ST 563 Project

AUTHOR
Matthieu Cartron

Data Processing

Background Information

The following data is sourced from two sources: the U.S Census and the Centers for Medicare and Medicaid Services. From this latter government entity I took two datasets: The FY 2025 SNF VBP Facility-Level Dataset (contains facility-specific performance results for the fiscal year (FY) 2025 Skilled Nursing Facility Value-Based Purchasing (SNF VBP) Program) and the Provider Information Dataset, which includes (as of March 1st) general information on currently active nursing homes, including number of certified beds, quality measure scores, staffing and other information used in the Five-Star Rating System.

From the U.S census I took ZIP-level median income and unemployment rates, and joined these by ZIP with the FY 2025 SNF VBP Facility-Level Dataset. This was then joined with the Provider Information data set.

Project Goal: I ultimately decided to look at the `Total_Weighted_Health_Survey_Score`; this will be the response variable in this analysis. The total weighted health survey score seems to give the best sense of how a skilled nursing facility might be operated. It is quite likely that some of the variable I include as predictors are used to construct the score to begin with, but this is what makes the analysis in large part interesting. I also include some economic variables that I thought could be related in some way.

Ingestion of Data and Data Cleaning:

Below the data sets are combined based on ZIP code and the provider name.

```r
library(tidycensus)
library(tidyverse)
library(RAQSAPI)
library(readr)

nursing_data <- read_csv("FY_2025_SNF_VBP_Facility_Performance.csv")

# bring in economic vars - API key not required
acs_data <- get_acs(
  geography = "zcta",
  variables = c(median_income = "B19013_001", unemployment_rate = "B23025_005"),
  year = 2023, survey = "acs5") |>
  select(GEOID, variable, estimate) |>
  pivot_wider(names_from = variable, values_from = estimate) |>
  rename(ZIP_Code = GEOID) |>
  mutate(ZIP_Code = as.double(ZIP_Code))



nursing_data <- nursing_data |>
  rename(Provider_Name = `Provider Name`,
         ZIP_Code = `ZIP Code`)

# Merge with nursing home data
nursing_data <- nursing_data |>
  left_join(acs_data, by = "ZIP_Code")



cms_nursing_data <- read_csv("NH_ProviderInfo_Mar2025.csv")

# Standardize provider names for better matching
nursing_data <- nursing_data |>
  mutate('Provider_Name' = tolower(trimws(Provider_Name)))

cms_nursing_data <- cms_nursing_data |>
  rename(Provider_Name = `Provider Name`,
         ZIP_Code = `ZIP Code`,
         CMS_Rating = `Overall Rating`)

cms_nursing_data <- cms_nursing_data |>
   mutate(ZIP_Code = as.double(ZIP_Code)) |>
   mutate(Provider_Name = tolower(trimws(Provider_Name)))

# Merge using BOTH Provider Name & ZIP Code
nursing_data <- nursing_data |>
  inner_join(cms_nursing_data, by = c("Provider_Name", "ZIP_Code"))


glimpse(nursing_data)
```

```
Rows: 9,762
Columns: 123
$ `SNF VBP Program Ranking`                                             <dbl> …
$ `Footnote -- SNF VBP Program Ranking`                                 <lgl> …
$ `CMS Certification Number (CCN).x`                                    <dbl> …
$ Provider_Name                                                         <chr> …
$ `Provider Address.x`                                                  <chr> …
$ `City/Town.x`                                                         <chr> …
$ State.x                                                               <chr> …
$ ZIP_Code                                                              <dbl> …
$ `Baseline Period: FY 2019 Risk-Standardized Readmission Rate`         <chr> …
$ `Footnote -- Baseline Period: FY 2019 Risk-Standardized Readmission Rate`     <chr> …
$ `Performance Period: FY 2023 Risk-Standardized Readmission Rate`      <dbl> …
$ `Footnote -- Performance Period: FY 2023 Risk-Standardized Readmission Rate` <lgl> …
$ `Achievement Score`                                                <dbl> …
$ `Footnote -- Achievement Score`                                    <lgl> …
$ `Improvement Score`                                                <chr> …
$ `Footnote -- Improvement Score`                                    <chr> …
$ `Performance Score`                                                <dbl> …
$ `Footnote -- Performance Score`                                    <lgl> …
$ `Incentive Payment Multiplier`                                     <dbl> …
$ `Footnote -- Incentive Payment Multiplier`                         <lgl> …
$ median_income                                                      <dbl> …
$ unemployment_rate                                                  <dbl> …
$ `CMS Certification Number (CCN).y`                                 <chr> …
$ `Provider Address.y`                                               <chr> …
$ `City/Town.y`                                                      <chr> …
$ State.y                                                            <chr> …
$ `Telephone Number`                                                 <dbl> …
$ `Provider SSA County Code`                                         <chr> …
$ `County/Parish`                                                    <chr> …
$ `Ownership Type`                                                   <chr> …
$ `Number of Certified Beds`                                         <dbl> …
$ `Average Number of Residents per Day`                              <dbl> …
$ `Average Number of Residents per Day Footnote`                     <dbl> …
$ `Provider Type`                                                    <chr> …
$ `Provider Resides in Hospital`                                     <chr> …
$ `Legal Business Name`                                              <chr> …
$ `Date First Approved to Provide Medicare and Medicaid Services`    <date> …
$ `Affiliated Entity Name`                                           <chr> …
$ `Affiliated Entity ID`                                             <dbl> …
$ `Continuing Care Retirement Community`                             <chr> …
$ `Special Focus Status`                                             <chr> …
$ `Abuse Icon`                                                       <chr> …
$ `Most Recent Health Inspection More Than 2 Years Ago`              <chr> …
$ `Provider Changed Ownership in Last 12 Months`                     <chr> …
