Challenge_2: Data Transformation(2), Pivot and Date-Time Data

AUTHOR
Muskan Dhar

PUBLISHED
June 14, 2024

Make sure you change the author's name.

Setup

If you have not installed the following packages, please install them before loading them.

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.4.4     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(readxl)
library(haven) #for loading other datafiles (SAS, STATA, SPSS, etc.)
library(stringr) # if you have not installed this package, please install it.
library(lubridate)
```

Challenge Overview

Building on the lectures in week#3 and week#4, we will continually practice the skills of different transformation functions with Challenge_2. In addition, we will explore the data more by conducting practices with pivoting data and dealing with date-time data.

There will be coding components and writing components. Please read the instructions for each part and complete your challenges.

Datasets

There are four datasets provided in this challenge. Please download the following dataset files from Google Classroom and save them to a folder within your project working directory (i.e.: "DACSS601_data"). If you don't have a folder to store the datasets, please create one.

- ESS_5.dta (Part 1) ⭐
- p5v2018.sav (Part 1)⭐
- austrlian_data.csv (Part 3)⭐
- FedFundsRate.csv (Part 4)⭐

Find the `_data` folder, then use the correct R command to read the datasets.

Part 1. Depending on the data you chose in Challenge#1 (ESS_5 or Polity V), please use that data to complete the following tasks

If you are using the ESS_5 Data:

1. **Read the dataset and keep the first 39 columns.**

```
ESS_5 <- read_dta("ESS_5.dta")[,1:39]
ESS_5
```

```
# A tibble: 52,458 × 39
    idno essround  male   age   edu income_10 eth_major  media  obey trust_court
   <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl>       <dbl>
 1 15906        5     0    14     1         2         1 0.312   1           1
 2 21168        5     0    14     1         2         1 0.438   1           0.75
 3    40        5     0    14     1         8        NA 0.375   0.5         0.5
 4  2108        5     0    14     1        NA         1 0.0625  0.75        0.75
 5   519        5     0    14     1        NA         1 0.125   1           1
 6  2304        5     0    14     1        NA         1 0.25    0.5         0.25
 7   290        5     0    14     1        NA         1 0.312   0.75        0.5
 8  3977        5     0    14     1        NA         1 0.375   0           0.5
 9 23244        5     0    14     1        NA         1 0.375   1           0.75
10 19417        5     0    14     1        NA         1 0.438   0.5         0.75
# ℹ 52,448 more rows
# ℹ 29 more variables: cntry <chr>, commonlaw <dbl>, PostComm <dbl>, tv <dbl>,
#   radio <dbl>, papers <dbl>, Internet <dbl>, name <chr>, edition <chr>,
#   proddate <chr>, tvtot <dbl+lbl>, tvpol <dbl+lbl>, rdtot <dbl+lbl>,
#   rdpol <dbl+lbl>, nwsptot <dbl+lbl>, nwsppol <dbl+lbl>, netuse <dbl+lbl>,
#   ppltrst <dbl+lbl>, pplfair <dbl+lbl>, pplhlp <dbl+lbl>, polintr <dbl+lbl>,
#   trstprl <dbl+lbl>, trstlgl <dbl+lbl>, trstplc <dbl+lbl>, …
```

2. **Conduct the following transformation for the data by using mutate() and other related functions :**

   (1) Create a new column named “YearOfBirth” using the information in the “age” column.

   (2) Create a new column named “adult” using the information in the “age” column.

   (3) Recode the “commonlaw” column: if the value is 0, recode it as “non-common-law”; if the value is 1, recode it as “common-law”.

(4) Recode the “vote” column: if the value is 3, recode it as 1; if the value is smaller than 3, recode it as 0. Make sure not to recode the NAs.

(5) Move the column “YearOfBirth”, “adult,” “commonlaw” and “vote” right before the “essround” column (the 2nd column in order).

(6) Answer the question: What is the data type of the “commonlaw” column before and after recoding? And what is the data type of the “vote” column before and after recoding?

