432 Class 13 Slides

thomaselove.github.io/432

2022-02-22

Today's Agenda

Fitting logistic regressions using `tidymodels` packages

- Pre-processing activities
- Model building (with multiple fitting engines)
- Measuring model effectiveness
- Creating a model workflow

Quiz 1 update

Setup

```
library(here); library(knitr)
library(magrittr); library(janitor)
library(naniar); library(equatiomatic)
library(rstanarm); library(rms)

library(tidymodels)
library(tidyverse)

theme_set(theme_bw())
```

Today's Data (from Classes 10 and 12)

```
fram_raw <- read_csv(here("data/framingham.csv")) %>%
    type.convert(as.is = FALSE) %>%
    clean_names()
```

The variables describe n = 4238 adults examined at baseline, then followed for 10 years to see if they developed incident coronary heart disease. Our outcome (below) has no missing values.

```
fram_raw %>% tabyl(ten_year_chd)
```

```
 ten_year_chd    n   percent
            0 3594 0.8480415
            1  644 0.1519585
```

Data Cleanup

```
fram_new <- fram_raw %>%
    rename(cigs = "cigs_per_day",
           stroke = "prevalent_stroke",
           hrate = "heart_rate",
           sbp = "sys_bp",
           chd10_n = "ten_year_chd") %>%
    mutate(educ = fct_recode(factor(education),
                      "Some HS" = "1",
                      "HS grad" = "2",
                      "Some Coll" = "3",
                      "Coll grad" = "4")) %>%
    mutate(chd10_f = fct_recode(factor(chd10_n),
                      "chd" = "1", "chd_no" = "0")) %>%
    select(subj_id, chd10_n, chd10_f, age,
           cigs, educ, hrate, sbp, stroke)
```

Data Descriptions (Main Variables Today)

The variables we'll use today are:

Variable	Description
`subj_id`	identifying code added by Dr. Love
`chd10_n`	(numeric) 1 = coronary heart disease in next 10 years
`chd10_f`	(factor) "chd" or "chd_no" in next ten years
`age`	in years (range is 32 to 70)
`cigs`	number of cigarettes smoked per day
`educ`	4-level factor: educational attainment
`hrate`	heart rate in beats per minute
`sbp`	systolic blood pressure in mm Hg
`stroke`	1 = history of stroke, else 0

Steps we'll describe today

1. Prepare our (binary) outcome.
2. Split the data into training and testing samples.
3. Build a recipe for our model.
    - Specify roles for outcome and predictors.
    - Deal with missing data in a reasonable way.
    - Complete all necessary pre-processing so we can fit models.
4. Specify a modeling engine for each fit we will create.
    - There are five available engines just for linear regression!
5. Create a workflow for each engine and fit model to the training data.
6. Compare coefficients graphically from two modeling approaches.
7. Assess performance in the models we create in the training data.
8. Compare multiple models based on their performance in test data.

Key Reference: Kuhn and Silge, *Tidy Modeling with R* or TMWR

Stage 1. Prepare our outcome.

To do logistic regression using `tidymodels`, we'll want our binary outcome to be a factor variable.

```
fram_new %$% str(chd10_f)
```

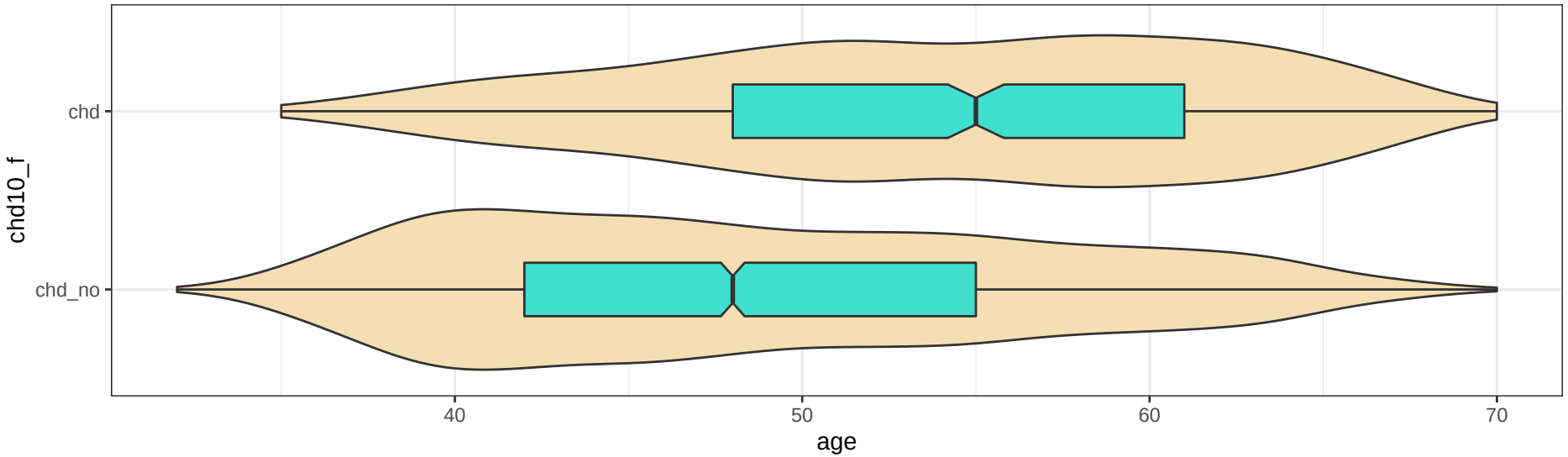
```
 Factor w/ 2 levels "chd_no","chd": 1 1 1 2 1 1 2 1 1 1 ...
```

```
fram_new %>% tabyl(chd10_f, chd10_n)
```

```
 chd10_f    0   1
  chd_no 3594   0
     chd    0 644
```