$ `With a Resident and Family Council`                               <chr> …
$ `Automatic Sprinkler Systems in All Required Areas`                <chr> …
$ CMS_Rating                                                         <dbl> …
$ `Overall Rating Footnote`                                          <dbl> …
$ `Health Inspection Rating`                                         <dbl> …
$ `Health Inspection Rating Footnote`                                <dbl> …
$ `QM Rating`                                                        <dbl> …
$ `QM Rating Footnote`                                               <dbl> …
$ `Long-Stay QM Rating`                                              <dbl> …
$ `Long-Stay QM Rating Footnote`                                     <dbl> …
$ `Short-Stay QM Rating`                                             <dbl> …
$ `Short-Stay QM Rating Footnote`                                    <dbl> …
$ `Staffing Rating`                                                  <dbl> …
$ `Staffing Rating Footnote`                                         <dbl> …
$ `Reported Staffing Footnote`                                       <dbl> …
$ `Physical Therapist Staffing Footnote`                             <dbl> …
$ `Reported Nurse Aide Staffing Hours per Resident per Day`          <dbl> …
$ `Reported LPN Staffing Hours per Resident per Day`                 <dbl> …
$ `Reported RN Staffing Hours per Resident per Day`                  <dbl> …
$ `Reported Licensed Staffing Hours per Resident per Day`                  <dbl> …
$ `Reported Total Nurse Staffing Hours per Resident per Day`               <dbl> …
$ `Total number of nurse staff hours per resident per day on the weekend`  <dbl> …
$ `Registered Nurse hours per resident per day on the weekend`             <dbl> …
$ `Reported Physical Therapist Staffing Hours per Resident Per Day`        <dbl> …
$ `Total nursing staff turnover`                                           <dbl> …
$ `Total nursing staff turnover footnote`                                  <dbl> …
$ `Registered Nurse turnover`                                              <dbl> …
$ `Registered Nurse turnover footnote`                                     <dbl> …
$ `Number of administrators who have left the nursing home`                <dbl> …
$ `Administrator turnover footnote`                                        <dbl> …
$ `Nursing Case-Mix Index`                                                 <dbl> …
$ `Nursing Case-Mix Index Ratio`                                           <dbl> …
$ `Case-Mix Nurse Aide Staffing Hours per Resident per Day`                <dbl> …
$ `Case-Mix LPN Staffing Hours per Resident per Day`                       <dbl> …
$ `Case-Mix RN Staffing Hours per Resident per Day`                        <dbl> …
$ `Case-Mix Total Nurse Staffing Hours per Resident per Day`               <dbl> …
$ `Case-Mix Weekend Total Nurse Staffing Hours per Resident per Day`       <dbl> …
$ `Adjusted Nurse Aide Staffing Hours per Resident per Day`                <dbl> …
$ `Adjusted LPN Staffing Hours per Resident per Day`                       <dbl> …
$ `Adjusted RN Staffing Hours per Resident per Day`                        <dbl> …
$ `Adjusted Total Nurse Staffing Hours per Resident per Day`               <dbl> …
$ `Adjusted Weekend Total Nurse Staffing Hours per Resident per Day`       <dbl> …
$ `Rating Cycle 1 Standard Survey Health Date`                             <date> …
$ `Rating Cycle 1 Total Number of Health Deficiencies`                     <dbl> …
$ `Rating Cycle 1 Number of Standard Health Deficiencies`                  <dbl> …
$ `Rating Cycle 1 Number of Complaint Health Deficiencies`                 <dbl> …
$ `Rating Cycle 1 Health Deficiency Score`                                 <dbl> …
$ `Rating Cycle 1 Number of Health Revisits`                               <dbl> …
$ `Rating Cycle 1 Health Revisit Score`                                    <dbl> …
$ `Rating Cycle 1 Total Health Score`                                      <dbl> …
$ `Rating Cycle 2 Standard Health Survey Date`                             <date> …
$ `Rating Cycle 2 Total Number of Health Deficiencies`                     <dbl> …
$ `Rating Cycle 2 Number of Standard Health Deficiencies`                  <dbl> …
$ `Rating Cycle 2 Number of Complaint Health Deficiencies`                 <dbl> …
$ `Rating Cycle 2 Health Deficiency Score`                                 <dbl> …
$ `Rating Cycle 2 Number of Health Revisits`                               <dbl> …
$ `Rating Cycle 2 Health Revisit Score`                                    <dbl> …
$ `Rating Cycle 2 Total Health Score`                                      <dbl> …
$ `Rating Cycle 3 Standard Health Survey Date`                             <date> …
$ `Rating Cycle 3 Total Number of Health Deficiencies`                     <dbl> …
$ `Rating Cycle 3 Number of Standard Health Deficiencies`                  <dbl> …
$ `Rating Cycle 3 Number of Complaint Health Deficiencies`                 <dbl> …
$ `Rating Cycle 3 Health Deficiency Score`                                 <dbl> …
$ `Rating Cycle 3 Number of Health Revisits`                               <dbl> …
$ `Rating Cycle 3 Health Revisit Score`                                    <dbl> …
$ `Rating Cycle 3 Total Health Score`                                      <dbl> …
$ `Total Weighted Health Survey Score`                                     <dbl> …
$ `Number of Facility Reported Incidents`                                  <dbl> …
$ `Number of Substantiated Complaints`                                     <dbl> …
$ `Number of Citations from Infection Control Inspections`                 <dbl> …
$ `Number of Fines`                                              <dbl> …
$ `Total Amount of Fines in Dollars`                             <dbl> …
$ `Number of Payment Denials`                                    <dbl> …
$ `Total Number of Penalties`                                    <dbl> …
$ Location                                                       <chr> …
$ Latitude                                                       <dbl> …
$ Longitude                                                      <dbl> …
$ `Geocoding Footnote`                                           <dbl> …
$ `Processing Date`                                              <date> …
```

```
#write.csv(nursing_data, file = "skilled_nursing_data_Cartron_ST563.csv",
          #row.names = FALSE)
```

