STAT 471: Midterm Exam Extra Credit

Ashley Clarke

Due: November 6, 2021 at 11:59pm; Time limit: 24 hours

Contents

Instructions

Download the `Rmd` file `midterm-exam-extra-credit.Rmd` from this link, and place it under `stat-471-fall-2021/midterm/midterm-fall-2021/`. Set the latter directory as your working directory. Note that the link above also contains the file `irf_data_tidy.tsv`, which is the correct output of the first section of the exam. You may use this to complete the remaining sections of the exam if you are unable to complete the first section of the exam.

Use this document as a starting point for your writeup, adding your solutions after "**Solution**". Add your R code using code chunks and add your text answers using **bold text**. Compile your writeup to PDF and submit to Gradescope.

You must complete this exam individually, but you may consult any course materials or the internet.

We'll need to use the following `R` packages and functions:

```
library(kableExtra)                    # for printing tables
library(cowplot)                       # for side by side plots
library(glmnetUtils)                   # to run ridge and lasso
source("../../functions/plot_glmnet.R") # for lasso/ridge trace plots
library(tidyverse)                     # for everything else
```

Medicare costs at inpatient rehabilitation facilities

According to medicare.gov, an inpatient rehabilitation facility (IRF) is "a hospital, or part of a hospital, that provides intensive rehabilitation to inpatients. Many patients with conditions like stroke or brain injury are transferred or admitted to an inpatient rehabilitation facility." IRFs can incur significant medical costs, as quantified by the Medicare spending per beneficiary (MSPB) measure. In this exam, we will be building predictive models for the MSPB based on several attributes of IRFs. We will use a dataset from The Centers for Medicare & Medicaid Services, which are available for download at the URL defined below. The variables of interest are coded in these data using "measure codes". For the purposes of this exam, we are interested in 15 of the measure codes, presented in Table 1 along with shortened variable names and descriptions. The first variable in Table 1 (MSPB) is the response and the remaining variables are features.

1 Wrangling (40 points for correctness; 0 points for presentation)

1.1 Import (8 points)

- (4 points) Import the IRF data directly from the URL below into a tibble called `irf_data_raw`. Print this tibble (no need to make a fancy table out of it).

```
# data URL
website = "https://data.cms.gov/provider-data/sites/default/files/resources"
resource_id = "6fe279310f5926cab64ad5b27b1da26a_1632923525"
filename = "Inpatient_Rehabilitation_Facility-Provider_Dec2020.csv"
url = paste(website, resource_id, filename, sep = "/")
#read IRF data from URL
irf_data_raw = read_csv(url)
print(irf_data_raw)
```

```
## # A tibble: 76,180 x 15
##    `CMS Certificatio~ `Facility Name`    `Address Line 1` `Address Line 2` City
##    <chr>              <chr>              <chr>            <chr>            <chr>
##  1 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  2 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  3 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  4 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  5 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  6 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  7 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  8 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
##  9 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
## 10 013025             ENCOMPASS HEALTH ~ 3800 RIDGEWAY D~ <NA>             BIRM~
## # ... with 76,170 more rows, and 10 more variables: State <chr>,
## #   Zip Code <chr>, County Name <chr>, PhoneNumber <chr>, CMS Region <chr>,
## #   Measure Code <chr>, Score <chr>, Footnote <dbl>, Start Date <chr>,
## #   End Date <chr>
```

Table 1: Features and response variable of interest in the inpatient rehabilitation facility data.

Measure code	Variable name	Variable description
I_020_01_MSPB_SCORE	mspb	Medicare Spending Per Beneficiary (MSPB)
I_006_01_SIR	uti_rate	Catheter-associated urinary tract infection rate
I_008_02_OBS_RATE	abilities_goals	Percentage of patients whose functional abilities were assessed and functional goals were included in their treatment plan
I_009_03_ADJ_CHG_SFCR_SCORE	self_care_ability_improved	Patients' ability to care for themselves changed between facility admission and discharge
I_010_03_ADJ_CHG_MOBL_SCORE	mobility_improved	Patients' ability to move around changed between facility admission and discharge
I_011_03_OBS_RATE	self_care_expectation_met	Percentage of patients who achieve or exceed a self-care ability expected for their condition at discharge
I_012_03_OBS_RATE	mobility_expectation_met	Percentage of patients who achieve or exceed the level of movement expected for their condition at discharge
I_013_01_OBS_RATE	fall_rate	Percentage of IRF patients who experience one or more falls with major injury during their IRF stay
I_015_01_SIR	cdi_rate	Clostridium difficile infection rate
I_016_01_OBS_RATE	flu_vax_rate	Influenza Vaccination Coverage Among Healthcare Personnel
I_017_01_PPR_PD_RSRR	readmission_rate_after	Rate of potentially preventable hospital readmissions 30 days after discharge from an IRF
I_018_01_PPR_WI_RSRR	readmission_rate_during	Rate of potentially preventable hospital readmissions during the IRF stay
I_019_02_DTC_RS_RATE	rate_return	Rate of successful return to home and community from an IRF
I_021_01_OBS_RATE	medications_reviewed	Percentage of patients whose medications were reviewed and who received follow-up care when medication issues were identified
I_022_01_ADJ_RATE	pressure_ulcers	Percentage of patients with pressure ulcers/injuries that are new or worsened

