Challenge_3: Joining Relational Data, Writing Your Own Functions, and String Operations

AUTHOR
Mehak Nargotra

PUBLISHED
March 1, 2024

Setup

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

```
library(tidyverse)
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

In this challenge, we will practice `join()` with relational data, use string functions to process, extract information, and mutate and clean data. We will also practice wring own functions.

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.

- Part 1 and 2: ESS_5.dta and p5v2018.sav (used in Challenge#1) ⭐⭐
- Part 3: babynames.csv (used in Challenge#1) ⭐

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

Part 1. Joining Individual-level and Country-Level Data

We have been working with ESS and Polity datasets in the previous two challenges and should be familiar with them. Suppose we have a research project that studies European citizens' social behaviors and public opinions, and we are interested in **how the countries that respondents live in influence their behavior and opinion**. In this case, we will need to combine the two data for future analysis.

1. **Read the two raw datasets.**

For ESS_5: (1) keep only the following columns: *idno, essround, male, age, edu, eth_major, income_10, cntry, vote.* **(2)** recode *essround* to 2010, and rename it as *year*.

For Polity V, keep the first 10 columns.

```
#Type your code here
ESS_data <- read_dta("~/Desktop/DACSS 601/DACSS_601_datasets/ESS_5.dta")
ESS_5_data <- ESS_data %>%
  select(idno, essround, male, age, edu, eth_major, income_10, cntry, vote)
ESS_5_data <- ESS_5_data %>%
  mutate(essround = case_when(essround == 5 ~ 2010,
                              TRUE ~ essround))
ESS_5_data
```

```
# A tibble: 52,458 × 9
    idno essround  male   age   edu eth_major income_10 cntry vote
   <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <chr> <dbl+lbl>
 1 15906     2010     0    14     1         1         2 GR    3 [Not eligible t…
 2 21168     2010     0    14     1         1         2 IE    3 [Not eligible t…
 3    40     2010     0    14     1        NA         8 LT    3 [Not eligible t…
 4  2108     2010     0    14     1         1        NA RU    3 [Not eligible t…
 5   519     2010     0    14     1         1        NA IL    2 [No]
 6  2304     2010     0    14     1         1        NA ES    3 [Not eligible t…
 7   290     2010     0    14     1         1        NA PT    2 [No]
 8  3977     2010     0    14     1         1        NA BG    3 [Not eligible t…
 9 23244     2010     0    14     1         1        NA IE    2 [No]
10 19417     2010     0    14     1         1        NA IE    3 [Not eligible t…
# i 52,448 more rows
```

```
p5v2018 <- read_sav("~/Desktop/DACSS 601/DACSS_601_datasets/p5v2018.sav")
polity_v <- p5v2018[, 1:10]
polity_v
```

```
# A tibble: 17,574 × 10
      p5   cyear ccode scode country      year  flag fragment democ autoc
   <dbl>   <dbl> <dbl> <chr> <chr>       <dbl> <dbl>    <dbl> <dbl> <dbl>
 1     0 7001800   700 AFG   Afghanistan  1800     0       NA     1     7
 2     0 7001801   700 AFG   Afghanistan  1801     0       NA     1     7
 3     0 7001802   700 AFG   Afghanistan  1802     0       NA     1     7
 4     0 7001803   700 AFG   Afghanistan  1803     0       NA     1     7
 5     0 7001804   700 AFG   Afghanistan  1804     0       NA     1     7
 6     0 7001805   700 AFG   Afghanistan  1805     0       NA     1     7
 7     0 7001806   700 AFG   Afghanistan  1806     0       NA     1     7
 8     0 7001807   700 AFG   Afghanistan  1807     0       NA     1     7
 9     0 7001808   700 AFG   Afghanistan  1808     0       NA     1     7
10     0 7001809   700 AFG   Afghanistan  1809     0       NA     1     7
# i 17,564 more rows
```

2. **Answer the following questions:**

(1) In this project, what is our unit of analysis? Which is the primary data, and which is the foreign data?

(2) What is(are) the key(s) for the two data?

