INFX 573: Problem Set 2 - Data Wrangling

Charudatta Deshpande

Due: Thursday, October 19, 2017

Collaborators: N/A

Setup:

In this problem set you will need, at minimum, the following R packages.

```
# Load standard libraries
library(tidyverse)
```

```
## Warning: Installed Rcpp (0.12.10) different from Rcpp used to build dplyr (0.12.11).
## Please reinstall dplyr to avoid random crashes or undefined behavior.
```

```
library(nycflights13)
library(jsonlite)
```

Problem 1: Open Government Data

Use the following code to obtain data on the Seattle Police Department Police Report Incidents.

```
police_incidents <- fromJSON("https://data.seattle.gov/resource/policereport.json")
```

(a) Describe, in detail, what the data represents.

Answer -

This data is sourced from City of Seattle's open data portal (data.city.gov). The data represents the details about criminal incidents in Seattle area which have been filed with the Seattle police department. The data provides high level information about of the incident, like the date, location, time, type of offense etc. There are 1000 records in the dataset. This data is likely used by Government agencies and general public to monitor crime statistics in Seattle area. The data originally is in JSON format, which means it does not have a single row for a single record. Rather, one record spans multiple rows and is separated by delimniters. The package JSONLITE converts it from JSON format to a format that R can understand.

(b) Describe each variable and what it measures. Be sure to note when data is missing. Confirm that each variable is appropriately cast - it has the correct data type. If any are incorrect, recast them to be in the appropriate format.

Answer -

This data has following variables -

year - Represents the year in which incident took place. Type - Character. This needs to be converted to Integer (done below).

zone_beat - Zone number of the location assigned by the police department. Type - Character. This is correct type.

latitude - Latitude of the crime location. Type - Character. This needs to be converted to Numeric (done below).

offense_code_extension - Extension of the field 'offense_code'. This is likely some supplementary field that provides additional information about the crime indicated in the primary field 'offense_code'. Type - Character. This needs to be converted to Integer (done below). 0 indicates absence of a value.

summarized_offense_description - Summarized description of the crime in plain English. Type - Character. This is correct type.

date_reported - Date and time of the crime. Type - Character. This is correct.

offense_type - The type of crime is indicated using certain codes assigned to types of crimes. Type - Character. This is correct.

occurred_date_or_date_range_start - Start Date and time of the crime. Type - Character. This is correct.

summary_offense_code - This field indicates the type of crime using a number. This is likely a tranlation of offense_type into a number using a coding system. Type - Character. 'X' represents abesence of a value. Because of presence of 'X', we will keep this field as character. Otherwise, we probably could have converted this to an Integerer data type.

occurred_date_range_end month - End Date and time of the crime. Type - Character. This is correct. NA indicates absence of a value.

general_offense_number - A number assined to each criminal incidene. Type - Character. This needs to be converted to Integer (done below).

census_tract_2000 - The area of the crime as identified under Census Tract 2000. Type - Character. This needs to be converted to Numeric (done below).

location.latitude - This is likely a repeatation of field 'latitude'. Latitude of the crime location. Type - Character. This needs to be converted to Numeric (done below).
Note - I tried to convert this to numeric, however was not successful. I will keep working on that, but for this variable I will keep it as Character.

location.needs_recoding - This indicates if the location needs to be recoded. Type - Character. This needs to be converted to Logical (done below).
Note - I tried to convert this to logical (true or false type) column, however was not successful. I will keep working on that, but for this variable I will keep it as Character.

location.longitude - Longitude of the crime location. Type - Character. This needs to be converted to Numeric (done below).
Note - I tried to convert this to numeric, however was not successful. I will keep working on that, but for this variable I will keep it as Character.

offense_code - This appears to be a slightly detailed version of 'offense_type'. This field appears to provide a more detailed coding of the type of criminal incident. Type - Character. This is correct. 'X' indicates absence of a value.

hundred_block_location - The hundred block location of the crime. The full address is likely not published to protect the identity of the victims. Type - Character. This is correct.

rms_cdw_id - This is another field that identifies the type of criminal incident. Type - Character. This needs to be converted to Integer (done below).

district_sector - A sector identifier of the crime location. Type - Character. This is correct.

