Demo 1: Biomedical & Clinical Informatics

Merrimack College DSE6630: Healthcare & Life Sciences Analytics

Team Gamma, Collins Njagi

12 May 2025

Contents

Introduction

You have just been brought on as a data science consultant for a patient advocacy watchdog. They have asked you to source information about hospital readmissions, because they would like to flag specific geographic areas or types of hospitals where hospital readmission is unexpectedly high. They have heard that datasets exist to investigate these types of questions, but they are not really sure where to begin.

You remember this one data science class you took - and the dataset from the **Centers for Medicare & Medicaid Services** managed by Medicare.gov. You start poking around the Hospitals dataset and decide it is exactly what you need.

The problem is, this dataset is massive, so you are not only tasked with identifying a reasonable question you can ask from the dataset but managing the cleaning, pre-processing, and exploratory data analysis on the merged datasets you choose to explore.

Question

For **Project 1**, you are going to choose your own question! But for this demo analysis, we are focusing on **pneumonia-related hospital readmissions**.

Why pneumonia-related readmissions?

Hospital readmission rates for patients with pneumonia is almost unbelievably high, with about $\frac{1}{5}$ or 20% of patients hospitalized with pneumonia re-admitted within 30 days (de Alba and Amin, 2014). Further, pneumonia is both a leading cause of death among the elderly as well as a commonly communicated disease in hospitals (as in, people may come into a hospital for a different condition but be readmitted for pneumonia).

Thus, given that pneumonia is the only communicable disease on this list, it is serving as our best proxy for diseases spread person-to-person within hospitals because of lack of availability.

Our research question: Can we predict pneumonia-related hospital readmissions and, more generally, disease transfer within hospitals, based on hospital-level characteristics, including patient-ratings, Medicare money received, and other indicators of general performance in disease diagnosis and prevention?

Our hypothesis & prediction: Hospitals are, by their very nature, hot zones for increased spread of disease, including pneumonia. We hypothesize that hospitals with lower indicators of patient care performance and lower Medicare funding rates will generally be unable to provide sufficient control of specific disease diagnosis and prevention, like pneumonia. This is because the same intrinsic and extrinsic factors that impact general patient quality of care is likely to extend to pneumonia, a *common, preventable, and treatable* disease. Thus, we specifically predict that factors indicating lower quality-of-care will predict higher than average rates of pneumonia-related hospital readmissions.

Objective

Our objective for the patient advocacy watchdog is clean, merge, explore, and predict which hospitals tend to have higher-than-average rates of pneumonia-related readmissions. This is to enable the watchdog group to put together a dashboard or otherwise deploy this information to share with patients so they know how their local hospitals are performing, allowing them to make healthier, more informed choices.

Data & Data Cleaning

1. Download the Hospitals dataset as a zip directory.
2. Unzip the directory.
3. Make sure to adjust the name and/or path of the directory in your directory if it is different from mine. *Note that mine is living on my Desktop.*

WARNING! Do not try to put these data onto GitHub. They are too large to push. If you need help adjusting your path or have questions, please let me know.

```
## Make sure to adjust your path!
filepath <- ("C:/data_science/DSE6630OM_SU2025S1/week1_Biomedical and Clinical Informatics, Part 1/hosp

  #"~/Desktop/school/Healthcare/Week1/Demo1_NotGitHub/hospitals_current_data/"
```

Read in the hospital-level data iteratively, while dynamically naming & storing each file as a dataframe.

You may have to dig through the accompanying data dictionary **HOSPITAL_Data_Dictionary.pdf** to get a better handle on what is in the hospital-level files.

Question 1: [1 point] Take a look at the chunk below, which performs **dynamic variable assignment** as it reads in each of the files. Your task is to add a comment to each line of code to describe what that line does.

Your answer in the code chunk as comments. Make sure to use the # sign!

```
# Greg A.

files <- list.files(path = filepath, pattern = "Hospital.csv") # Creates a variable which stores the na

for(f in 1:length(files)) { # Starts a for loop, for each entry in the files list
    dat <- clean_names(read_csv(paste0(filepath, files[f]), # Clean each column name in the file, start
```

```
                                show_col_types = FALSE), # Do not print out the names as they are read
                        case = "upper_camel")# Apply upper camel casing to each column name
    filename <- gsub(".Hospital\\.csv", "", files[f]) # Remove .Hospital.csv from each file name ##(".H
    assign(filename, dat) # Assign the cleaned name to the list
} # End the for loop

files <- gsub(".Hospital\\.csv", "", files) %>% data.frame() # Remove .Hospital.csv from each file name
names(files) <- "File Name" # Give the column a name of "File Name" to clarify what we are looking at
```

Now, let's look at a list of the files that we've read in: Table 1. List of hospital-level data files.

File Name
Complications_and_Deaths
FY_2025_HAC_Reduction_Program
FY_2025_Hospital_Readmissions_Reduction_Program
HCAHPS
Health_Equity
Healthcare_Associated_Infections
Maternal_Health
Medicare_Hospital_Spending_Per_Patient
Outpatient_Imaging_Efficiency
Payment_and_Value_of_Care
Promoting_Interoperability
Timely_and_Effective_Care
Unplanned_Hospital_Visits

