LibDataMarkDown

Ohood Al Qeshawi

2023-10-26

Assignment 2

#Question Num.1 #Import all the datasets in R and save them in separate objects.

Ontario Library Data year 2019 to year 2022

```
#Including library
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.3     ✓ readr     2.1.4
## ✓ 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(dplyr)
library(tidyr)
```

```
#installing package reader, to resolve a problem of special characters in the data
#install.packages("readr")
library(readr)
```

```
#reading data into objects

LibData2019 <- read.csv("LibData2019.csv", header = TRUE, fileEncoding = "UTF-8")
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## invalid input found on input connection 'LibData2019.csv'
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## Incomplete final line found by readTableHeader on 'LibData2019.csv'
```

```
#Reading data from 2019 had some problems and the following is the solution

##############################
#install.packages("stringi")
library(stringi)

# Read the CSV file as raw text
raw_text <- readLines("LibData2019.csv", warn = FALSE, encoding = "UTF-8")

# Replace or remove problematic characters
cleaned_text <- stri_trans_general(raw_text, "Latin-ASCII")

# Write the cleaned text back to the file
writeLines(cleaned_text, "LibData2019_cleaned.csv", useBytes = TRUE)

# Read the cleaned CSV file
LibData2019 <- read.csv("LibData2019_cleaned.csv", header = TRUE)

##############################

LibData2020 <- read.csv("LibData2020.csv")
LibData2021 <- read.csv("LibData2021.csv")

#encountered error in reading data from 2022
#the solution was using skip=1
LibData2022 <- read.csv("LibData2022.csv", skip = 1)

#(colnames(LibData2022))
(ncol(LibData2022))
```

```
## [1] 329
```

```
(nrow((LibData2022)))
```

```
## [1] 365
```

#select The required columns

```
selected_columns <- c("Survey.Year.From","A1.10.City.Town","Library.Number","A1.14..No..of.Active.Library.Cardholders","B2.9..Total.Operating.Revenues")
```

#Take a subset of each year with the selected columns

```
LibData2019 <- LibData2019[selected_columns]
LibData2020 <- LibData2020[selected_columns]
LibData2021 <- LibData2021[selected_columns]
LibData2022 <- LibData2022[selected_columns]
```

#Question Num.2 ##Create an object that merges all the files into one object. This is a tip in how to be efficient with your data – instead of having 4 separate objects, why not store them into 1 object? #data from 2019 to 2022 is included in object data_combined
##

```
common_columns <- Reduce(intersect, list(colnames(LibData2019),colnames(LibData2020),colnames(LibData2021),colnames(LibData2022)))

data_combined <- rbind(

  subset(LibData2019, select = common_columns),

  subset(LibData2020, select = common_columns),

  subset(LibData2021, select = common_columns),

  subset(LibData2022, select = common_columns)

)

(nrow(data_combined))
```

```
## [1] 1471
```

```
#remove NA values from the combined dataframe
data_combined_clean <- na.omit(data_combined)
(head(data_combined_clean))
```

```
##   Survey.Year.From      A1.10.City.Town Library.Number
## 1             2019             Flinton         L0005
## 2             2019           Alliston         L0003
## 3             2019            Douglas         L0002
## 4             2019               Ajax         L0032
## 5             2019 Township of Alberton         L1098
## 6             2019          Roseneath         L0390
##   A1.14..No..of.Active.Library.Cardholders B2.9..Total.Operating.Revenues
## 1                                      910                          92,020
## 2                                        0                          23,324
## 3                                      417                          41,516
## 4                                   37,004                       5,671,368
## 5                                        0                           4,000
## 6                                      190                          30,478
```

```
(nrow(data_combined_clean))
```

```
## [1] 1469
```

```
#(missing_values <- is.na(data_combined_clean))
```

#change the names of the columns

```
#rename the column "A1.10 City/Town" into city
#rename the column "survey year from" to year
#rename column A1.14.No..of.Active.Library.Cardholders into NumActiveCardHolders
#rename column Library.Number into LibraryNumber
#rename column B2.9..Total.Operating.Revenues into TotalOperatingRevenue

data_combined_clean <- data_combined_clean %>%
  rename(year = Survey.Year.From,
         city = A1.10.City.Town,
         NumActiveCardHolders = A1.14..No..of.Active.Library.Cardholders,
         LibraryNumber = Library.Number,
         TotalOperatingRevenue=B2.9..Total.Operating.Revenues
         )

#change its datatype from char to numeric
data_combined_clean <- data_combined_clean %>%
  mutate(NumActiveCardHolders = as.numeric(NumActiveCardHolders))
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `NumActiveCardHolders = as.numeric(NumActiveCardHolders)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
#it seems there is still na values in NumActiveCardHOlders Column
#data_combined_clean$NumActiveCardHolders <- #na.omit(data_combined_clean$NumActiveCardHolders)

#check how many row in each year
(data_combined_clean %>% group_by(year) %>% tally())
```

```
## # A tibble: 4 × 2
##    year     n
##   <int> <int>
## 1  2019   373
## 2  2020   369
## 3  2021   363
## 4  2022   364
```