longitude - This is likely a repeation of field location.longitude. Longitude of the crime location. Type - Character. This needs to be converted to Numeric (done below).

```
# Change data types of identified variables.
police_incidents$year <- as.integer(police_incidents$year)
police_incidents$latitude <- as.numeric(police_incidents$latitude)
police_incidents$offense_code_extension <- as.integer(police_incidents$offense_code_extension)
```

```
police_incidents$general_offense_number <- as.integer(police_incidents$general_offense_number)
police_incidents$census_tract_2000 <- as.numeric(police_incidents$census_tract_2000)
```

```
## Warning: NAs introduced by coercion
```

```
police_incidents$rms_cdw_id <- as.integer(police_incidents$rms_cdw_id)
police_incidents$longitude <- as.numeric(police_incidents$longitude)
```

(c) Produce a clean dataset, according to the rules of tidy data discussed in class. Export the data for future analysis using the Rdata format.

```
#Convert the JSON data into a data frame by using following command -
police_incidents <- as.data.frame(police_incidents)
#The above command creates a data frame structure, which is 'tidy version' of the
#original JSON data.
#
# Convert data to Rdata format and export on local drive.
save(police_incidents, file = "police report file.RData")
#
#File "police report file.RData" can be loaded again using 'load command'.
new_police_incidents <- load("police report file.RData")
#
#The loaded file, new_police_incidents, if viewed, displays a string 'police_incidents'.
#This can be used as a dataset in R, which is same as original dataset
#'police_incidents'.
```

(d) Describe any concerns you might have about this data. This may include biases, missing data, or ethical concerns.

Answer -
I noted following concerns/observations about this data after cleaning -

1. Latitude and Longitude are repeated in data, which means there is redundancy and the data is not completely tidy. After verification that the data fields are actually duplicates, two columns can be removed from the data making it easier for analysis.
2. Following fields have missing values for a considerable number of records. Depending on the type of analysis needed for these fields, decision will need to be taken about how to handle this missing data.

summary_offense_code
occured_date_range_end
offense_code

3. The latitude and longitude are sufficient to exactly identify the location of the crime and thus leading to possible identification of victim. Based on the type of crime, victims may not want to be identified. Though the street address is not fully published, that may not provide the victim sufficient protection against identity discovery. This is an ethical concern.
4. The dataset only has data about current month. There is no history. This makes it impossible to do trend analysis of the criminal incidents and perform any predictive analysis.

Problem 2: Wrangling the NYC Flights Data

In this problem set we will use the data on all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. You can find this data in the `nycflights13` R package.

(a) Importing Data:

Load the data.

```
# Load the dataset 'flights'.
data(flights)
```

(b) Data Manipulation:

Use the flights data to answer each of the following questions. Be sure to answer each question with a written response and supporting analysis.

- How many flights were there from NYC airports to LAX in 2013?

```
# Below code will count the number of flights that met the specified condition
#(destination = LAX).
flights %>%
filter(dest=="LAX") %>%
summarize(n=n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1 16174
```

```
#
#Answer -
#There were 16174 flights from NYC to LAX.
#
```

- How many airlines fly from NYC to LAX?

```
# This code will count the number of carriers from NYC to LAX
flights %>%
filter(dest=="LAX") %>%
count(carrier)
```

```
## # A tibble: 5 x 2
##   carrier     n
##     <chr> <int>
## 1      AA  3582
## 2      B6  1688
## 3      DL  2501
## 4      UA  5823
## 5      VX  2580
```

```
#
#Answer -
#An output tibble of 5 X 2 is created, specifying count for each airline. Since
#number of rows is 5, there is 5 airlines that travel from NYC to LAX.
```

- How many unique air planes fly from NYC to LAX?

```
# This code will count the number of airplanes from NYC to LAX. A
#unique airplane is one with a unique tail number.
flights %>%
filter(dest=="LAX") %>%
count(tailnum)
```

```
## # A tibble: 992 x 2
```

```
##    tailnum     n
##      <chr> <int>
##  1  N11206     6
##  2  N1200K     8
##  3  N1201P     6
##  4  N12109    34
##  5  N12114    16
##  6  N12116    22
##  7  N12125    28
##  8  N12216    12
##  9  N12218    10
## 10  N12221     9
## # ... with 982 more rows
```

```
#
#Answer -
#An output tibble of 992 X 2 is created, specifying count for each unique tail number.
#Since number of rows is 992, there is 992 unique airplanes that travel
#from NYC to LAX.
```

- What is the average arrival delay for flights from NYC to LAX?

```
# Below code will select flights to LAX with arrival delay
#not equal to zero, and calculate mean arrival delay.
flights %>%
filter(dest=="LAX") %>%
summarize(mean_delay=mean(arr_delay, na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   mean_delay
##        <dbl>
## 1  0.5471109
```

```
#
#Answer -
#0.547 minutes is the average arrival delay for flights from NYC to LAX.
#Note - Above code did not work if the condition na.rm=TRUE was not
#specified. It only worked with that condition. This means there are some NA
#values in the arr_delay field.
```

