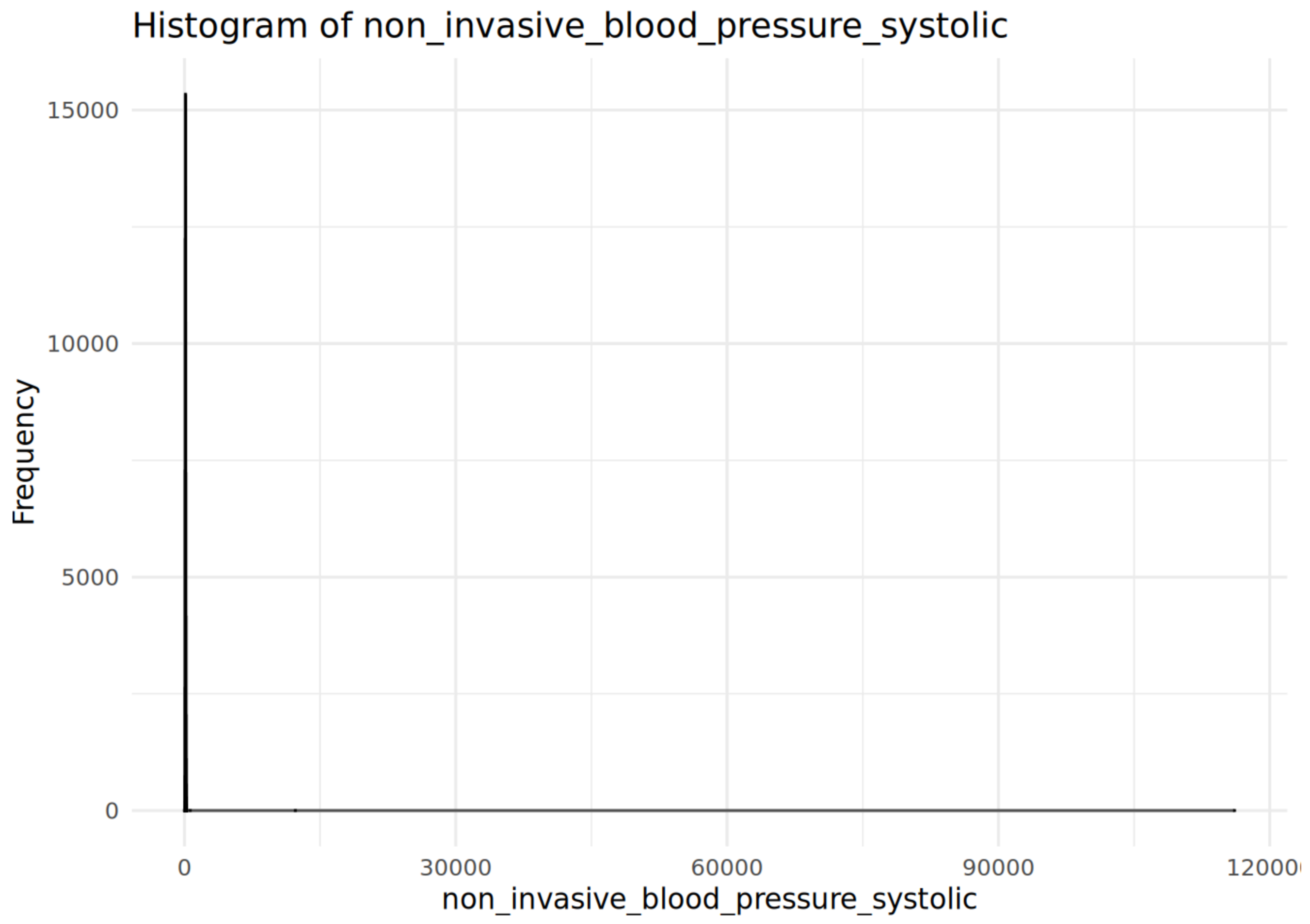
```
Warning: Removed 85 rows containing non-finite outside the scale range
(`stat_bin()`).
```



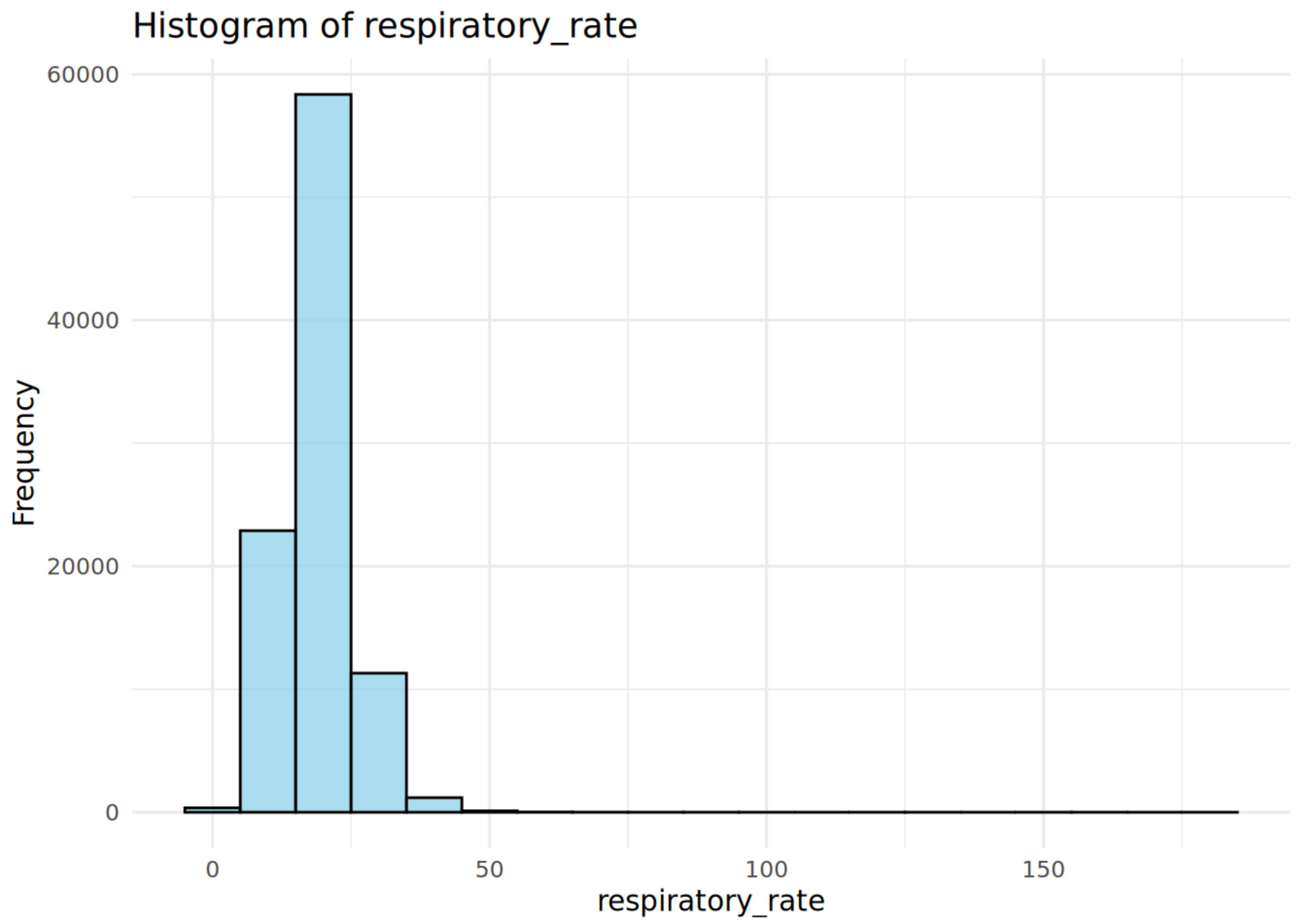
```
Warning: Removed 1365 rows containing non-finite outside the scale range
(`stat_bin()`).
```