Below variables are renamed to be more useable. In addition, only a subset of the cleaned and renamed variables are used - I looked for what I thought would be mostly uncorrelated, interesting, and important predictors.

NOTE: We drop rows with N/A values as there are not many and we have a large, robust data set even without these missing values. Given the sample size, losing some important observations should not be expected to have a material impact on the results.

```
nursing_data_cleaned <- nursing_data |>
  # Rename columns: Replace spaces with underscores and remove special characters
  rename(
    SNF_VBP_Program_Ranking = `SNF VBP Program Ranking`,
    Provider_Name = `Provider_Name`,
    Provider_Address = `Provider Address.x`,
    City_Town = `City/Town.x`,
    State = `State.x`,
    ZIP_Code = `ZIP_Code`,
    Readmission_Rate_2019 = `Baseline Period: FY 2019 Risk-Standardized Readmission Rate`,
    Readmission_Rate = `Performance Period: FY 2023 Risk-Standardized Readmission Rate`,
    Achievement_Score = `Achievement Score`,
    Improvement_Score = `Improvement Score`,
    Performance_Score = `Performance Score`,
    Incentive_Payment_Multiplier = `Incentive Payment Multiplier`,
    median_income = `median_income`,
    unemployment_rate = `unemployment_rate`,
    CMS_Certification_Number = `CMS Certification Number (CCN).y`,
    Provider_Address_y = `Provider Address.y`,
    City_Town_y = `City/Town.y`,
    State_y = `State.y`,
    Telephone_Number = `Telephone Number`,
    Provider_SSA_County_Code = `Provider SSA County Code`,
    County_Parish = `County/Parish`,
    Ownership_Type = `Ownership Type`,
    Number_of_Certified_Beds = `Number of Certified Beds`,
    Average_Number_of_Residents_per_Day = `Average Number of Residents per Day`,
    Provider_Type = `Provider Type`,
    Provider_Resides_in_Hospital = `Provider Resides in Hospital`,
    Affiliated_Entity_Name = `Affiliated Entity Name`,
    Continuing_Care_Retirement_Community = `Continuing Care Retirement Community`,
    Abuse_Icon = `Abuse Icon`,
    Most_Recent_Health_Inspection_More_Than_2_Years_Ago = `Most Recent Health Inspection More Than
    Provider_Changed_Ownership_in_Last_12_Months = `Provider Changed Ownership in Last 12 Months`,
    Resident_and_Family_Council = `With a Resident and Family Council`,
    Automatic_Sprinkler_Required_Areas = `Automatic Sprinkler Systems in All Required Areas`,
    CMS_Rating = `CMS_Rating`,
    Health_Inspection_Rating = `Health Inspection Rating`,
    QM_Rating = `QM Rating`,
    Long_Stay_QM_Rating = `Long-Stay QM Rating`,
    Short_Stay_QM_Rating = `Short-Stay QM Rating`,
    Staffing_Rating = `Staffing Rating`,
    Reported_Nurse_Aide_Staffing_Hours_per_Resident_per_Day = `Reported Nurse Aide Staffing Hours
    Reported_LPN_Staffing_Hours_per_Resident_per_Day = `Reported LPN Staffing Hours per Resident p
    Reported_RN_Staffing_Hours_per_Resident_per_Day = `Reported RN Staffing Hours per Resident per
    Reported_Licensed_Staffing_Hours_per_Resident_per_Day = `Reported Licensed Staffing Hours per
    Reported_Total_Nurse_Staffing_Hours_per_Resident_per_Day = `Reported Total Nurse Staffing Hour
    Total_Number_of_Nurse_Staff_Hours_per_Resident_per_Day_on_Weekend = `Total number of nurse sta
    Registered_Nurse_Hours_per_Resident_per_Day_on_Weekend = `Registered Nurse hours per resident
    Reported_Physical_Therapist_Staffing_Hours_per_Resident_Per_Day = `Reported Physical Therapist
    Total_Nursing_Staff_Turnover = `Total nursing staff turnover`,
    Registered_Nurse_Turnover = `Registered Nurse turnover`,
    Number_of_Administrators_Having_Left_the_Nursing_Home = `Number of administrators who have lef
    Nursing_Case_Mix_Index = `Nursing Case-Mix Index`,
    Nursing_Case_Mix_Index_Ratio = `Nursing Case-Mix Index Ratio`,
    Case_Mix_Nurse_Aide_Staffing_Hours_per_Resident_per_Day = `Case-Mix Nurse Aide Staffing Hours
    Case_Mix_LPN_Staffing_Hours_per_Resident_per_Day = `Case-Mix LPN Staffing Hours per Resident p
    Case_Mix_RN_Staffing_Hours_per_Resident_per_Day = `Case-Mix RN Staffing Hours per Resident per
    Case_Mix_Total_Nurse_Staffing_Hours_per_Resident_per_Day = `Case-Mix Total Nurse Staffing Hour
    Case_Mix_Weekend_Total_Nurse_Staffing_Hours_per_Resident_per_Day = `Case-Mix Weekend Total Nur
    Adjusted_Nurse_Aide_Staffing_Hours_per_Resident_per_Day = `Adjusted Nurse Aide Staffing Hours
    Adjusted_LPN_Staffing_Hours_per_Resident_per_Day = `Adjusted LPN Staffing Hours per Resident p
    Adjusted_RN_Staffing_Hours_per_Resident_per_Day = `Adjusted RN Staffing Hours per Resident per
    Adjusted_Total_Nurse_Staffing_Hours_per_Resident_per_Day = `Adjusted Total Nurse Staffing Hour
    Adjusted_Weekend_Total_Nurse_Staffing_Hours_per_Resident_per_Day = `Adjusted Weekend Total Nur
    Rating_Cycle_1_Total_Number_of_Health_Deficiencies = `Rating Cycle 1 Total Number of Health De
    Rating_Cycle_1_Number_of_Standard_Health_Deficiencies = `Rating Cycle 1 Number of Standard Hea
    Rating_Cycle_1_Number_of_Complaint_Health_Deficiencies = `Rating Cycle 1 Number of Complaint H
    Rating_Cycle_1_Health_Deficiency_Score = `Rating Cycle 1 Health Deficiency Score`,
    Rating_Cycle_1_Number_of_Health_Revisits = `Rating Cycle 1 Number of Health Revisits`,
    Rating_Cycle_1_Health_Revisit_Score = `Rating Cycle 1 Health Revisit Score`,
    Rating_Cycle_1_Total_Health_Score = `Rating Cycle 1 Total Health Score`,
    Rating_Cycle_2_Total_Number_of_Health_Deficiencies = `Rating Cycle 2 Total Number of Health De
    Rating_Cycle_2_Number_of_Standard_Health_Deficiencies = `Rating Cycle 2 Number of Standard Hea
    Rating_Cycle_2_Number_of_Complaint_Health_Deficiencies = `Rating Cycle 2 Number of Complaint H
    Rating_Cycle_2_Health_Deficiency_Score = `Rating Cycle 2 Health Deficiency Score`,
    Rating_Cycle_2_Number_of_Health_Revisits = `Rating Cycle 2 Number of Health Revisits`,
    Rating_Cycle_2_Health_Revisit_Score = `Rating Cycle 2 Health Revisit Score`,
    Rating_Cycle_2_Total_Health_Score = `Rating Cycle 2 Total Health Score`,
    Rating_Cycle_3_Total_Number_of_Health_Deficiencies = `Rating Cycle 3 Total Number of Health De
    Rating_Cycle_3_Number_of_Standard_Health_Deficiencies = `Rating Cycle 3 Number of Standard Hea
    Rating_Cycle_3_Number_of_Complaint_Health_Deficiencies = `Rating Cycle 3 Number of Complaint H
    Rating_Cycle_3_Health_Deficiency_Score = `Rating Cycle 3 Health Deficiency Score`,
    Rating_Cycle_3_Number_of_Health_Revisits = `Rating Cycle 3 Number of Health Revisits`,
    Rating_Cycle_3_Health_Revisit_Score = `Rating Cycle 3 Health Revisit Score`,
    Rating_Cycle_3_Total_Health_Score = `Rating Cycle 3 Total Health Score`,
    Total_Weighted_Health_Survey_Score = `Total Weighted Health Survey Score`,
    Number_of_Facility_Reported_Incidents = `Number of Facility Reported Incidents`,
    Number_of_Substantiated_Complaints = `Number of Substantiated Complaints`,
    Number_of_Fines = `Number of Fines`,
    Total_Amount_of_Fines_in_Dollars = `Total Amount of Fines in Dollars`,
    Number_of_Payment_Denials = `Number of Payment Denials`,
    Total_Number_of_Penalties = `Total Number of Penalties`
  ) |>
  # Select only necessary vars
  select(
    Readmission_Rate,
    median_income,
    unemployment_rate,
    Number_of_Certified_Beds,
    Average_Number_of_Residents_per_Day,
    Provider_Type,
    Continuing_Care_Retirement_Community,
    Abuse_Icon,
    Health_Inspection_Rating,
    Staffing_Rating,
    Total_Weighted_Health_Survey_Score,
    Number_of_Facility_Reported_Incidents,
    Number_of_Substantiated_Complaints,
    Total_Amount_of_Fines_in_Dollars,
    Number_of_Payment_Denials,
    Total_Number_of_Penalties
  ) |>
  # Convert categorical variables to indicators
  mutate(
    Provider_Type = ifelse(Provider_Type == "Medicare and Medicaid", 1, 0),
    Continuing_Care_Retirement_Community = ifelse(Continuing_Care_Retirement_Community == "Y", 1,
    Abuse_Icon = ifelse(Abuse_Icon == "Y", 1, 0)
  ) |>
  # Remove rows with missing values
  drop_na()
```