Question 2: **[1 point]** Take a look at some of the dataframes, e.g., `HCAHPS` and `FY_2024_Hospital_Readmissions_Reductio` What do you notice about the format / structure of the data? Does the data have an issue with the dimensionality? **Hint:** Check to see if each row represents a single line of data or not!

```
# Greg A

head(HCAHPS) # Look at the first 6 rows of the dataframe
```

```
## # A tibble: 6 x 22
##   FacilityId FacilityName            Address CityTown State ZipCode CountyParish
##   <chr>      <chr>                   <chr>   <chr>    <chr> <chr>   <chr>
## 1 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## 2 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## 3 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## 4 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## 5 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## 6 010001     SOUTHEAST HEALTH MEDIC~ 1108 R~ DOTHAN   AL    36301   HOUSTON
## # i 15 more variables: TelephoneNumber <chr>, HcahpsMeasureId <chr>,
## #   HcahpsQuestion <chr>, HcahpsAnswerDescription <chr>,
## #   PatientSurveyStarRating <chr>, PatientSurveyStarRatingFootnote <dbl>,
## #   HcahpsAnswerPercent <chr>, HcahpsAnswerPercentFootnote <chr>,
## #   HcahpsLinearMeanValue <chr>, NumberOfCompletedSurveys <chr>,
```

```
## #   NumberOfCompletedSurveysFootnote <chr>, SurveyResponseRatePercent <chr>,
## #   SurveyResponseRatePercentFootnote <chr>, StartDate <chr>, EndDate <chr>
```

```
head(FY_2025_Hospital_Readmissions_Reduction_Program) # Look at the first 6 rows of the dataframe
```

```
## # A tibble: 6 x 12
##   FacilityName          FacilityId State MeasureName NumberOfDischarges Footnote
##   <chr>                 <chr>      <chr> <chr>       <chr>                 <dbl>
## 1 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-A~ 296                      NA
## 2 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-C~ 151                      NA
## 3 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-H~ 681                      NA
## 4 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-H~ N/A                      NA
## 5 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-P~ 490                      NA
## 6 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-C~ 130                      NA
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

> Greg A. The data appears to be in long format, as opposed to the desired wide format. This means that each row does not represent a single hospital. Each row instead represents a single measure from a hospital.

Tidy the data response variables' dataframe: Fiscal Year 2024 Hospital Readmissions Reduction Program.

We are going to focus on the dataset `FY_2024_Hospital_Readmissions_Reduction_Program` to work on best practices to tidy our data. Although I do not personally choose to follow everything that Hadley Wickham and the **Tidy Data** Movement has espoused, I do think a lot of the best practices are useful principles especially when we are just getting use to working with data. Further, we can leverage a lot of the functions to make our lives MUCH easier! And who doesn't like easier?!

Question 3: [**1 point**] Take a deeper look at `FY_2024_Hospital_Readmissions_Reduction_Program` at some of the numeric variables in the dataset, e.g., `NumberOfReadmissions`?

```
#Greg A.

dim(FY_2025_Hospital_Readmissions_Reduction_Program) # Look at the dimensions of the dataframe
```

```
## [1] 18510    12
```

```
summary(FY_2025_Hospital_Readmissions_Reduction_Program) # Look at the summary statistics of the datafr
```

```
##  FacilityName        FacilityId           State           MeasureName
##  Length:18510       Length:18510       Length:18510       Length:18510
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  NumberOfDischarges    Footnote     ExcessReadmissionRatio
##  Length:18510       Min.   :1.000   Length:18510
##  Class :character   1st Qu.:1.000   Class :character
##  Mode  :character   Median :5.000   Mode  :character
##                     Mean   :3.188
##                     3rd Qu.:5.000
##                     Max.   :7.000
```

```
##                     NA's   :11927
##  PredictedReadmissionRate ExpectedReadmissionRate NumberOfReadmissions
##  Length:18510             Length:18510            Length:18510
##  Class :character         Class :character        Class :character
##  Mode  :character         Mode  :character        Mode  :character
##
##
##
##
##   StartDate           EndDate
##  Length:18510       Length:18510
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
head(FY_2025_Hospital_Readmissions_Reduction_Program) # Look at the first 6 rows of the dataframe
```

```
## # A tibble: 6 x 12
##   FacilityName          FacilityId State MeasureName NumberOfDischarges Footnote
##   <chr>                 <chr>      <chr> <chr>       <chr>                 <dbl>
## 1 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-A~ 296                      NA
## 2 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-C~ 151                      NA
## 3 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-H~ 681                      NA
## 4 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-H~ N/A                      NA
## 5 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-P~ 490                      NA
## 6 SOUTHEAST HEALTH MED~ 010001     AL    READM-30-C~ 130                      NA
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

```
# Greg A.

FY_2025_Hospital_Readmissions_Reduction_Program # Print the entire dataframe
```

```
## # A tibble: 18,510 x 12
##    FacilityName         FacilityId State MeasureName NumberOfDischarges Footnote
##    <chr>                <chr>      <chr> <chr>       <chr>                 <dbl>
##  1 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-A~ 296                      NA
##  2 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 151                      NA
##  3 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ 681                      NA
##  4 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ N/A                      NA
##  5 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-P~ 490                      NA
##  6 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 130                      NA
##  7 MARSHALL MEDICAL CE~ 010005     AL    READM-30-C~ N/A                       5
##  8 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ N/A                      NA
##  9 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ 176                      NA
## 10 MARSHALL MEDICAL CE~ 010005     AL    READM-30-P~ 305                      NA
## # i 18,500 more rows
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

Greg A. What should be numerical columns feature NA values written out as character data

instead of true NA values. In addition, there is a character value of 'Too Few to Report'.

Hint: Columns 5 & 7:10 have problems with the `NA` class.

Question 4: [**1 point**] Investigate the `replace_with_na_all()` function that is part of the `naniar` package. Use this function to fix the `NA` class in the aberrant columns in `FY_2024_Hospital_Readmissions_Reduction_Program`.

```
# Greg A.

?replace_with_na_all # Pull up documentation for replace_with_na_all function
FY_2025_Hospital_Readmissions_Reduction_Program <- replace_with_na_all( # Apply the replace_with_na_all
  FY_2025_Hospital_Readmissions_Reduction_Program, condition = ~.x == "N/A") # fill NA values where the

FY_2025_Hospital_Readmissions_Reduction_Program # Print the data so we can verify changes
```

```
## # A tibble: 18,510 x 12
##    FacilityName         FacilityId State MeasureName NumberOfDischarges Footnote
##    <chr>                <chr>      <chr> <chr>       <chr>                 <dbl>
##  1 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-A~ 296                      NA
##  2 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 151                      NA
##  3 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ 681                      NA
##  4 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ <NA>                     NA
##  5 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-P~ 490                      NA
##  6 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 130                      NA
##  7 MARSHALL MEDICAL CE~ 010005     AL    READM-30-C~ <NA>                      5
##  8 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ <NA>                     NA
##  9 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ 176                      NA
## 10 MARSHALL MEDICAL CE~ 010005     AL    READM-30-P~ 305                      NA
## # i 18,500 more rows
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

You will notice that this aberrant class of missing value **caused** the numeric columns to be coerced into character columns when they were imported by `read_csv()`. Although we could try using `as.numeric()` by itself, which would assign `NA` to any non-numeric entry in `FY_2024_Hospital_Readmissions_Reduction_Program`, the problem is that we could lose important information that way. Thus, as annoying as this is, we should fix it a little more specifically...

