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

AUTHOR
Mehak Nargotra

PUBLISHED
February 16, 2024

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     1.1.3     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.4     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.0
✔ 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.**

```r
#Type your code here
library(tidyverse)
library(haven)
ESS_data <- read_dta("~/Desktop/DACSS 601/DACSS_601_datasets/ESS_5.dta")
ESS_data_cleaned <- ESS_data %>%
  select(1:39)
ESS_data_cleaned
```

```
# 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?

```
#Type your code here
library(haven)
   #\(1\) Create a new column named "YearOfBirth" using the information in the "age'
ESS_data_cleaned <- ESS_data_cleaned %>%
  mutate(YearOfBirth = 2023 - age)
ESS_data_cleaned
```

```
# A tibble: 52,458 × 40
    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 30 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>, …
```

```
   #\(2\) Create a new column named "adult" using the information in the "age" colun
ESS_data_cleaned <- ESS_data_cleaned %>%
  mutate(adult = case_when(age >= 18 ~ "adult", age < 18 ~ "adolescent"))
ESS_data_cleaned
```

```
# A tibble: 52,458 × 41
    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 31 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>, …
```

```
    #\(3\) Recode the "commonlaw" column: if the value is 0, recode it as "non-commo
#ESS_data_cleaned <- ESS_data_cleaned %>%

ESS_data_cleaned <- ESS_data_cleaned %>%
  mutate(commonlaw1 = recode(commonlaw, '0' = 'non-common-law', '1' = 'common-law'))
ESS_data_cleaned
```

```
# A tibble: 52,458 × 42
    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 32 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>, …
```

```
    #\(4\) Recode the "vote" column: if the value is 3, recode it as 1; if the value
ESS_data_cleaned <- ESS_data_cleaned %>%
         mutate(vote1 = case_when(
                 vote == 3 ~ 1,
                 vote <3 ~ 0))
ESS_data_cleaned
```

```
# A tibble: 52,458 × 43
    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 33 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>, …
```

```
    #\(5\) Move the column "YearOfBirth", "adult," "commonlaw" and "vote" right befo

ESS_data_cleaned <- ESS_data_cleaned %>%
    relocate(YearOfBirth, adult,commonlaw,vote, .before = essround)
ESS_data_cleaned
```

```
# A tibble: 52,458 × 43
    idno YearOfBirth adult      commonlaw vote        essround  male   age   edu
   <dbl>       <dbl> <chr>          <dbl> <dbl+lbl>      <dbl> <dbl> <dbl> <dbl>
 1 15906        2009 adolescent         0 3 [Not eli…        5     0    14     1
 2 21168        2009 adolescent         1 3 [Not eli…        5     0    14     1
 3    40        2009 adolescent         0 3 [Not eli…        5     0    14     1
 4  2108        2009 adolescent         0 3 [Not eli…        5     0    14     1
 5   519        2009 adolescent         1 2 [No]             5     0    14     1
 6  2304        2009 adolescent         0 3 [Not eli…        5     0    14     1
 7   290        2009 adolescent         0 2 [No]             5     0    14     1
 8  3977        2009 adolescent         0 3 [Not eli…        5     0    14     1
 9 23244        2009 adolescent         1 2 [No]             5     0    14     1
10 19417        2009 adolescent         1 3 [Not eli…        5     0    14     1
# ℹ 52,448 more rows
# ℹ 34 more variables: income_10 <dbl>, 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>, …
```

```
    #\(6\) Answer the question: What is the data type of the "commonlaw" column befo
class(ESS_data_cleaned$commonlaw)
```

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

```
class(ESS_data_cleaned$commonlaw1)
```

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

```
class(ESS_data_cleaned$vote)
```

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

```
class(ESS_data_cleaned$vote1)
```

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

```
print("Data type of the 'commonlaw' column before recoding is: 'numeric'")
```

```
[1] "Data type of the 'commonlaw' column before recoding is: 'numeric'"
```

```
print("Data type of the 'commonlaw' column after recoding is: 'character'")
```

```
[1] "Data type of the 'commonlaw' column after recoding is: 'character'"
```

```
print("Data type of the 'vote' column before recoding is: 'haven_labelled', 'vctrs_v
```