Dealing with Multicollinearity

Below we remove highly correlated predictors (threshold above 0.7):

```
cor_matrix <- nursing_data_cleaned |>
  select(where(~ is.numeric(.))) |>  # Select numeric vars ONLY
  select(where(~ n_distinct(.) > 2)) |>
  cor(use = "pairwise.complete.obs")  # Compute correlation matrix


high_cor <- as.data.frame(as.table(cor_matrix)) |>
  filter(Var1 != Var2 & abs(Freq) > 0.7)  # Exclude diagonal & filter high correlations

# View highly correlated pairs
print(high_cor)
```

```
                                 Var1                               Var2
1 Average_Number_of_Residents_per_Day            Number_of_Certified_Beds
2            Number_of_Certified_Beds Average_Number_of_Residents_per_Day
       Freq
1 0.9229966
2 0.9229966
```

It turns out that `Number_of_Certified_Beds` and `Average_Number_of_Residents_per_Day` are highly correlated. While this may not be an issue for all of our models, if we want to ensure that the models with some interpretability do not have any issues with multicollinearity, we can combine these two variables into a single variable.

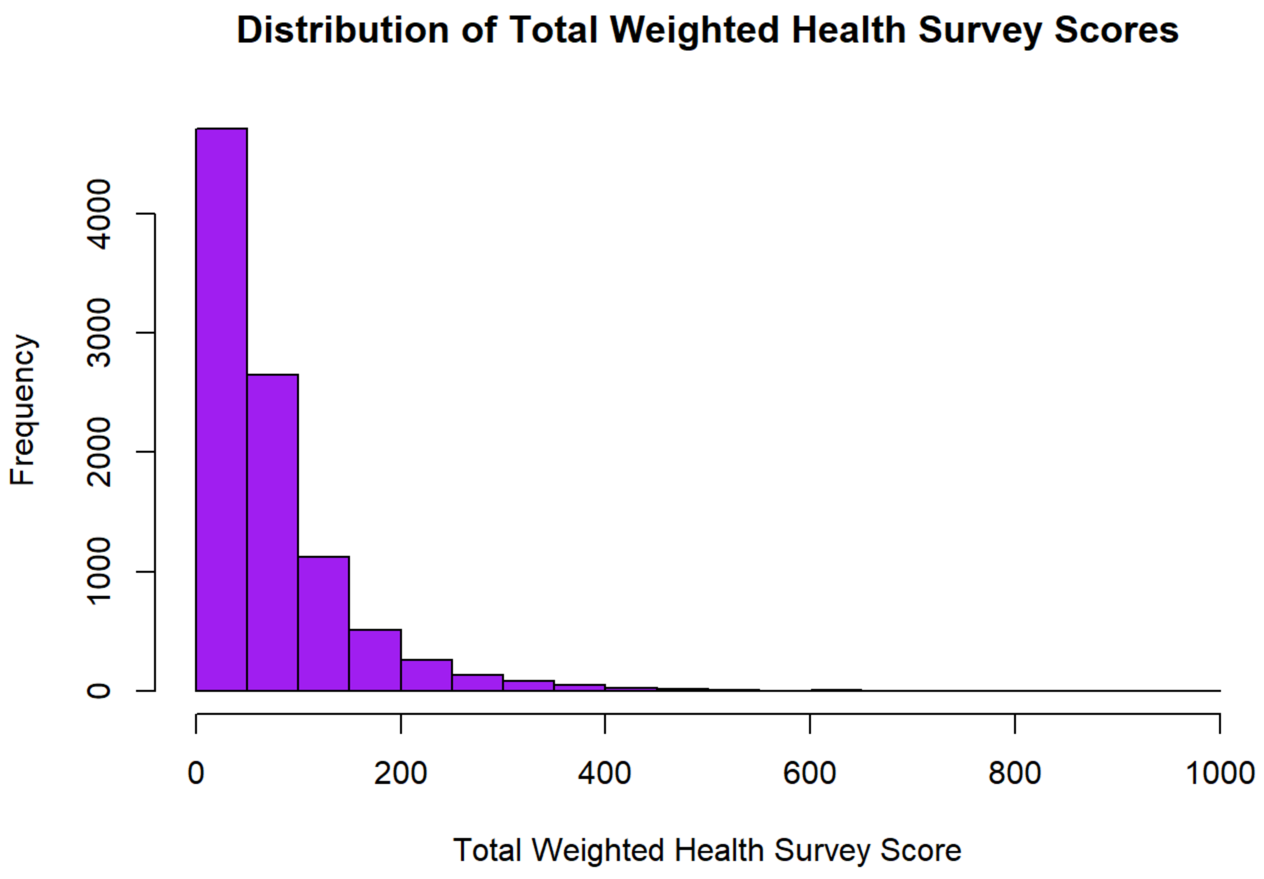
Let’s combine these into a new variable: `occupancy`

```
nursing_mlr_dat <- nursing_data_cleaned |>
  mutate("occupancy" = Average_Number_of_Residents_per_Day/Number_of_Certified_Beds) |>
  select(-c(Average_Number_of_Residents_per_Day, Number_of_Certified_Beds)) # remove pair
```

Transforming the Response

While not necessary for all of our model, for our parametric models that expect the response to be normally distributed, a check for normality is done via a histogram. Clearly, we do not have a normally distributed response.

```
hist(nursing_data_cleaned$Total_Weighted_Health_Survey_Score,
     main = "Distribution of Total Weighted Health Survey Scores",
     xlab = "Total Weighted Health Survey Score",
     ylab = "Frequency",
     col = "purple",
     border = "black")
```



We can use a Box-Cox transformation to find an optimal transformation (resulting in approximately normally distributed data), which is done below:

```
library(MASS)
```

```

Attaching package: 'MASS'

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

    select
```