Question 5: [**1 point**] Look more closely at the `NumberOfReadmissions` column in the `FY_2024_Hospital_Readmissions_R` What other issue do you see with this specific column that is coercing it to the character type, and why does it exist?

Hint: Make sure to use the accompanying data dictionary **HOSPITAL_Data_Dictionary.pdf** to also examine **WHY** this problem exists!

> Greg A. As noted earlier, there is another character entry of 'Too Few to Report'. This is causing the should-be numeric columns to be considered character. Looking at the data dictionary, and remembering that each row represents a single measure, we can see why this kind of entry is made. According to the data dictionary, this message is added for the following reasons: 1. When the number of cases is too few for public reporting 2. When the number of cases is too small to reliably tell performance 3. To protect personal health information

Question 6: [**1 point**] Investigate the `gsub()` function that is part of base `R` or any other function of your choosing. Use the function you choose to **replace** the problematic text in the `NumberOfReadmissions` column in `FY_2024_Hospital_Readmissions_Reduction_Program` with a **5**.

```
?gsub # Pull up documentation for gsub function
FY_2025_Hospital_Readmissions_Reduction_Program$NumberOfReadmissions <- gsub( # Apply the gsub function
  'Too Few to Report', 5, FY_2025_Hospital_Readmissions_Reduction_Program$NumberOfReadmissions) # Repla

FY_2025_Hospital_Readmissions_Reduction_Program # Print the data so we can verify changes
```

```
## # A tibble: 18,510 x 12
##    FacilityName         FacilityId State MeasureName NumberOfDischarges Footnote
##    <chr>                <chr>      <chr> <chr>       <chr>                 <dbl>
##  1 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-A~ 296                      NA
##  2 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 151                      NA
##  3 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ 681                      NA
##  4 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ <NA>                     NA
##  5 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-P~ 490                      NA
##  6 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 130                      NA
##  7 MARSHALL MEDICAL CE~ 010005     AL    READM-30-C~ <NA>                      5
##  8 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ <NA>                     NA
##  9 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ 176                      NA
## 10 MARSHALL MEDICAL CE~ 010005     AL    READM-30-P~ 305                      NA
## # i 18,500 more rows
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

Question 7: [**1 point**] Speculate as to why I asked you to replace the text with a **5** rather than a **0** or an `NA`. Can you think of any other logical choices for the substitution? Do you agree with my choice? (You're free to disagree, but please defend your decision either way!)

> Greg A. One reason to why a 5 was selected was so that we can later identify the value. Since 5 is a low value outside of the normal value range, it will not be confused for a legitimate entry. A 0 or NA might be confused as there being no value, which would cause a loss of information. Beyond a five, any small integer value above 0 could work, so long as it is outside of the normal value range.

Question 8: [**1 point**] Now, imagine an alternative scenario where instead of replacing it with a **5**, I instead asked you to replace it with a randomly sampled integer from 1 to 10. What would this do? Would `gsub()` work here or would I need to execute it a different way, e.g., with an `lapply()` function or a `for()` loop?

> Greg A. gsub() would not work for this application, because it expects a single value. Since it expects a single value, we are unable to apply a randomly sampled integer.

Question 9: [**3 points**] You probably anticipated this! We just decided that, rather than a **5** we want a randomly sampled integer from 1 to 10.

Hint 1: You might find it easier to read the data in from fresh and, after commenting out your `gsub` from Question 5, pattern match on the problematic text.

Hint 2: This is NOT the only way to do this, but I solved this using `gregexpr()`, `regmatches()`, and `lapply()`.

```
# Greg A.
# Before running this code, I reran the previous code, excluding the gsub chunk, as suggested.

FY_2025_Hospital_Readmissions_Reduction_Program$NumberOfReadmissions <- unlist(lapply( # Use lapply to
    FY_2025_Hospital_Readmissions_Reduction_Program$NumberOfReadmissions, function(x) { # For each row,
```

```
  if (is.na(x)) {x} # If the value is NA, return the original value
  else if (x == "Too Few to Report") {sample(1:10, 1)} # If the value is "Too Few to Report", replace i
  else {x} # If neither, return the original value
}))

FY_2025_Hospital_Readmissions_Reduction_Program # Print the data so we can verify changes
```

```
## # A tibble: 18,510 x 12
##    FacilityName         FacilityId State MeasureName NumberOfDischarges Footnote
##    <chr>                <chr>      <chr> <chr>       <chr>                 <dbl>
##  1 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-A~ 296                      NA
##  2 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 151                      NA
##  3 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ 681                      NA
##  4 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~ <NA>                     NA
##  5 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-P~ 490                      NA
##  6 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~ 130                      NA
##  7 MARSHALL MEDICAL CE~ 010005     AL    READM-30-C~ <NA>                      5
##  8 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ <NA>                     NA
##  9 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~ 176                      NA
## 10 MARSHALL MEDICAL CE~ 010005     AL    READM-30-P~ 305                      NA
## # i 18,500 more rows
## # i 6 more variables: ExcessReadmissionRatio <chr>,
## #   PredictedReadmissionRate <chr>, ExpectedReadmissionRate <chr>,
## #   NumberOfReadmissions <chr>, StartDate <chr>, EndDate <chr>
```

Question 10: [**1 point**] It is finally time to fix those aberrant character columns by making them numeric. However, just using `as.numeric()` would require us to do it over and over on each of the columns in multiple lines of code. Instead, investigate the function `mutate_at()` which will allow you to pass it a list of columns you want to convert with the function you want to use.

```
# Greg A.