```r
#1
new_ESS_5 <- mutate(ESS_5, YearOfBirth = 2023 - age)
#2
new_ESS_5 <- new_ESS_5 %>%
  mutate(adult = case_when(
    age >= 18 ~ "Adult",
    age < 18 ~ "Young"))
#3
new_ESS_5 <- mutate(new_ESS_5,
                    commonlaw = case_when(
                      commonlaw == 0 ~ "non-common-law",
                      commonlaw == 1 ~ "common-law",))
#4
new_ESS_5 <- new_ESS_5 %>%
  mutate(vote = case_when(
    vote == 3 ~ 1,
    vote < 3 ~ 0))

#5
new_ESS_5 <- new_ESS_5 %>%
    relocate(
      YearOfBirth, adult, commonlaw, vote, before = essround )
head(new_ESS_5)
```

```
# A tibble: 6 × 41
  YearOfBirth adult commonlaw       vote before  idno  male   age   edu income_10
        <dbl> <chr> <chr>          <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>
1        2009 Young non-common-l…      1      5 15906     0    14     1         2
2        2009 Young common-law         1      5 21168     0    14     1         2
3        2009 Young non-common-l…      1      5    40     0    14     1         8
4        2009 Young non-common-l…      1      5  2108     0    14     1        NA
5        2009 Young common-law         0      5   519     0    14     1        NA
6        2009 Young non-common-l…      1      5  2304     0    14     1        NA
# ℹ 31 more variables: eth_major <dbl>, media <dbl>, obey <dbl>,
#   trust_court <dbl>, cntry <chr>, PostComm <dbl>, tv <dbl>, radio <dbl>,
#   papers <dbl>, Internet <dbl>, name <chr>, edition <chr>, proddate <chr>,
#   tvtot <dbl+lbl>, tvpol <dbl+lbl>, rdtot <dbl+lbl>, rdpol <dbl+lbl>,
#   nwsptot <dbl+lbl>, nwsppol <dbl+lbl>, netuse <dbl+lbl>, ppltrst <dbl+lbl>,
#   pplfair <dbl+lbl>, pplhlp <dbl+lbl>, polintr <dbl+lbl>, trstprl <dbl+lbl>,
#   trstlgl <dbl+lbl>, trstplc <dbl+lbl>, trstplt <dbl+lbl>, …
```

```r
#6
```

```
print("Datatype of commonlaw column before recoding")
```

```
[1] "Datatype of commonlaw column before recoding"
```

```
print(class(ESS_5$commonlaw))
```

```
[1] "numeric"
```

```
print("Datatype of commonlaw column after recoding")
```

```
[1] "Datatype of commonlaw column after recoding"
```

```
print(class(new_ESS_5$commonlaw))
```

```
[1] "character"
```

```
print("Datatype of vote column before recoding")
```

```
[1] "Datatype of vote column before recoding"
```

```
print(class(ESS_5$vote))
```

```
[1] "haven_labelled" "vctrs_vctr"     "double"
```

```
print("Datatype of vote column after recoding")
```

```
[1] "Datatype of vote column after recoding"
```

```
print(class(new_ESS_5$vote))
```

```
[1] "numeric"
```

If you are using the Polity V Data:

1. **Read the dataset and keep the first 11 columns.**

```
#Type your code here
```

2. **Conduct the following transformation for the data by using mutate() and other related functions :**

   (1) Create a new column named “North America” using the information in the “country” column. Note: “United States,” “Mexico,” or “Canada” are the countries in North America. In the new “North America” column, if a country is one of the above three countries, it should be coded as 1, otherwise as 0.

(2) Recode the "democ" column: if the value is 10, recode it as "Well-Functioning Democracy"; if the value is greater than 0 and smaller than 10, recode it as "Either-Autocracy-or-Democracy"; if the value is 0, recode it as "Non-democracy"; if the value is one of the following negative integers (-88, -77, and -66), recode it as "Special-Cases."

(3) Move the column "North America" and "democ" right before the "year" column (the 6th column in order).

(4) Answer the question: What is the data type of the "North America" column? What is the data type of the "democ" column before and after recoding?

```
#Type your code here
```

Part 2. Generate your own Data

1. **Generate an untidy data that includes 10 rows and 10 columns. In this dataset, column names are not names of variables but a value of a variable.**

   *Note: do not ask ChatGPT to generate a dataframe for you. I have already checked the possible questions and answers generated by AI.