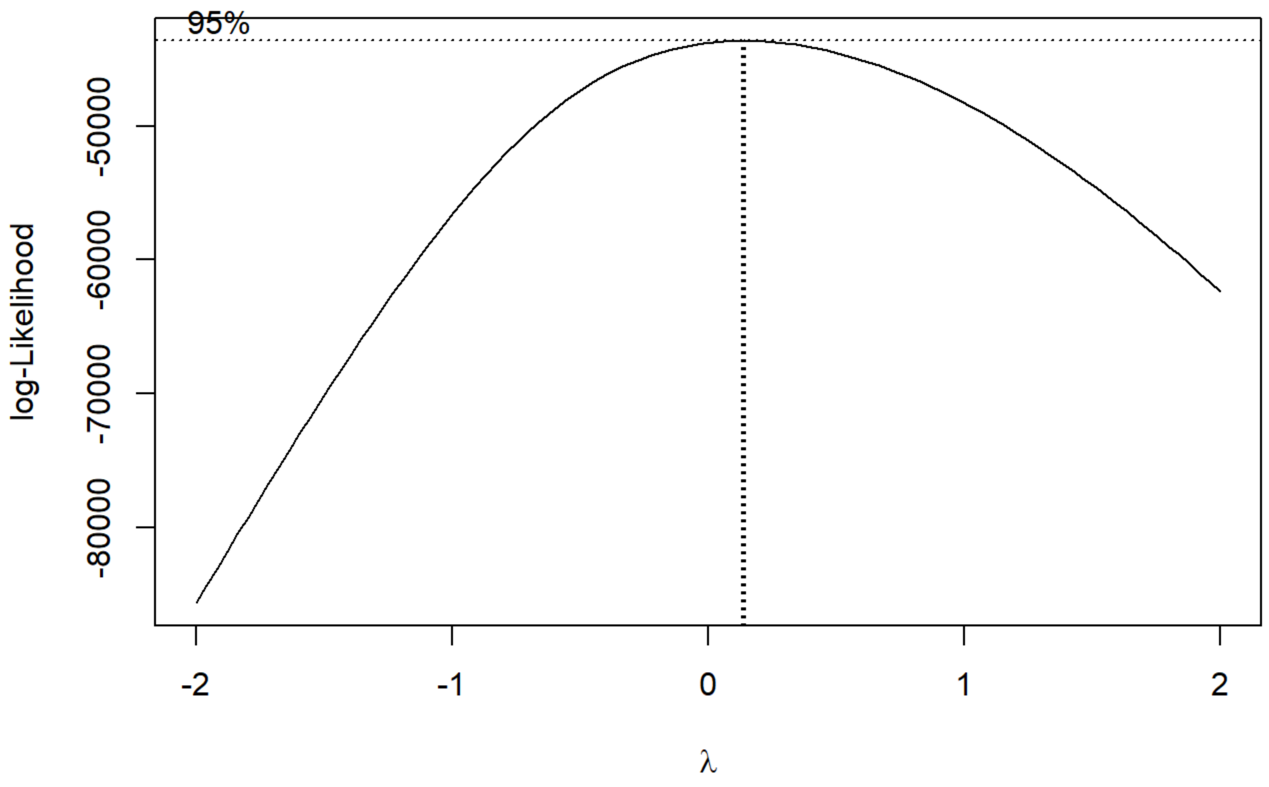
```
# Check if the response variable is > 0 - shift if needed

if(any(nursing_data_cleaned$Total_Weighted_Health_Survey_Score <= 0)) {
  offset <- abs(min(nursing_data_cleaned$Total_Weighted_Health_Survey_Score)) + 1
  nursing_data_cleaned$Total_Weighted_Health_Survey_Score <-
    nursing_data_cleaned$Total_Weighted_Health_Survey_Score + offset
  cat("Response variable adjusted by offset:", offset, "\n")
}
```

```
Response variable adjusted by offset: 1
```

```
model <- lm(Total_Weighted_Health_Survey_Score ~ 1, data = nursing_data_cleaned)

# Box-Cox transformation
boxcox_result <- boxcox(model, lambda = seq(-2, 2, by = 0.1))
```



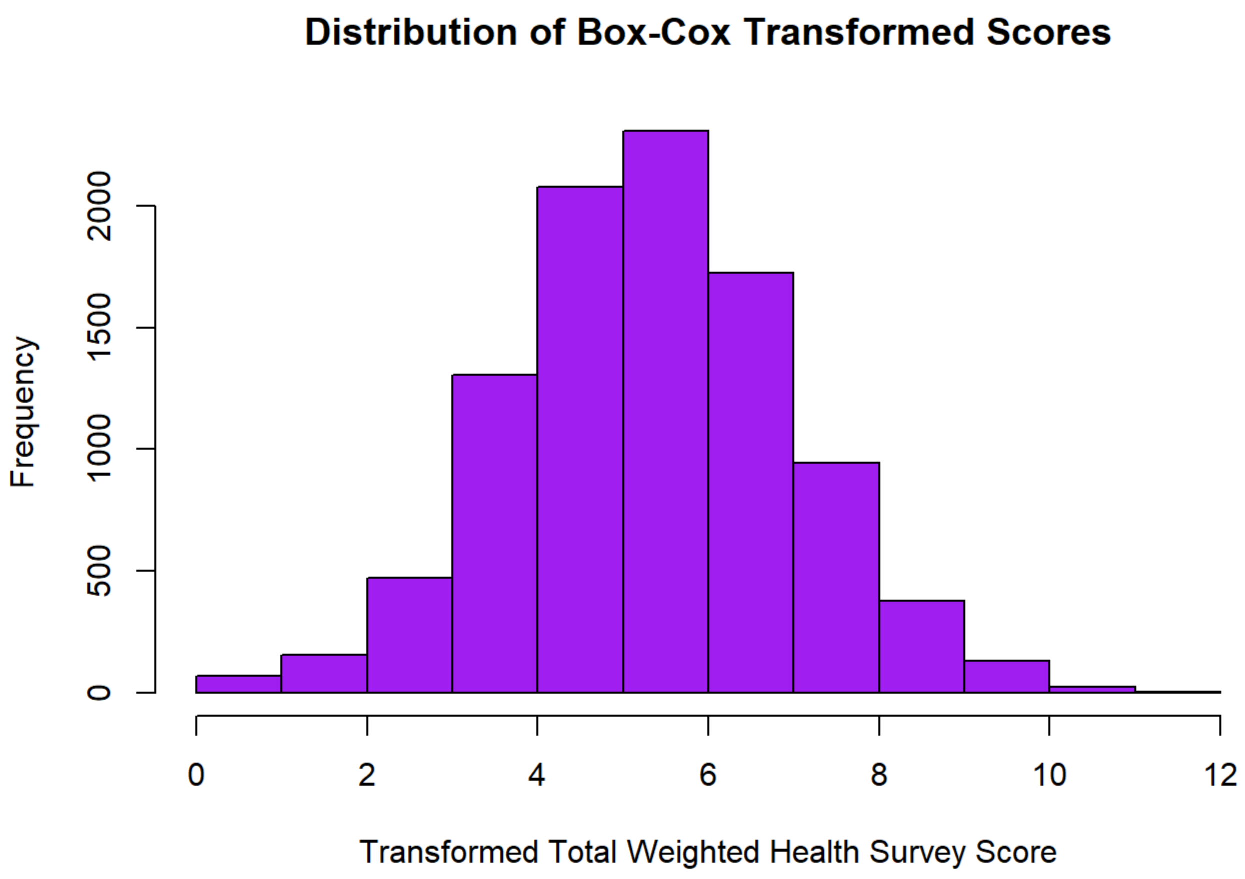
```
best_lambda <- boxcox_result$x[which.max(boxcox_result$y)]
cat("Best lambda:", best_lambda, "\n")
```

```
Best lambda: 0.1414141
```

```
# Apply the Box-Cox transformation using the best lambda (found above).

if (abs(best_lambda) < 1e-6) {
  transformed_data <- log(nursing_data_cleaned$Total_Weighted_Health_Survey_Score)
} else {
  transformed_data <- (nursing_data_cleaned$Total_Weighted_Health_Survey_Score^best_lambda - 1) /
}

# Plot the histogram of the transformed data - looks much better!
hist(transformed_data,
     main = "Distribution of Box-Cox Transformed Scores",
     xlab = "Transformed Total Weighted Health Survey Score",
     col = "purple",
     border = "black")
```



```
print(best_lambda)
```

```
[1] 0.1414141
```