?mutate_at # Pull up documentation for mutate_at function
FY_2025_Hospital_Readmissions_Reduction_Program <- FY_2025_Hospital_Readmissions_Reduction_Program %>% 
  mutate_at(vars(NumberOfDischarges, ExcessReadmissionRatio, PredictedReadmissionRate, # Alter the foll
                 ExpectedReadmissionRate, NumberOfReadmissions), as.numeric) # Set the columns as numer

FY_2025_Hospital_Readmissions_Reduction_Program # Print the data so we can verify changes
```

```
## # A tibble: 18,510 x 12
##    FacilityName         FacilityId State MeasureName NumberOfDischarges Footnote
##    <chr>                <chr>      <chr> <chr>                   <dbl>    <dbl>
##  1 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-A~               296       NA
##  2 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~               151       NA
##  3 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~               681       NA
##  4 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-H~                NA       NA
##  5 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-P~               490       NA
##  6 SOUTHEAST HEALTH ME~ 010001     AL    READM-30-C~               130       NA
##  7 MARSHALL MEDICAL CE~ 010005     AL    READM-30-C~                NA        5
##  8 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~                NA       NA
##  9 MARSHALL MEDICAL CE~ 010005     AL    READM-30-H~               176       NA
## 10 MARSHALL MEDICAL CE~ 010005     AL    READM-30-P~               305       NA
## # i 18,500 more rows
## # i 6 more variables: ExcessReadmissionRatio <dbl>,
```

```
## #   PredictedReadmissionRate <dbl>, ExpectedReadmissionRate <dbl>,
## #   NumberOfReadmissions <dbl>, StartDate <chr>, EndDate <chr>
```

Let's look more closely at Measure Names. Notice, that what I have done here first is used the `gsub()` function to `mutate()` the `MeasureName` column to remove the text that flanked the medical conditions.

```
FY_2025_Hospital_Readmissions_Reduction_Program <-  FY_2025_Hospital_Readmissions_Reduction_Program %>%
  mutate(MeasureName = gsub("READM-30-", "", MeasureName)) %>%
  mutate(MeasureName = gsub("-HRRP", "", MeasureName))
```

Note: If at this point you've had any issues with any of the previous problems 2-10, you can load the cleaned readmissions data, `readmissionsClean.Rdata` so that you can proceed with the assignment.

```
load(file = "readmissionsClean.Rdata")
```

Question 11: [1 point] Investigate the function `pivot_wider()` or `spread()` from the `tidyr` package, which comes bundled with `tidyverse`. Try pivoting the `FY_2024_Hospital_Readmissions_Reduction_Program` dataset wider. Make sure you correctly identify the `names_from` and `values_from` column(s). **Make sure to save it as a separate dataframe called `wideDF`.** Use the `dim()` function to prove to me that you successfully pivoted wider.

```
# Your code here.
#Collins Njagi

?pivot_wider #to check the documentation of pivot_wider


wideDF <- FY_2025_Hospital_Readmissions_Reduction_Program %>%
  pivot_wider(names_from = MeasureName,values_from = c(NumberOfDischarges, Footnote,
                                                       ExcessReadmissionRatio, PredictedReadmissionRate
                                                       ExpectedReadmissionRate, NumberOfReadmissions))

dim(wideDF)
```

```
## [1] 3085   41
```

Filtering for specific conditions.

Ideally, though, before we'd pivot wider we would decide if we were going to filter for any specific hospital readmission conditions. For your **Project 1**, you will be choosing which medical condition(s) you want to focus on for predicting hospital readmission. However, for the remainder of this demo, we're going to focus on just **pneumonia**.

Ideas you might consider for your Project_1 are:

- surgical interventions (`HIP-KNEE`, `CABG`)
- heart-related conditions (`HF`, `AMI`, & `CABG`)
- all conditions (warning: this might be too unwieldy!)
- any other condition(s) that you motivate with a statement or two in support

Therefore, let's filter for just pneumonia-related readmissions which will actually eliminate the need to pivot wider in just this case (because we only chose a single condition).

```
readmissionsClean <-
  readmissionsClean %>%
  filter(MeasureName == "PN")
```

Table 2. Acronyms of medical conditions for which hospital readmissions are tracked.

Acronym

Definition

HIP-KNEE

Total Hip/Knee Arthroplasty

HF

Heart Failure

COPD

Chronic Obstructive Pulmonary Disease

AMI

Acute Myocardial Infarction

CABG

Coronary Artery Bypass Graft

PN

Pneumonia

It's important to understand that these medical conditions are the **ones that Medicaid/Medicare tracks for hospital readmissions**. These are not the only conditions in which a patient might be readmitted, but these are the ones that the agency uses to keep track of hospital performance.

Dynamic creation of merged dataframe objects: supporting dataframes.

We will often be asked to deal with large, unwieldy datasets and sometimes we need to be able to handle them dynamically. **Dynamic variable assignment**, as I demonstrated when we read in the files, is going to come in handy - but it can sometimes be tricky to accomplish. The whole point, though, is to create code that can handle ANYTHING you throw at it.

In a similar vein, we can also perform **dynamic dataframe creation** when we need to modify and stitch together more than one pre-existing dataframe - ESPECIALLY when we might decide down the line to alter our choices by adding or removing other dataframes.

Writing your own function to create objects dynamically. We are going to take all the cleaning steps we did above that was not specific to the `FY_2025_Hospital_Readmissions_Reduction_Program` dataframe and tidy, pivot, & filter for specific readmission conditions, and iteratively join as many dataframes as needed. The goal is to make something flexible enough that, if you change a set of inputs, it will create a joined table from ANY set of dataframe inputs. There is more than one way to do this; but leveraging some of the existing `tidyverse` and `tidyr` functions will likely help you make this very flexible. Please note, however, that you do not have to use those packages.

I start you out with something I find helpful when joining together dataframes that sometimes have *redundant* columns BUT not every column is in every dataframe. I start by making a dataframe called `hospitalInfo` that contains the pertinent information on each hospital in the dataset, so you can drop those columns from the other dataframes to prevent inflation or issues when you later join the dataframes together. You may want to join each of the other dataframes to `hospitalInfo`.

As I was writing my own version of this function, I found that the `Payment and Value of Care` table must be separated into two tables to join well, at least with how I wrote my function. So, I do that for you below. I also clean up `HCAHPS` to make it easier to work for the function I wrote, so it may also be useful for you. Note that what I'm doing in these cases is making sure that `MeasureName` is the title of the column **I will join my tables on.** In this way, I'm using the `MeasureName` as the **key** for my table joins.

First, make sure to separate the Payment and Values tables, giving each facility IDs:

