INFX 573: Problem Set 2 - Data Wrangling

TAPASVI BANSAL

Due: Thursday, October 19, 2017

Collaborators:

Instructions:

Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `problemset2.Rmd` file from Canvas. Open `problemset2.Rmd` in RStudio and supply your solutions to the assignment by editing `problemset2.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text.
4. Collaboration on problem sets is acceptable, and even encouraged, but each student must turn in an individual write-up in his or her own words and his or her own work. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the R Markdown file to `YourLastName_YourFirstName_ps2.Rmd`, knit a PDF and submit the PDF file on Canvas.

Setup:

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

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

```
## Warning: package 'tidyr' was built under R version 3.4.2
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
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")
table(police_incidents$year)
```

```
##
## 2017
## 1000
```

```
#table(police_incidents$summarized_offense_description)
nrow(table(police_incidents$rms_cdw_id))
```

```
## [1] 1000
```

```
nrow(table(police_incidents$district_sector))
```

```
## [1] 19
```

```
#table(police_incidents$offense_code)
#View(police_incidents$location)
#names(police_incidents)
#nrow(police_incidents)
#str(police_incidents)
#table(police_incidents$location$needs_recoding)
```

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

The data represents a piece of police report containing a thousand records all dated to the year of 2017. These records are incidents that are reported to or recorded by police, within the city. Each record is defined by 19 vectors, these vectors define 3 aspects of the incident: 1) When? 2) Where? 3) & What?

The "What" is defined by offense_code_extension, summarized_offense_description, offense_type, summary, offense_code, general_offense_number, census_tract_2000, offense_code, rms_cdw_id.

The "When" is defined by year, date_reported, occured_date_or_date_range_start, month, occured_date_range_end.

The "Where" aspects are defined by hundred_block_location, zone_beat, latitude, district_sector, longitude. The data also has a location data frame with 3 vectors, latitude, longitude and needs_recoding associated with each record in the data. The latitude vector is referenced from the location table into the police_incidents main data frame as location.latitude.

(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.

[1] "year" - Describes the year of incidents, in this data frame all belong to the year 2017.
[2] "zone_beat" - Describes the zone/ Beat of the incident into 53 categories.
[3] "latitude" - Describes the location of the incident in terms of latitude.
[4] "offense_code_extension" - Describes the extension of offense code of the incident into 17 categories.
[5] "summarized_offense_description" - Summarizes the offense description of the incident into 33 categories.
[6] "date_reported" - Describes the reporting date of the incident in the form of a TIMESTAMP.
[7] "offense_type" - Describes the offense type of the incident into roughly 70 categories.
[8] "occurred_date_or_date_range_start" -Describes the occurred date or start date of the incident in the form of a TIMESTAMP.

[9] "summary_offense_code" - Describes the offense code of the incident into 21 categories.
[10] "month" - Describes the month of incidents, in this data frame, all belong to the 10th month (October).

[11] "general_offense_number" - Describes the offense general number of the incident. As the name suggests these are not unique as they are general. Existing records are attributed to 605 different types of general offense numbers.

[12] "census_tract_2000" - Describes the census tract (population range) for the area where the incident occurred.

[13] "location" - Data frame with 3 vectors, latitude, longitude and needs_recording associated with each record in the data.

```
[13.1] Latitude- Describes the latitude of the incident.
[13.2] Longitude- Describes the longitude of the incident.
```

[13.3] needs_recording - Described as FALSE for all records, could be an indicator of changes that are

[14] "offense_code" - Describes the code for the type of the incident into 51 categories.

[15] "hundred_block_location" - Describes the block location of the incident in the city.

[16] "rms_cdw_id" - Provides the UNIQUE identification for each incident that occurred and was recorded.

[17] "district_sector" - Describes the district/sector of the incident into 19 categories.

[18] "longitude" - Describes the location of the incident in terms of longitude.
[19] "occurred_date_range_end" - Describes the end date with time (TIMESTAMP) of the incident.