```
[1] "Data type of the 'vote' column before recoding is: 'haven_labelled',
'vctrs_vctr','double' which is basically a Vector."
```

```
print("Data type of the 'vote' column after recoding is: 'numeric'")
```

```
[1] "Data type of the 'vote' column after recoding is: '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.

```
#Type your code here
CompanyOffices <- tibble(
Companys = c("Boston", "Seattle", "Dallas", "Chicago", "Washington D.C.", "Californi
"Google" = c(2,4,1,1,2,3,3,2,1,2),
"Microsoft" = c(1,3,2,1,1,2,3,2,1,2),
"Optiver" =  c(0,1,0,1,0,0,0,1,0,1),
"Nvidia" = c(1,2,3,2,1,2,3,1,1,1),
"Intel" = c(2,1,1,2,3,2,3,1,2,1),
"Tesla" = c(1,3,4,1,2,2,3,3,2,1),
"Meta" = c(1,3,2,1,2,2,1,1,1,1),
"Apple" = c(2,2,0,2,2,1,0,1,2,1),
"Samsung" = c(2,1,0,2,1,2,1,2,2,1))

CompanyOffices
```

```
# A tibble: 10 × 10
   Companys        Google Microsoft Optiver Nvidia Intel Tesla  Meta Apple Samsung
   <chr>            <dbl>     <dbl>   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
 1 Boston               2         1       0      1     2     1     1     2       2
 2 Seattle              4         3       1      2     1     3     3     2       1
 3 Dallas               1         2       0      3     1     4     2     0       0
 4 Chicago              1         1       1      2     2     1     1     2       2
 5 Washington D…        2         1       0      1     3     2     2     2       1
 6 California           3         2       0      2     2     2     2     1       2
 7 San Francisco        3         3       0      3     3     3     1     0       1
 8 Las Vegas            2         2       1      1     1     3     1     1       2
 9 New York             1         1       0      1     2     2     1     2       2
10 Los Angeles          2         2       1      1     1     1     1     1       1
```

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

```
CompanyOffices_long <- CompanyOffices|>
pivot_longer(
cols = "Google":"Samsung",
names_to = "Company",
values_to = "No_of_Offices"
)
CompanyOffices_long
```

```
# A tibble: 90 × 3
   Companys Company   No_of_Offices
   <chr>    <chr>             <dbl>
 1 Boston   Google                2
 2 Boston   Microsoft             1
 3 Boston   Optiver               0
 4 Boston   Nvidia                1
 5 Boston   Intel                 2
 6 Boston   Tesla                 1
 7 Boston   Meta                  1
 8 Boston   Apple                 2
 9 Boston   Samsung               2
10 Seattle  Google                4
# 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
school_data <- data.frame(
  Names = c("Liam", "Liam", "Ethan", "Ethan", "Noah", "Noah", "Charlotte", "Charlott
  "School" = rep("Army Public School", times = 10),
  "Divison" = rep("A", times = 10),
  "Variables" = rep(c("English", "Math"), times = 5),
  "Values" = sample(70:100, 10, replace = TRUE)
)
school_data
```

```
       Names             School Divison Variables Values
1       Liam Army Public School       A   English     94
2       Liam Army Public School       A      Math     86
3      Ethan Army Public School       A   English     96
4      Ethan Army Public School       A      Math     78
5       Noah Army Public School       A   English     79
6       Noah Army Public School       A      Math     72
7  Charlotte Army Public School       A   English     81
8  Charlotte Army Public School       A      Math     83
9      Emily Army Public School       A   English     80
10     Emily Army Public School       A      Math     86
```

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

```
school_data_wide <- school_data %>%
pivot_wider(names_from = Variables, values_from = Values)
head(school_data_wide)
```

```
# A tibble: 5 × 5
  Names     School             Divison English  Math
  <chr>     <chr>              <chr>     <int> <int>
1 Liam      Army Public School A            94    86
2 Ethan     Army Public School A            96    78
3 Noah      Army Public School A            79    72
4 Charlotte Army Public School A            81    83
5 Emily     Army Public School A            80    86
```