- What proportion of flights to LAX come from each NYC airport?

```
# Below code will create a separate table named t for records with
#destination as LAX. Then it will calculate proportions of values
#based on origin airport.
t <- flights %>%
     filter(dest=="LAX")
prop.table(table(t$origin))
```

```
##
##       EWR       JFK
## 0.3036973 0.6963027
```

```
#
#Answer -
#EWR - 0.3036973 = 30.36973% of flights to LAX from NYC airports come from EWR.
#JFK - 0.6963027 = 69.63027% of flights to LAX from NYC airports come from JFK.
```

Problem 3: Pipes and Diamonds

The following questions relate to the "diamonds" dataset included in the ggplot2 package.

(a) Importing Data:

Load and describe the data

```
# Following code will load the data.
library(ggplot2)
data(diamonds)
#
#Answer - Description of data -
#
#This dataset contains prices and other attributes of 53,940 diamonds.
#There are 10 variables. Some of the variables include price, carat, cut,
#color etc. Additional documentation can be found with help("diamonds")
#command.
#
```

(b) Exploring the price column:

Create a new variable which contains the average `price` for each `cut` of diamond in decreasing order. Does this follow the ordering what you would expect?

```
# Following table will create the new variable mean_price_table which
#contains the average price for each cut in descending order.
mean_price_table <- diamonds %>%
                    group_by(cut) %>%
                    summarize(mean_price=mean(price)) %>%
                    arrange(desc(mean_price))
mean_price_table
```

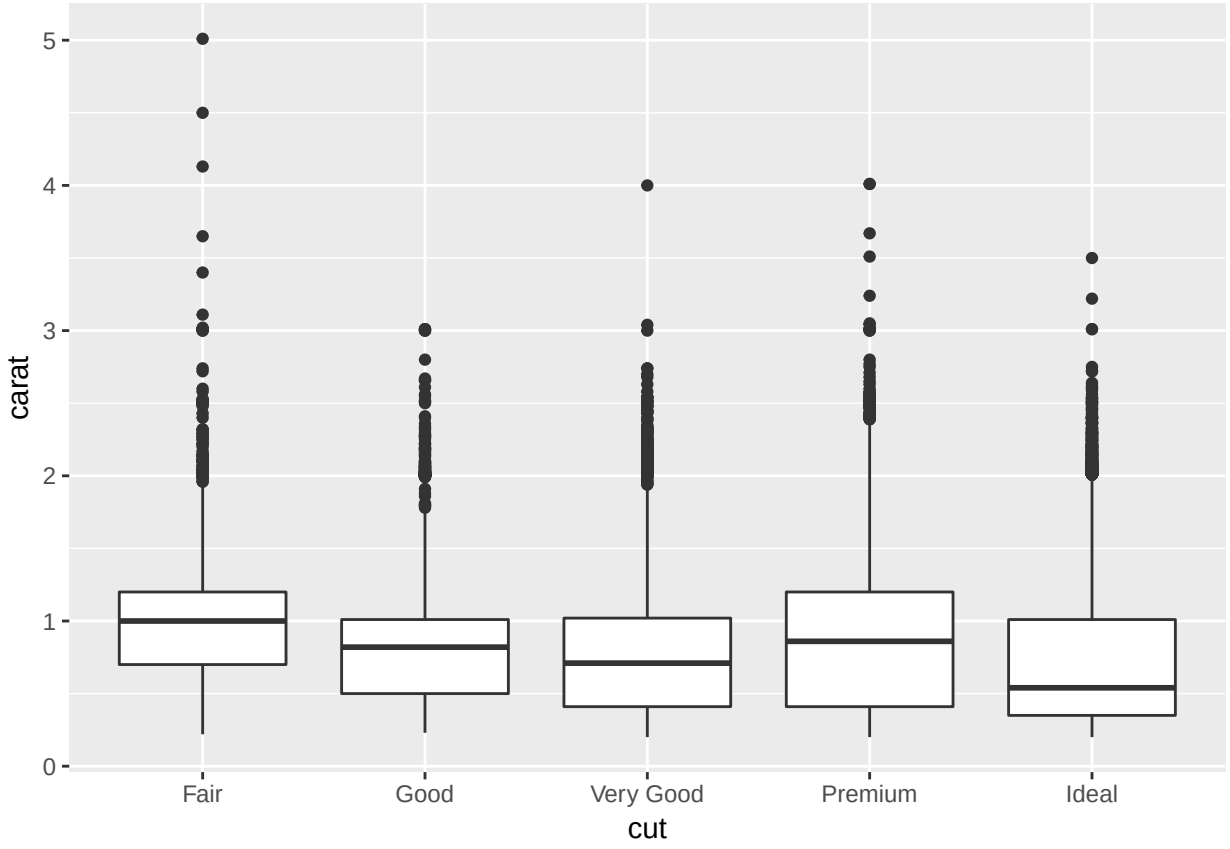
```
## # A tibble: 5 x 2
##         cut mean_price
##       <ord>      <dbl>
## 1   Premium   4584.258
## 2      Fair   4358.758
## 3 Very Good   3981.760
## 4      Good   3928.864
## 5     Ideal   3457.542
```

```
#
#Answer -
#
#This does not follow the ordering I would expect. One would expect the price
# to increase in order Fair, Good, Very Good, Premium, Ideal.
#However here, Ideal diamonds are the cheapest.
#But this is not a complete piture. We have not considered the carat weight
#without which no conclusions can be made about a diamond being
#'cheap' or 'expensive'.
```