```
#\(1\) In this project, what is our unit of analysis? Which is the primary data, and
print("In this project, the unit of analysis is the individual respondents that is t
The primary data is the ESS (European Social Survey) dataset (ESS_5), which contains
```

```
[1] "In this project, the unit of analysis is the individual respondents that is the
European citizens. \nThe primary data is the ESS (European Social Survey) dataset
(ESS_5), which contains information about individual respondents social behaviors
and opinions to investigate how they interact with Europes changing institutions.
The foreign data is the Polity V dataset, which provides information about the
political characteristics of all major, independent states in the global system."
```

```
# \(2\) What is(are) the key(s) for the two data?
print("In ESS_5 dataset, 'idno' is the key. In Polity_v dataset, 'cyear' and 'countr
```

```
[1] "In ESS_5 dataset, 'idno' is the key. In Polity_v dataset, 'cyear' and 'country'
is the key."
```

3. **Suppose we have a theory that a country's level of democracy (*democ* in Polity V) affects an individual's electoral participation (*vote* in ESS 5). We must first conduct some necessary data transformation before merging the two data.**

   (1) Countries in ESS_5 are coded with their 2-digit codes (ISO-3166-1) in the *cntry* column. It is difficult to identify from these two-letter abbreviations. Let's first transform the *cntry* column by changing it from the abbreviations to the full country names and renaming the column as *country.*

   Please refer to this website for the list of countries with their 2-letter abbreviations. Read the country list (csv) file, into RStudio, and merge it with the ESS_5 data. By doing so, you add a new "country" column to the existing ESS_5 data.

   ```
   #Type your code here
   country_data <- read_csv("~/Desktop/DACSS 601/DACSS_601_datasets/data.csv")
   ```

   ```
   Rows: 249 Columns: 2
   ── Column specification ────────────────────────────────────────────────────────
   Delimiter: ","
   chr (2): Name, Code

   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.
   ```

   ```
   country_data
   ```

   ```
   # A tibble: 249 × 2
      Name                Code
      <chr>               <chr>
    1 Afghanistan         AF
    2 Åland Islands       AX
   ```

```
 3 Albania             AL
 4 Algeria             DZ
 5 American Samoa      AS
 6 Andorra             AD
 7 Angola              AO
 8 Anguilla            AI
 9 Antarctica          AQ
10 Antigua and Barbuda AG
# ℹ 239 more rows
```

```r
ess_5_merged <- ESS_5_data %>%
  left_join(country_data, by = c("cntry" = "Code")) %>%
  rename(Country = Name)
ess_5_merged
```

```
# A tibble: 52,458 × 10
    idno essround  male   age   edu eth_major income_10 cntry vote       Country
   <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <chr> <dbl+lbl>  <chr>
 1 15906     2010     0    14     1         1         2 GR    3 [Not el… Greece
 2 21168     2010     0    14     1         1         2 IE    3 [Not el… Ireland
 3    40     2010     0    14     1        NA         8 LT    3 [Not el… Lithua…
 4  2108     2010     0    14     1         1        NA RU    3 [Not el… Russia…
 5   519     2010     0    14     1         1        NA IL    2 [No]     Israel
 6  2304     2010     0    14     1         1        NA ES    3 [Not el… Spain
 7   290     2010     0    14     1         1        NA PT    2 [No]     Portug…
 8  3977     2010     0    14     1         1        NA BG    3 [Not el… Bulgar…
 9 23244     2010     0    14     1         1        NA IE    2 [No]     Ireland
10 19417     2010     0    14     1         1        NA IE    3 [Not el… Ireland
# ℹ 52,448 more rows
```

```r
dim(ess_5_merged)
```

```
[1] 52458    10
```

(2) What column(s) will we use as a matching key(s) for combining the updated ESS_5 dataset and Polity V dataset? Note: you can use multiple matching strategies, but I suggest we create a common matching key for both data if there are none.

```r
print("We will use country column and essround column from the ESS_5 dataset. From t
```

```
[1] "We will use country column and essround column from the ESS_5 dataset. From the Polity V dataset, we will use country and year column. Since, ESS_5 dataset contains only data of year 2010, we want to filter and fetch only that dataset from Polity V. Hence, using year column."
```

\(3\) Join the two data (updated ESS_5 and Polity V). Please print the first few entries as a sanity check. Name the joined data as "ESS_Polity"

::: {.cell}