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":**

```
#Type your code here
library(readr)
australian_data <- read_csv("~/Desktop/DACSS 601/DACSS_601_datasets/australian_data.
```

```
New names:
Rows: 150 Columns: 7
── Column specification
──────────────────────────────────────────────────────── Delimiter: "," chr
(2): District, Division dbl (5): ...1, Yes, No, Illegible, No Response
i Use `spec()` to retrieve the full column specification for this data. i
Specify the column types or set `show_col_types = FALSE` to quiet this message.
• `` -> `...1`
```

```
print(australian_data)
```

```
# A tibble: 150 × 7
    ...1 District    Yes    No Illegible `No Response` Division
   <dbl> <chr>     <dbl> <dbl>     <dbl>         <dbl> <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
7      7 Calare    54091 35779       285          25342 New South Wales Divisions
8      8 Chifley   32871 46702       263          28180 New South Wales Divisions
9      9 Cook      47505 38804       229          18713 New South Wales Divisions
10    10 Cowper    57493 38317       315          25197 New South Wales Divisions
# i 140 more rows
```

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

```
#Type your code here
 #\(1\) What is the dimension of the data (# of rows and columns)?
  dim_data <- dim(australian_data)
  print("1. Dimension of the data (# of rows and columns):")
```

```
[1] "1. Dimension of the data (# of rows and columns):"
```

```
  print(paste("Number of rows:", dim_data[1]))
```

```
[1] "Number of rows: 150"
```

```
  print(paste("Number of columns:", dim_data[2]))
```

```
[1] "Number of columns: 7"
```

```
#\(2\) What do the rows and columns mean in this data?
  print("2.   Rows and columns meaning in this data:")
```

```
[1] "2.   Rows and columns meaning in this data:"
```

```
  print("The rows represent different observations or instances in the dataset.
```

```
[1] "The rows represent different observations or instances in the dataset. Each
row represents a district within a division whereas the district column
represents the name of each district and the divison column represents the
division each district belongs to. The different other columns represents the
voting behavior in each district i.e., number of yes votes, number of no votes,
number of No answers and illegible people."
```

(1) What is the dimension of the data (# of rows and columns)?

(2) What do the rows and columns mean in this data?

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

```
#Type your code here
#\(1\) Reshape the dataset to longer format
australian_data <- australian_data[, -1]
```

```
australian_data_reshaped <- australian_data %>%
  pivot_longer(
    cols = Yes:`No Response`,
    names_to = "Response",
    values_to = "Count")
head(australian_data_reshaped)
```

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

```
#\(2\) How many districts and divisions are in the data?
australian_data_reshaped %>%
  summarise(unique_division = length(unique(Division)))
```

```
# A tibble: 1 × 1
  unique_division
            <int>
1               8
```

```
australian_data_reshaped
```

```
# A tibble: 600 × 4
   District  Division                  Response    Count
   <chr>     <chr>                     <chr>       <dbl>
 1 Banks     New South Wales Divisions Yes         37736
 2 Banks     New South Wales Divisions No          46343
 3 Banks     New South Wales Divisions Illegible     247
 4 Banks     New South Wales Divisions No Response 20928
 5 Barton    New South Wales Divisions Yes         37153
 6 Barton    New South Wales Divisions No          47984
 7 Barton    New South Wales Divisions Illegible     226
 8 Barton    New South Wales Divisions No Response 24008
 9 Bennelong New South Wales Divisions Yes         42943
10 Bennelong New South Wales Divisions No          43215
# i 590 more rows
```

```
australian_data_reshaped %>%
  summarise(unique_district = length(unique(District)))
```

```
# A tibble: 1 × 1
  unique_district
            <int>
1             150
```

```
australian_data_reshaped
```

```
# A tibble: 600 × 4
   District  Division                  Response    Count
   <chr>     <chr>                     <chr>       <dbl>
 1 Banks     New South Wales Divisions Yes         37736
 2 Banks     New South Wales Divisions No          46343
 3 Banks     New South Wales Divisions Illegible     247
 4 Banks     New South Wales Divisions No Response 20928
 5 Barton    New South Wales Divisions Yes         37153
 6 Barton    New South Wales Divisions No          47984
 7 Barton    New South Wales Divisions Illegible     226
 8 Barton    New South Wales Divisions No Response 24008
 9 Bennelong New South Wales Divisions Yes         42943
10 Bennelong New South Wales Divisions No          43215
# i 590 more rows
```

```
#\(3\) Use mutate() to create a new column "district turnout(%)". This column sh
australian_data_turnout <- australian_data %>%
  mutate(turnout = (Yes + No + Illegible) / (Yes + No + Illegible + `No Response

head(australian_data_turnout)
```

```
# A tibble: 6 × 7
  District    Yes    No Illegible `No Response` Division                 turnout
  <chr>     <dbl> <dbl>     <dbl>         <dbl> <chr>                      <dbl>
1 Banks     37736 46343       247         20928 New South Wales Divisio…   0.801
2 Barton    37153 47984       226         24008 New South Wales Divisio…   0.780
3 Bennelong 42943 43215       244         19973 New South Wales Divisio…   0.812
4 Berowra   48471 40369       212         16038 New South Wales Divisio…   0.847
5 Blaxland  20406 57926       220         25883 New South Wales Divisio…   0.752
6 Bradfield 53681 34927       202         17261 New South Wales Divisio…   0.837
```

