Ryan Timbrook

Applied Data Science

IST687 Intro to Data Science, Spring 2019
Due Date: 05/8/2019
Homework: 5

NetID: RTIMBROO
SUID: 386792749

#R Code - unexecuted

```
## Homework Week 5: JSON & tapply: Accident Analysis

# ---Preprocess Steps:------------------------------------------------------------------

### Clear objects from Memory
rm(list=ls())
### Clear Console:
cat("\014")

### Global Variable Assignments
url.accident.analysis <- "http://opendata.maryland.gov/resource/pdvh-tf2u.json"

### Load Required Packages
if(!require("RCurl")){install.packages("RCurl")}
if(!require("curl")){install.packages("curl")}
if(!require("RJSONIO")){install.packages("RJSONIO")}
if(!require("jsonlite")){install.packages("jsonlite")}
if(!require("sqldf")){install.packages("sqldf")}
if(!require("stringr")){install.packages("stringr")}

# ----Step 1: Load the data---------------------------------------------------------------------

## 1.1: Function to fetch the Accident JSON dataset
fetchJSONDataSet <- function(ds.url,as.json=TRUE){
  u <- getURL(ds.url) # send URL to internet

  if(as.json){
    ds <- fromJSON(u)
  }else{
    ds <- data.frame(fromJSON(u), stringsAsFactors = FALSE)
  }

  return(ds)
}


# Get the Accident Data from URL as JSON
ds.accidental <- fetchJSONDataSet(url.accident.analysis, FALSE)
#str(ds.accidental)
#View(ds.accidental)

# -----Step 2: Clean the data-----------------------------------------------------------------
## 2.1: Function to Clean the data
### Remove last 8 columns, Rename the rest of the columns
```

```
clean.accidental.ds <- function(ds, removeFirst = TRUE){

  #Make all empty cells equal to NA
  ds[ds==""] <- NA

  #Clean NA Columns from Dataframe
  ds <- ds[ ,!apply(ds,2,function(x) all(is.na(x)))]

  #Clean empty Rows from Dataframe
  ds <- ds[!apply(ds,1,function(x) all(is.na(x))),]

  ## Rename remaning columns
  new.col.names <- c(
  "ACC_DATE",
  "ACC_TIME",
  "ACC_TIME_CODE",
  "BARRACK",
  "CASE_NUMBER",
  "CITY_NAME",
  "COLLISION_WITH_1",
  "COLLISION_WITH_2",
  "COUNTY_CODE",
  "COUNTY_NAME",
  "DAY_OF_WEEK",
  "DIST_DIRECTION",
  "DIST_FROM_INTERSECT",
  "INJURY",
  "INTERSECT_ROAD",
  "PROP_DEST",
  "ROAD",
  "VEHICLE_COUNT"

  )

  colnames(ds) <- new.col.names

  #Handle NA values
  #ds$ACC_DATE[is.na(ds$ACC_DATE)] <- 'NOT_LISTED'
  #ds$ACC_TIME[is.na(ds$ACC_TIME)] <- 'NOT_LISTED'
  ds$ACC_TIME_CODE[is.na(ds$ACC_TIME_CODE)] <- 'NOT_LISTED'
  ds$BARRACK[is.na(ds$BARRACK)] <- 'NOT_LISTED'
  ds$CASE_NUMBER[is.na(ds$CASE_NUMBER)] <- 'NOT_LISTED'
  ds$CITY_NAME[is.na(ds$CITY_NAME)] <- 'NOT_LISTED'
  ds$COLLISION_WITH_1[is.na(ds$COLLISION_WITH_1)] <- 'NOT_LISTED'
  ds$COLLISION_WITH_2[is.na(ds$COLLISION_WITH_2)] <- 'NOT_LISTED'
  ds$COUNTY_CODE[is.na(ds$COUNTY_CODE)] <- 'NOT_LISTED'
```

```
  ds$COUNTY_NAME[is.na(ds$COUNTY_NAME)] <- 'NOT_LISTED'
  ds$DAY_OF_WEEK[is.na(ds$DAY_OF_WEEK)] <- 'NOT_LISTED'
  ds$DAY_OF_WEEK <- str_replace_all(ds$DAY_OF_WEEK," ","")
  ds$DAY_OF_WEEK <- str_to_upper(ds$DAY_OF_WEEK)
  ds$DAY_OF_WEEK <- as.factor(ds$DAY_OF_WEEK)
  ds$INJURY[is.na(ds$INJURY)] <- 'NO'
  ds$VEHICLE_COUNT[is.na(ds$VEHICLE_COUNT)] <- "1"
  ds$VEHICLE_COUNT <- as.numeric(ds$VEHICLE_COUNT)

  ds$DIST_FROM_INTERSECT <- as.numeric(ds$DIST_FROM_INTERSECT)
  ds$INJURY <- as.factor(ds$INJURY)
  ds$DIST_DIRECTION <- as.factor(ds$DIST_DIRECTION)

  # Remove Columns
  l <- length(ds)
  if(removeFirst){
    ## Remove first 8 columns
    ds <- ds[,-c(1:8)]
  }else{
    ## Remove last 8 columns
    ds <- ds[,-c((l-7):l)]
  }

  return(ds)
}

# Execute Clean DataSet Functino
cds <- clean.accidental.ds(ds.accidental)


# -----Step 3: Understand the data using SQL (via SQLDF)----------------------------------------------------------
--------------------
# 3.1: How many accidents happen on SUNDAY - An accident can have 1 or more vehicals involved
sql.sun.acdnt.cnt <- sqldf("select count(*) as 'SUNDAY_ACCT_CNT' from cds where
DAY_OF_WEEK = 'SUNDAY'")
sql.sun.acdnt.cnt

# 3.2: How many accidents had injuries
sql.inj.acdnt.cnt <- sqldf("select count(*) as 'INJURY_ACCT_CNT' from cds where INJURY = 'YES'")
sql.inj.acdnt.cnt

# 3.3: List the injuries by day
sql.inj.by.day <- sqldf("select count(*) as 'INJURY_CNT', DAY_OF_WEEK from cds where INJURY
= 'YES' group by DAY_OF_WEEK order by INJURY_CNT")
sql.inj.by.day
```

```
# -----Step 4: Understand the data using tapply----------------------------------------------------------
-----
# 4.1: How many accidents happen on Sunday
tapp.sun.acdnt.cnt <- tapply(cds$DAY_OF_WEEK, cds$DAY_OF_WEEK=='SUNDAY', length)
tapp.sun.acdnt.cnt["TRUE"]

# 4.2: How many accidents had injuries
tapp.inj.acdnt.cnt <- tapply(cds$INJURY, cds$INJURY=='YES', length)
tapp.inj.acdnt.cnt["TRUE"]

# 4.3: List the injuries by day
tapp.inj.by.day <- tapply(cds$INJURY, list(cds$INJURY=='YES', cds$DAY_OF_WEEK), length)
tapp.inj.by.day["TRUE",]
```