#Question Num.3 #Write a sequence of code which will create a single data set that can be used to output a table that lists the number of libraries in each city for the selected years.

```
#group the combined data by city and year
city_library_counts <- data_combined_clean %>%
  group_by(city, year) %>%
  summarise(number_of_libraries = n()) %>%
  ungroup() %>%
  spread(key = year, value = number_of_libraries, fill = 0)
```

```
## `summarise()` has grouped output by 'city'. You can override using the
## `.groups` argument.
```

```
city_library_counts
```

```
## # A tibble: 334 x 5
##    city          `2019` `2020` `2021` `2022`
##    <chr>          <dbl>  <dbl>  <dbl>  <dbl>
##  1 Addison            1      1      1      1
##  2 Ajax               1      1      1      1
##  3 Algoma Hills       1      1      1      1
##  4 Alliston           2      2      2      2
##  5 Almonte            1      1      1      1
##  6 Amaranth           1      1      1      1
##  7 Angus              1      1      1      1
##  8 Apsley             1      1      1      1
##  9 Arnprior           2      2      2      2
## 10 Astor-ville        1      1      1      1
## # ℹ 324 more rows
```

#this is to make sure the result is right #i will check the number of libraries in city Arnprior in year 2020

```
(Alison2020Libraries = data_combined_clean %>% filter(year == 2020, city == "Arnprior") %>% nrow())
```

```
## [1] 2
```

#Question Num.4 #Write a sequence of code that shows the total number of active cardholders for each library for the selected years

```
NumberOfCardHoldersEachLibraryEachYear <- data_combined_clean %>%
  group_by(LibraryNumber, year) %>%
  summarise(number_of_card_holders =sum(NumActiveCardHolders)) %>%
  spread(key = year, value = number_of_card_holders, fill = 0) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'LibraryNumber'. You can override using the
## `.groups` argument.
```

```
NumberOfCardHoldersEachLibraryEachYear
```

```
## # A tibble: 377 x 5
##    LibraryNumber `2019` `2020` `2021` `2022`
##    <chr>          <dbl>  <dbl>  <dbl>  <dbl>
##  1 L0002            417    366    342    333
##  2 L0003              0      0      0      0
##  3 L0004              0      0      0      0
##  4 L0005            910    910    925    969
##  5 L0010              0      0      0      0
##  6 L0011              0      0      0      0
##  7 L0012              0      0      0      0
##  8 L0013            258    260    159    292
##  9 L0014              0      0      0      0
## 10 L0016            606    565    636    812
## # ℹ 367 more rows
```

#Question Num.5 #Write a sequence of code that lists the top 5 libraries with the highest average Total Operating Revenues from selected years

```
#for this we need the column TotalOperatingRevenue

Top5LibrariesTOR <- data_combined_clean %>%
  group_by(year) %>%  # Group by YEAR if you want to calculate the average for each library
  summarise(AvgTOR = mean(as.numeric(TotalOperatingRevenue), na.rm = TRUE)) %>%  # Calculate average TotalOperatingRevenue a fter casting it to numeric, for each year
  arrange(desc(AvgTOR)) %>%  # Arrange by descending average TotalOperatingRevenue
  slice_head(n = 5)
```

```
## Warning: There were 4 warnings in `summarise()`.
## The first warning was:
## ℹ In argument: `AvgTOR = mean(as.numeric(TotalOperatingRevenue), na.rm =
##   TRUE)`.
## ℹ In group 1: `year = 2019`.
## Caused by warning in `mean()`:
## ! NAs introduced by coercion
## ℹ Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```
Top5LibrariesTOR
```

```
## # A tibble: 4 × 2
##    year AvgTOR
##   <int>  <dbl>
## 1  2022  509.
## 2  2019  508.
## 3  2020  508.
## 4  2021  501.
```