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

#-   In total, how many people support same-sex marriage in Australia, and how m

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

#-   Which *division* has the highest approval rate (% of "yes" in the total cas

#-   Hint: Do NOT take the average of the district approval rate. Each district

    australian_data %>%
        summarise(Total_Yes = sum(Yes),
        Total_No = sum(No))
```

```
# A tibble: 1 × 2
  Total_Yes Total_No
      <dbl>    <dbl>
1   7817247  4873987
```

```
  australian_data|>
    arrange(desc(Yes))
```

```
# A tibble: 150 × 6
   District          Yes    No Illegible `No Response` Division
   <chr>           <dbl> <dbl>     <dbl>         <dbl> <chr>
 1 Canberra(d)     89590 31361       281         24399 Australian Capital Terri…
 2 Fenner(e)       85869 30159       253         26196 Australian Capital Terri…
 3 Melbourne       81287 15839       182         20154 Victoria Divisions
 4 Sydney          76144 14860       146         22093 New South Wales Divisions
 5 McEwen          73705 39007       377         26966 Victoria Divisions
 6 Grayndler       73208 18429       136         16074 New South Wales Divisions
 7 Brisbane        72812 18762       159         20656 Queensland Divisions
 8 Newcastle       71158 23999       232         19970 New South Wales Divisions
 9 Melbourne Ports 70589 15523       198         18745 Victoria Divisions
10 Higgins         70059 19375       180         16615 Victoria Divisions
# i 140 more rows
```

```
  australian_data_approval <- australian_data %>%
   mutate(approval = Yes / (Yes + No + Illegible)*100)
   australian_data_approval <- australian_data_approval %>%
   group_by(Division)%>%
   summarise(Approval = sum(Yes)/(sum(Yes) + sum(No) + sum(Illegible))*100,
   approval_incorrect = mean(approval)) %>%
   arrange(desc(Approval))
   head(australian_data_approval)
```

```
# A tibble: 6 × 3
  Division                                 Approval approval_incorrect
  <chr>                                       <dbl>              <dbl>
1 Australian Capital Territory Divisions       73.9               73.9
2 Victoria Divisions                           64.7               64.4
3 Western Australia Divisions                  63.6               63.4
4 Tasmania Divisions                           63.5               63.2
5 South Australia Divisions                    62.3               62.1
6 Queensland Divisions                         60.6               60.2
```

```
  australian_data_approval |>
   summarise(average_approval = mean(Approval),
   average_approaval_incorrect = mean(approval_incorrect))
```

```
# A tibble: 1 × 2
  average_approval average_approaval_incorrect
             <dbl>                       <dbl>
1             63.3                        63.0
```

(1) Reshape the dataset to longer format

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

(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.

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

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”:**

```
#Type your code here
  FedFundsRate <- read_csv("~/Desktop/DACSS 601/DACSS_601_datasets/FedFundsRate.csv'
```