#R Code – executed

```
> ### Global Variable Assignments
> url.accident.analysis <- "http://opendata.maryland.gov/resource/pdvh-tf2u.j
son"
>
> ### Load Required Packages
> if(!require("RCurl")){install.packages("RCurl")}
> if(!require("curl")){install.packages("curl")}
> if(!require("RJSONIO")){install.packages("RJSONIO")}
> if(!require("jsonlite")){install.packages("jsonlite")}
> if(!require("sqldf")){install.packages("sqldf")}
> if(!require("stringr")){install.packages("stringr")}
>
> # ----Step 1: Load the data-----------------------------------------------
-----------------------------
>
> ## 1.1: Function to fetch the Accident JSON dataset
> fetchJSONDataSet <- function(ds.url,as.json=TRUE){
+   u <- getURL(ds.url) # send URL to internet
+
+   if(as.json){
+     ds <- fromJSON(u)
+   }else{
+     ds <- data.frame(fromJSON(u), stringsAsFactors = FALSE)
+   }
+
+   return(ds)
+ }
>
>
> # Get the Accident Data from URL as JSON
> ds.accidental <- fetchJSONDataSet(url.accident.analysis, FALSE)
> #str(ds.accidental)
> #View(ds.accidental)
>
> # -----Step 2: Clean the data---------------------------------------------
-----------------------------
> ## 2.1: Function to Clean the data
> ### Remove last 8 columns, Rename the rest of the columns
> clean.accidental.ds <- function(ds, removeFirst = TRUE){
+
+   #Make all empty cells equal to NA
+   ds[ds==""] <- NA
+
```

```
+   #Clean NA Columns from Dataframe
+   ds <- ds[ ,!apply(ds,2,function(x) all(is.na(x)))]
+
+   #Clean empty Rows from Dataframe
+   ds <- ds[!apply(ds,1,function(x) all(is.na(x))),]
+
+   ## Rename remaning columns
+   new.col.names <- c(
+   "ACC_DATE",
+   "ACC_TIME",
+   "ACC_TIME_CODE",
+   "BARRACK",
+   "CASE_NUMBER",
+   "CITY_NAME",
+   "COLLISION_WITH_1",
+   "COLLISION_WITH_2",
+   "COUNTY_CODE",
+   "COUNTY_NAME",
+   "DAY_OF_WEEK",
+   "DIST_DIRECTION",
+   "DIST_FROM_INTERSECT",
+   "INJURY",
+   "INTERSECT_ROAD",
+   "PROP_DEST",
+   "ROAD",
+   "VEHICLE_COUNT"
+
+   )
+
+   colnames(ds) <- new.col.names
+
+   #Handle NA values
+   #ds$ACC_DATE[is.na(ds$ACC_DATE)] <- 'NOT_LISTED'
+   #ds$ACC_TIME[is.na(ds$ACC_TIME)] <- 'NOT_LISTED'
+   ds$ACC_TIME_CODE[is.na(ds$ACC_TIME_CODE)] <- 'NOT_LISTED'
+   ds$BARRACK[is.na(ds$BARRACK)] <- 'NOT_LISTED'
+   ds$CASE_NUMBER[is.na(ds$CASE_NUMBER)] <- 'NOT_LISTED'
+   ds$CITY_NAME[is.na(ds$CITY_NAME)] <- 'NOT_LISTED'
+   ds$COLLISION_WITH_1[is.na(ds$COLLISION_WITH_1)] <- 'NOT_LISTED'
+   ds$COLLISION_WITH_2[is.na(ds$COLLISION_WITH_2)] <- 'NOT_LISTED'
+   ds$COUNTY_CODE[is.na(ds$COUNTY_CODE)] <- 'NOT_LISTED'