```{.r .cell-code}
```

```
#Type your code here

ESS_Polity <- ess_5_merged %>%
  left_join(polity_v, by = c("Country" = "country", "essround" = "year" ))
head(ESS_Polity)
```

::: {.cell-output .cell-output-stdout}
```
# A tibble: 6 × 18
   idno essround  male   age   edu eth_major income_10 cntry vote        Country
  <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <chr> <dbl+lbl>   <chr>
1 15906     2010     0    14     1         1         2 GR    3 [Not eli… Greece
2 21168     2010     0    14     1         1         2 IE    3 [Not eli… Ireland
3    40     2010     0    14     1        NA         8 LT    3 [Not eli… Lithua…
4  2108     2010     0    14     1         1        NA RU    3 [Not eli… Russia…
5   519     2010     0    14     1         1        NA IL    2 [No]      Israel
6  2304     2010     0    14     1         1        NA ES    3 [Not eli… Spain
# ℹ 8 more variables: p5 <dbl>, cyear <dbl>, ccode <dbl>, scode <chr>,
#   flag <dbl>, fragment <dbl>, democ <dbl>, autoc <dbl>
```
:::
:::

\(4\) Save the joined data *ESS_Polity* to your local directory using the following code. We will be using this joined data to explore visualization in future challenges. (This is for future usage. No need to submit the saved joined data.)

::: {.cell}

```{.r .cell-code}
write_csv(ESS_Polity, "ESS_Polity.csv")
```
:::

4. **Describe the data structure of the newly joined data *ESS_Polity*. What is its dimension (# of rows and # of columns)? What is its unit of observation? Compared to the original ESS_5 data, does the above data combination change the dimension and unit of observation?**

```
#Type your code here
ESS_Polity_dimensions <- dim(ESS_Polity)
print("1. Dimension of the ESS_Polity data (# of rows and columns):")
```

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

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

```
[1] "Number of rows: 52458"
```

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

```
[1] "Number of columns: 18"
```

```
ESS_5_dimensions <- dim(ESS_5_data)
print("1. Dimension of the ESS_5 data (# of rows and columns):")
```

```
[1] "1. Dimension of the ESS_5 data (# of rows and columns):"
```

```
print(paste("Number of rows:", ESS_5_dimensions[1]))
```

```
[1] "Number of rows: 52458"
```

```
print(paste("Number of columns:", ESS_5_dimensions[2]))
```

```
[1] "Number of columns: 9"
```

```
print("The ESS_Polity data has more number of columns as compared to ESS_5 data.
```

```
[1] "The ESS_Polity data has more number of columns as compared to ESS_5 data.
This is beacuse the ESS_5 data has been left joined with the polity data which
has 10 columns and joining with the ESS_5 data (comprising of 9 columns) added
more columns to the ESS_Polity dataset. We can see the additional 9 columns
after merging the 2 datasets by the year and country columns."
```

Part 2. Writing Your Own Functions

Please use the joined data **ESS_Polity* in Part 1 and write** ONE* function to complete all the following tasks:

(1) Estimate the range, average, standard deviation, number of NAs, and the number of unique values of any given numeric-type (double or integer) columns.

(2) Test your function with any four columns of your choice.

```
print("Attempted this question in 2 ways. (1) by printing data inside the function f
      (2) by creating a tibble and returning a summary of all the values together.")
```

```
[1] "Attempted this question in 2 ways. (1) by printing data inside the function for
every column.  \n      (2) by creating a tibble and returning a summary of all the
values together."
```

```
print("METHOD (1)")
```

```
[1] "METHOD (1)"
```

```r
estimated_statistics <- function(data, columns) {
  for (col in columns) {
    col_data <- data[[col]]

      print("*****************************************************")
      range <- range(col_data, na.rm = TRUE)
      print("range: ")
      print(range)
      avg <- mean(col_data, na.rm = TRUE)
      print("average: ")
      print(avg)
      sd <- sd(col_data, na.rm = TRUE)
      print("standard deviation")
      print(sd)
      nas <- sum(is.na(col_data))
      print("number of NAs")
      print(nas)
      unique_values <- length(unique(col_data))
      print("number of unique values")
      print(unique_values)
  }
}

test_columns <- c('age', 'income_10', 'eth_major', 'cyear')
estimated_statistics(ESS_Polity, test_columns)
```