Working with Binary Outcome Models

Does Pr(CHD in next ten years) look higher for *older* or *younger* people?



chd10_f	n	mean(age)	sd(age)	median(age)
chd_no	3594	48.77	8.41	48
chd	644	54.15	8.01	55

So what do we expect in this model?

Pr(CHD in next ten years) looks higher for *older* people?

If we predict log(odds(CHD in next ten years)), we want to ensure that value will be **rising** with increased age.

So, for the `mage_1` model below, what sign do we expect for the slope of age?

```
mage_1 <- glm(chd10_f ~ age, family = binomial,
              data = fram_new)
```

Results for mage_1

```
tidy(mage_1) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-5.558	0.284	-19.585	0
age	0.075	0.005	14.166	0

```
tidy(mage_1, exponentiate = TRUE) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.004	0.284	-19.585	0
age	1.077	0.005	14.166	0

Stage 2. Split the data into training/test samples.

```
set.seed(2022432)

fram_splits <-
    initial_split(fram_new, prop = 3/4, strata = chd10_f)

fram_train <- training(fram_splits)
fram_test <- testing(fram_splits)
```

Did the stratification work?

```
fram_train %>% tabyl(chd10_f)
```

```
 chd10_f    n   percent
  chd_no 2695 0.8480176
     chd  483 0.1519824
```

```
fram_test %>% tabyl(chd10_f)
```

```
 chd10_f   n   percent
  chd_no 899 0.8481132
     chd 161 0.1518868
```

Stage 3. Build a recipe for our model.

```
fram_rec <-
    recipe(chd10_f ~ age + cigs + educ + hrate +
                sbp + stroke, data = fram_new) %>%
    step_impute_bag(all_predictors()) %>%
    step_dummy(all_nominal(), -all_outcomes()) %>%
    step_normalize(all_predictors())
```

1. Specify the roles for the outcome and the predictors.
2. Use bagged trees to impute missing values in predictors.
3. Form dummy variables to represent all categorical variables.
    - Forgetting the `-all_outcomes()` wasted a half hour of my life, so learn from my mistake.
4. Normalize (subtract mean and divide by SD) all quantitative predictors.

Stage 4. Specify engines for our fit(s).

```
fram_glm_model <-
    logistic_reg() %>%
    set_engine("glm")

prior_dist <- rstanarm::normal(0, 3)

fram_stan_model <- logistic_reg() %>%
    set_engine("stan",
               prior_intercept = prior_dist,
               prior = prior_dist)
```

Working with `rstanarm`

- I recommend How To Use the `rstanarm` Package at http://mc-stan.org/rstanarm/articles/rstanarm.html
- `rstanarm` models have default prior distributions for their parameters. These are discussed at http://mc-stan.org/rstanarm/articles/priors.html

In general, the default priors are *weakly informative* rather than flat. They are designed to help stabilize computation.

Stage 5. Create a workflow and fit model(s).

```
fram_glm_wf <- workflow() %>%
    add_model(fram_glm_model) %>%
    add_recipe(fram_rec)

fram_stan_wf <- workflow() %>%
    add_model(fram_stan_model) %>%
    add_recipe(fram_rec)
```

Ready to fit the models?

