ForteLab12.R

andre

2022-10-02

#install.packages("RCurl")  
## Loading required package: bitops  
#install.packages("RJSONIO")  
#install.packages("tidyverse")  
#install.packages("dplyr")  
library(RCurl)

## Warning: package 'RCurl' was built under R version 4.2.1

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.1

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(RJSONIO)  
library(sqldf)

## Warning: package 'sqldf' was built under R version 4.2.1

## Loading required package: gsubfn

## Warning: package 'gsubfn' was built under R version 4.2.1

## Loading required package: proto

## Warning: package 'proto' was built under R version 4.2.1

## Loading required package: RSQLite

## Warning: package 'RSQLite' was built under R version 4.2.1

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.2.1

## ── Attaching packages  
## ───────────────────────────────────────  
## tidyverse 1.3.2 ──

## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.8 ✔ stringr 1.4.1  
## ✔ tidyr 1.2.0 ✔ forcats 0.5.2  
## ✔ readr 2.1.2

## Warning: package 'ggplot2' was built under R version 4.2.1

## Warning: package 'tibble' was built under R version 4.2.1

## Warning: package 'tidyr' was built under R version 4.2.1

## Warning: package 'readr' was built under R version 4.2.1

## Warning: package 'purrr' was built under R version 4.2.1

## Warning: package 'stringr' was built under R version 4.2.1

## Warning: package 'forcats' was built under R version 4.2.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ tidyr::complete() masks RCurl::complete()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(lubridate)

## Warning: package 'lubridate' was built under R version 4.2.1

##   
## Attaching package: 'lubridate'  
##   
## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

fileURL <- "https://opendata.maryland.gov/resource/rqid-652u.json"  
#fileURL <- "http://opendata.maryland.gov/resource/rqid-652u.json"  
#just in case the link above does not work, use the json file provided instead.   
  
######### 1. investigate the url above.  
######### 2. covert the JSON format dataset into R objects  
  
mydata <- fromJSON(fileURL)  
  
# look into the data summary  
  
#summary(mydata)  
  
######### 3. what is the data type of "mydata" after reading the URL using the appropriate function.   
  
# str(mydata) # Investigating the structure of mydata reveals that it is a nested list.  
  
######## 4. Print the number of rows below.  
  
numRows <- sapply(mydata, nrow)  
  
# nameList1 <-names(mydata[[1]])  
  
# Step 2: Clean the data  
  
#If you investigate mydata, you will find the length of each element is different. We must make it consistent.  
#try map\_df function to flatten mydata to a dataframe  
  
mydata2 <- map(mydata, as.list)  
mydata2 <- map\_df(mydata2, flatten)  
  
#then create a dataframe from it.  
  
df1 <- data.frame(mydata2, stringsAsFactors = FALSE)  
  
#change the name of each variables with meaningful names (column names of the original data)  
#extract meaningful names from the original data. You need to create two separate columns to include both longitude and latitude  
#The total number of columns would be seven columns  
#create day of week variable  
#convert characters to the proper data formats (numeric or date format)  
  
df1$date <- as.Date(df1$date)  
df1$day\_of\_week <- wday(df1$date, label = TRUE, abbr = FALSE)  
df1$day\_of\_week <- as.character(df1$day\_of\_week)  
df1$X..computed\_region\_r4de\_cuuv <- as.numeric(df1$X..computed\_region\_r4de\_cuuv)  
df1$latitude <- as.numeric(df1$latitude)  
df1$longitude <- as.numeric(df1$longitude)  
names(df1)[names(df1) == "X..computed\_region\_r4de\_cuuv"] <- "computed\_region\_r4de\_cuuv"  
df1 <- df1[!is.na(df1$cc\_number),]  
  
#convert/standardize all the accident\_type values from acronym to meaningful terms.  
#The Maryland Open Data Portal documentation is not correct.   
#So, you have to print out all the unique values included in the accident\_type column and their frequency.   
############## Clean the data   
#4.1 Print out all the values and their frequencies here..  
  
sqldf("SELECT accident\_type, COUNT(accident\_type) FROM df1 GROUP BY accident\_type")