```
# dataset of Grammy Awards won by singers from the year 2015 to 2023
singers <- c("Beyonce", "Taylor Swift", "Adele", "Bruno Mars", "Lady Gaga","Ed Sheer
awards <- data.frame(Singers = singers)
year <- c("2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023")
awards[, year] <- NA
awards[1, 2:10] <- c(2, 1, 1, 0, 4, 3, 1, 5, 1)
awards[2, 2:10] <- c(5, 2, 2, 1, 1, 0, 2, 3, 4)
awards[3, 2:10] <- c(4, 1, 2, 0, 2, 1, 3, 2, 0)
awards[4, 2:10] <- c(0, 3, 1, 2, 4, 2, 2, 4, 1)
awards[5, 2:10] <- c(1, 2, 3, 0, 1, 1, 4, 3, 0)
awards[6, 2:10] <- c(1, 0, 2, 5, 2, 4, 0, 1, 3)
awards[7, 2:10] <- c(3, 4, 1, 2, 3, 0, 3, 1, 5)
awards[8, 2:10] <- c(2, 2, 4, 1, 5, 1, 2, 0, 4)
awards[9, 2:10] <- c(0, 1, 5, 3, 0, 2, 4, 1, 2)
awards[10,2:10] <- c(5, 4, 3, 1, 2, 3, 1, 2, 0)
awards
```

```
         Singers 2015 2016 2017 2018 2019 2020 2021 2022 2023
1        Beyonce    2    1    1    0    4    3    1    5    1
2   Taylor Swift    5    2    2    1    1    0    2    3    4
3          Adele    4    1    2    0    2    1    3    2    0
4     Bruno Mars    0    3    1    2    4    2    2    4    1
5      Lady Gaga    1    2    3    0    1    1    4    3    0
6     Ed Sheeran    1    0    2    5    2    4    0    1    3
7        Rihanna    3    4    1    2    3    0    3    1    5
8  Justin Bieber    2    2    4    1    5    1    2    0    4
9     Katy Perry    0    1    5    3    0    2    4    1    2
10 Ariana Grande    5    4    3    1    2    3    1    2    0
```

2. **Use the correct pivot command to convert the data to tidy data.**

```
awards_pivot <- awards %>%
  pivot_longer(cols = year, names_to = "Year", values_to = "Count")
```

```
Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
i Please use `all_of()` or `any_of()` instead.
  # Was:
  data %>% select(year)

  # Now:
  data %>% select(all_of(year))

See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
awards_pivot
```

```
# A tibble: 90 × 3
   Singers      Year  Count
   <chr>        <chr> <dbl>
 1 Beyonce      2015      2
 2 Beyonce      2016      1
 3 Beyonce      2017      1
 4 Beyonce      2018      0
 5 Beyonce      2019      4
 6 Beyonce      2020      3
 7 Beyonce      2021      1
 8 Beyonce      2022      5
 9 Beyonce      2023      1
10 Taylor Swift 2015      5
# i 80 more rows
```

3. **Generate an untidy data that includes 10 rows and 5 columns. In this dataset, an observation is scattered across multiple rows.**

```
#Type your code here

names <- c("Beyonce", "Beyonce", "Taylor", "Taylor", "Adele", "Adele", "Rihanna", "R
cols_names <- c("Age", "Awards_Won", "Total_Albums","Information")
info <- data.frame(Names = names)
info[, cols_names] <- NA
info[1, 2:5] <- c(40, 5, 5, "verified")
info[2, 2:5] <- c(35, 4, 3, "not verified")
info[3, 2:5] <- c(33, 4, 8, "verified")
info[4, 2:5] <- c(30, 3, 6, "not verified")
info[5, 2:5] <- c(32, 6, 4, "verified")
info[6, 2:5] <- c(29, 3, 3, "not verified")
info[7, 2:5] <- c(35, 2, 3, "verified")
info[8, 2:5] <- c(30, 1, 2, "not verified")
info[9, 2:5] <- c(31, 3, 6, "verified")
info[10,2:5] <- c(28, 1, 3, "not verified")
```

```
info
```

```
      Names Age Awards_Won Total_Albums  Information
1   Beyonce  40          5            5     verified
2   Beyonce  35          4            3 not verified
3    Taylor  33          4            8     verified
4    Taylor  30          3            6 not verified
5     Adele  32          6            4     verified
6     Adele  29          3            3 not verified
7   Rihanna  35          2            3     verified
8   Rihanna  30          1            2 not verified
9    Ariana  31          3            6     verified
10   Ariana  28          1            3 not verified
```

3. **Use the correct pivot command to convert the data to tidy data.**