Fit the glm and stan models

```
fit_A <- fit(fram_glm_wf, fram_train)

set.seed(432)
fit_B <- fit(fram_stan_wf, fram_train)
```

Produce tidied coefficients (log odds scale)

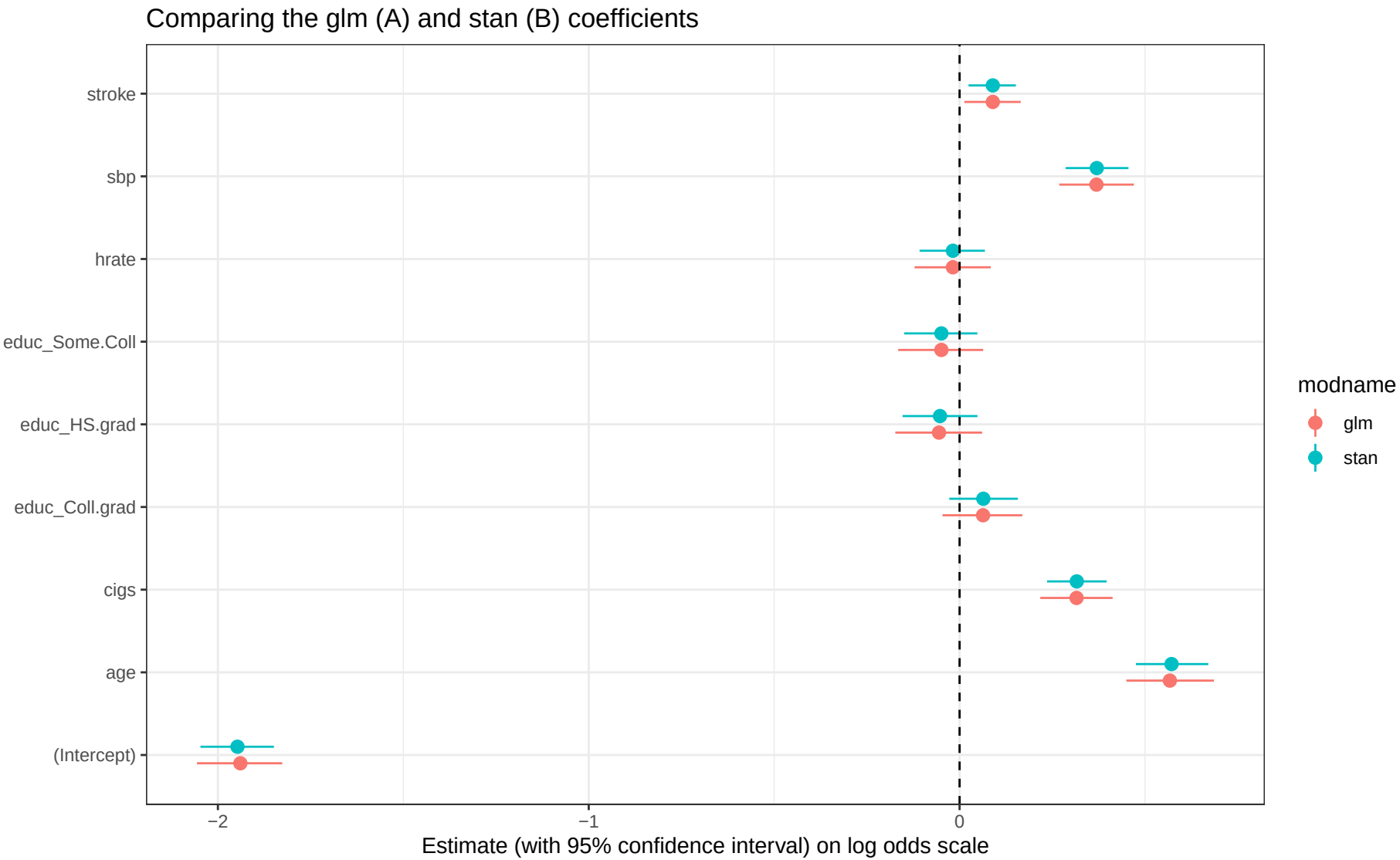
```
A_tidy <- tidy(fit_A, conf.int = T) %>%
    mutate(modname = "glm")

B_tidy <- broom.mixed::tidy(fit_B, conf.int = T) %>%
    mutate(modname = "stan")

coefs_comp <- bind_rows(A_tidy, B_tidy)
```

That's set us up for some plotting.

Stage 6. Compare coefficients of the fits.