```
[1] "*****************************************************"
[1] "range: "
[1]  14 101
[1] "average: "
[1] 47.91529
[1] "standard deviation"
[1] 18.79573
[1] "number of NAs"
[1] 137
[1] "number of unique values"
[1] 88
[1] "*****************************************************"
[1] "range: "
[1]  1 10
[1] "average: "
[1] 5.048622
[1] "standard deviation"
[1] 2.787532
[1] "number of NAs"
[1] 12620
[1] "number of unique values"
[1] 11
[1] "*****************************************************"
[1] "range: "
[1] 0 1
[1] "average: "
```

```
[1] 0.9369868
[1] "standard deviation"
[1] 0.2429891
[1] "number of NAs"
[1] 1310
[1] "number of unique values"
[1] 3
[1] "**********************************************************"
[1] "range: "
[1] 2002010 6662010
[1] "average: "
[1] 3164547
[1] "standard deviation"
[1] 1028503
[1] "number of NAs"
[1] 4451
[1] "number of unique values"
[1] 26
```

```r
print("METHOD (2)")
```

```
[1] "METHOD (2)"
```

```r
estimated_statistics <- function(data, columns) {
  summary <- tibble(
    column = character(),
    range = numeric(),
    average = numeric(),
    sd = numeric(),
    na = numeric(),
    unique_val = numeric()
  )

  for (col in columns) {
    col_data <- data[[col]]

    range_val <- range(col_data, na.rm = TRUE)
    avg <- mean(col_data, na.rm = TRUE)
    sd_val <- sd(col_data, na.rm = TRUE)
    nas <- sum(is.na(col_data))
    unique_vals <- length(unique(col_data))

    summary <- bind_rows(summary, tibble(
      column = col,
      range = range_val[2] - range_val[1],
      average = avg,
      sd = sd_val,
      na = nas,
      unique_val = unique_vals
    ))
  }
```

```
  return(summary)
}

#range -> tibble
test_columns <- c('age', 'income_10', 'eth_major', 'cyear')
estimated_statistics(ESS_Polity, test_columns)
```

```
# A tibble: 4 × 6
  column      range     average          sd    na unique_val
  <chr>       <dbl>       <dbl>       <dbl> <dbl>      <dbl>
1 age            87      47.9        18.8     137         88
2 income_10       9       5.05        2.79  12620         11
3 eth_major       1       0.937       0.243  1310          3
4 cyear     4660000 3164547.    1028503.     4451         26
```

Part 3. Practicing Stringr Package with Babynames

1. **Import the babynames data:**

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

```
Rows: 2084710 Columns: 4
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Name, Sex
dbl (2): Occurrences, Year

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.
```

```
  head(babynames)
```

```
# A tibble: 6 × 4
  Name      Sex    Occurrences  Year
  <chr>     <chr>        <dbl> <dbl>
1 Mary      Female        7065  1880
2 Anna      Female        2604  1880
3 Emma      Female        2003  1880
4 Elizabeth Female        1939  1880
5 Minnie    Female        1746  1880
6 Margaret  Female        1578  1880
```

2. **Use different string functions from stringr package to answer the following questions:**

   (1) Find and list the longest names using count() and a string function.

   (2) Use a string function to detect if the following names are present in the data:

   “Ronaldo”, “Messi”, “Wayne”, “Clarck”, “Rick”, and “Morty”.

```
    #Use the Anchoring (a way of regular expression), "^name$", to specify the name

#\(1\) Find and list the longest names using [count()](https://dplyr.tidyverse.org/r
longest_names <- babynames %>%
  mutate(name_length = str_length(Name)) %>%
  count(Name, name_length)

max_length <- max(longest_names$name_length)

longest_names <- longest_names %>%
  filter(name_length == max_length)

print("Longest names are:    ")
```

```
[1] "Longest names are:    "
```

```
print(longest_names)
```

```
# A tibble: 37 × 3
   Name            name_length     n
   <chr>                 <int> <int>
 1 Ashleyelizabeth          15     1
 2 Christianalexan          15     1
 3 Christiananthon          15     2
 4 Christiandaniel          15     1
 5 Christianjoseph          15     4
 6 Christianjoshua          15     1
 7 Christianmichae          15     2
 8 Christopheranth          15     1
 9 Christopherdavi          15     1
10 Christopherjame          15    16
# ℹ 27 more rows
```

```
num_unique_longest_names <- longest_names %>%
  summarize(num_unique = n_distinct(Name))
print("Number of unique longest names")
```