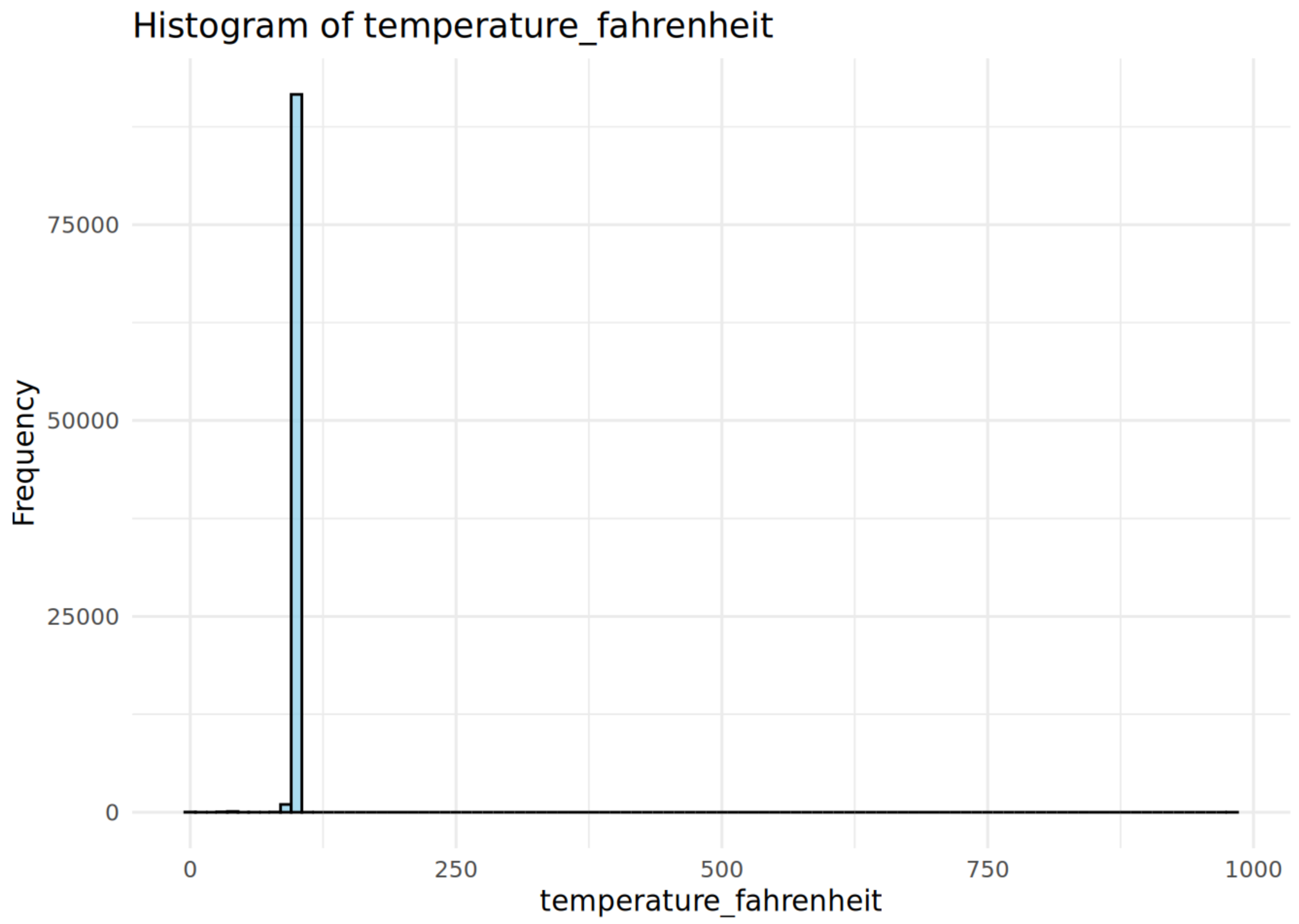
```
Warning: Removed 1360 rows containing non-finite outside the scale range
(`stat_bin()`).
```



```
Warning: Removed 195 rows containing non-finite outside the scale range
(`stat_bin()`).
```



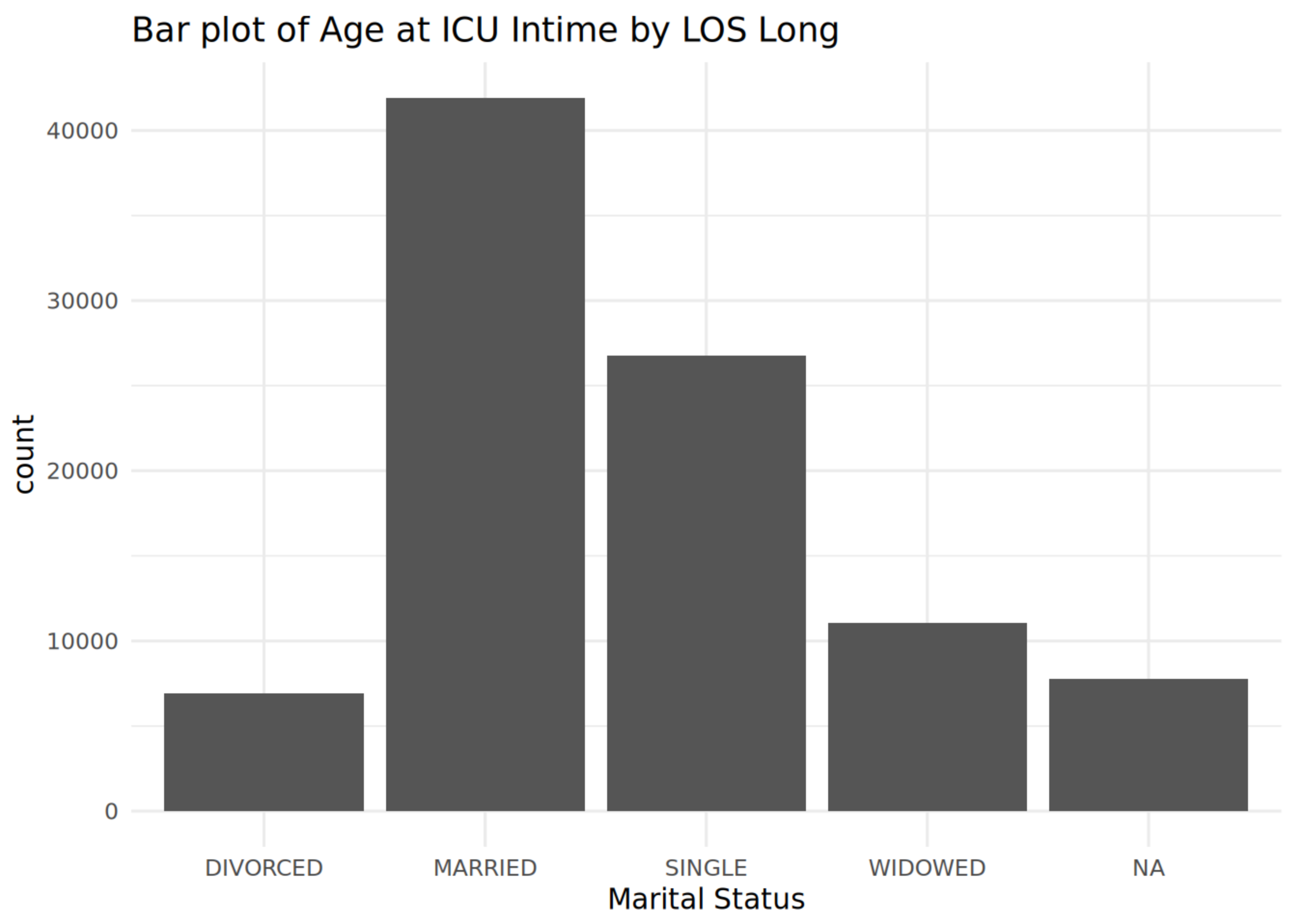
```
Warning: Removed 1616 rows containing non-finite outside the scale range
(`stat_bin()`).
```



Mean imputation should be used for normally distributed variables, `bicarbonate`, `chloride`, `hematocrit`, `sodium`, and `heart_rate`.

Median imputation should be used for skewed variables, `creatinine`, `glucose`, `potassium`, `wbc`, `non_invasive_blood_pressure_diastolic`, `non_invasive_blood_pressure_systolic`, `respiratory_rate`, and `temperature_fahrenheit`.

```r
ggplot(mimiciv_icu_cohort, aes(x = marital_status)) +
  geom_bar() +
  labs(title = "Bar plot of Age at ICU Intime by LOS Long",
       x = "Marital Status") +
  theme_minimal()
```



KNN imputation should be used for categorical variables, `marital_status`.

```r
rm(cohort_data)
gc()
```

```
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells 2892146 154.5    4883167 260.8  4883167 260.8
Vcells 6748227  51.5   23923775 182.6 23923775 182.6
```

2. Partition data into 50% training set and 50% test set.

Stratify partitioning according to `los_long`. For grading purpose, sort the data by `subject_id`, `hadm_id`, and `stay_id` and use the seed `203` for the initial data split. Below is the sample code.

```r
set.seed(203)

# sort
mimiciv_icu_cohort <- mimiciv_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id) |>
  # remove subject_id, hadm_id, stay_id
  select(-subject_id, -hadm_id, -stay_id)

data_split <- initial_split(
  mimiciv_icu_cohort,
  # stratify by los_long
  strata = "los_long",
  prop = 0.5
  )

# data_split
mimiciv_icu_cohort_train <- training(data_split)
dim(mimiciv_icu_cohort_train)
```

```
[1] 47221    19
```

```
mimiciv_icu_cohort_test <- testing(data_split)
dim(mimiciv_icu_cohort_test)
```

```
[1] 47223    19
```

```
rm(mimiciv_icu_cohort)
gc()
```

```
         used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells 2897657 154.8    4883167 260.8  4883167 260.8
Vcells 9806467  74.9   23923775 182.6 23923775 182.6
```