Can we compare coefficients as odds ratios?

```
A_odds <- A_tidy %>%
    mutate(odds = exp(estimate),
           odds_low = exp(conf.low),
           odds_high = exp(conf.high)) %>%
    filter(term != "(Intercept)") %>%
    select(modname, term, odds, odds_low, odds_high)

head(A_odds, 2)
```

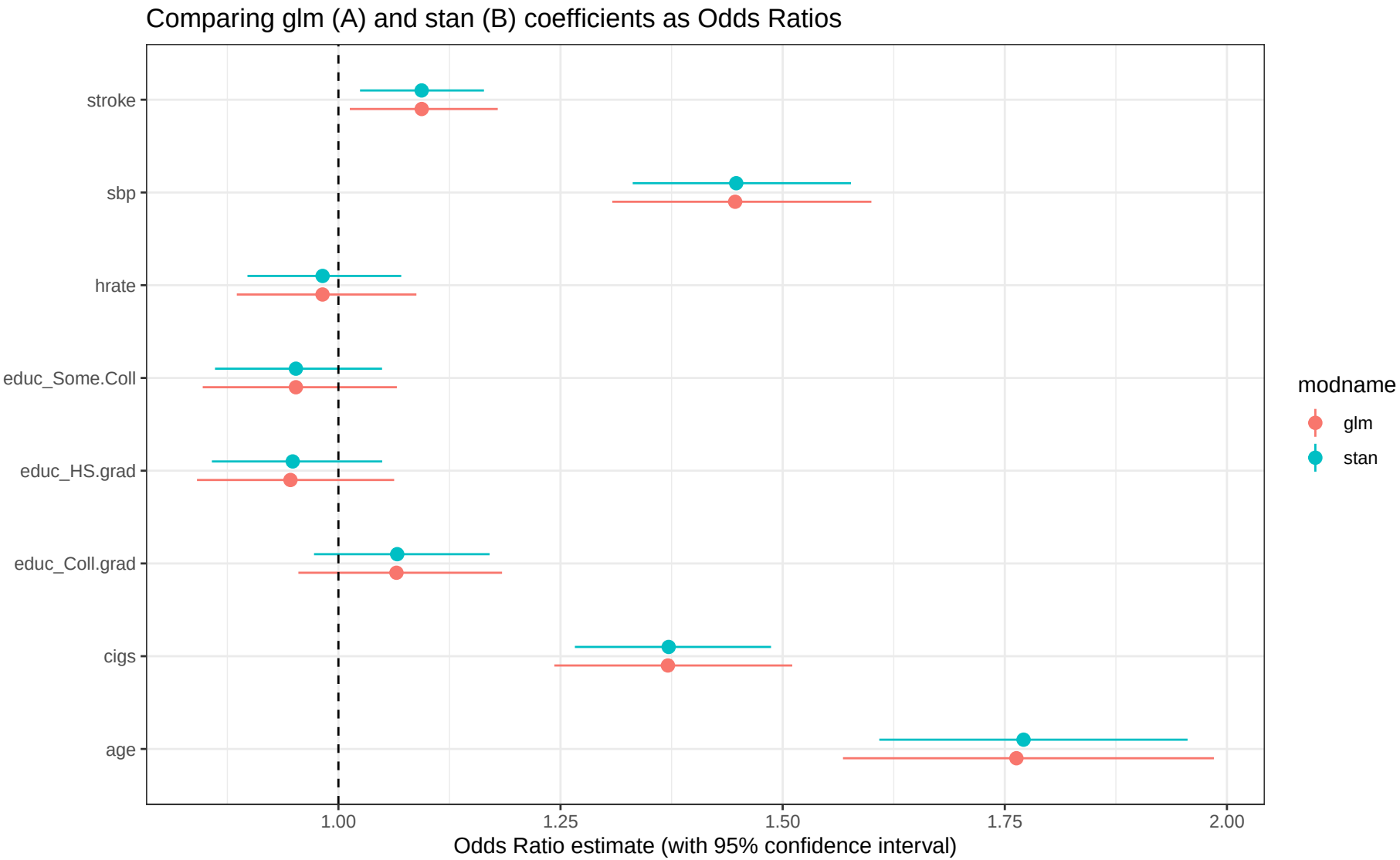
```
# A tibble: 2 x 5
  modname term   odds odds_low odds_high
  <chr>   <chr> <dbl>    <dbl>     <dbl>
1 glm     age    1.76     1.57      1.99
2 glm     cigs   1.37     1.24      1.51
```

Then repeat to create B_odds (see next slide)

Creating B_odds

```
B_odds <- B_tidy %>%
    mutate(odds = exp(estimate),
           odds_low = exp(conf.low),
           odds_high = exp(conf.high)) %>%
    filter(term != "(Intercept)") %>%
    select(modname, term, odds, odds_low, odds_high)
```

Combined Results Plotted on Odds Ratio scale



Stage 7. Assess training sample performance.