```
info_ <- info |>
  pivot_wider(
    names_from = Information,
    values_from = c("Age","Awards_Won", "Total_Albums"),
  )

info_
```

```
# A tibble: 5 × 7
  Names   Age_verified `Age_not verified` Awards_Won_verified
  <chr>   <chr>        <chr>              <chr>
1 Beyonce 40           35                 5
2 Taylor  33           30                 4
3 Adele   32           29                 6
4 Rihanna 35           30                 2
5 Ariana  31           28                 3
# ℹ 3 more variables: `Awards_Won_not verified` <chr>,
#   Total_Albums_verified <chr>, `Total_Albums_not verified` <chr>
```

Part 3. The Australian Data

This is another tabular data source published by the [Australian Bureau of Statistics]() that requires a decent amount of cleaning. In 2017, Australia conducted a postal survey to gauge citizens' opinions towards same sex marriage: "Should the law be changed to allow same-sex couples to marry?" All Australian citizens are required to vote in elections, so citizens could respond in one of four ways: vote yes, vote no, vote in an unclear way (illegible), or fail to vote. (See the "Explanatory Notes" sheet for more details.)

I have already cleaned up the data for you and you can directly import it. We will come back to clean and process the original "messy" data after we learn some string functions in the later weeks.

1. **Read the dataset "australian_data.csv":**

```r
data <- read.csv("australian_data.csv")
head(data)
```

```
  X  District   Yes    No Illegible No.Response                  Division
1 1     Banks 37736 46343       247       20928 New South Wales Divisions
2 2    Barton 37153 47984       226       24008 New South Wales Divisions
3 3 Bennelong 42943 43215       244       19973 New South Wales Divisions
4 4   Berowra 48471 40369       212       16038 New South Wales Divisions
5 5  Blaxland 20406 57926       220       25883 New South Wales Divisions
6 6 Bradfield 53681 34927       202       17261 New South Wales Divisions
```

- **Data Description: Please use the necessary commands and codes and briefly describe this data with a short writing paragraph answering the following questions.**

  ```r
  dim(data)
  ```

  ```
  [1] 150   7
  ```

  (1) What is the dimension of the data (# of rows and columns)? The given dataset has 150 rows and 7 columns.

  (2) What do the rows and columns mean in this data? Each row here represents a new district. The columns represent the voting data related to each district, like the no. of people who answered Yes and No, or people who gave illegible or no responses.

- **Data Transformation: use necessary commands and codes and answer the following questions.**

  (1) Reshape the dataset to longer format

  ```r
  data_ <- pivot_longer(data,
                        cols = c(`Yes`, `No`, `Illegible`, `No.Response`),
                        names_to = "Response_Type",
                        values_to = "Count")

  head(data_)
  ```

  ```
  # A tibble: 6 × 5
        X District Division                  Response_Type Count
    <int> <chr>    <chr>                     <chr>         <int>
  1     1 Banks    New South Wales Divisions Yes           37736
  2     1 Banks    New South Wales Divisions No            46343
  3     1 Banks    New South Wales Divisions Illegible       247
  4     1 Banks    New South Wales Divisions No.Response   20928
  5     2 Barton   New South Wales Divisions Yes           37153
  6     2 Barton   New South Wales Divisions No            47984
  ```

  (2) How many districts and divisions are in the data?

```r
summary <- data %>%
  summarise(
```

```
    districts = n_distinct(District),
    divisions = n_distinct(Division)
  )
print(summary$districts)
```

```
[1] 150
```

```
print(summary$divisions)
```

```
[1] 8
```

\(3\) Use mutate() to create a new column "district turnout(%)". This column should be the voting turnout in a given district, or the proportion of people cast votes (yes, no and illegible) in the total population of a district.

```
data <- data %>%
  group_by(District) %>%
  mutate(`district turnout(%)` = sum(Yes + No + Illegible) /
                                 sum(Yes + No + Illegible + No.Response) * 100) %>%
  ungroup()

# Print the first few rows to check the results
head(data)
```

```
# A tibble: 6 × 8
      X District    Yes    No Illegible No.Response Division
  <int> <chr>     <int> <int>     <int>       <int> <chr>
1     1 Banks     37736 46343       247       20928 New South Wales Divisions
2     2 Barton    37153 47984       226       24008 New South Wales Divisions
3     3 Bennelong 42943 43215       244       19973 New South Wales Divisions
4     4 Berowra   48471 40369       212       16038 New South Wales Divisions
5     5 Blaxland  20406 57926       220       25883 New South Wales Divisions
6     6 Bradfield 53681 34927       202       17261 New South Wales Divisions
# i 1 more variable: `district turnout(%)` <dbl>
```

\(4\) please use summarise() to estimate the following questions:

- In total, how many people support same-sex marriage in Australia, and how many people oppose it?