3. Train and tune the models using the training set.

Logistic regression with enet regularization

```
logit_rec <- recipe(los_long ~ ., data = mimiciv_icu_cohort_train) |>
  # Mean inputation for normal variables
  step_impute_mean(bicarbonate, chloride, hematocrit, sodium, heart_rate) |>

  # Median imputation for skewed variables
  step_impute_median(creatinine, glucose, potassium, wbc,
                     non_invasive_blood_pressure_diastolic,
                     non_invasive_blood_pressure_systolic,
                     respiratory_rate, temperature_fahrenheit) |>

  # KNN imputation for categorical variables
  step_impute_knn(marital_status) |>

  # create traditional dummy variables
  step_dummy(all_nominal_predictors()) |>
  # zero-variance filter
  step_zv(all_nominal_predictors()) |>
  # center and scale numeric data
  step_normalize(all_numeric_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────

── Inputs

Number of variables by role

outcome:    1
predictor: 18

── Operations

• Mean imputation for: bicarbonate, chloride, hematocrit, sodium, ...

• Median imputation for: creatinine, glucose, potassium, wbc, ...

• K-nearest neighbor imputation for: marital_status

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_nominal_predictors()

• Centering and scaling for: all_numeric_predictors()
```

```r
logit_mod <- logistic_reg(
  penalty = tune(),
  mixture = tune()
) |>
  set_engine("glmnet", standardize = FALSE) |>
  print()
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
logit_wf <- workflow() |>
  add_recipe(logit_rec) |>
  add_model(logit_mod) |>
  print()
```

```
══ Workflow ════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────
6 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_dummy()
• step_zv()
• step_normalize()

── Model ───────────────────────────────────────────────────────────────────
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
logit_grid <- grid_regular(
  penalty(range = c(-4, 1)),
  mixture(),
  levels = c(50, 5)
  ) |>
  print()
```

```
# A tibble: 250 × 2
    penalty mixture
      <dbl>   <dbl>
 1 0.0001         0
 2 0.000126       0
 3 0.000160       0
 4 0.000202       0
 5 0.000256       0
 6 0.000324       0
 7 0.000409       0
 8 0.000518       0
 9 0.000655       0
10 0.000829       0
# ℹ 240 more rows
```

```r
set.seed(203)
```

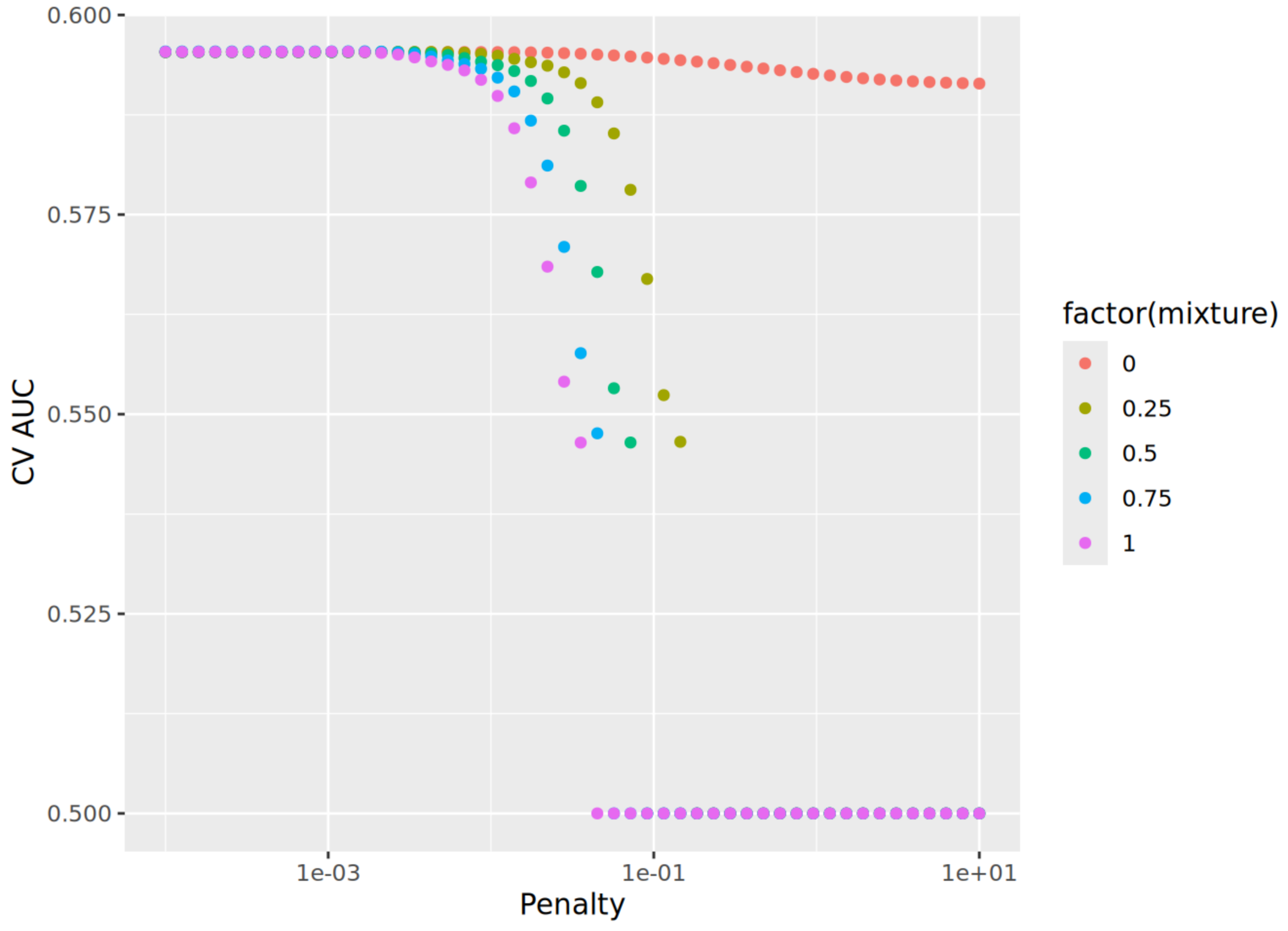
```r
folds <- vfold_cv(mimiciv_icu_cohort_train, v = 5, strata = los_long)
folds
```

```
#  5-fold cross-validation using stratification
# A tibble: 5 × 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37776/9445]> Fold2
3 <split [37776/9445]> Fold3
4 <split [37778/9443]> Fold4
5 <split [37778/9443]> Fold5
```

```r
logit_fit <- logit_wf |>
  tune_grid(
    resamples = folds,
    grid = logit_grid,
    metrics = metric_set(roc_auc, accuracy)
  )
```

```r
logit_fit |>
  # aggregate metrics from K folds
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = penalty, y = mean, color = factor(mixture))) +
  geom_point() +
  labs(x = "Penalty", y = "CV AUC") +
  scale_x_log10()
```

```
# A tibble: 500 × 8
    penalty mixture .metric  .estimator  mean     n  std_err
      <dbl>   <dbl> <chr>    <chr>      <dbl> <int>    <dbl>
 1 0.0001         0 accuracy binary     0.570     5 0.000757
 2 0.0001         0 roc_auc  binary     0.595     5 0.00119
 3 0.000126       0 accuracy binary     0.570     5 0.000757
 4 0.000126       0 roc_auc  binary     0.595     5 0.00119
 5 0.000160       0 accuracy binary     0.570     5 0.000757
 6 0.000160       0 roc_auc  binary     0.595     5 0.00119
 7 0.000202       0 accuracy binary     0.570     5 0.000757
 8 0.000202       0 roc_auc  binary     0.595     5 0.00119
 9 0.000256       0 accuracy binary     0.570     5 0.000757
10 0.000256       0 roc_auc  binary     0.595     5 0.00119
   .config
   <chr>
 1 Preprocessor1_Model001
 2 Preprocessor1_Model001
 3 Preprocessor1_Model002
 4 Preprocessor1_Model002
 5 Preprocessor1_Model003
 6 Preprocessor1_Model003
 7 Preprocessor1_Model004
 8 Preprocessor1_Model004
 9 Preprocessor1_Model005
10 Preprocessor1_Model005
# ℹ 490 more rows
```



```
logit_fit |> show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
   penalty mixture .metric .estimator  mean     n std_err .config
     <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.00168     0.75 roc_auc binary     0.595     5 0.00113 Preprocessor1_Model163
2 0.00133     0.75 roc_auc binary     0.595     5 0.00115 Preprocessor1_Model162
3 0.00105     0.75 roc_auc binary     0.595     5 0.00117 Preprocessor1_Model161
4 0.000829    0.75 roc_auc binary     0.595     5 0.00120 Preprocessor1_Model160
5 0.0001      0.75 roc_auc binary     0.595     5 0.00120 Preprocessor1_Model151
```

```
logit_best <- logit_fit |>
  select_best(metric = "roc_auc")

logit_best
```

```
# A tibble: 1 × 3
  penalty mixture .config
    <dbl>   <dbl> <chr>
1 0.00168    0.75 Preprocessor1_Model163
```

```r
logit_final <- logit_wf |>
  finalize_workflow(logit_best)
logit_final
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
6 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_dummy()
• step_zv()
• step_normalize()

── Model ───────────────────────────────────────────────────────────────────────
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = 0.00167683293681101
  mixture = 0.75

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```r
logit_fit_final <- logit_final |>
  last_fit(data_split)
```