```
Rows: 904 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (10): Year, Month, Day, Federal Funds Target Rate, Federal Funds Upper T...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
  FedFundsRate
```

```
# A tibble: 904 × 10
    Year Month   Day `Federal Funds Target Rate` `Federal Funds Upper Target`
   <dbl> <dbl> <dbl>                       <dbl>                        <dbl>
 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
 7  1955     1     1                          NA                           NA
 8  1955     2     1                          NA                           NA
 9  1955     3     1                          NA                           NA
10  1955     4     1                          NA                           NA
```

```
# i 894 more rows
# i 5 more variables: `Federal Funds Lower Target` <dbl>,
#   `Effective Federal Funds Rate` <dbl>, `Real GDP (Percent Change)` <dbl>,
#   `Unemployment Rate` <dbl>, `Inflation Rate` <dbl>
```

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

```
#Type your code here
#\(1\) What is the dimension of the data (# of rows and columns)?
  dimension <- dim(FedFundsRate)
  print(" Dimension of the data:")
```

```
[1] " Dimension of the data:"
```

```
  print(dimension)
```

```
[1] 904  10
```

```
#\(2\) What do the rows and columns mean in this data?
column_names <- colnames(FedFundsRate)
column_names
```

```
 [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"
```

```
print(" The rows in the dataset shows what were the federal funds rate, targets
```

```
[1] " The rows in the dataset shows what were the federal funds rate, targets
both lower and upper, by how much percent gdp changed and wht was the inflation
rate on a particular day of the month in an year. The columns in the dataset
represent the category of data i.e., Federal Funds Target Rate, Federal Funds
Upper Rate, Federal Funds Lower Target, Effective Federal Funds Rate, Real GDP
(Percent Change), Unemployment Rate and Inflation Rate in a Day of a onth of an
Year."
```

```
#\(3\) What is the unit of observation? In other words, what does each case mean
  print(" The unit of observation is each day.")
```

```
[1] " The unit of observation is each day."
```

(1) What is the dimension of the data (# of rows and columns)?

(2) What do the rows and columns mean in this data?

(3) What is the unit of observation? In other words, what does each case mean in this data?

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.

```
library(stringr)
fed_rates<-FedFundsRate %>%
  mutate(Date = str_c(Year, Month, Day, sep="-"))
print(fed_rates)
```

```
# A tibble: 904 × 11
    Year Month   Day `Federal Funds Target Rate` `Federal Funds Upper Target`
   <dbl> <dbl> <dbl>                       <dbl>                        <dbl>
 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
 7  1955     1     1                          NA                           NA
 8  1955     2     1                          NA                           NA
 9  1955     3     1                          NA                           NA
10  1955     4     1                          NA                           NA
# i 894 more rows
# i 6 more variables: `Federal Funds Lower Target` <dbl>,
#   `Effective Federal Funds Rate` <dbl>, `Real GDP (Percent Change)` <dbl>,
#   `Unemployment Rate` <dbl>, `Inflation Rate` <dbl>, Date <chr>
```

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

```
  FedFundsRate<-fed_rates %>%
 relocate(Date, .before = Year)
FedFundsRate
```

```
# A tibble: 904 × 11
   Date       Year Month   Day Federal Funds Target Rat…¹ Federal Funds Upper …²
   <chr>     <dbl> <dbl> <dbl>                      <dbl>                  <dbl>
 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
 7 1955-1-1    1955     1     1                         NA                     NA
 8 1955-2-1    1955     2     1                         NA                     NA
 9 1955-3-1    1955     3     1                         NA                     NA
10 1955-4-1    1955     4     1                         NA                     NA
# i 894 more rows
# i abbreviated names: ¹`Federal Funds Target Rate`,
#   ²`Federal Funds Upper Target`
# i 5 more variables: `Federal Funds Lower Target` <dbl>,
#   `Effective Federal Funds Rate` <dbl>, `Real GDP (Percent Change)` <dbl>,
#   `Unemployment Rate` <dbl>, `Inflation Rate` <dbl>
```

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

```
#Type your code here
print("Data Type of the 'date' column is: character. ")
```

```
[1] "Data Type of the 'date' column is: character. "
```

```
print(class(FedFundsRate$Date))
```

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

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