1. We'll make predictions for the training sample using each model, and use them to find the C statistic and plot the ROC curve.
2. We'll show some other summaries of performance in the training sample.

Make Predictions with `fit_A`

We'll start by using the `glm` model `fit_A` to make predictions.

```
glm_probs <-
    predict(fit_A, fram_train, type = "prob") %>%
    bind_cols(fram_train %>% select(chd10_f))

head(glm_probs, 4)
```

```
# A tibble: 4 x 3
  .pred_chd_no .pred_chd chd10_f
         <dbl>     <dbl> <fct>
1        0.759    0.241  chd
2        0.917    0.0826 chd
3        0.911    0.0892 chd
4        0.889    0.111  chd
```

Obtain C statistic for `fit_A`

Next, we'll use `roc_auc` from `yardstick`. This assumes that the first level of `chd10_f` is the thing we're trying to predict. Is that true in our case?

```
fram_train %>% tabyl(chd10_f)
```

```
 chd10_f    n   percent
  chd_no 2695 0.8480176
     chd  483 0.1519824
```

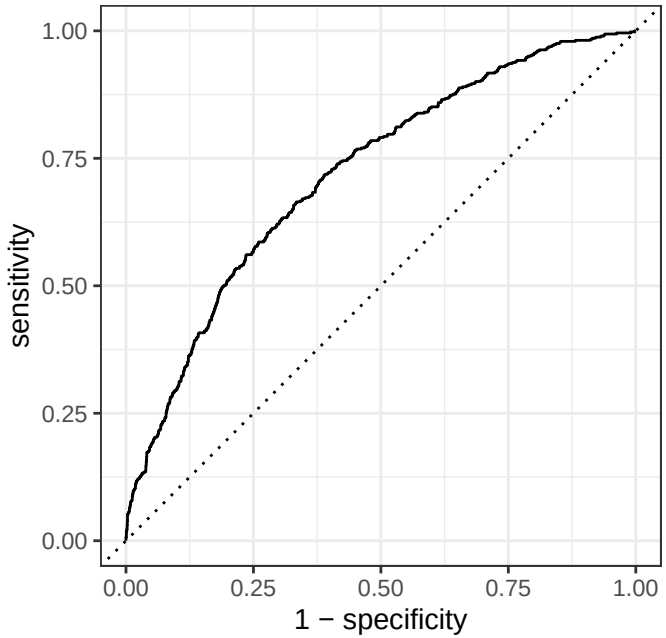
No. We want to predict the second level: `chd`. So we need to switch the `event_level` to "second", like this.

```
glm_probs %>% roc_auc(chd10_f, .pred_chd,
                      event_level = "second") %>%
    kable(dig = 5)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7186

Can we plot the ROC curve for `fit_A`?

```
glm_roc <- glm_probs %>%
    roc_curve(chd10_f, .pred_chd, event_level = "second")
autoplot(glm_roc)
```



- We saw on the prior slide that our C statistic for the `glm` fit is 0.719.

Make Predictions with `fit_B`

We'll use the `stan` model `fit_B` to make predictions.

```
stan_probs <-
    predict(fit_B, fram_train, type = "prob") %>%
    bind_cols(fram_train %>% select(chd10_f))
```

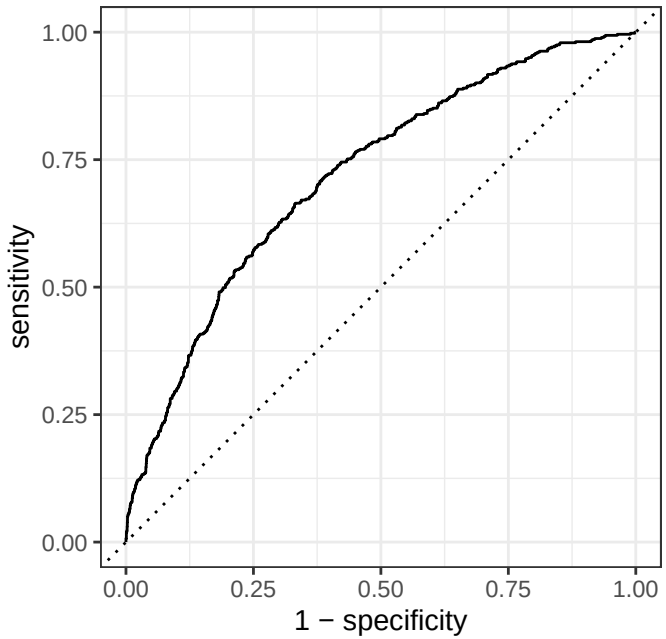
Now, we'll obtain the C statistic for `fit_B`

```
stan_probs %>%
    roc_auc(chd10_f, .pred_chd,
                     event_level = "second") %>%
    kable(dig = 5)
```

.metric	.estimator	.estimate
roc_auc	binary	0.71861

Plotting the ROC curve for `fit_B`?