Number IRFs	Number Measure Codes
1172	65

- (4 points) How many IRFs are represented in these data? Note that each IRF is coded using the `CMS Certification Number (CCN)`. How many measure codes are represented in these data?

```
num_irf = irf_data_raw %>%
  group_by(`CMS Certification Number (CCN)`) %>%
  #create column equal to one for every CCN
  summarise(unique_code_indictor = 1) %>%
  #calc number of unique codes
  summarise('Number IRFs' = sum(unique_code_indictor))

num_measure_code = irf_data_raw %>%
  group_by(`Measure Code`) %>%
  #create column equal to one for every measure
  summarise(unique_measure_indictor = 1) %>%
  #calc number of unique measures
  summarise('Number Measure Codes' = sum(unique_measure_indictor))

tibble(num_irf, num_measure_code) %>%
  kable(format = "latex", row.names = NA,
        booktabs = TRUE,
        digits = 4) %>%
 kable_styling(position = "center")
```

There are 1172 IRFs and 65 measure codes

1.2 Clean (24 points)

- (4 points) Create a new tibble called `irf_data_1` from `irf_data_raw` that contains only the columns `CMS Certification Number (CCN)`, `Facility Name`, `City`, `State`, `Measure Code`, `Score`, renamed to `facility_id`, `facility_name`, `city`, `state`, `measure_code`, `score`, respectively. Print this tibble (no need to create a fancy table).

```
irf_data_1 = irf_data_raw %>%
  #select and rename desired columns
  summarise('facility_id' = `CMS Certification Number (CCN)`, 'facility_name' = `Facility Name`,
            city = City, state = State, 'measure_code' = `Measure Code`, score = Score)
print(irf_data_1) #print
```

```
## # A tibble: 76,180 x 6
##    facility_id facility_name           city   state measure_code   score
##    <chr>       <chr>                   <chr>  <chr> <chr>          <chr>
##  1 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_CI_L~ 0.368
##  2 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_CI_U~ 3.935
##  3 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_COMP~ No Different~
##  4 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_DOPC~ 1,920
##  5 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_ELIG~ 2.075
##  6 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_NUME~ 3
##  7 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_006_01_SIR   1.446
##  8 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_008_02_DENO~ 2,253
##  9 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_008_02_NUME~ 2,246
## 10 013025      ENCOMPASS HEALTH LAKES~ BIRMI~ AL    I_008_02_OBS_~ 99.7
```

Table 2: Fraction of IRFs with Missing Scores

Fraction of IRFS with Missing Scores
0.712

```
## # ... with 76,170 more rows
```

- (4 points) The column `measure_code` contains the measure codes. Create a new tibble called `irf_data_2` from `irf_data_1` that contains only the rows representing measure codes in Table 1. Print this tibble (no need to create a fancy table). [Hint: Use the tibble `variables` underlying Table 1.]

```
irf_data_2 = irf_data_1 %>%
  # only select measure codes in variables table
  filter(measure_code %in% variables$measure_code)
print(irf_data_2) #print
```

```
## # A tibble: 17,580 x 6
##    facility_id facility_name               city    state measure_code      score
##    <chr>       <chr>                       <chr>   <chr> <chr>             <chr>
##  1 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_006_01_SIR      1.446
##  2 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_008_02_OBS_RATE 99.7
##  3 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_009_03_ADJ_CHG~ 12.8
##  4 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_010_03_ADJ_CHG~ 29.3
##  5 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_011_03_OBS_RATE 59.3
##  6 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_012_03_OBS_RATE 49.8
##  7 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_013_01_OBS_RATE 0.0
##  8 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_015_01_SIR      0.775
##  9 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_016_01_OBS_RATE 69.2
## 10 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_017_01_PPR_PD_~ 6.66
## # ... with 17,570 more rows
```

- (4 points) Table 8 of the data dictionary contains all of the measure codes in the original data (listed as "Provider Variables"). Why might we want to use only the subset of measure codes in Table 1 for our predictive models?

Many of the measure codes we selected to use in our predictive model are either percentages or rates calculated based on the measure codes we excluded. If we did not exclude these measure codes, we would be inserting the same variable into our analysis twice, which could create multi-collinearity issues.