```
#Type your code here
FedFundsRate$Date <- as.Date(fed_rates$Date)
print(class(FedFundsRate$Date))
```

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

```
print("Data Type of the 'date' column is: Date ")
```

```
[1] "Data Type of the 'date' column is: Date "
```

```
FedFundsRate
```

```
# A tibble: 904 × 11
   Date        Year Month   Day Federal Funds Target Ra…¹ Federal Funds Upper …²
   <date>     <dbl> <dbl> <dbl>                     <dbl>                  <dbl>
 1 1954-07-01  1954     7     1                        NA                     NA
 2 1954-08-01  1954     8     1                        NA                     NA
 3 1954-09-01  1954     9     1                        NA                     NA
 4 1954-10-01  1954    10     1                        NA                     NA
 5 1954-11-01  1954    11     1                        NA                     NA
 6 1954-12-01  1954    12     1                        NA                     NA
 7 1955-01-01  1955     1     1                        NA                     NA
 8 1955-02-01  1955     2     1                        NA                     NA
 9 1955-03-01  1955     3     1                        NA                     NA
10 1955-04-01  1955     4     1                        NA                     NA
# i 894 more rows
# i abbreviated names: ¹`Federal Funds Target Rate`,
#   ²`Federal Funds Upper Target`
# i 5 more variables: `Federal Funds Lower Target` <dbl>,
#   `Effective Federal Funds Rate` <dbl>, `Real GDP (Percent Change)` <dbl>,
#   `Unemployment Rate` <dbl>, `Inflation Rate` <dbl>
```

7. **Conduct following statistics:**

```
#Type your code here
#\(1\) On which *date* has the highest unemployment rate? and the lowest?
highest_unemployment_date <- FedFundsRate %>%
  filter(`Unemployment Rate` == max(`Unemployment Rate`, na.rm = TRUE)) %>%
  pull(Date)
  highest_unemployment_date
```

```
[1] "1982-11-01" "1982-12-01"
```

```
lowest_unemployment_date <- FedFundsRate %>%
  filter(`Unemployment Rate` == min(`Unemployment Rate`, na.rm = TRUE)) %>%
  pull(Date)
  lowest_unemployment_date
```

```
[1] "1968-09-01" "1968-10-01" "1968-11-01" "1968-12-01" "1969-01-01"
[6] "1969-02-01" "1969-03-01" "1969-04-01" "1969-05-01"
```

```
#\(2\) (Optional) Which *decade* has the highest average unemployment rate?
FedFundsRate <- FedFundsRate %>%
  mutate(Decade = cut(Year, breaks = seq(1950, 2020, by = 10), format = "%Y")) %
  group_by(Decade) %>%
  mutate(mean = mean(`Unemployment Rate`, na.rm = TRUE))%>%
  arrange(desc(mean))
  head(FedFundsRate)
```

```
# A tibble: 6 × 13
# Groups:   Decade [1]
  Date        Year Month   Day Federal Funds Target Rat…¹ Federal Funds Upper …²
  <date>     <dbl> <dbl> <dbl>                      <dbl>                  <dbl>
1 1981-01-01  1981     1     1                         NA                     NA
2 1981-02-01  1981     2     1                         NA                     NA
3 1981-03-01  1981     3     1                         NA                     NA
4 1981-04-01  1981     4     1                         NA                     NA
5 1981-05-01  1981     5     1                         NA                     NA
6 1981-06-01  1981     6     1                         NA                     NA
# i abbreviated names: ¹`Federal Funds Target Rate`,
#   ²`Federal Funds Upper Target`
# i 7 more variables: `Federal Funds Lower Target` <dbl>,
#   `Effective Federal Funds Rate` <dbl>, `Real GDP (Percent Change)` <dbl>,
#   `Unemployment Rate` <dbl>, `Inflation Rate` <dbl>, Decade <fct>, mean <dbl>
```

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

(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:

```
#fed_rates <- fed_rates |>
#  mutate(Decade = cut(Year, breaks = seq(1954, 2017, by = 10), labels = format(


##Note: the cut() a baseR function that we don't generally use. Basically, it al
```