Incorporate the transformation to the data set:

```
nursing_mlr_dat <- nursing_mlr_dat |>
  mutate(Transformed_Health_Survey_Score = (Total_Weighted_Health_Survey_Score^0.14 - 1) / 0.14)
  dplyr::select(-one_of("Total_Weighted_Health_Survey_Score"))
```

Enable Parallel Processing

Enabling parallel processing will improve the runtime of the models fitted later in the document.

```
library(doParallel)
```

```
Warning: package 'doParallel' was built under R version 4.4.3
```

```
Loading required package: foreach


Attaching package: 'foreach'

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

    accumulate, when

Loading required package: iterators

Loading required package: parallel
```

```r
library(foreach)

cl <- makePSOCKcluster(6)  # use 6 cores
registerDoParallel(cl)
```

Modeling

Below we partition our data into training and test sets. These sets will be used to train our models and evaluate them on unseen data, respectively.

```r
library(caret)
```

```
Loading required package: lattice


Attaching package: 'caret'

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

    lift
```

```r
set.seed(456)

in_train <- createDataPartition(nursing_mlr_dat$Transformed_Health_Survey_Score,
                                p = 0.7,
                                list = FALSE)


nursing_train <- nursing_mlr_dat[in_train, ]
nursing_test <- nursing_mlr_dat[-in_train, ]
```

kNN Model

The K-Nearest Neighbors is a regression and classification method that uses the closest k observations to make predictions (using a distance measure - usually Euclidean distance)

The k-Nearest Neighbor method is:

- non-parametric
- has a single tuning parameter, *k*, which specifies the number of nearest observations ('neighbors') off of which predictions are made for a new observation x_0.
- This is not a particularly interpretable model, and we cannot use it for meaningful inferences
- This model does not perform variable selection

The kNN has the advantage of not being overly complex and is easy to use to demonstrate various machine learning ideas.

```r
set.seed(56)



fit_control_knn <- trainControl(method = 'cv',
             number = 10,
             allowParallel = TRUE,
             verboseIter = TRUE)

tune_knn <- expand.grid(k = seq(1, 500, by = 10))

kNN_out <- train(Transformed_Health_Survey_Score ~.,
                 data = nursing_train,
                 method = 'knn',
                 trControl = fit_control_knn,
                 tuneGrid = tune_knn,
                 verbose = TRUE,
                 allowParallel = TRUE
                 )
```

```
Aggregating results
Selecting tuning parameters
Fitting k = 151 on full training set
```

LASSO Model

Least absolute shrinkage and selection operator regression (LASSO) is a regression method that uses L1 regularization in determining the model fit. It has the notable quality of shrinking unimportant predictors toward (and to) zero, allowing for built-in variable selection and added interpretability.

The LASSO model is - parametric - has the tuning parameter λ which controls the level of regularization or shrinkage. - is interpretable, but because of the shrinkage, we cannot perform the same statistical inference as we are able to with regular least squares regression - requires the standardization of predictors

```r
set.seed(90)
train_control_lasso <- trainControl(method = 'cv',
                                    number = 10)
tune_grid_lasso <- expand.grid(alpha = 1, lambda = seq(0.00001, 0.025, length = 1000))



lasso_out <- train(Transformed_Health_Survey_Score ~.,
                   data = nursing_train,
                   preProcess = c("center", "scale"),
                   method = "glmnet",
                   trControl = train_control_lasso,
                   tuneGrid = tune_grid_lasso,
                   allowParallel = TRUE)
print(lasso_out$bestTune)
```

```
    alpha      lambda
140     1 0.003487087
```

Model 3: GAM

The smoothing spline model is a generalized additive model (GAM) whose fit complexity is regularized by the tuning parameter λ, penalizing the curvature in the function (by having a penalty proportional to the integral of the squared second derivative) which allows for the balancing of the flexibility and smoothness of the fit. The chosen λ specifies the effective degrees of freedom.

Smoothing splines - are non-parametric - has the tuning parameter λ (see the paragraph above) - are less interpretable than parametric regression methods but do allow for some inference - in particular with comparing a model with a spline term (or additional spline term) with one that does not include that term. This is like an anova F-test. - do not inherently perform variable selection - do not require the standarization of predictors

Below we fit a smoothing spline model with basis functions applied to a number of the predictors. The `REML` method is used to find the optimal $lambda$ for each spline. `REML` is restricted maximum likelihood (can be used for estimating smoothing params), but it is not the only metric that can be used here.

```r
set.seed(90)

library(mgcv)
```

```
Warning: package 'mgcv' was built under R version 4.4.3

Loading required package: nlme


Attaching package: 'nlme'
```

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

    collapse

This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```r
gam_out <- gam(
  Transformed_Health_Survey_Score ~
    s(median_income) +
    s(unemployment_rate) +
    Health_Inspection_Rating +
    Staffing_Rating +
    s(Number_of_Facility_Reported_Incidents) +
    s(Number_of_Substantiated_Complaints) +
    s(Total_Amount_of_Fines_in_Dollars) +
    Number_of_Payment_Denials +
    factor(Provider_Type) +
    factor(Abuse_Icon) +
    factor(Continuing_Care_Retirement_Community),
  data = nursing_train,
  method = "REML"  #example method - REML is a placeholder here
)

print(gam_out$sp)
```

```
                       s(median_income)
                             5.54732077
                    s(unemployment_rate)
                             0.90087741
s(Number_of_Facility_Reported_Incidents)
                             0.15338638
   s(Number_of_Substantiated_Complaints)
                             0.00915701
     s(Total_Amount_of_Fines_in_Dollars)
                             0.27795709
```

Single Tree Model (Regression)

Tree-based methods split the predictor space into regions, and within each region the predicted value of the response is the average (value that minimizes the residual sum of squares) of the training data. With multiple predictors, as is the case in the fitted model below, we use recursive binary splitting to fit the model so that we can circumvent the impossible task of considering all possible partitions of the predictor space. Recursive binary splitting involves searching the training set for the predictor and split point that creates the two initial regions (first split) minimizing the total sum of squares - this process is then repeated until some stopping criterion is met. A single tree provides the foundation of ensemble tree methods.

A single regression tree

- Is a non-parametric method
- Consists of the tuning parameter `cp`, or cost complexity, which controls the trade off between the complexity of sub-trees and their fit to the data.
- Is a highly interpretable method but is not generally best for inference in the statistical sense (hypothesis testing, confidence intervals, etc.)
- Performs variable selection in the sense that we can look at certain measures of variable importance (Gini Importance, for example) and see what variables were used for splits (some may not have been used, or were hardly used)
- does not require the standardization of predictors