```
paymentOnly <- Payment_and_Value_of_Care %>%
  select(FacilityId, PaymentMeasureName, PaymentCategory, Payment) %>%
  mutate(Payment = gsub("\\$", "", Payment)) %>% # Remove the dollar sign
  mutate(Payment = gsub("\\,", "", Payment)) %>% # Remove the comma
  rename(MeasureName = PaymentMeasureName)  # Make consistent with other tables

valueOnly <- Payment_and_Value_of_Care %>%
  select(FacilityId, ValueOfCareDisplayName, ValueOfCareCategory) %>%
  rename(MeasureName = ValueOfCareDisplayName)  # Make consistent with other tables
```

HCAHPS is also a mess. Remove the columns that we will never use in any analysis, make a `MeasureName` column, and also drop the " - linear mean score" from some rows in the newly minted `MeasureName`

```
HCAHPS <- HCAHPS %>%
  select(-HcahpsMeasureId, -PatientSurveyStarRatingFootnote,
         -HcahpsAnswerPercentFootnote, -NumberOfCompletedSurveysFootnote,
         -SurveyResponseRatePercentFootnote, -HcahpsAnswerDescription,
         -PatientSurveyStarRating) %>%  ## Drops the columns we don't need
  rename(MeasureName = HcahpsQuestion) %>%  ## Makes a new MeasureName column
  mutate(MeasureName = gsub(" - linear mean score", "", MeasureName))
  ## Takes off the phrase that we don't want
```

Lastly, pull the hospital information off of the Payment and Value table because it is complete there. We can then drop this information from all the other tables:

```
hospitalInfo <- Payment_and_Value_of_Care %>%
              ## Information about the hospitals
                select(FacilityId, FacilityName, Address, CityTown,
                         State, ZipCode, CountyParish, TelephoneNumber)
```

Question 12: [5 points] Write a function that will:

1. Fix any issues identified previously, e.g., issues with the `NA` class or pivoting, if needed
2. Join any number of a list of dataframes you give it. For example, it should be able to join these 8 tables:

```
datList <- list(Healthcare_Associated_Infections,
                paymentOnly,
                Outpatient_Imaging_Efficiency,
                Complications_and_Deaths,
                Medicare_Hospital_Spending_Per_Patient,
                Timely_and_Effective_Care,
                Unplanned_Hospital_Visits,
                HCAHPS)
```

3. Filters the data for given criteria and condition(s) prior to pivoting. For example, for the 8 tables listed above, perhaps to filter for these possible measures before pivoting, so that I'm not pivoting ALL of those measures! (That would likely crash R!). Notice that what we're filtering for - is really a selection criteria, but because the data are in **wide format**, we are selecting them by row (i.e., "filter") rather than selection by column (i.e., "select").

```
filterList <- list("MRSA Bacteremia",
                   "Payment for pneumonia patients",
                   "Abdomen CT Use of Contrast Material",

                   c("Death rate for pneumonia patients",
```

```
                      "Perioperative pulmonary embolism or deep vein thrombosis rate",
                      "CMS Medicare PSI 90: Patient safety and adverse events composite",
                      "Postoperative respiratory failure rate"),

                    "Medicare spending per patient",

                    c("Healthcare workers given influenza vaccination",
                      "Percentage of healthcare personnel who completed COVID-19 primary vaccination seri
                      "Average (median) time patients spent in the emergency department before leaving fi
                      "Left before being seen",
                      "Venous Thromboembolism Prophylaxis",
                      "Intensive Care Unit Venous Thromboembolism Prophylaxis",
                      "Emergency department volume"),

                    "Hospital return days for pneumonia patients",

                    c("Nurse communication",
                      "Doctor communication",
                      "Staff responsiveness",
                      "Communication about medicines",
                      "Discharge information",
                      "Care transition",
                      "Cleanliness",
                      "Quietness",
                      "Overall hospital rating",
                      "Recommend hospital")
)
```

4. For full credit, make sure that your function can clean up and join together at least three of the dataframes. **Note**: Make sure to join with `readmissionsClean` at the end!

Perhaps you will choose to start your function like this (although you do not have to):

```
tidyNjoin <- function(datList, filterList, hospitalInfo) {

} #Collins Njagi, my experience with functions is abit wanting! I will have to dig deep
```

- **Hint 1:** Feel free to drop the `startDate` and `endDate` columns, as well as the `footnote` column. I would even suggest dropping `MeasureId`.
- **Hint 2:** `select(-any_of(c(...)))` can be used to drop any of a list of columns, regardless of whether it exists in every dataframe or not.
- **Hint 3:** If you choose to drop all those extraneous columns, then the column you will need to merge/join on will always be in the second position, `MeasureName`.
- **Hint 4:** `full_join()` in `dplyr` is likely what you will need; you will want to execute the join on `FacilityId`.
- **Hint 5:** Make sure to check for and/or remove duplicate rows.
- **Hint 6:** If you are really stuck, move ahead and I give you a cleaned, merged data set to work with.

Okay, take a stab at it!

```
# Your function here.
```

Question 13: [**1 point**] Why do you think I just made you work through this function? (**Hint**: What are you going to be asked to do in Project 1?)

#By Collins Njagi. > Your answer here. This function will be very handy in project 1 because we will be focusing on one or two specific conditions to predict hospital readimission

The full, merged, tidied dataset for pneumonia.

Please load if you are struggling to get a function that performs the task or if you aren't sure if it does what's expected.