Random Forest

```r
rf_rec <- recipe(los_long ~ ., data = mimiciv_icu_cohort_train) |>
  # Mean inputation for normal variables
  step_impute_mean(bicarbonate, chloride, hematocrit, sodium, heart_rate) |>

  # Median inputation for skewed variables
  step_impute_median(creatinine, glucose, potassium, wbc,
                     non_invasive_blood_pressure_diastolic,
                     non_invasive_blood_pressure_systolic,
                     respiratory_rate, temperature_fahrenheit) |>
  # KNN imputation for categorical variables
  step_impute_knn(marital_status) |>

  # zero-variance filter
  step_zv(all_nominal_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 18


── Operations

• Mean imputation for: bicarbonate, chloride, hematocrit, sodium, ...

• Median imputation for: creatinine, glucose, potassium, wbc, ...

• K-nearest neighbor imputation for: marital_status

• Zero variance filter on: all_nominal_predictors()
```

```
rf_mod <- rand_forest(
  mode = "classification",
  mtry = tune(), # number of predictors randomly sampled in each split
  trees = tune() # number of trees in ensemble
) |>
  set_engine("ranger")
rf_mod
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Computational engine: ranger
```

```
rf_wf <- workflow() |>
  add_recipe(rf_rec) |>
  add_model(rf_mod)
rf_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
4 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Computational engine: ranger
```

```
rf_grid <- grid_regular(
  trees(range = c(200L, 800L)),
  mtry(range = c(1L, 8L)),
  levels = c(2, 4)
  )
rf_grid
```

```
# A tibble: 8 × 2
  trees  mtry
  <int> <int>
1   200     1
2   800     1
3   200     3
4   800     3
5   200     5
6   800     5
7   200     8
8   800     8
```

```
set.seed(203)

folds <- vfold_cv(mimiciv_icu_cohort_train, v = 5, strata = los_long)
folds
```

```
#  5-fold cross-validation using stratification
# A tibble: 5 × 2
  splits                id
  <list>                <chr>
```

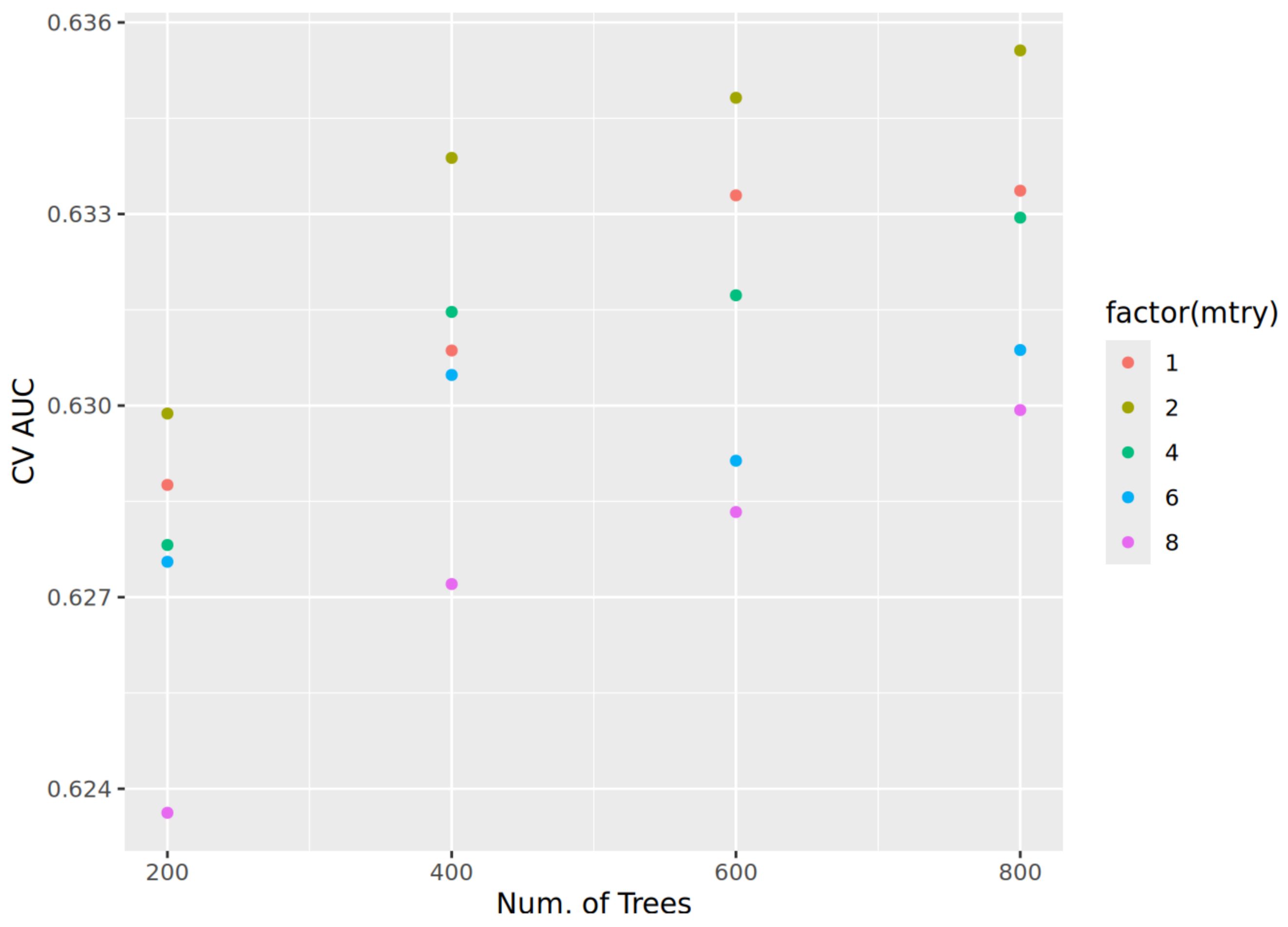
```
1 <split [37776/9445]> Fold1
2 <split [37776/9445]> Fold2
3 <split [37776/9445]> Fold3
4 <split [37778/9443]> Fold4
5 <split [37778/9443]> Fold5
```

```r
rf_fit <- rf_wf |>
  tune_grid(
    resamples = folds,
    grid = rf_grid,
    metrics = metric_set(roc_auc, accuracy)
  )
rf_fit
```

```
# Tuning results
# 5-fold cross-validation using stratification
# A tibble: 5 × 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [40 × 6]> <tibble [0 × 3]>
2 <split [37776/9445]> Fold2 <tibble [40 × 6]> <tibble [0 × 3]>
3 <split [37776/9445]> Fold3 <tibble [40 × 6]> <tibble [0 × 3]>
4 <split [37778/9443]> Fold4 <tibble [40 × 6]> <tibble [0 × 3]>
5 <split [37778/9443]> Fold5 <tibble [40 × 6]> <tibble [0 × 3]>
```

```r
rf_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = trees, y = mean, color = factor(mtry))) +
  geom_point() +
  labs(x = "Num. of Trees", y = "CV AUC")
```

```
# A tibble: 40 × 8
    mtry trees .metric  .estimator  mean     n std_err .config
   <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
 1     1   200 accuracy binary     0.592     5 0.00269 Preprocessor1_Model01
 2     1   200 roc_auc  binary     0.629     5 0.00281 Preprocessor1_Model01
 3     1   400 accuracy binary     0.591     5 0.00225 Preprocessor1_Model02
 4     1   400 roc_auc  binary     0.631     5 0.00207 Preprocessor1_Model02
 5     1   600 accuracy binary     0.595     5 0.00270 Preprocessor1_Model03
 6     1   600 roc_auc  binary     0.633     5 0.00246 Preprocessor1_Model03
 7     1   800 accuracy binary     0.596     5 0.00232 Preprocessor1_Model04
 8     1   800 roc_auc  binary     0.633     5 0.00243 Preprocessor1_Model04
 9     2   200 accuracy binary     0.593     5 0.00274 Preprocessor1_Model05
10     2   200 roc_auc  binary     0.630     5 0.00248 Preprocessor1_Model05
# ℹ 30 more rows
```



```r
rf_fit |> show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
   mtry trees .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2   800 roc_auc binary     0.636     5 0.00220 Preprocessor1_Model08
2     2   600 roc_auc binary     0.635     5 0.00210 Preprocessor1_Model07
3     2   400 roc_auc binary     0.634     5 0.00266 Preprocessor1_Model06
4     1   800 roc_auc binary     0.633     5 0.00243 Preprocessor1_Model04
5     1   600 roc_auc binary     0.633     5 0.00246 Preprocessor1_Model03
```

```r
rf_best <- rf_fit |>
  select_best(metric = "roc_auc")
rf_best
```

```
# A tibble: 1 × 3
   mtry trees .config
  <int> <int> <chr>
1     2   800 Preprocessor1_Model08
```

```r
rf_final <- rf_wf |>
  finalize_workflow(rf_best)
```

```r
rf_final
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
4 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Random Forest Model Specification (classification)

Main Arguments:
  mtry = 2
  trees = 800

Computational engine: ranger
```

```r
rf_fit_final <- rf_final |>
  last_fit(data_split)
rf_fit_final
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits                id              .metrics .notes   .predictions .workflow
  <list>                <chr>           <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

XGBoost

```r
gb_rec <- recipe(los_long ~ ., data = mimiciv_icu_cohort_train) |>
  # Mean inputation for normal variables
  step_impute_mean(bicarbonate, chloride, hematocrit, sodium, heart_rate) |>

  # Median imputation for skewed variables
  step_impute_median(creatinine, glucose, potassium, wbc,
                     non_invasive_blood_pressure_diastolic,
                     non_invasive_blood_pressure_systolic,
                     respiratory_rate, temperature_fahrenheit) |>

  # KNN imputation for categorical variables
  step_impute_knn(marital_status) |>
  # create dummy variable
  step_dummy(all_nominal_predictors()) |>

  # zero-variance filter
  step_zv(all_nominal_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 18


── Operations

• Mean imputation for: bicarbonate, chloride, hematocrit, sodium, ...