```
[1] "Number of unique longest names"
```

```
print(num_unique_longest_names)
```

```
# A tibble: 1 × 1
  num_unique
       <int>
1         37
```

```
num_people_longest_names <- sum(longest_names$n)
print("Number of people with longest names")
```

```
[1] "Number of people with longest names"
```

```r
print(num_people_longest_names)
```

```
[1] 138
```

```r
#\(2\) Use a string function to detect if the following names are present in the dat
# "Ronaldo", "Messi", "Wayne", "Clarck", "Rick", and "Morty".
presence_check <- babynames %>%
  mutate(name_presence = str_detect(Name, "^Ronaldo$|^Messi$|^Wayne$|^Clarck$|^Rick$
  arrange(desc(name_presence))
print(presence_check)
```

```
# A tibble: 2,084,710 × 5
   Name  Sex   Occurrences  Year name_presence
   <chr> <chr>       <dbl> <dbl> <lgl>
 1 Wayne Male           23  1880 TRUE
 2 Wayne Male           30  1881 TRUE
 3 Wayne Male           22  1882 TRUE
 4 Wayne Male           25  1883 TRUE
 5 Wayne Male           34  1884 TRUE
 6 Wayne Male           24  1885 TRUE
 7 Wayne Male           23  1886 TRUE
 8 Wayne Male           29  1887 TRUE
 9 Wayne Male           41  1888 TRUE
10 Wayne Male           32  1889 TRUE
# i 2,084,700 more rows
```

\(3\) Create a column *LastName* with just one value, "LastName". Next, create another column *FullName,* by combing the strings of columns *name* and *LastName*, separating by a period. For example, a value in this new column should be like "Jacky.LastName".

\(4\) Find all "Elizabeth" in the data and replace "Elizabeth" with "Liz".

```r
#Type your code here
#\(3\) Create a column *LastName* with just one value, "LastName". Next, create anot
babynames <- babynames %>%
  mutate(LastName = str_c("LastName"),
         FullName = str_c(Name, LastName, sep = "."))
print(babynames)
```

```
# A tibble: 2,084,710 × 6
   Name      Sex    Occurrences  Year LastName FullName
   <chr>     <chr>        <dbl> <dbl> <chr>    <chr>
 1 Mary      Female        7065  1880 LastName Mary.LastName
 2 Anna      Female        2604  1880 LastName Anna.LastName
 3 Emma      Female        2003  1880 LastName Emma.LastName
 4 Elizabeth Female        1939  1880 LastName Elizabeth.LastName
 5 Minnie    Female        1746  1880 LastName Minnie.LastName
 6 Margaret  Female        1578  1880 LastName Margaret.LastName
 7 Ida       Female        1472  1880 LastName Ida.LastName
 8 Alice     Female        1414  1880 LastName Alice.LastName
 9 Bertha    Female        1320  1880 LastName Bertha.LastName
```

```
10 Sarah     Female        1288  1880 LastName Sarah.LastName
# ℹ 2,084,700 more rows
```

```r
#\(4\) Find all "Elizabeth" in the data and replace "Elizabeth" with "Liz".
babynames <- babynames %>%
  mutate(
    Name = str_replace_all(Name, "Elizabeth", "Liz"),
    FullName = str_replace_all(FullName, "Elizabeth", "Liz")
  )
print(babynames)
```

```
# A tibble: 2,084,710 × 6
   Name     Sex    Occurrences  Year LastName FullName
   <chr>    <chr>        <dbl> <dbl> <chr>    <chr>
 1 Mary     Female        7065  1880 LastName Mary.LastName
 2 Anna     Female        2604  1880 LastName Anna.LastName
 3 Emma     Female        2003  1880 LastName Emma.LastName
 4 Liz      Female        1939  1880 LastName Liz.LastName
 5 Minnie   Female        1746  1880 LastName Minnie.LastName
 6 Margaret Female        1578  1880 LastName Margaret.LastName
 7 Ida      Female        1472  1880 LastName Ida.LastName
 8 Alice    Female        1414  1880 LastName Alice.LastName
 9 Bertha   Female        1320  1880 LastName Bertha.LastName
10 Sarah    Female        1288  1880 LastName Sarah.LastName
# ℹ 2,084,700 more rows
```