```
set.seed(58)

train_control_stree <- trainControl(method = "cv",
                                    number = 5)

single_tree_out <- train(Transformed_Health_Survey_Score ~ .,
                  data = nursing_train,
                  method = "rpart",
                  tuneLength = 50, # number of cp values considered
                  trControl = train_control_stree)
```

```
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
: There were missing values in resampled performance measures.
```

```
print(single_tree_out$bestTune)
```

```
         cp
39 0.003794
```

Random Forest Model

The Random Forest model is a special case of a bagged tree where, for each split of each tree, a random subset of the predictors is used rather than all predictors. This subsetting helps the model train on less influential predictors - including all models with each tree means that the most important predictors will dominate those that are less important, and thus we lose out on some of the signal in these less influential predictors.

The random forest

- Is a non-parametric method
- Has the same tuning parameters as a bagged tree (minimum size of the terminal nodes, the number of trees, etc.), but most importantly it requires a natural number for the number of subsetted variables used to fit each tree (`mtry`) (at each split). We will use the default bagging hyperparameter values but will allow the value for the subet `mtry` to vary.
- Is not particularly useful when the objective is inference due to a lack of interpretability (though there is some research positing statistical methods for evaluating variable importance in models like the

random forest: https://pmc.ncbi.nlm.nih.gov/articles/PMC6428414/) Thank you Dr. Post for the link!)
- does perform variable selection in the sense that we get an idea of the important variables in the model (think Gini importance or some other measure) and those that are unimportant (and not used)
- does not require the standardization of the predictors.

```
train_control_rf <- trainControl(
  method = "cv",
  number = 5,
  allowParallel = TRUE,
  verboseIter = TRUE
)

tune_grid_rf <- expand.grid(
  mtry = 1:(ncol(nursing_train) - 1) # number to be subsetted
)


rf_out <- train(
  Transformed_Health_Survey_Score ~ .,
  data = nursing_train,
  method = "rf",
  trControl = train_control_rf,
  tuneGrid = tune_grid_rf
)
```

```
Aggregating results
Selecting tuning parameters
Fitting mtry = 3 on full training set
```

```
print(rf_out$bestTune)
```

```
  mtry
3    3
```

Support Vector Machine

The support vector machine (SVM) is an extension of the support vector classifier that enlarges the predictor space via some family of kernel functions.

- SVMs will be nonparametric when the number of parameters depends on the size of the training set; this is the case with the radial basis function, which is the kernel method chosen for this model.
- The SVM using the RBF kernel has two parameters: 1). The regularization parameter (controlling the flexibility of the decision boundary) C and the kernel coefficient σ, which controls the influence of the training observations on the model.
- SVMs are generally not as well-suited to inferential questions as they are to prediction
- SVMs do not inherently perform feature selection

- Because SVM kernels are calculated using some kind of distance measure, the predictors must be scaled.

Below we use ten combinations of the hyperparameters along with 5-fold cross validation to select our best SVM model:

```
library(kernlab)
```

```

Attaching package: 'kernlab'

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

    cross

The following object is masked from 'package:ggplot2':

    alpha
```

```
svm_out <- train(
  Transformed_Health_Survey_Score ~ .,
  data = nursing_train,
  method = "svmRadial",  # Radial basis function kernel
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv",
                           number = 5),
  tuneLength = 10 # 10 combinations of the tuning params
)
print(svm_out$bestTune)
```

```
       sigma C
4 0.08503423 2
```

Results

Below we evaluate our models on the test set using `rmse`, or root mean square error as our metric.

```
knn_preds <- predict(kNN_out, nursing_test)
lasso_preds <- predict(lasso_out, nursing_test)
single_tree_preds <- predict(single_tree_out, nursing_test)
rf_preds <- predict(rf_out, nursing_test)
svm_preds <- predict(svm_out, nursing_test)


# [1] to grab just rmse
knn_rmse <- postResample(pred = knn_preds,
                         obs = nursing_test$Transformed_Health_Survey_Score)[1]
lasso_rmse <- postResample(pred = lasso_preds,
                           obs = nursing_test$Transformed_Health_Survey_Score)[1]
single_tree_rmse <- postResample(pred = single_tree_preds,
                                 obs = nursing_test$Transformed_Health_Survey_Score)[1]
rf_rmse <- postResample(pred = rf_preds,
                        obs = nursing_test$Transformed_Health_Survey_Score)[1]
svm_rmse <- postResample(pred = svm_preds,
                         obs = nursing_test$Transformed_Health_Survey_Score)[1]




result_list <- list(
  "The optimal kNN model" = knn_rmse,
  "The optimal LASSO model" = lasso_rmse,
  "The optimal single Tree" = single_tree_rmse,
  "The optimal Random Forest model" = rf_rmse,
  "The optimal SVM model" = svm_rmse
)

# loop that prints results

for (name in names(result_list)) {
  print(paste("The", name, "rmse is", result_list[[name]]))
}
```

```
[1] "The The optimal kNN model rmse is 1.48086819265042"
[1] "The The optimal LASSO model rmse is 1.03435783750333"
[1] "The The optimal single Tree rmse is 1.01394944167336"
[1] "The The optimal Random Forest model rmse is 0.962617711171913"
[1] "The The optimal SVM model rmse is 1.01115983844883"
```

```
best_mod <- min(unlist(result_list))
```

The model that performed best on the test set was the random forest model. This is the model I would choose if I wanted to ensure that I had the most accurate predictions. I also, however, like the LASSO model that I constructed. It seems to perform reasonably well in a predictive sense, but it also gives me insight into the variables that mattered in forming those predictions. For this latter reason, I'll investigate this model a little further after refitting to the entire data set.

Refit to Full Data Set

For computational savings and because I preferred the interpretability of the LASSO model over the random forest, I refit the LASSO model to the entire data set.