## accident\_type COUNT(accident\_type)  
## 1 F 5  
## 2 IS 1  
## 3 Injury Crash 11  
## 4 PD 657  
## 5 PI 308  
## 6 Property Damage Crash 16  
## 7 pd 2

#4.2 Merge the values so that the final values include only three categories:   
#######Property Damage (pd, PD, Property Damage Crash), Personal Injury (PI and Injury Crash), and Fatal Crash (F)  
  
df1$accident\_type <- gsub("pd", "Property Damage", df1$accident\_type)  
df1$accident\_type <- gsub("PD", "Property Damage", df1$accident\_type)  
df1$accident\_type <- gsub("Property Damage Crash", "Property Damage", df1$accident\_type)  
  
df1$accident\_type <- gsub("PI", "Personal Injury", df1$accident\_type)  
df1$accident\_type <- gsub("Injury Crash", "Personal Injury", df1$accident\_type)  
df1$accident\_type <- gsub("IS", "Personal Injury", df1$accident\_type)  
  
df1$accident\_type <- gsub("F", "Fatal Crash", df1$accident\_type)  
  
# Step 3: Understand the data using SQL (via SQLDF)  
############## 5. how many accidents happen on Sunday?  
# Use sql to count how many accidents on "Sunday"  
  
sun\_acc <- sqldf("SELECT COUNT(accident\_type) FROM df1 WHERE day\_of\_week = 'Sunday'")  
  
# Print the result  
  
print(sun\_acc)

## COUNT(accident\_type)  
## 1 132

############## 6. how many accidents had injuries? Read the documentation from the Maryland Open Data portal.  
#Use sql to count how many observations meet the criterion that accident type is Injury Crash  
inj\_acc <- sqldf("SELECT COUNT(accident\_type) FROM df1 WHERE accident\_type = 'Personal Injury'")  
# Print the result  
print(inj\_acc)

## COUNT(accident\_type)  
## 1 320

# list the injuries by day  
# count the number of injuries for each day of the week  
  
list\_inj <- sqldf("SELECT day\_of\_week, COUNT(accident\_type) FROM df1 WHERE accident\_type == 'Personal Injury' GROUP BY day\_of\_week")  
  
# Print the result  
print(list\_inj)

## day\_of\_week COUNT(accident\_type)  
## 1 Friday 57  
## 2 Monday 41  
## 3 Saturday 36  
## 4 Sunday 41  
## 5 Thursday 50  
## 6 Tuesday 51  
## 7 Wednesday 44

# Step 4: Understand the data using tapply  
###########7.how many accidents happen on SUNDAY?  
# tapply(Summary Variable, Group Variable, Function):  
# apply the length function on the "Sunday" subset of the column day\_of\_week  
  
tapply(df1$day\_of\_week, as.vector(df1$day\_of\_week == "Sunday"), length)

## FALSE TRUE   
## 868 132

# how many accidents had injuries  
# apply the length function  
  
tapply(df1$accident\_type, as.vector(df1$accident\_type == "Personal Injury"), length)

## FALSE TRUE   
## 680 320

# list the injuries by day  
# apply the length function on subset of the column accident\_type broken down by the value in Wday\_of\_week   
# and accident\_type == "Injury Crash"  
  
tapply(df1$accident\_type == "Personal Injury", df1$day\_of\_week, length)

## Friday Monday Saturday Sunday Thursday Tuesday Wednesday   
## 178 124 122 132 161 143 140

tapply(df1$accident\_type, df1$day\_of\_week, length)

## Friday Monday Saturday Sunday Thursday Tuesday Wednesday   
## 178 124 122 132 161 143 140

###########8: What is the percentage of injury for all accidents?  
  
percentInjury <- inj\_acc / length(df1$accident\_type) \* 100  
percentInjury

## COUNT(accident\_type)  
## 1 32

########## 9. Which day of a week do you observe the most injury?  
  
list\_inj[which.max(list\_inj$`COUNT(accident\_type)`), ]

## day\_of\_week COUNT(accident\_type)  
## 1 Friday 57