```
load("pneumoniaFull.Rdata")
dim(pneumoniaFull)
```

```
## [1] 4816  100
```

As I mentioned in Question 12, I ultimately chose to merge 8 of the datasets together for this dataset, although I selectively filtered to focus on my question: **Can we predict pneumonia-related hospital readmissions based on factors that might indicate how well the hospital is doing at diagnostics and disease prevention?**

Whew! Notice that this dataset currently has **100 features** and **4,816 rows**. That's definitely too many features, so let's perform some additional feature selection.

Let's start by examining the "big picture" of what's in this giant dataset:

Table 3. List of hospital-level data sets chosen for the pneumonia-related readmissions analysis.

Dataset

Information Used

Rationale

Healthcare_Associated_Infections

MRSA Bacteremia

Hospital-transmitted disease rate

paymentOnly

Mean Medicare payment per patient received by hospital

Outpatient_Imaging_Efficiency

Abdomen CT Use of Contrast Material

Proxy for imaging ability to diagnose pneumonia appropriately

Complications_and_Deaths

Death rate for pneumonia patients

Complications_and_Deaths

Perioperative pulmonary embolism or deep vein thrombosis rate

Diagnostic ability of hospital

Complications_and_Deaths

CMS Medicare PSI 90: Patient safety and adverse events composite

General hospital safety

Complications_and_Deaths

Postoperative respiratory failure rate

General ability for respiratory diagnostics

Medicare_Hospital_Spending_Per_Patient

Score

Score rather than $ amount

Timely_and_Effective_Care

Healthcare workers given influenza vaccination

Likelihood of influenza transmission

Timely_and_Effective_Care

% healthcare personnel completed COVID-19 primary vaccination series

Likelihood of COVID-19 transmission

Timely_and_Effective_Care

Average (median) minutes patients spent in the ED before leaving (lower is better)

Proxy for overall effective diagnosis & treatment

Timely_and_Effective_Care

Left before being seen in ED

Proxy for overall effective diagnosis & treatment

Timely_and_Effective_Care

Venous Thromboembolism Prophylaxis

Diagnostic ability for another critical and silent disease

Timely_and_Effective_Care

ICU Thromboembolism Prophylaxis

Diagnostic ability for another critical and silent disease

Timely_and_Effective_Care

Emergency department volume

Proxy for how overwhelmed the hospital is

Unplanned_Hospital_Visits

Hospital return days for pneumonia patients

HCAHPS

Nurse communication

Overall rating as linear score

HCAHPS

Doctor communication

Overall rating as linear score

HCAHPS

Staff responsiveness

Overall rating as linear score

HCAHPS

Communication about medicines

Overall rating as linear score

HCAHPS

Discharge information

Overall rating as linear score

HCAHPS

Care transition

Overall rating as linear score

HCAHPS

Cleanliness

Overall rating as linear score

HCAHPS

Quietness

Overall rating as linear score

HCAHPS

Overall hospital rating

Overall rating as linear score

HCAHPS

Recommend hospital

Overall rating as linear score

Pre-processing & Feature Selection (Round 1)

WARNING: Typically we would do a lot of this work AFTER splitting into a training and testing set. I'm not having you do that today so that you can write code that will help you with your Project1. Just be aware that this goes against the typical workflow a bit.

Feature selection can take many forms, from what we've already done (choosing features relevant to our question) to culling non-informative columns to using more machine-guided approaches. We're going to leverage all types here.

Additionally, we are going to perform **encoding** to our categorical variables. It will be important to perform encoding **BEFORE** using automated methods for feature selection.

Culling Features (and, yes, more tidying!)

Although we ideally could just include these steps (and perhaps you did!) in our tidying & joining function from Question 12, I wanted to make sure we talk about this explicitly.

Question 14: [**1 point**] Looking through the feature names, you probably notice names like `LowerEstimate_Death rate for pneumonia patients`. We do not need the higher and lower estimates in this case; just the point estimates. Use the `contains()` function to **drop** any columns that contain `LowerEstimate` or `HigherEstimate`. Let's also go ahead and drop any columns containing `Denominator` and `HcahpsAnswerPercent` as well. How many features did you drop?

Your answer here. 25 features

```
# Your code here.
# by Collins Njagi
?contains #checking how this function works in documentation

pneumoniaFull <- pneumoniaFull %>%
  select(-contains(c("LowerEstimate", "HigherEstimate", "Denominator", "HcahpsAnswerPercent"), ignore.ca

dim(pneumoniaFull)
```

```
## [1] 4816   75
```

Note: Please drop `FacilityName`, `TelephoneNumber`, `Address`, & `CityTown` at this stage as well, if they're still in your full dataset.

```
#Collins Njagi
# Your code here.

pneumoniaFull <- pneumoniaFull%>%
  select(-c(FacilityName, TelephoneNumber, Address, CityTown))

dim(pneumoniaFull)
```

```
## [1] 4816   71
```

Encoding

Mystery encoding! (Well, it won't be a mystery for long.)

Question 15: [**1 point**] We need to perform encoding on all of the categorical columns, ultimately. However, we will focus first on the columns that begin with '`ComparedToNational_`, e.g., `ComparedToNational_Death rate for pneumonia patients`. What kind of encoding should we perform on these columns? Why?

#Collins Njagi

```
unique(pneumoniaFull$`PaymentCategory_Payment for pneumonia patients`)
```

```
## [1] "No Different Than the National Average Payment"
## [2] NA
## [3] "Greater Than the National Average Payment"
## [4] "Less Than the National Average Payment"
```

```
unique(pneumoniaFull$`ComparedToNational_MRSA Bacteremia`)
```

```
## [1] "No Different than National Benchmark"
## [2] NA
## [3] "Worse than the National Benchmark"
## [4] "Better than the National Benchmark"
```

```
unique(pneumoniaFull$`ComparedToNational_Death rate for pneumonia patients`)
```

```
## [1] "No Different Than the National Rate" "Worse Than the National Rate"
## [3] NA                                    "Better Than the National Rate"
```

```
unique(pneumoniaFull$`ComparedToNational_Postoperative respiratory failure rate`)
```

```
## [1] "No Different Than the National Rate" NA
## [3] "Worse Than the National Rate"        "Better Than the National Rate"
```

```
unique(pneumoniaFull$`ComparedToNational_Perioperative pulmonary embolism or deep vein thrombosis rate`)
```

```
## [1] "No Different Than the National Rate" NA
## [3] "Worse Than the National Rate"        "Better Than the National Rate"
```

```
unique(pneumoniaFull$`ComparedToNational_CMS Medicare PSI 90: Patient safety and adverse events composi
```

```
## [1] "No Different Than the National Value"
## [2] NA
## [3] "Worse Than the National Value"
## [4] "Better Than the National Value"
```

> Your answer here. I woulod prefer Ordinal Encoding because both features seems to have some order of ranking ie. No different than the national rate, worse than the national rate. Meaning one category is worse off than the other.Ordinal encoding therefore would be the best because it respects this ranking within the categories without implying a quantitative relationship that parctically doesn't exist

Question 16: [**1 point**] What other column do we also need to perform this kind of encoding on?

#Collins Njagi > Your answer here. paymentCategory_paymentfor pneumonia patients.

Question 17: [**3 points**] You probably expected this - attempt to encode those columns the method you selected. It's okay if you can only think of a way to brute-force this for now; I will show you code to help you do it faster after the assignment. :)

- **Hint 1:** You could use the `contains()` or `startsWith()` functions to help you quickly grab those columns so you can make the changes you want to make.
- **Hint 2:** You may want to explore the `grepl()` function or regular expressions, as they can allow us to match in a "fuzzy" way.
- **Hint 3:** You may want to create a temporary dataframe that you execute the changes on and then replace the original columns from there. That way, you don't have to keep reloading the original data if you make mistakes!
- **Hint 3:** For full credit, don't forget to double check that you managed to preserve all the data! The `table()` function is sufficient here.