```
lasso_full_out <- train(Transformed_Health_Survey_Score ~ .,
                        data = nursing_mlr_dat,
                        preProcess = c("center", "scale"),
                        method = "glmnet",
                        trControl = trainControl(method = "none"),  # no cross-validation
```

```
                           tuneGrid = expand.grid(alpha = 1, lambda = best_lambda),
                           allowParallel = TRUE)
```

We can also now stop the parallel processing:

```
stopCluster(cl) # stop parallel
```

From the coefficient traceplot, we can see how the different predictors were shrunk.

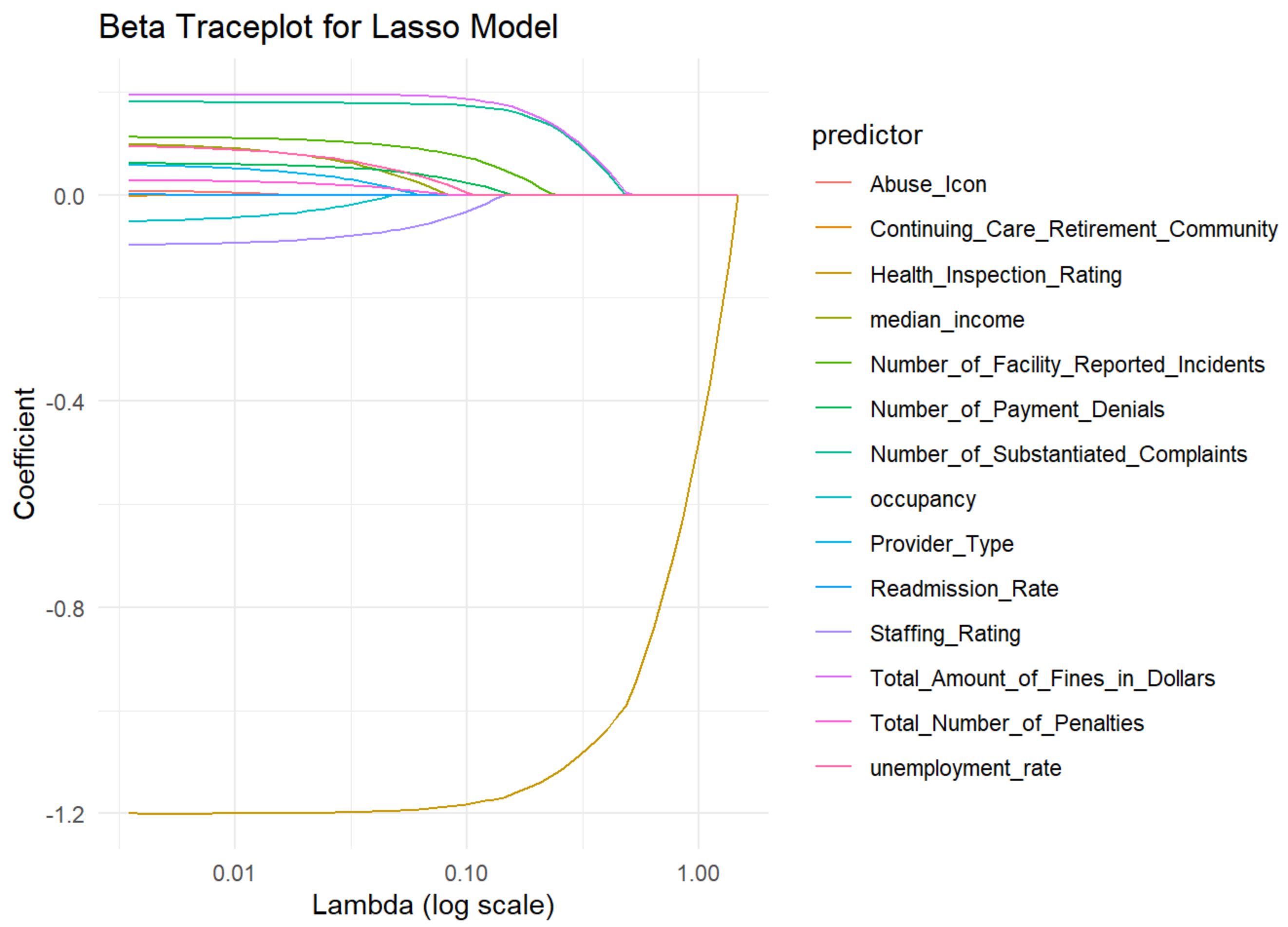
```
final_model <- lasso_full_out$finalModel
lambda_seq <- final_model$lambda

coefs <- as.matrix(final_model$beta)

coefs_df <- data.frame(lambda = lambda_seq, t(coefs))


coefs_long <- pivot_longer(coefs_df,
                           cols = -lambda,
                           names_to = "predictor",
                           values_to = "coefficient")


ggplot(coefs_long, aes(x = lambda, y = coefficient, color = predictor)) +
  geom_line() +
  scale_x_log10() +  # log-scale preferable
  labs(title = "Beta Traceplot for Lasso Model",
       x = "Lambda (log scale)",
       y = "Coefficient") +
  theme_minimal()
```



The beta coefficient are also printed out:

```
optimal_lambda <- lasso_full_out$bestTune$lambda

lasso_summary <- coef(lasso_full_out$finalModel, s = optimal_lambda)


print(lasso_summary)
```

```
15 x 1 sparse Matrix of class "dgCMatrix"
                                                s1
(Intercept)                            5.235566606
Readmission_Rate                       .
median_income                          .
unemployment_rate                      .
Provider_Type                          .
Continuing_Care_Retirement_Community   .
Abuse_Icon                             .
Health_Inspection_Rating              -1.168986189
Staffing_Rating                       -0.002090015
Number_of_Facility_Reported_Incidents  0.050225727
Number_of_Substantiated_Complaints     0.166615618
Total_Amount_of_Fines_in_Dollars       0.175913987
Number_of_Payment_Denials              0.006028798
```

```
Total_Number_of_Penalties             .
occupancy                             .
```

```
beta_df <- data.frame(
  predictor = rownames(lasso_summary),
  coefficient = as.vector(lasso_summary)
)
print(beta_df)
```

```
                             predictor  coefficient
1                          (Intercept)  5.235566606
2                     Readmission_Rate  0.000000000
3                        median_income  0.000000000
4                    unemployment_rate  0.000000000
5                        Provider_Type  0.000000000
6   Continuing_Care_Retirement_Community  0.000000000
7                           Abuse_Icon  0.000000000
8              Health_Inspection_Rating -1.168986189
9                       Staffing_Rating -0.002090015
10 Number_of_Facility_Reported_Incidents  0.050225727
11    Number_of_Substantiated_Complaints  0.166615618
12      Total_Amount_of_Fines_in_Dollars  0.175913987
13            Number_of_Payment_Denials  0.006028798
14            Total_Number_of_Penalties  0.000000000
15                            occupancy  0.000000000
```

From the traceplot and the regularized β values, we can see that variables like `Health_Inspection_Rating`, `Number_of_Facility_Reported_Incidents`, `Number_of_Substantiated_Complaints`, and `Total_Amount_of_Fines_in_Dollars` are important (not as much shrinkage), while a number of the others were shrunk to effectively zero or near zero, like `Readmission_Rate`.