+   ds$COUNTY_NAME[is.na(ds$COUNTY_NAME)] <- 'NOT_LISTED'
+   ds$DAY_OF_WEEK[is.na(ds$DAY_OF_WEEK)] <- 'NOT_LISTED'
+   ds$DAY_OF_WEEK <- str_replace_all(ds$DAY_OF_WEEK," ","")
+   ds$DAY_OF_WEEK <- str_to_upper(ds$DAY_OF_WEEK)
+   ds$DAY_OF_WEEK <- as.factor(ds$DAY_OF_WEEK)
+   ds$INJURY[is.na(ds$INJURY)] <- 'NO'
+   ds$VEHICLE_COUNT[is.na(ds$VEHICLE_COUNT)] <- "1"
+   ds$VEHICLE_COUNT <- as.numeric(ds$VEHICLE_COUNT)
+
+   ds$DIST_FROM_INTERSECT <- as.numeric(ds$DIST_FROM_INTERSECT)
+   ds$INJURY <- as.factor(ds$INJURY)
+   ds$DIST_DIRECTION <- as.factor(ds$DIST_DIRECTION)
+
+   # Remove Columns
+   l <- length(ds)
+   if(removeFirst){
+     ## Remove first 8 columns
+     ds <- ds[,-c(1:8)]
+   }else{
+     ## Remove last 8 columns
+     ds <- ds[,-c((l-7):l)]
+   }
+
```

```
+   return(ds)
+ }
>
> # Execute Clean DataSet Functino
> cds <- clean.accidental.ds(ds.accidental)
>
>
> # -----Step 3: Understand the data using SQL (via SQLDF)------------------
-------------------------------------------------------
> # 3.1: How many accidents happen on SUNDAY - An accident can have 1 or more
vehicals involved
> sql.sun.acdnt.cnt <- sqldf("select count(*) as 'SUNDAY_ACCT_CNT' from cds w
here DAY_OF_WEEK = 'SUNDAY'")
> sql.sun.acdnt.cnt
  SUNDAY_ACCT_CNT
1              95
>
> # 3.2: How many accidents had injuries
> sql.inj.acdnt.cnt <- sqldf("select count(*) as 'INJURY_ACCT_CNT' from cds w
here INJURY = 'YES'")
> sql.inj.acdnt.cnt
  INJURY_ACCT_CNT
1             301
>
> # 3.3: List the injuries by day
> sql.inj.by.day <- sqldf("select count(*) as 'INJURY_CNT', DAY_OF_WEEK from
cds where INJURY = 'YES' group by DAY_OF_WEEK order by INJURY_CNT")
> sql.inj.by.day
  INJURY_CNT DAY_OF_WEEK
1         23      SUNDAY
2         41      MONDAY
3         42    SATURDAY
4         42     TUESDAY
5         49      FRIDAY
6         50    THURSDAY
7         54   WEDNESDAY
>
> # -----Step 4: Understand the data using tapply---------------------------
-----------------------------------------------
> # 4.1: How many accidents happen on Sunday
> tapp.sun.acdnt.cnt <- tapply(cds$DAY_OF_WEEK, cds$DAY_OF_WEEK=='SUNDAY', le
ngth)
> tapp.sun.acdnt.cnt["TRUE"]
TRUE
  95
>
> # 4.2: How many accidents had injuries
> tapp.inj.acdnt.cnt <- tapply(cds$INJURY, cds$INJURY=='YES', length)
> tapp.inj.acdnt.cnt["TRUE"]
TRUE
 301
>
> # 4.3: List the injuries by day
> tapp.inj.by.day <- tapply(cds$INJURY, list(cds$INJURY=='YES', cds$DAY_OF_WE
EK), length)
> tapp.inj.by.day["TRUE",]
   FRIDAY    MONDAY  SATURDAY    SUNDAY  THURSDAY   TUESDAY WEDNESDAY
       49        41        42        23        50        42        54
```