- (4 points) The column `score` contains the value of each measure, with missing values indicated using the string `Not Available`. What fraction of IRFs have any missing scores? [Hint: Use the `any` function within `summarise`; see `?any`.]

```
irf_data_2 %>%
  group_by(facility_id) %>%
  #determine if any measures are missing for given facility id
  summarise(na_score = any(score == "Not Available")) %>%
  #calculate fraction of facility ids with missing scores
  summarise('Fraction of IRFS with Missing Scores'= mean(na_score == TRUE)) %>%
  kable(format = "latex", row.names = NA,
        booktabs = TRUE, digits = 4,
        caption = "Fraction of IRFs with Missing Scores") %>%
  kable_styling(position = "center")
```

71.2% of IRFs have missing scores

- (4 points) Create a new tibble called `irf_data_3` from `irf_data_2` that contains only those IRFs for which none of the scores are missing. Print this tibble (no need to create a fancy table). How many IRFs remain? [Hint: First create a list of IRFs for which none of the scores are missing using code from the previous bullet point.]

```
irf_no_missing = irf_data_2 %>%
  group_by(facility_id) %>%
  #find facility ids with missing scores
  summarise(missing_score = any(score == "Not Available")) %>%
  #keep only those without missing scores
  filter(missing_score == FALSE) %>%
  select(facility_id) %>% pull()

irf_data_3= irf_data_2 %>%
  #keep only IRFs with no missing
  filter(facility_id %in% irf_no_missing)

print(irf_data_3) #print
```

```
## # A tibble: 5,070 x 6
##    facility_id facility_name               city    state measure_code      score
##    <chr>       <chr>                       <chr>   <chr> <chr>             <chr>
##  1 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_006_01_SIR      1.446
##  2 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_008_02_OBS_RATE 99.7
##  3 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_009_03_ADJ_CHG~ 12.8
##  4 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_010_03_ADJ_CHG~ 29.3
##  5 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_011_03_OBS_RATE 59.3
##  6 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_012_03_OBS_RATE 49.8
##  7 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_013_01_OBS_RATE 0.0
##  8 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_015_01_SIR      0.775
##  9 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_016_01_OBS_RATE 69.2
## 10 013025      ENCOMPASS HEALTH LAKESHORE~ BIRMIN~ AL    I_017_01_PPR_PD_~ 6.66
## # ... with 5,060 more rows
```

- (4 points) Convert the `score` column from `character` to `numeric`. Replace the `measure_code` column with a column named `feature`, which is a factor with `levels` equal to the first column of Table 1 and `labels` equal to the the second column of Table 1. [Hint: You can create a factor variable out of a character variable with given `levels` and `labels` using the `factor()` function; check out `?factor`.] Store the resulting tibble in `irf_data_4`. Print this tibble (no need to create a fancy table).

```
irf_data_4 <- irf_data_3 %>%
  mutate(score = as.numeric(score),
         #changes measure code to factor and assign levels
          measure_code = factor(measure_code,
                                levels = variables$measure_code)) %>%
  rename(feature = measure_code) #rename to feature
print(irf_data_4) # print
```

```
## # A tibble: 5,070 x 6
##    facility_id facility_name                city    state feature          score
##    <chr>       <chr>                        <chr>   <chr> <fct>            <dbl>
##  1 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_006_01_SIR     1.45
##  2 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_008_02_OBS_R~ 99.7
##  3 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_009_03_ADJ_C~ 12.8
```

```
##  4 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_010_03_ADJ_C~ 29.3
##  5 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_011_03_OBS_R~ 59.3
##  6 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_012_03_OBS_R~ 49.8
##  7 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_013_01_OBS_R~  0
##  8 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_015_01_SIR     0.775
##  9 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_016_01_OBS_R~ 69.2
## 10 013025      ENCOMPASS HEALTH LAKESHORE ~ BIRMIN~ AL    I_017_01_PPR_P~  6.66
## # ... with 5,060 more rows
```

1.3 Tidy (8 points)

- (4 points) Tidy the tibble `irf_data_4`, storing the result in a new tibble called `irf_data_tidy`. Print this tibble (no need to create a fancy table).

```
irf_data_tidy <- irf_data_4 %>%
  #create a column for each feature and assign value to columns
  pivot_wider(names_from = feature, values_from = score) %>%
  #rename measure_codes to new_names
  rename_at(vars(variables$measure_code), ~variables$new_name)
print(irf_data_tidy) #print
```

```
## # A tibble: 338 x 19
##    facility_id facility_name              city    state uti_rate abilities_goals
##    <chr>       <chr>                      <chr>   <chr>    <dbl>           <dbl>
##  1 013025      ENCOMPASS HEALTH LAKESHOR~ BIRMIN~ AL       1.45            99.7
##  2 013028      ENCOMPASS REHABILITATION ~ MONTGO~ AL       0.358           99.7
##  3 013029      ENCOMPASS HEALTH REHABILI~ HUNTSV~ AL       0               99.8
##  4 013030      ENCOMPASS HEALTH REHABILI~ DOTHAN  AL       0               99.8
##  5 013032      ENCOMPASS HEALTH REHABILI~ GADSDEN AL       0               99.9
##  6 013033      REGIONAL REHABILITATION H~ PHENIX~ AL       1.64            99.8
##  7 01T033      SPAIN REHABILITATION CENT~ BIRMIN~ AL       1.30           100
##  8 01T092      DCH REHABILITATION CENTER  TUSCAL~ AL       1.33            99.5
##  9 01T113      MOBILE INFIRMARY           MOBILE  AL       1.87           100
## 10 01T157      JW SOMMER REHABILITATION ~ MUSCLE~ AL       1.48            99.6
## # ... with 328 more rows, and 13 more variables:
## #   self_care_ability_improved <dbl>, mobility_improved <dbl>,
## #   self_care_expectation_met <dbl>, mobility_expectation_met <dbl>,
## #   fall_rate <dbl>, cdi_rate <dbl>, flu_vax_rate <dbl>,
## #   readmission_rate_after <dbl>, readmission_rate_during <dbl>,
## #   rate_return <dbl>, mspb <dbl>, medications_reviewed <dbl>,
## #   pressure_ulcers <dbl>
```

- (4 points) What does each row in `irf_data_tidy` represent? How many rows are there?