(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.

```
police_incidents_1 <-police_incidents
police_incidents_1$occurred_date_range_end <-NULL
police_incidents_1$latitude <- NULL
police_incidents_1$longitude <- NULL
#table(police_incidents_1$location$needs_recoding)
police_incidents_1$location$needs_recoding <- NULL
#names(police_incidents_1)
police_incidents_2 <- police_incidents_1[,c("rms_cdw_id", "summarized_offense_description", "offense_ty
Police_Incident_3 <- na.omit(police_incidents_2)

#Police_Incident_3  # Clean dataset, please check if need be.

saveRDS(Police_Incident_3, file = "Police_Incident_TapasviB.rds") # Exporting the dataset for future us
```

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

The data though helpful in analysis from academic point of view has provided the locations of incidents which might be helpful to a student in analysing the patterns of various type of crimes committed in various locations but it also subjects the mind of person analysing to be skeptical of living or even visiting such locations. If a larger population learns about this and be able to verify the analysis, serious damages could be caused to the growth of any sector/district of the city.

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.

```
#install.packages("nycflights13")
library(nycflights13)
table(nycflights13::flights$year)
```

```
##
##   2013
## 336776
```

```
?nycflights13::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?

```
nyclaxflights <- data.frame(nycflights13::flights %>% filter(dest == "LAX" & year == 2013))
#nyclaxflights
length(na.omit(nyclaxflights$flight))
```

```
## [1] 16174
```

There were 16174 flights from NYC airports to LAX in 2013

- How many airlines fly from NYC to LAX?

```
table(nyclaxflights$carrier)
```

```
##
##   AA   B6   DL   UA   VX
## 3582 1688 2501 5823 2580
```

```
length(unique(nyclaxflights$carrier))
```

```
## [1] 5
```

There are 5 airlines (AA, B6, DL, UA, VX) from NYC to LAX

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

```
#table(nyclaxflights$tailnum)
#nrow(table(nyclaxflights$tailnum))
length(unique(nyclaxflights$tailnum))
```

```
## [1] 992
```

The number of unique air planes from NYC to LAX is 992

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

```
nyclaxflightsclean<-na.omit(nyclaxflights)
mean(nyclaxflightsclean$arr_delay)
```

```
## [1] 0.5471109
```

```
nyclaxflights_arrdelay <- na.omit(nyclaxflights$arr_delay)
mean(nyclaxflights_arrdelay)
```

```
## [1] 0.5471109
```

The average arrival delay for flights from NYC to LAX is 0.55 Minutes

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

```
prop.table(table(nyclaxflights$origin))
```

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

```
nyclaxflights %>% group_by(origin) %>% summarise( n=n()) %>% mutate(Proportion = n/sum(n))
```

```
## # A tibble: 2 x 3
##   origin     n Proportion
##    <chr> <int>      <dbl>
## 1    EWR  4912  0.3036973
## 2    JFK 11262  0.6963027
```

The proportions are approximately EWR: 0.30, JFK: 0.70 with Zero flights flying out of NYX to LAX from the LGA airport of NYC

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

```
data(diamonds)
#diamonds
#?diamonds
table(diamonds$clarity)
```

```
##
##    I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   741  9194 13065 12258  8171  5066  3655  1790
```

The diamonds data frame consist of 53940 rows/records with 10 variables. There are 3 factor variables (Cut, Color and Clarity). These variable define the quality of diamond cut, the color of diamond (from J (worst) to D (best)) and how clear the diamond is(I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best)). Other variables like x, y, z, depth and table defines the structure of the dimaond. The price variable defines the worth of each diamond desribed by the other variables in US dollars.

(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?