• Median imputation for: creatinine, glucose, potassium, wbc, ...

• K-nearest neighbor imputation for: marital_status

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_nominal_predictors()
```

```
gb_mod <- boost_tree(
  mode = "classification",
  trees = 1000,
  tree_depth = tune(),
  learn_rate = tune()
) |>
  set_engine("xgboost")
gb_mod
```

```
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 1000
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```r
gb_wf <- workflow() |>
  add_recipe(gb_rec) |>
  add_model(gb_mod)
gb_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: boost_tree()

── Preprocessor ────────────────────────────────────────────────────────────────
5 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 1000
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```r
gb_grid <- grid_regular(
  tree_depth(range = c(3L, 10L)),
  learn_rate(range = c(0.01, 0.3)),
  levels = c(2, 2)
  )
gb_grid
```

```
# A tibble: 4 × 2
  tree_depth learn_rate
       <int>      <dbl>
1          3       1.02
2         10       1.02
3          3       2.00
4         10       2.00
```

```r
set.seed(203)

folds <- vfold_cv(mimiciv_icu_cohort_train, v = 5, strata = los_long)
folds
```

```
#  5-fold cross-validation using stratification
# A tibble: 5 × 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37776/9445]> Fold2
3 <split [37776/9445]> Fold3
4 <split [37778/9443]> Fold4
5 <split [37778/9443]> Fold5
```

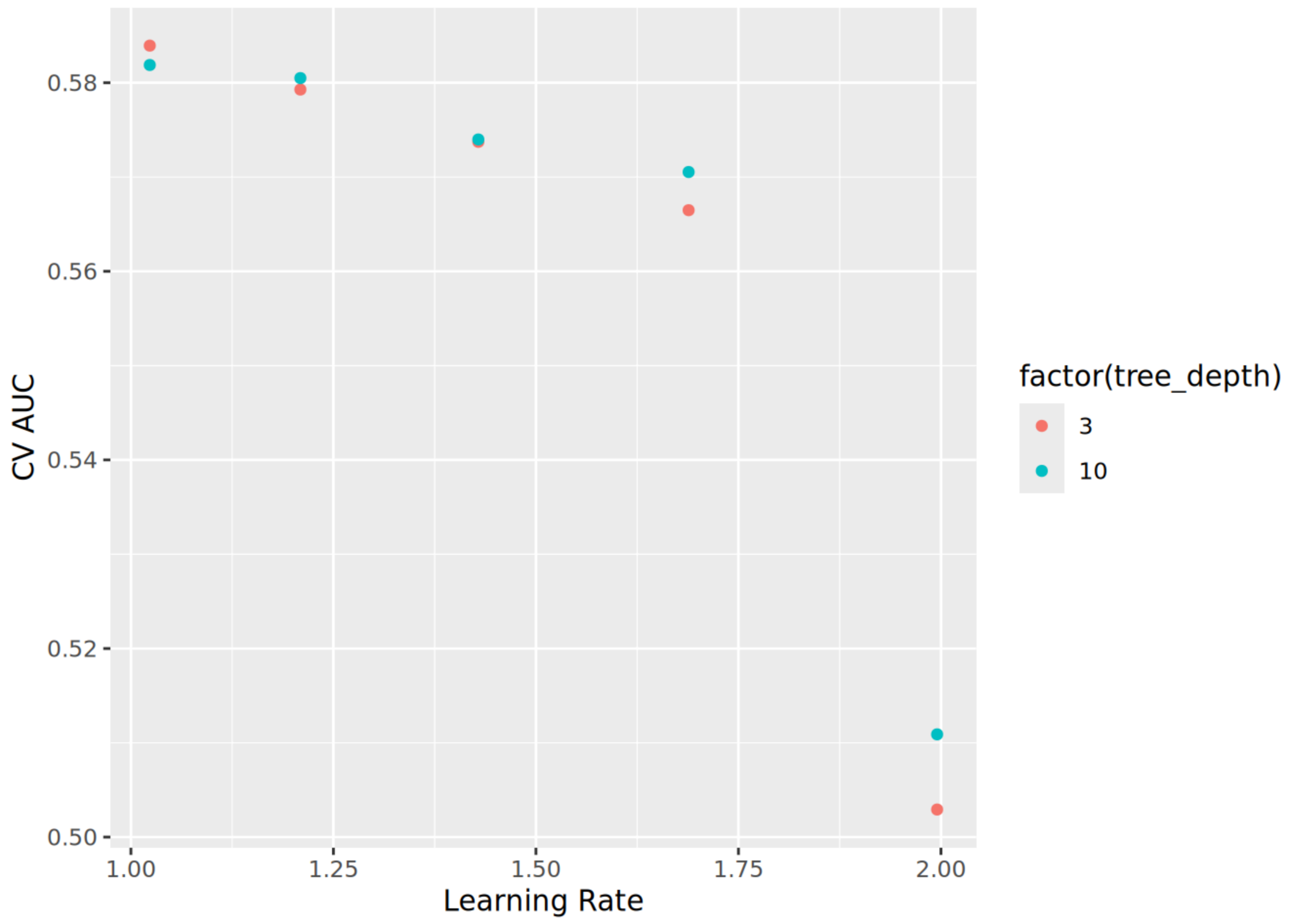
```
gb_fit <- gb_wf |>
  tune_grid(
    resamples = folds,
    grid = gb_grid,
    metrics = metric_set(roc_auc, accuracy)
  )
gb_fit
```

```
# Tuning results
# 5-fold cross-validation using stratification
# A tibble: 5 × 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [20 × 6]> <tibble [0 × 3]>
2 <split [37776/9445]> Fold2 <tibble [20 × 6]> <tibble [0 × 3]>
3 <split [37776/9445]> Fold3 <tibble [20 × 6]> <tibble [0 × 3]>
4 <split [37778/9443]> Fold4 <tibble [20 × 6]> <tibble [0 × 3]>
5 <split [37778/9443]> Fold5 <tibble [20 × 6]> <tibble [0 × 3]>
```

```
gb_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = learn_rate, y = mean, color = factor(tree_depth))) +
  geom_point() +
  labs(x = "Learning Rate", y = "CV AUC")
```

```
# A tibble: 20 × 8
   tree_depth learn_rate .metric  .estimator  mean     n  std_err
        <int>      <dbl> <chr>    <chr>      <dbl> <int>    <dbl>
 1          3       1.02 accuracy binary     0.560     5 0.00207
 2          3       1.02 roc_auc  binary     0.584     5 0.00245
 3         10       1.02 accuracy binary     0.560     5 0.00165
 4         10       1.02 roc_auc  binary     0.582     5 0.00135
 5          3       1.21 accuracy binary     0.559     5 0.00146
 6          3       1.21 roc_auc  binary     0.579     5 0.00180
 7         10       1.21 accuracy binary     0.557     5 0.00105
 8         10       1.21 roc_auc  binary     0.580     5 0.00115
 9          3       1.43 accuracy binary     0.556     5 0.00158
```