```
#Collins Njagi
# Your code here.

pneumoniaFull_encoded <- pneumoniaFull%>%
  mutate(
    across(c(contains("ComparedToNational"), "PaymentCategory_Payment for pneumonia patients"), ~case_wh
      grepl("Better", .)~1,
      grepl("No Different", .) ~0,
      grepl("Worse", .) ~ -1,
      TRUE ~ NA_real_
    ))
  )

table(pneumoniaFull_encoded$`ComparedToNational_Death rate for pneumonia patients`, useNA = "ifany")
```

```
##
##   -1    0    1 <NA>
##  133 3165  216 1302
```

Note: If you struggled with Question 17, load the data below and keep going:

```
load("pneumoniaFullEncoded.Rdata")
```

Frequency encoding state and county.

We also decide that we want a way to try to preserve and analyze the geographic information without causing terrible overfitting. So, we will try to apply **frequency encoding** to those two categorical columns.

Question 18: [**1 points**] Go through the **frequency encoding** code chunk below and comment each line. What does it do? *Make sure to comment why you think the choice was made*, if appropriate.

Your answer in comments in the code chunk. By Collins Njagi

```
cols2encode <- c("State",
                 "CountyParish") #defines the feactures (columns) to be frequency encoded

temp <- pneumoniaFullEncoded[, names(pneumoniaFullEncoded) %in% cols2encode] #creating a temporary data

#defines a function to add frequency encoding to a dataframe
add_freq <- function(data, column_name) {
  frequency_map <- table(data[[column_name]], useNA = "always") #creates a frequency map of the values
  data[[column_name]] <- frequency_map[match(data[[column_name]],# Replaces the values in the column wi
                                             names(frequency_map))]
  return(data) #returns the modified dataframe
} #

for (col in names(temp)) {
  temp <- add_freq(temp, col)
} #applies the add_freq function to each column in the temp dataframe

for (c in 1:length(cols2encode)) {
  pneumoniaFullEncoded[, cols2encode[c]] <- temp[, cols2encode[c]]
} #replaces the original columns in pneumoniaFullEncoded withthe frequency encoded columns
```

Question 19: [**1 point**] Use your finally & freshly encoded version of `pneumoniaFullEncoded` to answer this vital question for `State` and `CountyParish` features: are we justified using these features? Why or why not? You will need to write code to answer this question.

Hint 1: Use your original `pneumoniaFull` dataset to compare the features in `pneumoniaFullEncoded`.

Hint 2: **Heavily** duplicated frequency values can make it ill-advised to proceed with a frequency encoded feature. Low to moderate duplication can be kept if there is a strong justification for doing so.

Hint 3: If you decide it is ill-advised for either or both columns, make sure to remove those features from the dataset!

```
# Your code here. By Collins Njagi

Unique_values <- data.frame(
  features = c("State", "CountyParish"), unique_values = c(
    length(unique(pneumoniaFull$State)),
    length(unique(pneumoniaFull$CountyParish)),
```

```
    length(unique(pneumoniaFullEncoded$State)),
    length(unique(pneumoniaFullEncoded$CountyParish))
  )
)

print(Unique_values)
```

```
##       features unique_values
## 1        State            57
## 2 CountyParish          1549
## 3        State            50
## 4 CountyParish            36
```

#Collins Njagi > Your answer here. Frequency encoding is justified because the number of unique values in encoded dataframe (pneumoniaFullEncoded, state 50 and CountyParish 36) is less than the unique values in the original dataframe as shown above state 57 and countyParish 1549

Finishing touches

```
load("pneumoniaAnalyze.Rdata")

pneumoniaAnalyze <- pneumoniaAnalyze %>%
  mutate(across(where(is.character), as.numeric))
```

1. Ensure that every feature is numeric.

```
pneumoniaAnalyze <- pneumoniaAnalyze %>%
  ## arbitrarily chose as the rep as they are identical
  mutate(NumberSurveysCompleted = NumberOfCompletedSurveys_Cleanliness,
         SurveyResponseRate = SurveyResponseRatePercent_Cleanliness/100) %>%       ## Turned into an ac
  select(-contains(c("NumberOfCompletedSurveys_", "SurveyResponseRatePercent_")))   ## drop the others
```

2. Collapse the `NumberOfCompletedSurveys_...` and `SurveyResponseRatePercent_...` features into a single one, as they are identical / redundant: Excellent! Now we are down to just 51 features...

3. Identify the target variable. Our very, very last steps! Your task is to identify the appropriate target variable from the dataset. I have purposefully not gone into great detail about what it would be because choosing the best target is sometimes tricky.

Question 20: [2 points] Use `pneumoniaAnalyze` to calculate an `observed_readmission_rate` as the number of readmissions divided by the number of discharges, multiplied by 100, then drop the two columns used to make this new one. Now, using the data and the data dictionary, try to understand the relationship between `observed_readmission_rate` that we just created, the `PredictedReadmissionRate`, `ExpectedReadmissionRate`, and the `ExcessReadmissionRatio`. What is the relationship? What is the appropriate target to choose and why?

Collins Njagi In my opinion I would pick observed_readimission_rate as my target variable. This is because observed_readmission is the actual rate and therefore any good model should predict a value close to the actual observed rate.