There are 338 rows, and each row in irf_data_tidy represents a different inpatient rehabilitation facility.

2 Exploration (25 points for correctness; 5 points for presentation)

The tidied dataset `irf_data_tidy` is provided for you here. You may use this to complete the remainder of the exam if you are unable to complete the tidying yourself.

2.1 Response distribution (15 points)

- (5 points) Print the mean, standard deviation, minimum, and maximum of `mspb` in a nice table.

Table 3: Summary of Medicare Spending per Beneficiary

Mean	Standard Deviation	Minimum	Maximum
1	0.07	0.79	1.34

```
irf_data_tidy %>%
  summarise(Mean = mean(mspb), 'Standard Deviation' = sd(mspb), Minimum =
              min(mspb), Maximum = max(mspb)) %>%
  kable(format = "latex", row.names = NA,
        booktabs = TRUE, digits = 2,
        caption = "Summary of Medicare Spending per Beneficiary") %>%
  kable_styling(position = "center")
```

- (5 points) Produce a histogram of `mspb`, with a vertical dashed line at the mean. Comment on the shape of this distribution.

```
# plot histogram ofmedical spending per beneficiary
median_mspb = irf_data_tidy %>% summarise(median(mspb)) %>% pull()
irf_data_tidy %>%
  ggplot(aes(x = mspb)) +
  geom_histogram(bins = 20, fill = "grey", col = "black") +
  geom_vline(xintercept = median_mspb,
             linetype = "dashed") +
  labs(x = "Medical Spending Per Beneficiary",
       y = "Number of IRFS") +
  theme_bw()
```

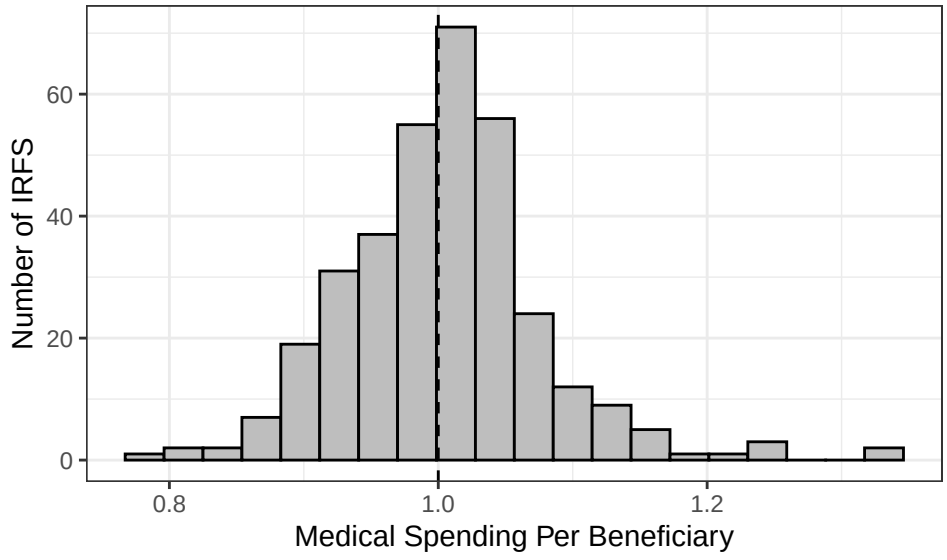

Figure 1: Distribution of medical spending per beneficiary; vertical dashed line indicates the median.

1 shows that the distribution of medical spending per beneficiary is slightly skewed to the right, with a median of 1. This indicates that there are a few high outliars

- (5 points) Produce a (nice) table of the IRFs with the top 5 `mspb` values; for each IRF print its `facility_name`, `city`, `state`, and `mspb`. What cities (or city) do the two most costly IRFs come from?