```
10          3       1.43 roc_auc  binary     0.574     5 0.00120 
11         10       1.43 accuracy binary     0.550     5 0.00153 
12         10       1.43 roc_auc  binary     0.574     5 0.000296
13          3       1.69 accuracy binary     0.552     5 0.00186 
14          3       1.69 roc_auc  binary     0.566     5 0.00200 
15         10       1.69 accuracy binary     0.549     5 0.00100 
16         10       1.69 roc_auc  binary     0.571     5 0.000592
17          3       2.00 accuracy binary     0.502     5 0.0103  
18          3       2.00 roc_auc  binary     0.503     5 0.00766 
19         10       2.00 accuracy binary     0.512     5 0.00688 
20         10       2.00 roc_auc  binary     0.511     5 0.00669 
   .config              
   <chr>                
 1 Preprocessor1_Model01
 2 Preprocessor1_Model01
 3 Preprocessor1_Model02
 4 Preprocessor1_Model02
 5 Preprocessor1_Model03
 6 Preprocessor1_Model03
 7 Preprocessor1_Model04
 8 Preprocessor1_Model04
 9 Preprocessor1_Model05
10 Preprocessor1_Model05
11 Preprocessor1_Model06
12 Preprocessor1_Model06
13 Preprocessor1_Model07
14 Preprocessor1_Model07
15 Preprocessor1_Model08
16 Preprocessor1_Model08
17 Preprocessor1_Model09
18 Preprocessor1_Model09
19 Preprocessor1_Model10
20 Preprocessor1_Model10
```



```
gb_fit |> show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
  tree_depth learn_rate .metric .estimator  mean     n  std_err .config
       <int>      <dbl> <chr>   <chr>      <dbl> <int>    <dbl> <chr>
1          3       1.02 roc_auc binary     0.584     5 0.00245  Preprocessor1_M…
2         10       1.02 roc_auc binary     0.582     5 0.00135  Preprocessor1_M…
3         10       1.21 roc_auc binary     0.580     5 0.00115  Preprocessor1_M…
4          3       1.21 roc_auc binary     0.579     5 0.00180  Preprocessor1_M…
5         10       1.43 roc_auc binary     0.574     5 0.000296 Preprocessor1_M…
```

```
gb_best <- gb_fit |>
  select_best(metric = "roc_auc")
gb_best
```

```
# A tibble: 1 × 3
  tree_depth learn_rate .config
       <int>      <dbl> <chr>
1          3       1.02 Preprocessor1_Model01
```

```
gb_final <- gb_wf |>
  finalize_workflow(gb_best)
```

```r
gb_final
```

```
══ Workflow ════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: boost_tree()

── Preprocessor ────────────────────────────────────────────────────────────
5 Recipe Steps

• step_impute_mean()
• step_impute_median()
• step_impute_knn()
• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 1000
  tree_depth = 3
  learn_rate = 1.02329299228075

Computational engine: xgboost
```

```r
gb_fit_final <- gb_final |>
  last_fit(data_split)
gb_fit_final
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits                id              .metrics .notes   .predictions .workflow
  <list>                <chr>           <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

Model Stacking

```r
stacks_recipe <- recipe(los_long ~ ., data = mimiciv_icu_cohort_train) |>
  # Mean inputation for normal variables
  step_impute_mean(bicarbonate, chloride, hematocrit, sodium, heart_rate) |>

  # Median imputation for skewed variables
  step_impute_median(creatinine, glucose, potassium, wbc,
                     non_invasive_blood_pressure_diastolic,
                     non_invasive_blood_pressure_systolic,
                     respiratory_rate, temperature_fahrenheit) |>

  # KNN imputation for categorical variables
  step_impute_knn(marital_status) |>

  # create dummy variable
  step_dummy(all_nominal_predictors()) |>

  # zero-variance filter
  step_zv(all_nominal_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 18


── Operations

• Mean imputation for: bicarbonate, chloride, hematocrit, sodium, ...

• Median imputation for: creatinine, glucose, potassium, wbc, ...

• K-nearest neighbor imputation for: marital_status

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_nominal_predictors()
```

```r
stacks_recipe
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 18


── Operations

• Mean imputation for: bicarbonate, chloride, hematocrit, sodium, ...
```

- Median imputation for: creatinine, glucose, potassium, wbc, ...
- K-nearest neighbor imputation for: marital_status
- Dummy variables from: all_nominal_predictors()
- Zero variance filter on: all_nominal_predictors()

```r
set.seed(203)
folds <- vfold_cv(mimiciv_icu_cohort_train, v = 2)
```

```r
logit_mod <- logistic_reg(
  penalty = tune(),
  mixture = tune()
) |>
  set_engine("glmnet", standardize = TRUE)

logit_wf <- workflow() |>
  add_recipe(stacks_recipe) |>
  add_model(logit_mod)

logit_grid <- grid_regular(
  penalty(range = c(-4, 1)),
  mixture(),
  levels = c(10, 3)
  )

logit_res <-
  tune_grid(
    object = logit_wf,
    resamples = folds,
    grid = logit_grid,
    control = control_stack_grid()
  )
```

```
ℹ The workflow being saved contains a recipe, which is 5.84 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```r
logit_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 × 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [90 × 6]> <tibble [0 × 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [90 × 6]> <tibble [0 × 3]> <tibble>
```

```r
rf_mod <- rand_forest(
  mode = "classification",
  mtry = tune(), # number of predictors randomly sampled in each split
  trees = tune() # number of trees in ensemble
) |>
  set_engine("ranger")

rf_wf <- workflow() |>
  add_recipe(stacks_recipe) |>
  add_model(rf_mod)

rf_grid <- grid_regular(
  trees(range = c(200L, 1000L)),
  mtry(range = c(1L, 8L)),
  levels = c(2, 2)
  )

rf_res <- tune_grid(
  object = rf_wf,
  resamples = folds,
  grid = rf_grid,
  control = control_stack_grid()
)
```

```
ℹ The workflow being saved contains a recipe, which is 5.84 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```r
rf_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 × 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [12 × 6]> <tibble [0 × 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [12 × 6]> <tibble [0 × 3]> <tibble>
```

```r
gb_mod <- boost_tree(
  mode = "classification",
  trees = 1000,
  tree_depth = tune(),
  learn_rate = tune()
) |>
  set_engine("xgboost")
gb_mod
```

```
Boosted Tree Model Specification (classification)
```

```
Main Arguments:
  trees = 1000
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```
gb_wf <- workflow() |>
  add_recipe(stacks_recipe) |>
  add_model(gb_mod)

gb_grid <- grid_regular(
  tree_depth(range = c(3L, 10L)),
  learn_rate(range = c(0.01, 0.3)),
  levels = c(1, 2)
  )

gb_res <- tune_grid(
  object = gb_wf,
  resamples = folds,
  grid = gb_grid,
  control = control_stack_grid()
  )
```

```
ℹ The workflow being saved contains a recipe, which is 5.84 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```
gb_res
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 × 5
  splits                id    .metrics         .notes           .predictions
  <list>                <chr> <list>           <list>           <list>
1 <split [23610/23611]> Fold1 <tibble [6 × 6]> <tibble [0 × 3]> <tibble>
2 <split [23611/23610]> Fold2 <tibble [6 × 6]> <tibble [0 × 3]> <tibble>
```

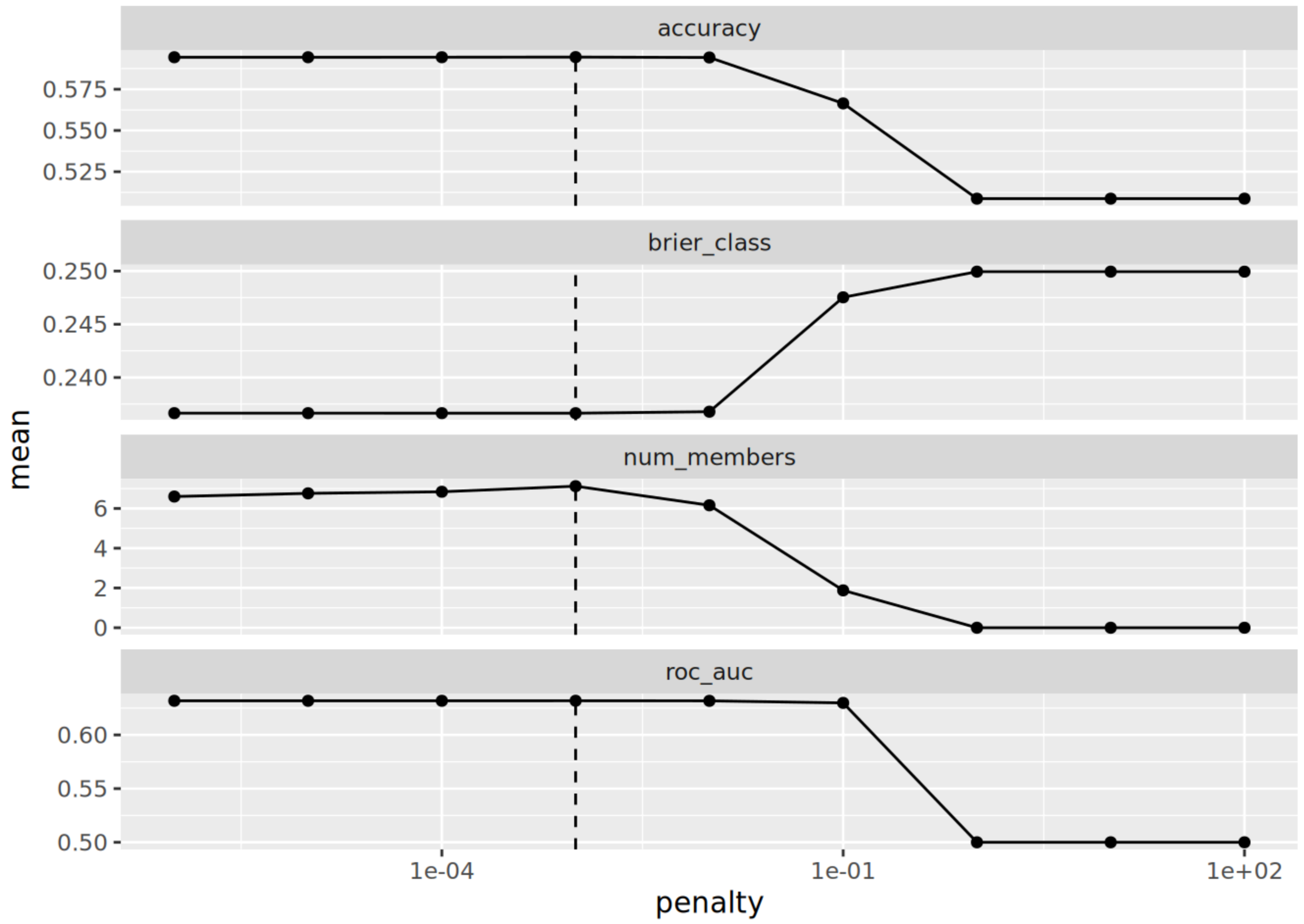
```
model_st <- stacks() |>
  add_candidates(logit_res) |>
  add_candidates(rf_res) |>
  add_candidates(gb_res) |>
  # determine how to combine their predictions
  blend_predictions(
    penalty = 10^(-6:2),
    metrics = c("roc_auc")
  ) |>
  # fit candidates with nonzero stacking coefficients
  fit_members()
```

```
Warning: Predictions from 26 candidates were identical to those from existing candidates
and were removed from the data stack.

Warning: The `...` are not used in this function but one or more arguments were
passed: 'metrics'
```