```
  supporting <- data %>%
  summarise(supporting = sum(Yes))
print(supporting)
```

```
# A tibble: 1 × 1
  supporting
       <int>
1    7817247
```

```
opposing <- data %>%
  summarise(opposing = sum(No))
```

```r
print(opposing)
```

```
# A tibble: 1 × 1
  opposing
     <int>
1  4873987
```

-   Which *district* has ***most people*** supporting the policy, and how many?

```r
max_yes_district <- data %>%
  arrange(desc(Yes)) %>%
  summarise(District = first(District), Max_Yes_Votes = first(Yes))
print(max_yes_district)
```

```
# A tibble: 1 × 2
  District    Max_Yes_Votes
  <chr>               <int>
1 Canberra(d)         89590
```

-   Which *division* has the highest approval rate (% of "yes" in the total casted votes)? And what is the average approval rate at the *division level?*

    -   Hint: Do NOT take the average of the district approval rate. Each district has a different number of population. The raw approval rate at the district level is not weighted by its population.

::: {.cell}

```{.r .cell-code}
  division_approval <- data %>%
  group_by(Division) %>%
  summarise(
total_yes = sum(Yes),
total_casted = sum(Yes + No + Illegible)
  ) %>%
  mutate(approval_rate = (total_yes / total_casted) * 100)

# Find the division with the highest approval rate
max_approval_division <- division_approval %>%
  filter(approval_rate == max(approval_rate))

print(max_approval_division)
```

::: {.cell-output .cell-output-stdout}
```
# A tibble: 1 × 4
  Division                               total_yes total_casted approval_rate
  <chr>                                      <int>        <int>         <dbl>
1 Australian Capital Territory Divisions    175459       237513          73.9
```

:::
:::

```
average_approval_rate <- division_approval %>%
  summarise(average_approval_rate = mean(approval_rate))
print(average_approval_rate)
```

```
# A tibble: 1 × 1
  average_approval_rate
                  <dbl>
1                  63.3
```

Part 4. The Marco-economic Data

This data set runs from July 1954 to March 2017, and includes daily macroeconomic indicators related to the *effective federal funds rate* - or the interest rate at which banks lend money to each other in order to meet mandated reserve requirements.

1. **Read the dataset “FedFundsRate.csv”:**

```
data1 <- read.csv("FedFundsRate.csv")
head(data1)
```

```
  Year Month Day Federal.Funds.Target.Rate Federal.Funds.Upper.Target
1 1954     7   1                        NA                         NA
2 1954     8   1                        NA                         NA
3 1954     9   1                        NA                         NA
4 1954    10   1                        NA                         NA
5 1954    11   1                        NA                         NA
6 1954    12   1                        NA                         NA
  Federal.Funds.Lower.Target Effective.Federal.Funds.Rate
1                         NA                         0.80
2                         NA                         1.22
3                         NA                         1.06
4                         NA                         0.85
5                         NA                         0.83
6                         NA                         1.28
  Real.GDP..Percent.Change. Unemployment.Rate Inflation.Rate
1                       4.6               5.8             NA
2                        NA               6.0             NA
3                        NA               6.1             NA
4                       8.0               5.7             NA
5                        NA               5.3             NA
6                        NA               5.0             NA
```

2. **Data Description: Please use the necessary commands and codes and briefly describe this data with a short writing paragraph answering the following questions.**

   ```
   dim(data1)
   ```

```
[1] 904  10
```

(1) What is the dimension of the data (# of rows and columns)? Given data has 904 rows and 10 columns

```r
colnames(data1)
```

```
 [1] "Year"                         "Month"
 [3] "Day"                          "Federal.Funds.Target.Rate"
 [5] "Federal.Funds.Upper.Target"   "Federal.Funds.Lower.Target"
 [7] "Effective.Federal.Funds.Rate" "Real.GDP..Percent.Change."
 [9] "Unemployment.Rate"            "Inflation.Rate"
```

```
\(2\) What do the rows and columns mean in this data?
Each row in the dataset represents the data recorded for a specific date, and the
columns provide various indicators related to the effective federal funds rate.