```
diamonds %>% group_by(cut) %>% summarise( Avg_price = mean(price)) %>% arrange(desc(Avg_price))
```

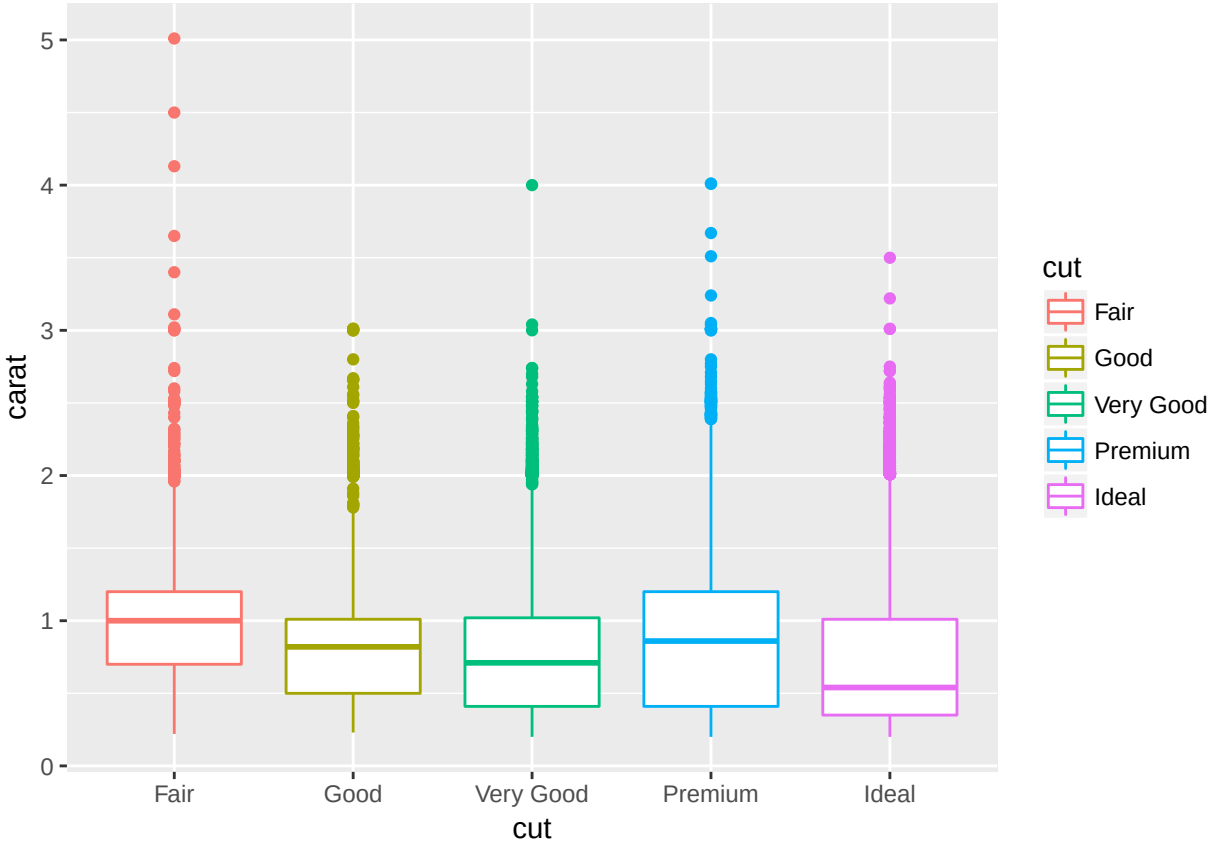
```
## # A tibble: 5 x 2
##         cut Avg_price
##       <ord>     <dbl>
## 1   Premium  4584.258
## 2      Fair  4358.758
## 3 Very Good  3981.760
## 4      Good  3928.864
## 5     Ideal  3457.542
```

No, it doesn't give the order I expect i.e Ideal, Premium, Very Good, Good, Fair, which should be the correct order if you arrange the cut list in descending order.

(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`.

```
diamonds %>% ggplot(aes(x=cut,y=carat, col = cut)) + geom_boxplot()
```



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

```
diamonds %>% group_by(cut) %>% summarise( Avg_price = mean(price/carat), Mean_Carat= mean(carat), Mean_
```