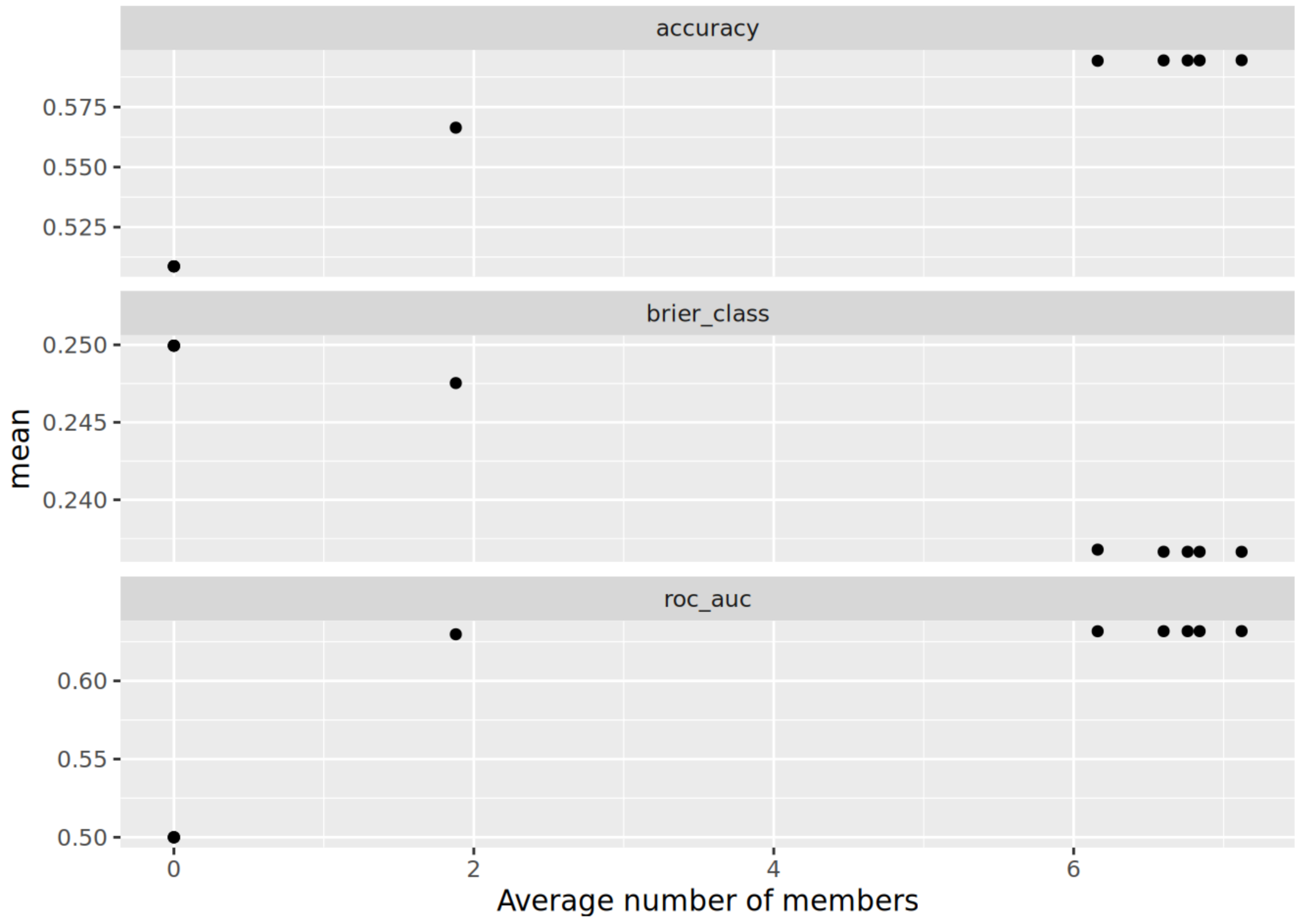
```r
model_st
```

```
── A stacked ensemble model ─────────────────────────────────────


Out of 23 possible candidate members, the ensemble retained 7.

Penalty: 0.001.

Mixture: 1.


The 7 highest weighted member classes are:

# A tibble: 7 × 3
  member                     type           weight
  <chr>                      <chr>           <dbl>
1 .pred_FALSE_rf_res_1_2     rand_forest  5.03
2 .pred_FALSE_rf_res_1_4     rand_forest  2.15
3 .pred_FALSE_logit_res_1_21 logistic_reg 0.376
4 .pred_FALSE_rf_res_1_1     rand_forest  0.326
5 .pred_FALSE_rf_res_1_3     rand_forest  0.317
6 .pred_FALSE_gb_res_1_1     boost_tree   0.119
7 .pred_FALSE_logit_res_1_11 logistic_reg 0.00994
```