```
stan_roc <- stan_probs %>%
    roc_curve(chd10_f, .pred_chd, event_level = "second")
autoplot(stan_roc)
```



- Our C statistic for the `stan` fit is also 0.719.

Other available summaries from `yardstick`

For a logistic regression where we're willing to specify a decision rule, we can consider:

- `Conf_mat` which produces a confusion matrix if we specify a decision rule.
  - There is a way to tidy a confusion matrix, summarize it with `summary()` and autoplot it with either a mosaic or a heatmap.
- `accuracy` = proportion of the data that are predicted correctly
- `kap` is very similar to `accuracy` but is normalized by the accuracy that would be expected by chance alone and is most useful when one or more classes dominate the distribution - attributed to Cohen (1960)
- `sens` = sensitivity and `spec` specificity
- `ppv` positive predictive value and `npv` negative predictive value

Establishing a decision rule for the glm fit

Let's use .pred_chd > 0.2 for now to indicate a prediction of chd.

```
glm_probs <-
    predict(fit_A, fram_train, type = "prob") %>%
    bind_cols(fram_train %>% select(chd10_f)) %>%
    mutate(chd10_pre =
                ifelse(.pred_chd > 0.2, "chd", "chd_no")) %>%
    mutate(chd10_pre = fct_relevel(factor(chd10_pre),
                                   "chd_no"))

glm_probs %>% tabyl(chd10_pre, chd10_f)
```

```
 chd10_pre chd_no chd
    chd_no   2144 234
       chd    551 249
```

Why didn't I use .pred_chd > 0.5?

```
glm_probs5 <-
    predict(fit_A, fram_train, type = "prob") %>%
    bind_cols(fram_train %>% select(chd10_f)) %>%
    mutate(chd10_pre =
                ifelse(.pred_chd > 0.5, "chd", "chd_no")) %>%
    mutate(chd10_pre = fct_relevel(factor(chd10_pre),
                                   "chd_no"))

glm_probs5 %>% tabyl(chd10_pre)
```

```
 chd10_pre    n    percent
    chd_no 3138 0.98741347
       chd   40 0.01258653
```

What can we run now?

```
conf_mat(glm_probs, truth = chd10_f, estimate = chd10_pre)
```

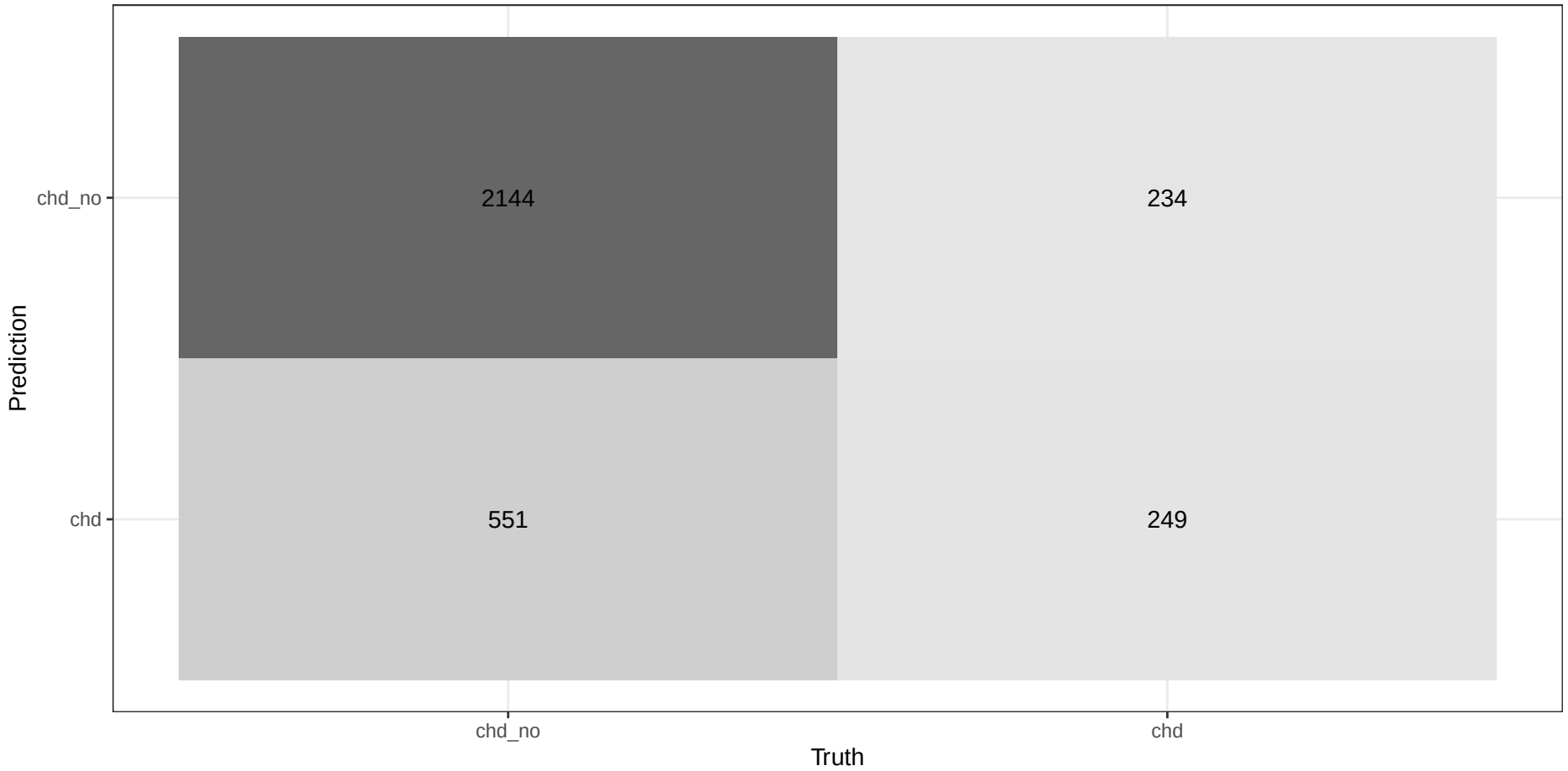
```
          Truth
Prediction chd_no  chd
    chd_no   2144  234
    chd       551  249
```