Table 4: Top 5 IRFs by Medicare Spending per Beneficiary

facility_name	city	state	mspb
HOSPITAL OF THE UNIVERSITY OF PENNSYLVANIA	PHILADELPHIA	PA	1.34
MAGEE REHABILITATION HOSPITAL	PHILADELPHIA	PA	1.33
LAC/RANCHO LOS AMIGOS NATIONAL MED CTR	DOWNEY	CA	1.25
SOUTHERN CALIFORNIA HOSPITAL AT CULVER CITY	CULVER CITY	CA	1.24
HOLY CROSS HOSPITAL	FORT LAUDERDALE	FL	1.24

```
irf_data_tidy %>%
  select(facility_name, city, state, mspb) %>%
  arrange(desc(mspb)) %>%
  head(5) %>%
  kable(format = "latex", row.names = NA,
        booktabs = TRUE, digits = 2,
        caption = "Top 5 IRFs by Medicare Spending per Beneficiary") %>%
  kable_styling(position = "center")
```

The two most costly IRFS come from Philadelphia, PA

2.2 Relationships among features (10 points)

- (5 points) Some of the features appear to be related to each other, e.g. `self_care_ability_improved` and `self_care_expectation_met`; `mobility_improved` and `mobility_expectation_met`; `readmission_rate_during` and `readmission_rate_after`. Create scatter plots of these three pairs of features, displaying all three side by side in a single figure, and comment on the degree to which each pair appears to be correlated.

```
# plot self_care_ability_improved against self_care_expectation_met
p1 = irf_data_tidy %>%
  ggplot(aes(x = self_care_ability_improved, y = self_care_expectation_met)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y~x", se = FALSE) +
  labs(x = "Self Care Ability Improved",
       y = "Self Care Expectation Met") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

# plot mobility_improved against mobility_expectation_met
p2 = irf_data_tidy %>%
  ggplot(aes(x = mobility_improved, y = mobility_expectation_met)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y~x", se = FALSE) +
  labs(x = "Mobility Improved",
       y = "Mobility Expectation Met") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

# plot readmission_rate_during against readmission_rate_after
p3 = irf_data_tidy %>%
  ggplot(aes(x = readmission_rate_during, y = readmission_rate_after)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y~x", se = FALSE) +
```

```
  labs(x = "Readmission Rate During Stay",
       y = "Readmission Rate After Stay") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

# combine the plots
cowplot::plot_grid(p1, p2, p3, nrow = 1)
```

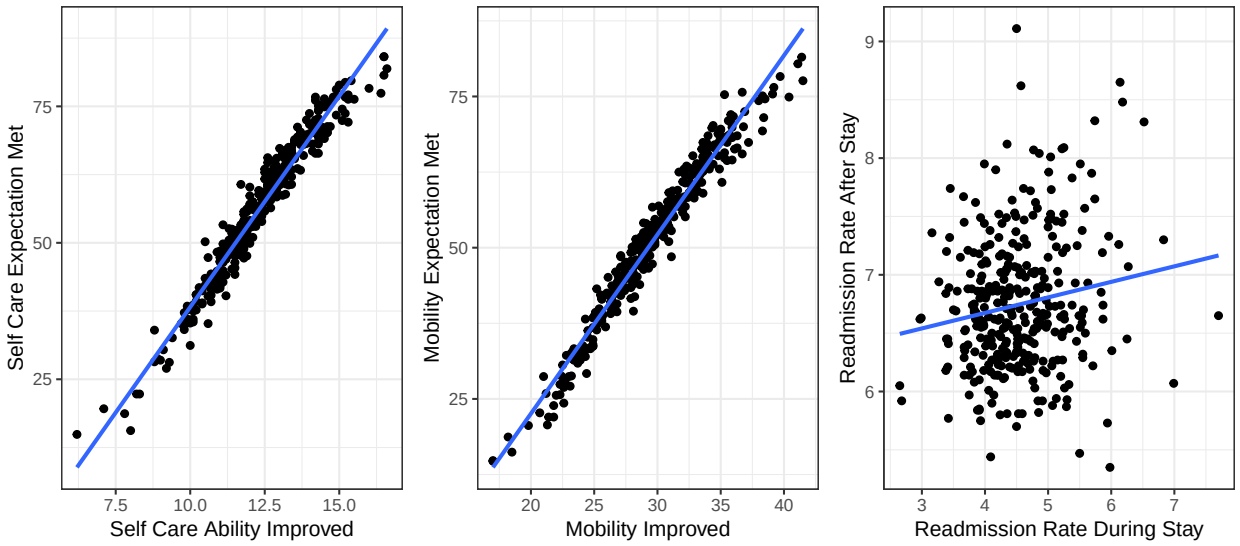

Figure 2: Scatter plots of three pairs of features: self care ability improved and self care ability met; mobility improved and mobility expectation met; readmission rate during and readmission rate after

The percentage of patients who achieve or exceed a self-care ability expected for their condition at discharge has a strong, positive relationship with how much the patients' ability to care for themselves changed between facility admission and discharge

Likewise, the percentage of patients who achieve or exceed the level of movement expected for their condition at discharge has a strong, postive relationship with how much the patients' ability to move around changed between facility admission and discharge.