(c) Correcting via carats:

One possible explaination for this is that the `carat` values might not be distributed evenly across different cuts. To check this, create a chart with a boxplot of the `carat` distribution for each `cut`.

```
#Below code will create a box plot of `carat` distribution for each `cut`.
ggplot(diamonds, aes(cut, carat)) + geom_boxplot()
```




```
#
#Answer -
#It can be seen that ideal diamonds tend to be less in weight, while fair diamonds
#tend to be heavier. This causes the anamoly that we see above.
```

With this in mind, update the variable created in part (b) to weight the `price` by the `carat` value. Did this solve the issue?

```
#Below code will create a weighted mean for price by carat.
weighted_mean_price_table <- diamonds %>%
                    group_by(cut) %>%
                    summarize(weighted_mean_price=weighted.mean(price, carat)) %>%
                    arrange(desc(weighted_mean_price))
weighted_mean_price_table
```

```
## # A tibble: 5 x 2
##         cut weighted_mean_price
##       <ord>               <dbl>
## 1   Premium            6908.103
## 2 Very Good            6058.932
## 3      Fair            5868.048
## 4      Good            5744.404
## 5     Ideal            5641.596
```

```
#
#Answer -
#This still does not resolve the anamoly. The ideal diamonds are still
#cheapest even with weighted price by carat. There must be another factor
#that is skewing the results.
```

(d) Further exploration:

Further refine your analysis by including at least one other varaible in the dataset. Does this improve your results? What factors might still be skewing them? Feel free to use visualizations, but be sure to include dplyr manipulations in your analysis.

```
#We will consider other two factors, clarity and color of the diamond.
#Clarity is a a measurement of how clear the diamond is (I1 (worst),
#SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best)).
#Color ranges from J (worst) to D (best).
#We will create following dataset -
extended_mean_price_table <- diamonds %>%
                    group_by(cut, clarity, color) %>%
                    summarize(weighted_mean_price=weighted.mean(price, carat)) %>%
                    arrange(desc(weighted_mean_price))
extended_mean_price_table
```

```
## # A tibble: 276 x 4
## # Groups:   cut, clarity [40]
##          cut clarity color weighted_mean_price
##        <ord>   <ord> <ord>               <dbl>
##  1     Ideal      I1     J           13084.995
##  2      Good      IF     D           12602.620
##  3   Premium      IF     D           12490.774
##  4 Very Good      IF     D           12340.653
##  5   Premium      IF     J           11039.065
##  6   Premium    VVS1     J            9940.023
##  7      Fair      I1     D            9891.253
##  8   Premium     VS2     I            9637.069
##  9     Ideal     SI2     I            9323.096
## 10   Premium     SI2     J            9233.506
## # ... with 266 more rows
```

```
#
#Answer -
#This extended_mean_price_table contains cut, clarity, color and weighted
#mean price per carat. A review of the table suggest following -
#1. The most expensive diamonds per carat are ideal, but with worst color
#and worst quality. They are more expensive than good and premium cut with better
#color and clarity. This is as expected, because they are ideal.
#2. But it still does not explain why ideal diamonds with better color and
#better clarity are cheaper than ideal diamonds with worst color and
#clarity.
#3. In general, this provides a better distribution of prices, but this is
#still no conclusive. There must be other reasons for this anamoly.
#4. Future exploration could include analyis of x, y, z fields and
#checking if they influence the prices of the diamonds.
```