```
## # A tibble: 5 x 4
##         cut Avg_price Mean_Carat Mean_Price
##       <ord>     <dbl>      <dbl>      <dbl>
## 1   Premium  4222.905  0.8919549   4584.258
## 2 Very Good  4014.128  0.8063814   3981.760
## 3     Ideal  3919.700  0.7028370   3457.542
## 4      Good  3860.028  0.8491847   3928.864
## 5      Fair  3767.256  1.0461366   4358.758
```

No, it doesn't help in coming up with a correct order, with the type Ideal in the middle.

(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.

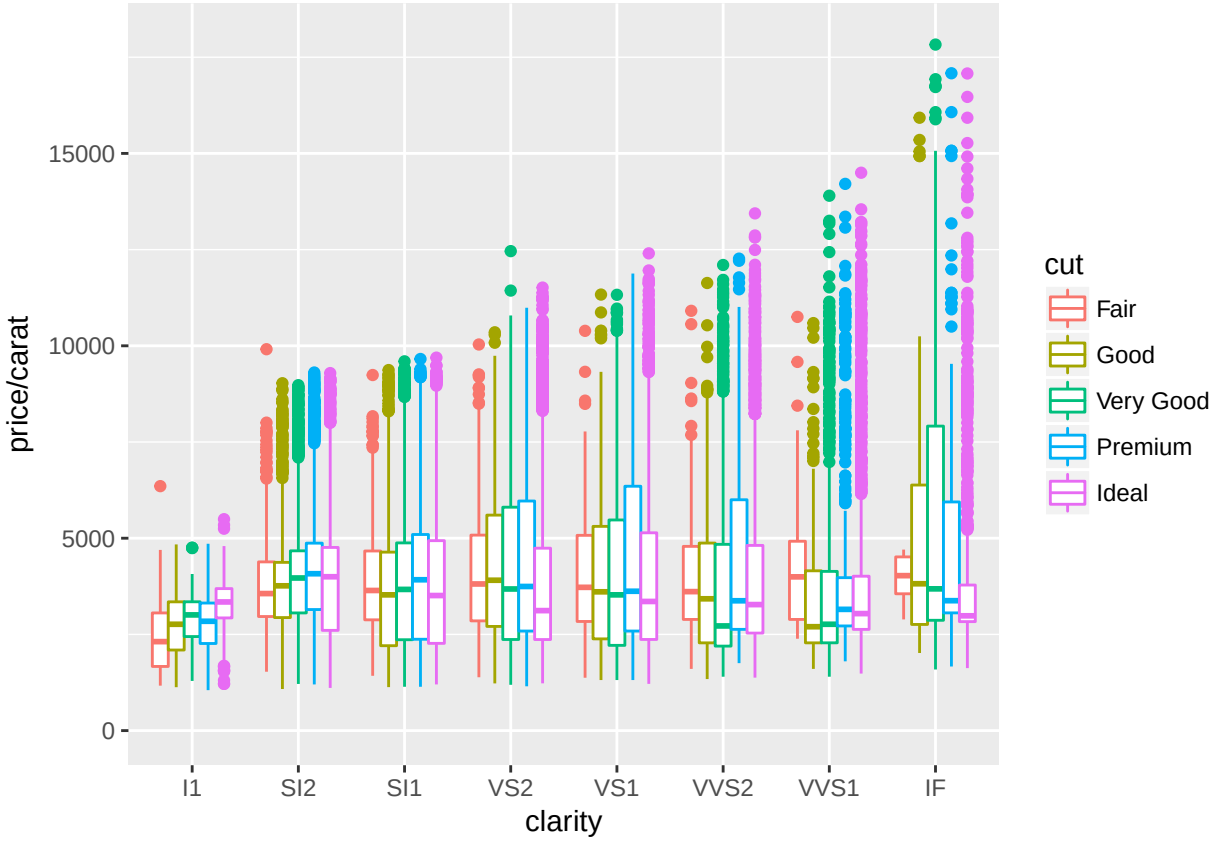
```
diamonds %>% group_by(clarity) %>% mutate(avg=mean(price/carat)) %>% summarise( Avg_price = mean(avg)) %
```

```
## # A tibble: 8 x 2
```

```
##   clarity Avg_price
##     <ord>     <dbl>
## 1      IF  4259.932
## 2    VVS2  4204.166
## 3     VS1  4155.817
## 4     VS2  4080.527
## 5     SI2  4010.854
## 6    VVS1  3851.411
## 7     SI1  3849.078
## 8      I1  2796.296
```

```
#ggplot(data=diamonds) + geom_histogram(binwidth = 50, aes(x=price/carat, fill = clarity)) + coord_cart

ggplot(data=diamonds) + geom_boxplot(aes (x = clarity, y =price/carat, col = cut)) + coord_cartesian(yl
```



Additional variable of Clarity is used as it's one of the 3 factor variables. It helps to identify how different cuts do with clarity with respect to the price per carat of the diamonds. The influencing could be the buying pattern of customers who don't understand completely about the nature of diamonds.

Through the box plot we can see that for example that the maximum/ Greatest price value per carat value for all the cuts with Included (I1) type clarity is below 5000 US dollars (with some outliers). Surprisingly, the maximum/ greatest price value for Internally Flawless (IF) but with a Fair type cut diamond is also under 5000 US dollars.