```
#Collins Njagi
# feature engineering to create the new calculated field (feature)
pneumoniaAnalyze$observed_readmission_rate <- (pneumoniaAnalyze$NumberOfReadmissions / pneumoniaAnalyze
```

```r
pneumoniaAnalyze <- pneumoniaAnalyze %>%
  select(-c("NumberOfReadmissions", "NumberOfDischarges")) #drop the two base columns
```

Hint: Think about (or perhaps try to investigate) why the **predicted** rate exists. What would be the (dis)advantage to using the predicted vs. the newly created `observed_readmission_rate`?

Your answer here.

Question 21: **[1 points]** Why did I have you drop the two columns used to make `observed_readmission_rate`?

#Collins Njagi > Your answer here. By dropping the two base columns used to create`observed_readmission_rate` we reduce the chance of high correlation

Exploratory Data Analysis

For your final question, you will kick start our EDA by focusing on the target variable that you identified in Question 20. **You will not be penalized for choosing the wrong target as long as you made an attempt to choose and defend a rational choice.**

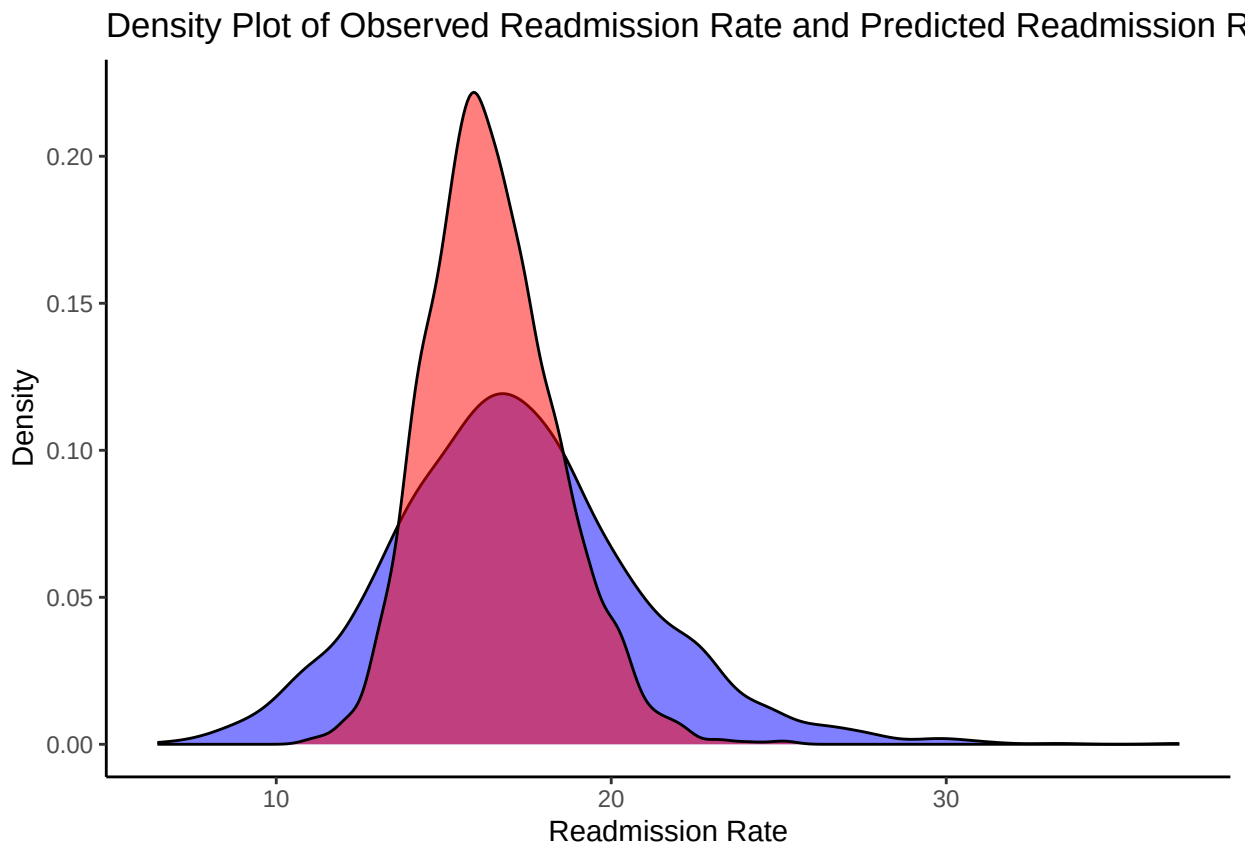
Question 22: **[3 points]** Explore the target variable you identified in Question 20 with at least one other variable. Try to push yourself to try a new style of plot; for example, have you ever made a `density` or `violin` plot? These can be excellent choices when exploring the distribution of a target variable against another variable. Note that, if you choose to compare the target against one of the encoded categorical variables, you will need to properly re-label the categories for your plot.

Full points are awarded for professional plots with axis labels, labeled legends (if appropriate), and creative use of multivariable information. **One bonus point will be awarded if you make a faceted plot or otherwise include information from at least 3 variables.**

Need inspiration that comes with code?!? Check out the R Graph Gallery!

```r
# Your code here.By Collins Njagi

ggplot(pneumoniaAnalyze, aes(x = observed_readmission_rate)) +
  geom_density(alpha = 0.5, fill = "blue") +
  geom_density(aes(x = PredictedReadmissionRate), alpha = 0.5, fill = "red") +
  labs(title = "Density Plot of Observed Readmission Rate and Predicted Readmission Rate",
       x = "Readmission Rate",
       y = "Density") +
  theme_classic()
```



Lastly, make sure to **interpret** your graphic.

#Collins Njagi > Your answer here. 1. the predicted readmission rate has a narrow range and a higher peak, indicating that the predicted values are more concentrated around the mean 2. the observed readmission rate (blue) has a wider range and a lower peak, indicating that the actual values are more spread out 3. the observed readmission rate plot is shifted slightly to the right, indicating that the actual readmission rates tend to be higher than the predicted rates. 4. There is some overlap between the two plots, indicating that there is some agreement between the predicted and observed readmission rates. Therefore, in conclusipn the density plot suggest that the predicted readmission rates tend to be lower and more concentrated than the observed readmission rates.

References

De Alba, Israel, and Alpesh Amin. 2014. "Pneumonia Readmissions: Risk Factors and Implications." *Ochsner Journal* 14 (4): 649–54.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10): 1–23. https://doi.org/10.18637/jss.v059.i10.