\(3\) What is the unit of observation? In other words, what does each case mean in
this data?
The unit of observation in this dataset is a specific day.
```

3. **Generating a date column:**

   Notice that the year, month, and day are three different columns. We will first have to use a string function called "str_c()" from the "stringr" library to combine these three columns into one "date" column. Please delete the # in the following code chunk.

```r
fed_rates<-data1 |>
  mutate(date = str_c(Year, Month, Day, sep="-"))
head(fed_rates)
```

```
  Year Month Day Federal.Funds.Target.Rate Federal.Funds.Upper.Target
1 1954     7   1                        NA                         NA
2 1954     8   1                        NA                         NA
3 1954     9   1                        NA                         NA
4 1954    10   1                        NA                         NA
5 1954    11   1                        NA                         NA
6 1954    12   1                        NA                         NA
  Federal.Funds.Lower.Target Effective.Federal.Funds.Rate
1                         NA                         0.80
2                         NA                         1.22
3                         NA                         1.06
4                         NA                         0.85
5                         NA                         0.83
6                         NA                         1.28
  Real.GDP..Percent.Change. Unemployment.Rate Inflation.Rate      date
1                       4.6               5.8             NA  1954-7-1
2                        NA               6.0             NA  1954-8-1
3                        NA               6.1             NA  1954-9-1
4                       8.0               5.7             NA 1954-10-1
```

```
5                        NA                5.3               NA 1954-11-1
6                        NA                5.0               NA 1954-12-1
```

4. **Move the new created "date" column to the beginning as the first column of the data.**

```
fed_rates <- fed_rates %>%
  relocate(date, .before = Year)
head(fed_rates)
```

```
       date Year Month Day Federal.Funds.Target.Rate Federal.Funds.Upper.Target
1  1954-7-1 1954     7   1                        NA                         NA
2  1954-8-1 1954     8   1                        NA                         NA
3  1954-9-1 1954     9   1                        NA                         NA
4 1954-10-1 1954    10   1                        NA                         NA
5 1954-11-1 1954    11   1                        NA                         NA
6 1954-12-1 1954    12   1                        NA                         NA
  Federal.Funds.Lower.Target Effective.Federal.Funds.Rate
1                         NA                         0.80
2                         NA                         1.22
3                         NA                         1.06
4                         NA                         0.85
5                         NA                         0.83
6                         NA                         1.28
  Real.GDP..Percent.Change. Unemployment.Rate Inflation.Rate
1                       4.6               5.8             NA
2                        NA               6.0             NA
3                        NA               6.1             NA
4                       8.0               5.7             NA
5                        NA               5.3             NA
6                        NA               5.0             NA
```

5. **What is the data type of the new "date" column?**

```
class(fed_rates$date)
```

```
[1] "character"
```

6. **Transform the "date" column to a <date> data.**

```
fed_rates <- fed_rates %>%
  mutate(date = as.Date(date, format="%Y-%m-%d"))
class(fed_rates$date)
```

```
[1] "Date"
```

7. **Conduct following statistics:**

(1) On which *date* has the highest unemployment rate? and the lowest?

```
rows_max <- fed_rates %>%
  filter(`Unemployment.Rate` == max(`Unemployment.Rate`)) %>%
```

```r
  select(date)
print(rows_max)
```

```
[1] date
<0 rows> (or 0-length row.names)
```

```r
rows_min <- fed_rates %>%
  filter(`Unemployment.Rate` == min(`Unemployment.Rate`)) %>%
  select(date)
print(rows_min)
```

```
[1] date
<0 rows> (or 0-length row.names)
```

```
\(2\) (Optional) Which *decade* has the highest average unemployment rate?

Here is a template for you to create a decade column to allow you to group the data
by decade. You can use it for the optional question in Challenge#1:


::: {.cell}

```{.r .cell-code}
#fed_rates <- fed_rates |>
#  mutate(Decade = cut(Year, breaks = seq(1954, 2017, by = 10), labels =
format(seq(1954, 2017, by = 10), format = "%Y")))


##Note: the cut() a baseR function that we don't generally use. Basically, it allows
us divides the range of Year into intervals and codes the values in Year according
to which interval (1954 and 2017) they fall; the break argument specifies how we
segment the sequence of Year (by a decade)
```
:::
```