Finally, the rate of potentially preventable hospital readmissions 30 days after discharge from an IRF has a moderate positive relationship with the rate of potentially preventable hospital readmissions during the IRF stay

- (5 points) Discuss how the findings in the above bullet point might impact the fitted coefficients of ridge and lasso regressions. **Lasso and ridge regressions treat correlated features differently, but both try and deal with multi-collinearity issues. The ridge regression provides are a more stable approach to dealing with correlated functions by "splitting the credit" among correlated features. On the other hand, lasso coefficients are unstable for correlated features because the lasso penalty does not help "break the tie" between which figure should have more importance in the model. Instead, lasso often chooses one of the two features arbitrarily. However, it is important to note that coefficient instability doesn't necessarily translate into prediction instability.**

3 Elastic net regression (25 points for correctness; 5 points for presentation)

Next, let's train penalized regression models to predict the case-fatality ratio based on the available features. We use the following train-test split:

```
num_train_samples = 300
set.seed(1) # seed set for reproduciblity (DO NOT CHANGE)

#uncomment the three commented lines below to create train-test split
train_samples = sample(1:nrow(irf_data_tidy), num_train_samples)
irf_train = irf_data_tidy %>% filter(row_number() %in% train_samples)
irf_test = irf_data_tidy %>% filter(!(row_number() %in% train_samples))
```

3.1 Training and tuning (15 points)

- (3 points) Using `irf_train`, fit a 10-fold cross-validated elastic net regression of `mspb` on the 14 other features in Table 1, where the cross-validation is over `alpha` as well as `lambda`.

```
set.seed(127) # set seed for reproducibility (DO NOT CHANGE)
#Fit cross-validated elastic net regression, leaving out facility_name, city, and state
elnet_fit = cva.glmnet(mspb ~ . -facility_id - facility_name - city - state,
                       nfolds = 10,        # number of folds
                       data = irf_train)   # data to run on
```

- (3 points) Produce a plot of the CV error (minimized across `lambda` for each `alpha`) versus `alpha`. Which value of alpha performs best? What does this suggest about the number of features actually impacting the response?

```
plot_cva_glmnet(elnet_fit)
```

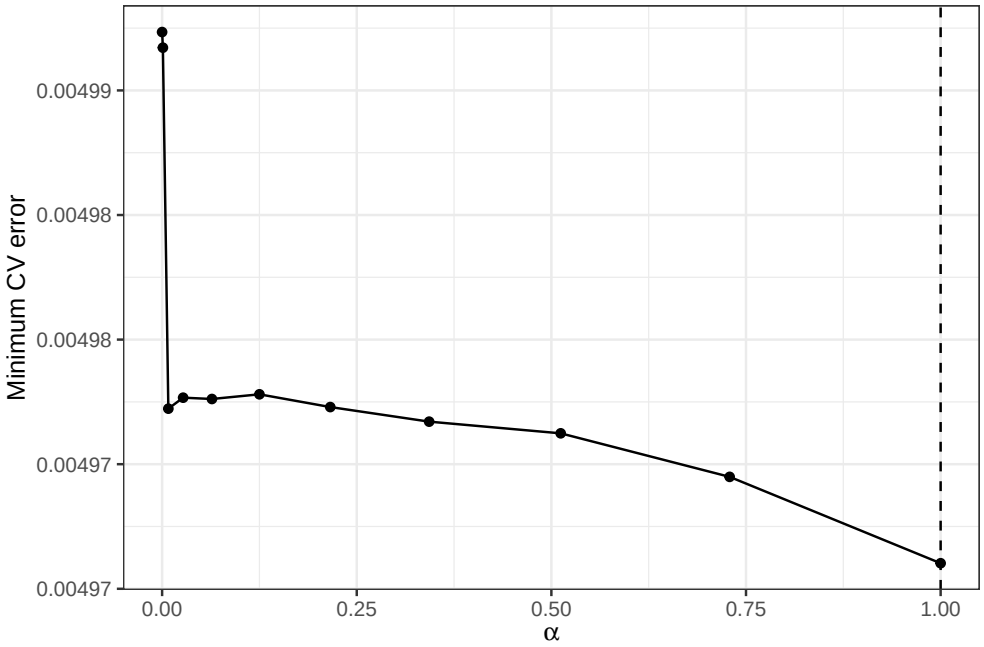

Figure 3: Elastic Net Error Plot.

```
elnet_fit_best = extract_best_elnet(elnet_fit)
best_alpha = elnet_fit_best$alpha
```

Alpha equals 1 performs best, which suggests that lasso model gives lower error than any combination of ridge and lasso. Lasso tends to do well if there are a small number of significant parameters and the others are all close to 0. In contrast, ridge works better when there are many large parameters of about the same value. Thus, this suggests that there are only a small number of significant parameters

- (3 points) Produce a regular CV plot (showing CV error as a function of lambda) for the alpha selected in the previous bullet point. Based on this plot, how many features have nonzero coefficients in the model selected by the one-standard error rule?

```
plot(elnet_fit_best)
```