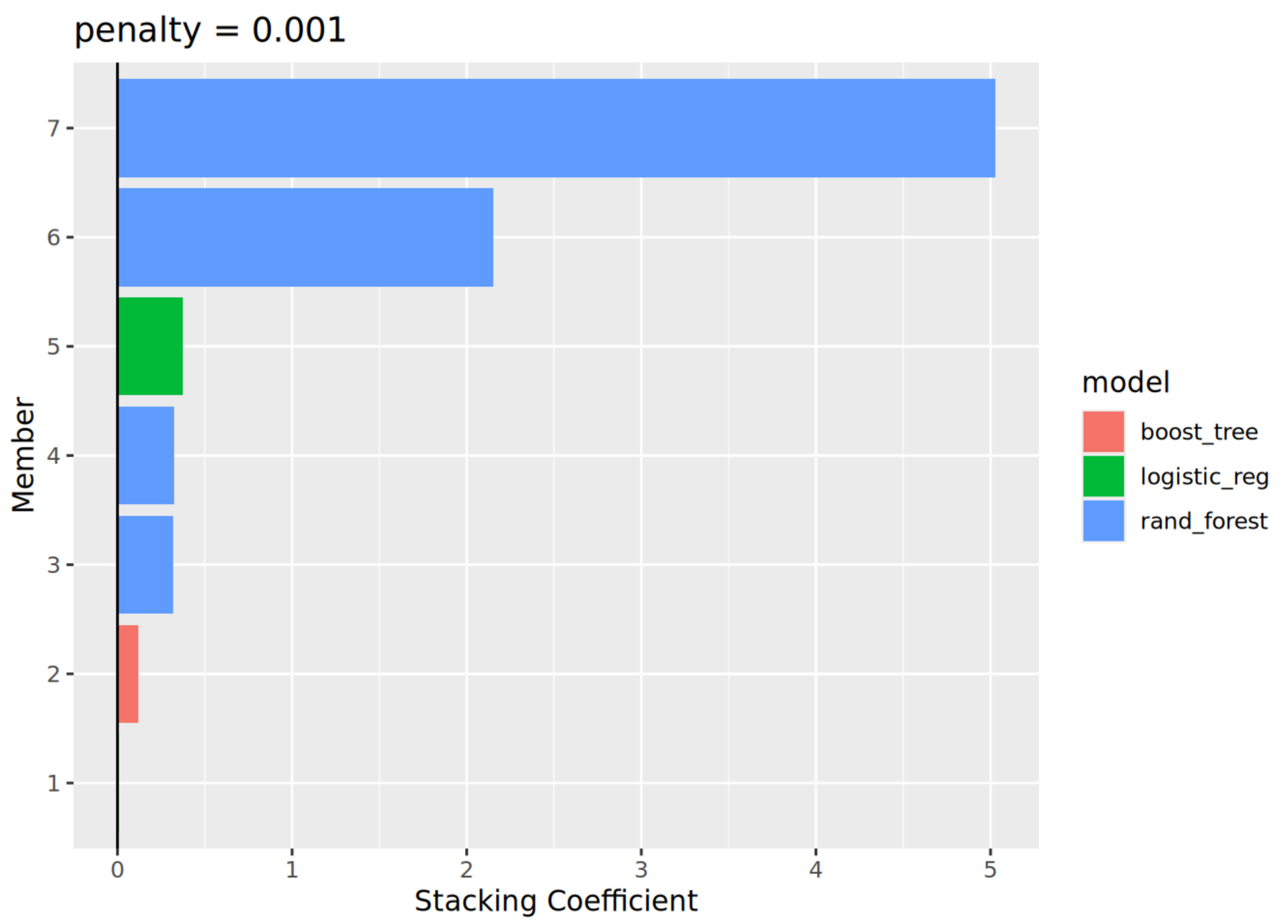
```r
autoplot(model_st)
```



```r
autoplot(model_st, type = "members")
```



```r
autoplot(model_st, type = "weights")
```



```r
collect_parameters(model_st, "rf_res")
```

```
# A tibble: 4 × 5
  member      mtry trees terms                  coef
  <chr>      <int> <int> <chr>                 <dbl>
1 rf_res_1_1     1   200 .pred_FALSE_rf_res_1_1 0.326
2 rf_res_1_2     1  1000 .pred_FALSE_rf_res_1_2 5.03
3 rf_res_1_3     8   200 .pred_FALSE_rf_res_1_3 0.317
4 rf_res_1_4     8  1000 .pred_FALSE_rf_res_1_4 2.15
```

```r
mimic_pred <- mimiciv_icu_cohort_test %>%
  bind_cols(predict(model_st, ., type = "prob")) %>%
  print(width = Inf)
```

```
# A tibble: 47,223 × 21
   los_long gender age_intime marital_status race
   <fct>    <fct>       <int> <fct>          <fct>
 1 FALSE    F              52 WIDOWED        white
 2 FALSE    F              46 MARRIED        white
 3 FALSE    F              57 SINGLE         other
 4 TRUE     M              56 <NA>           other
 5 FALSE    F              83 MARRIED        white
 6 TRUE     F              82 MARRIED        white
 7 TRUE     F              81 WIDOWED        white
 8 TRUE     M              90 WIDOWED        white
 9 TRUE     M              53 SINGLE         white
10 FALSE    F              58 <NA>           white
   first_careunit                                    bicarbonate chloride
   <fct>                                                   <dbl>    <dbl>
 1 Medical Intensive Care Unit (MICU)                         25       95
 2 Medical/Surgical Intensive Care Unit (MICU/SICU)           NA       98
 3 Cardiac Vascular Intensive Care Unit (CVICU)               24      102
 4 Other                                                      18       NA
 5 Medical Intensive Care Unit (MICU)                         26       85
 6 Medical/Surgical Intensive Care Unit (MICU/SICU)           23       98
 7 Medical Intensive Care Unit (MICU)                         27      111
 8 Other                                                      23      102
 9 Other                                                      18      106
10 Cardiac Vascular Intensive Care Unit (CVICU)               NA       NA
   creatinine glucose hematocrit potassium sodium   wbc heart_rate
        <dbl>   <dbl>      <dbl>     <dbl>  <dbl> <dbl>      <dbl>
 1        0.7     102       41.1       6.7    126   6.9       91
 2       NA        NA       NA         4.1    139  NA         86
 3        0.9     288       34.9       3.5    137   7.2       80
 4        3.1      95       34.3       6.5    125  16.8      110.
 5        1.4     133       22.4       5.7    120   9.8      114
 6        2.8     117       25.5       4.9    135  17.9       91
 7        0.6     173       34.7       4.4    144  10.5      106.
 8        1.9     105       29.9       4.4    140   5.1       93.5
 9        0.9     269       43.1       5.3    135  16.9      106
10       NA        NA       NA        NA       NA  NA         80
   non_invasive_blood_pressure_diastolic non_invasive_blood_pressure_systolic
                                   <dbl>                                <dbl>
 1                                    48                                 84
 2                                    56                                 73
 3                                    62                                 98.5
 4                                    80                                112
 5                                    65                                109
 6                                    51                                118
 7                                    51                                102
 8                                    61                                108
 9                                    99                                140
10                                    72                                109
   respiratory_rate temperature_fahrenheit .pred_TRUE .pred_FALSE
              <dbl>                  <dbl>      <dbl>       <dbl>
 1             24                     98.7      0.575       0.425
 2             19                     97.7      0.470       0.530
 3             14                     97.2      0.436       0.564
 4             21                     97.9      0.643       0.357
 5             24                     97.7      0.571       0.429
 6             18                     96.9      0.547       0.453
 7             25                     98.6      0.522       0.478
 8             22.5                   98.1      0.523       0.477
 9              12                         96.7      0.487       0.513
10              17                         99        0.611       0.389
# ℹ 47,213 more rows
```

4. Compare model classification performance on the test set.

Report both the area under ROC curve and accuracy for each machine learning algorithm and the model stacking. Interpret the results. What are the most important features in predicting long ICU stays? How do the models compare in terms of performance and interpretability?

Logistic Regression

```r
logit_fit_final |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.567 Preprocessor1_Model1
2 roc_auc     binary         0.592 Preprocessor1_Model1
3 brier_class binary         0.244 Preprocessor1_Model1
```

The best logistic regression model has accuracy of 0.567 and roc_auc of 0.59. The results shows that 56.7% of ICU stay length are correctly classified by the model. The ROC AUC measures the model's ability to distinguish between classes. The score of 0.59 suggests that the model performs slightly better than random guessing (0.5).

Random Forest

```r
rf_fit_final |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.590 Preprocessor1_Model1
2 roc_auc     binary         0.629 Preprocessor1_Model1
3 brier_class binary         0.237 Preprocessor1_Model1
```

The accuracy of random forest model is 0.58, meaning 58% of models are correctly predicted. The ROC AUC is 0.62, which is means 62% of the time the model distinguish the classes successfully.

XGBoost

```r
gb_fit_final |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.560 Preprocessor1_Model1
2 roc_auc     binary         0.580 Preprocessor1_Model1
3 brier_class binary         0.282 Preprocessor1_Model1
```

The accuracy of XGBoost model is 0.56%, meaning 56% of los_long are correctly predicted. The ROC AUC of 0.58 means that 58% of the time the model distinguish the classes successfully.

Model Stacking

```
yardstick::roc_auc(
  mimic_pred,
  truth = los_long,
  contains(".pred_FALSE")
)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.370
```

The ROC_AUC of model stacking is 0.3687729. which shows that it's not successfully predicted the los_long.

Most important features

```
logit_fit_final |>
  extract_fit_parsnip() |>
  tidy() |>
  arrange(desc(estimate))
```

```
# A tibble: 27 × 3
   term                                                     estimate penalty
   <chr>                                                       <dbl>   <dbl>
 1 first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.S… 0.160    0.00168
 2 hematocrit                                               0.0977   0.00168
 3 non_invasive_blood_pressure_systolic                     0.0974   0.00168
 4 first_careunit_Medical.Intensive.Care.Unit..MICU.        0.0942   0.00168
 5 chloride                                                 0.0915   0.00168
 6 (Intercept)                                              0.0374   0.00168
 7 marital_status_WIDOWED                                   0.0172   0.00168
 8 race_black                                               0.000442 0.00168
 9 bicarbonate                                              0        0.00168
10 glucose                                                  0        0.00168
# ℹ 17 more rows
```

Based on the best logistic model, the most important features with the largest estimates are first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.SICU., hematocrit, non_invasive_blood_pressure_systolic, first_careunit_Medical.Intensive.Care.Unit..MICU., and chloride.

Compare performance and interpretablity

Comparing the accuracy and roc auc of 4 models, the first 3 models, logistic regression, random forest, and XGBoost, have similar performance in accuracy. Random Forest has the highest roc auc among the 3 models. The stacking model performed poorly, which an ROC AUC below 0.5, indicating it performs worse than random guessing. This suggests issues with the model blending or that the individual models may not complement each other well.

Logistic regression is the most interpretable among all 4 models. The coefficients directly show the effect of each feature on the outcome. It's excellent for understanding relationships in data. **Random Forest** provides variable importance, but understanding individual predictions can be difficult. **XGBoost** is even more complex, though feature importance and SHAP values can offer insights into how predictions are made. **Model Stacking** is the hardest to interpret since it's a combination of other models, making it a "black-box" approach.