```
metrics(glm_probs, truth = chd10_f, estimate = chd10_pre)
```

```
# A tibble: 2 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.753
2 kap      binary         0.245
```

Plot a confusion matrix for the `glm` fit?

```
conf_mat(glm_probs,
         truth = chd10_f, estimate = chd10_pre) %>%
    autoplot(type = "heatmap")
```



More Confusion Matrix Summaries?

Other available metrics include:

- sensitivity, specificity, positive predictive value, negative predictive value, and the statistics below.

```
conf_mat(glm_probs, truth = chd10_f, estimate = chd10_pre) %>%
    summary() %>% slice(7:13)
```

```
# A tibble: 7 x 3
  .metric              .estimator .estimate
  <chr>                <chr>          <dbl>
1 mcc                  binary         0.257
2 j_index              binary         0.311
3 bal_accuracy         binary         0.656
4 detection_prevalence binary         0.748
5 precision            binary         0.902
6 recall               binary         0.796
7 f_meas               binary         0.845
```

Establishing a decision rule for the stan fit

Let's also use .pred_chd > 0.2 to indicate a prediction of chd.

```
stan_probs <-
    predict(fit_B, fram_train, type = "prob") %>%
    bind_cols(fram_train %>% select(chd10_f)) %>%
    mutate(chd10_pre =
               ifelse(.pred_chd > 0.2, "chd", "chd_no")) %>%
    mutate(chd10_pre = fct_relevel(factor(chd10_pre),
                                   "chd_no"))
```

Confusion Matrix and Basic Metrics