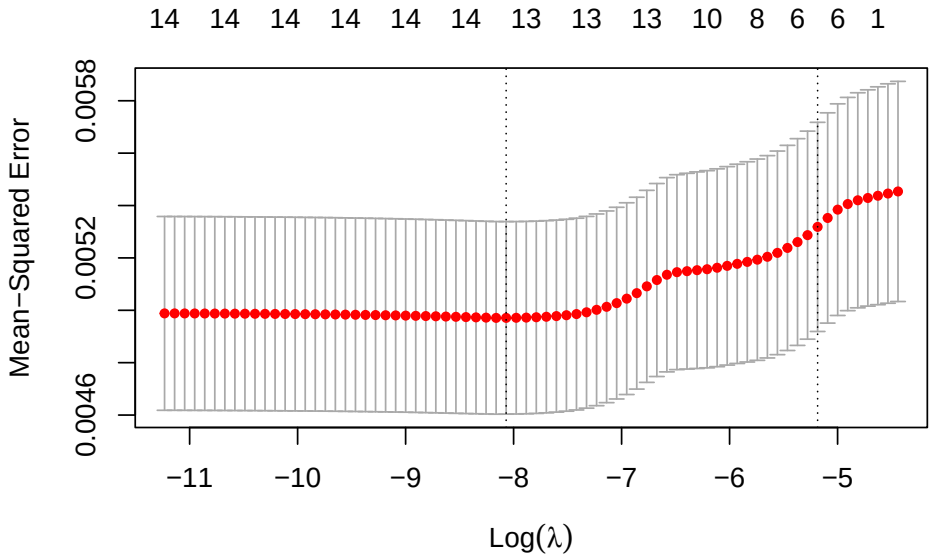

Figure 4: Elastic Net CV plot where alpha is set to minimizes CV error

```
nzero_1se = elnet_fit_best$nzero[elnet_fit_best$lambda == elnet_fit_best$lambda.1se]
```

6 features have nonzero coefficients when the number of features is determined by using the one-standard error rule.This is seen in 4 and calculated manually

- (3 points) Produce the trace plot for the `glmnet` model from the previous bullet point. [Hint: To make the plot easier to read, change the y axis scale by appending `+ scale_y_continuous(limits = c(-0.02, 0.02))` after your `plot_glmnet` call.] What is the first feature that comes in with a positive coefficient? What is the first feature that comes in with a negative coefficient? Do the signs of these coefficients make sense to you? Why or why not?

```
plot_glmnet(elnet_fit_best, irf_train, features_to_plot = 6) +
  scale_y_continuous(limits = c(-0.02, 0.02))
```

```
## Warning: Removed 96 row(s) containing missing values (geom_path).
```

The first positive coefficient to come into the model is cdi_rate. It makes sense to me that when the Clostridium difficile infection rate increases, medicare spending per beneficary would

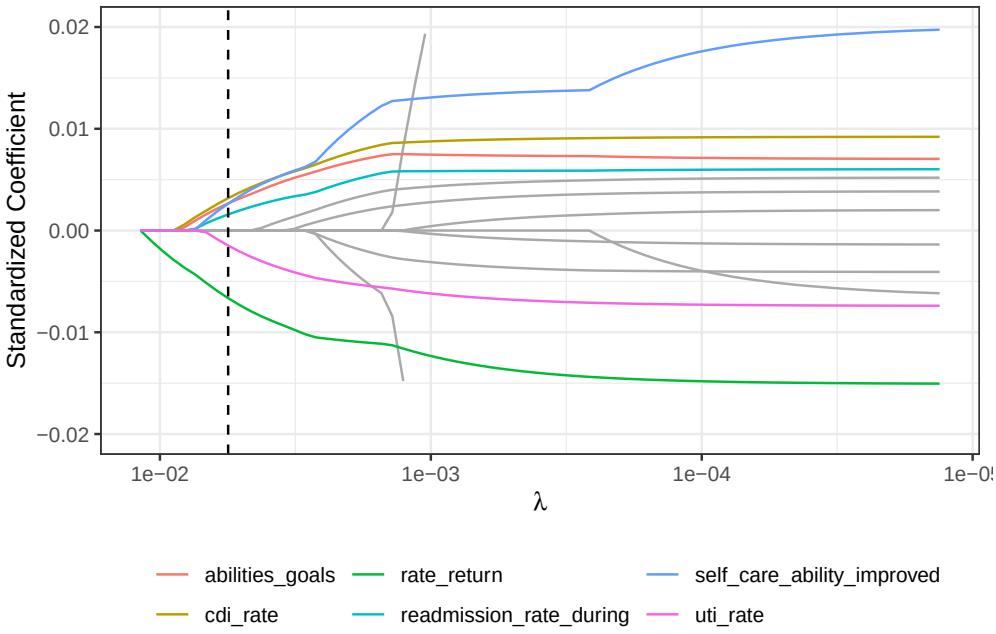

Figure 5: Elastic Net trace plot where 6 features are plotted

also increase because Clostridium difficile colitis results from disruption of normal healthy bacteria in the colon. C. difficile can also be transmitted from person to person by spores. It can cause severe damage to the colon and even be fatal. Thus, those with C. difficile are likely to spend more on their healthcare

The first negative coefficient to come into the model is rate_return. This makes sense because when a patient is succesffully returned to their home and community from an IRF, they are no longer incurring medical bills. The longer a patient stays in the facility, the more they will pay.

- (3 points) Produce a nice table with the features selected by the above `glmnet` model and their standardized coefficients, ordered by decreasing magnitude. According to these coefficients, if an IRF increases its `rate_return` by 10 while keeping other things equal, how would this change the expected `mspb`?

```
beta_hat_std = extract_std_coefs(elnet_fit_best, irf_train)
beta_hat_std %>%
  filter(coefficient != 0) %>%
  arrange(desc(abs(coefficient))) %>%
  kable(format = "latex", row.names = NA,
        booktabs = TRUE, digits = 4,
        caption = "Features and Standardized Coefficents") %>%
  kable_styling(position = "center")
```

```
#calc standard deviation of rate_return
sd_rate_return = irf_train %>%
  summarise(sd(rate_return)) %>% pull
#multiple the change in standard deviation by the coefficent
change_rate_return_10 = beta_hat_std$coefficient[(which(beta_hat_std$feature
                        == "rate_return"))]*(10/sd_rate_return)
```

The coefficient for rate_return tells us that for every increase of one standard deviation in

Table 5: Features and Standardized Coefficents

feature	coefficient
rate_return	-0.0066
cdi_rate	0.0032
abilities_goals	0.0027
self_care_ability_improved	0.0026
readmission_rate_during	0.0016
uti_rate	-0.0015

rate_return, mspb would decrease by 0.00665. Since the standard deviation for rate_return is equal to 5.47, an increase in rate return represent a 1.83 increase in the standard deviation. Thus, mspb will decrease by 0.0121 (0.00665 x 1.83) as a result of a 10 unit increase in mspb

3.2 Performance evaluation (10 points)

Let's compare the performance of elastic net models for different `alpha`. While the fit objects for each `alpha` can be extracted from the original fit object from Section 3.1, we haven't learned how to do this. We will instead separately re-train models for different `alpha`.

- (5 points) Train three cross-validated elastic net models—one for `alpha` = 0, one for `alpha` = 0.5, and one for `alpha` = 1—this time cross-validating over lambda only.

```
set.seed(127) # set seed for reproducibility for alpha = 0 (DO NOT CHANGE)
# Fit elastic net for alpha = 0
elnet_fit_0 = cva.glmnet(mspb ~ . -facility_id - facility_name - city - state,
                         alpha = 0,          # set alpha to 0
                         nfolds = 10,        # number of folds
                         data = irf_train)   # data to run on

set.seed(127) # set seed for reproducibility for alpha = 0.5 (DO NOT CHANGE)
# Fit elastic net for alpha = 0.5
elnet_fit_0.5 = cva.glmnet(mspb ~ . -facility_id - facility_name - city - state,
                         alpha = 0.5,        # set alpha to 0.5
                         nfolds = 10,        # number of folds
                         data = irf_train)   # data to run on

set.seed(127) # set seed for reproducibility for alpha = 1 (DO NOT CHANGE)
# Fit elastic net for alpha = 1
elnet_fit_1 = cva.glmnet(mspb ~ . -facility_id - facility_name - city - state,
                         alpha = 1,          # set alpha to 1
                         nfolds = 10,        # number of folds
                         data = irf_train)   # data to run on
```

- (5 points) Compute the RMSE prediction error for each of these three models, with `lambda` chosen based on the one-standard-error rule. Print these in a nice table, together with the corresponding `alpha` values (use `digits = 4` in your call to `kable`). Comment on the ordering among the prediction errors, and the extent to which it is consistent with the CV error versus `alpha` plot from Section 3.1.

```
predictions_0 = predict(elnet_fit_0,
                         newdata = irf_test,
                         alpha = 0,
                         s = "lambda.1se") %>% as.numeric()
```

Table 6: Root-mean-squared prediction errors for elastic net model when alpha is equal to 0, 0.5, and 1

Alpha = 0	Alpha = 0.5	Alpha = 1
0.0613	0.0598	0.0597

```
predictions_0.5 = predict(elnet_fit_0.5,
                          newdata = irf_test,
                          alpha = 0.5,
                          s = "lambda.1se") %>% as.numeric()
predictions_1 = predict(elnet_fit_1,
                        newdata = irf_test,
                        alpha = 1,
                        s = "lambda.1se") %>% as.numeric()

RMSE_0 = sqrt(mean((predictions_0-irf_test$mspb)^2))
RMSE_0.5 = sqrt(mean((predictions_0.5 -irf_test$mspb)^2))
RMSE_1 = sqrt(mean((predictions_1 -irf_test$mspb)^2))

# Format table
tibble('Alpha = 0'= RMSE_0, 'Alpha = 0.5'= RMSE_0.5, 'Alpha = 1'= RMSE_1) %>%
   kable(format = "latex", row.names = NA,
       booktabs = TRUE, digits = 4,
       caption = "Root-mean-squared prediction errors for elastic net model
       when alpha is equal to 0, 0.5, and 1") %>%
 kable_styling(position = "center")
```

Alpha = 1 performs the best. Alpha = 1 (lasso) performs marginally better than when Alpha = 0 (ridge), and Alpha = 0.5 (combination of lasso and ridge) performs better than ridge alone but worse than lasso alone.. Similar to 3, the RMSE suggests that lasso alone performs better than any combination of lasso and ridge, which implies that there are a small number of significant parameters. This is consistent with consistent with the CV error versus `alpha` plot from Section 3.1