```
conf_mat(stan_probs, truth = chd10_f, estimate = chd10_pre)
```

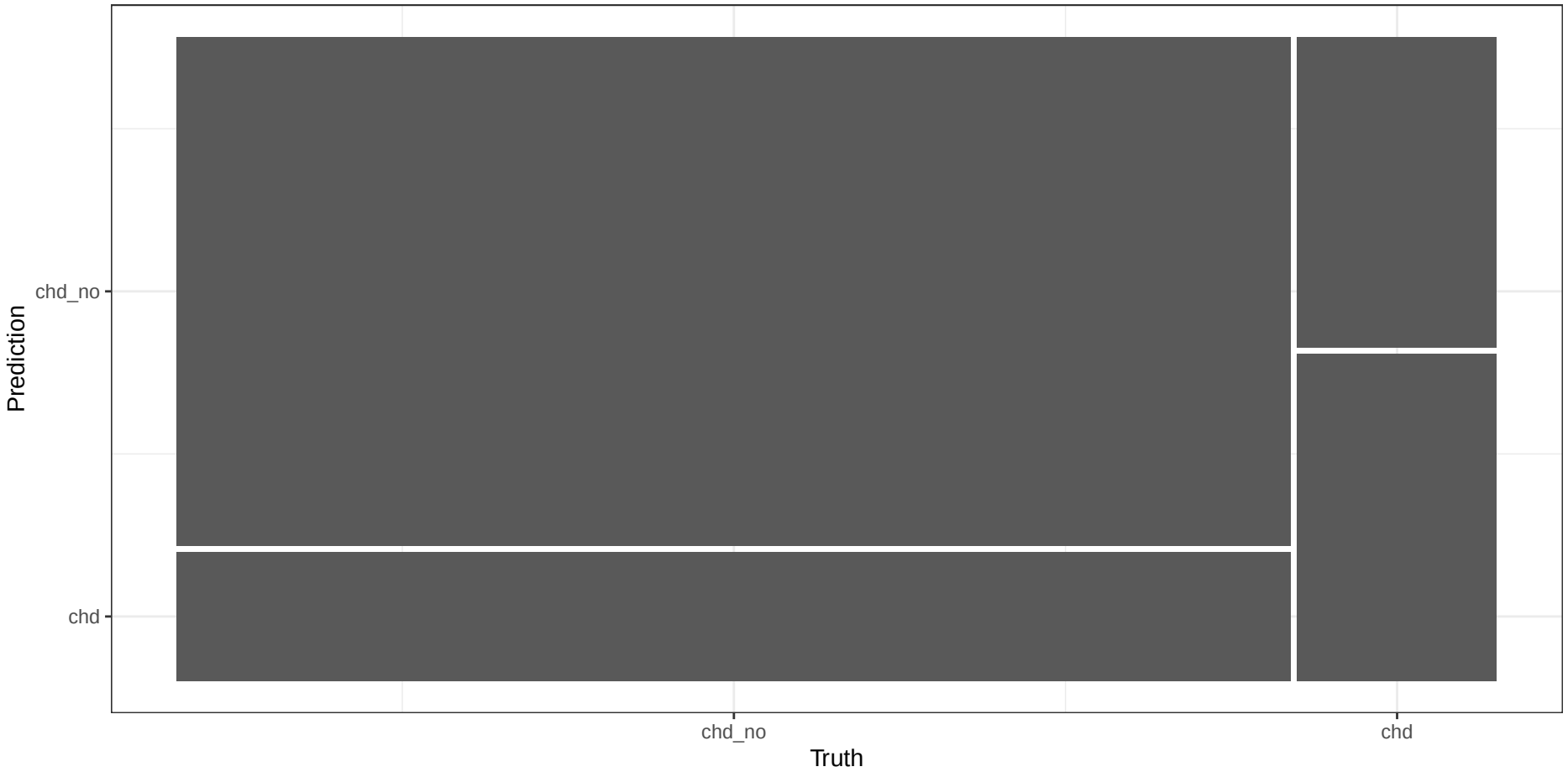
```
          Truth
Prediction chd_no  chd
    chd_no   2150  235
    chd       545  248
```

```
metrics(stan_probs, truth = chd10_f, estimate = chd10_pre)
```

```
# A tibble: 2 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.755
2 kap      binary         0.246
```

Plot a confusion matrix?

```
conf_mat(stan_probs,
         truth = chd10_f, estimate = chd10_pre) %>%
    autoplot(type = "mosaic")
```



More Confusion Matrix Summaries?

```
conf_mat(stan_probs,
         truth = chd10_f, estimate = chd10_pre) %>%
    summary()
```

```
# A tibble: 13 x 3
   .metric              .estimator .estimate
   <chr>                <chr>          <dbl>
 1 accuracy             binary         0.755
 2 kap                  binary         0.246
 3 sens                 binary         0.798
 4 spec                 binary         0.513
 5 ppv                  binary         0.901
 6 npv                  binary         0.313
 7 mcc                  binary         0.258
 8 j_index              binary         0.311
 9 bal_accuracy         binary         0.656
10 detection_prevalence binary         0.750
```

Stage 8. Assess test sample performance.

```
glm_test <-
    predict(fit_A, fram_test, type = "prob") %>%
    bind_cols(fram_test %>% select(chd10_f))

stan_test <-
    predict(fit_B, fram_test, type = "prob") %>%
    bind_cols(fram_test %>% select(chd10_f))
```

Test Sample C statistic comparison?

```
glm_test %>% roc_auc(chd10_f, .pred_chd,
                     event_level = "second") %>%
    kable(dig = 4)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7231

```
stan_test %>% roc_auc(chd10_f, .pred_chd,
                      event_level = "second") %>%
    kable(dig = 4)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7229

No class Thursday.

- Next Tuesday, we'll discuss Regression on Count Outcomes