

COMPSCIX 415.2 Homework 5/Midterm

Rajat Jain

July 10th, 2018

Contents

Exercises (Total Points - 30)	1
My Git Repository	1
RStudio and R Markdown (3 points)	1
The tidyverse packages (3 points)	2
R Basics (1.5 points)	2
Data import/export (3 points)	3
Visualization (6 points)	5
Data munging and wrangling (6 points)	7
EDA (6 points)	9
Git and Github (1.5 points)	12

Exercises (Total Points - 30)

My Git Repository

My git repository is located at this Github location: <https://github.com/rajatmnnit/compscix-415-2-assignments>

All the assignments are organized under their individual directories containing rmarkdown (.Rmd) file(s) and the knitted pdf version along with all the required data files and generated output files.

Midterm assignment can be found under midterm directory.

RStudio and R Markdown (3 points)

1. **Use markdown headers in your document to clearly separate each midterm question and add a table of contents to your document.**

Each section in this document starts with a `##` Heading 2 style section title.

You can use the table of contents on the first page to navigate to various sections directly.

Wherever relevant, hyperlinks are clickable.

The tidyverse packages (3 points)

By now you've used at least five different packages from the tidyverse for plotting, data munging, reshaping data, importing/exporting data, and using tibbles (the tibble package is used for this without you even realizing it's there).

1. Can you name which package is associated with each task below?

Here is the list of tidyverse packages:

- Plotting - `ggplot2`
- Data munging/wrangling - `dplyr`
- Reshaping (speading and gathering) data - `tidyr`
- Importing/exporting data - `readr`

2. Now can you name two functions that you've used from each package that you listed above for these tasks?

Some of the functions we have used from each of these packages are:

- Plotting - `ggplot()`, `geom_point()`
- Data munging/wrangling - `filter()`, `summarise()`
- Reshaping data - `spread()`, `gather()`
- Importing/exporting data - `read_csv()`, `read_delim()`

R Basics (1.5 points)

1. Fix this code *with the fewest number of changes possible* so it works:

```
My_data.name___is.too00ooLong! <- c( 1 , 2 , 3 )
```

While the variable name is too long and doesn't follow any conventions, it is not invalid to use it. The only problem is the ! mark in the variable name. Just removing it is the smallest change which makes this code work.

```
My_data.name___is.too00ooLong <- c( 1 , 2 , 3 )
```

2. Fix this code so it works:

```
my_string <- C('has', 'an', 'error', 'in', 'it')
```

Lower case the concat function `c()` and complete quote on the last value to fix this code.

```
my_string <- c('has', 'an', 'error', 'in', 'it')
```

3. Look at the code below and comment on what happened to the values in the vector.

```
my_vector <- c(1, 2, '3', '4', 5)
my_vector
```

```
## [1] "1" "2" "3" "4" "5"
```

In R the vectors can contain values of only a single data type. Since we are trying to construct a vector with mixed values of numbers and characters (quoted numbers), R automatically converts everything to character. Hence all values are returned as characters on printing `my_vector`.

Data import/export (3 points)

1. Download the rail_trail.txt file from Canvas (in the Midterm Exam section) and successfully import it into R. Prove that it was imported successfully by including your import code and taking a glimpse of the result.

```
# File has been downloaded from Canvas to the midterm directory.
```

```
rail_trail <- read_delim("rail_trail.txt", delim = "|")
```

```
## Parsed with column specification:
```

```
## cols(
```

```
##   hightemp = col_integer(),
```

```
##   lowtemp = col_integer(),
```

```
##   avgtemp = col_double(),
```

```
##   spring = col_integer(),
```

```
##   summer = col_integer(),
```

```
##   fall = col_integer(),
```

```
##   cloudcover = col_double(),
```

```
##   precip = col_double(),
```

```
##   volume = col_integer(),
```

```
##   weekday = col_integer()
```

```
## )
```

```
glimpse(rail_trail)
```

```
## Observations: 90
```

```
## Variables: 10
```

```
## $ hightemp <int> 83, 73, 74, 95, 44, 69, 66, 66, 80, 79, 78, 65, 41,...
```

```
## $ lowtemp <int> 50, 49, 52, 61, 52, 54, 39, 38, 55, 45, 55, 48, 49,...
```

```
## $ avgtemp <dbl> 66.5, 61.0, 63.0, 78.0, 48.0, 61.5, 52.5, 52.0, 67.0,...
```

```
## $ spring <int> 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, ...
```

```
## $ summer <int> 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, ...
```

```
## $ fall <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, ...
```

```
## $ cloudcover <dbl> 7.6, 6.3, 7.5, 2.6, 10.0, 6.6, 2.4, 0.0, 3.8, 4.1, ...
```

```
## $ precip <dbl> 0.00, 0.29, 0.32, 0.00, 0.14, 0.02, 0.00, 0.00, 0.00, ...
```

```
## $ volume <int> 501, 419, 397, 385, 200, 375, 417, 629, 533, 547, 4...
```

```
## $ weekday <int> 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, ...
```

2. Export the file into a comma-separated file and name it “rail_trail.csv”. Make sure you define the path correctly so that you know where it gets saved. Then reload the file. Include your export and import code and take another glimpse.

```
# Write the new CSV file locally to the midterm directory.
```

```
write_csv(rail_trail, path = "rail_trail.csv")
```

```
# Read back from midterm directory.
```

```
rail_trail_csv <- read_csv("rail_trail.csv")
```

```
## Parsed with column specification:
```

```
## cols(
```

```
##   hightemp = col_integer(),
```

```
##   lowtemp = col_integer(),
```

```
##   avgtemp = col_double(),
```

```
##   spring = col_integer(),
```

```
##   summer = col_integer(),
```

```
##   fall = col_integer(),
```

```
##   cloudcover = col_double(),
```

```
##   precip = col_double(),
```

```
##   volume = col_integer(),
```

```
##   weekday = col_integer()
```

```
## )
```

```
glimpse(rail_trail_csv)
```

```
## Observations: 90
```

```
## Variables: 10
```

```
## $ hightemp    <int> 83, 73, 74, 95, 44, 69, 66, 66, 80, 79, 78, 65, 41, ...
```

```
## $ lowtemp     <int> 50, 49, 52, 61, 52, 54, 39, 38, 55, 45, 55, 48, 49, ...
```

```
## $ avgtemp     <dbl> 66.5, 61.0, 63.0, 78.0, 48.0, 61.5, 52.5, 52.0, 67.0, ...
```

```
## $ spring      <int> 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, ...
```

```
## $ summer      <int> 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, ...
```

```
## $ fall        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, ...
```

```
## $ cloudcover  <dbl> 7.6, 6.3, 7.5, 2.6, 10.0, 6.6, 2.4, 0.0, 3.8, 4.1, ...
```

```
## $ precip      <dbl> 0.00, 0.29, 0.32, 0.00, 0.14, 0.02, 0.00, 0.00, 0.00, ...
```

```
## $ volume      <int> 501, 419, 397, 385, 200, 375, 417, 629, 533, 547, 4...
```

```
## $ weekday     <int> 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, ...
```

Visualization (6 points)

1. Critique this graphic: give only three examples of what is wrong with this graphic. Be concise.

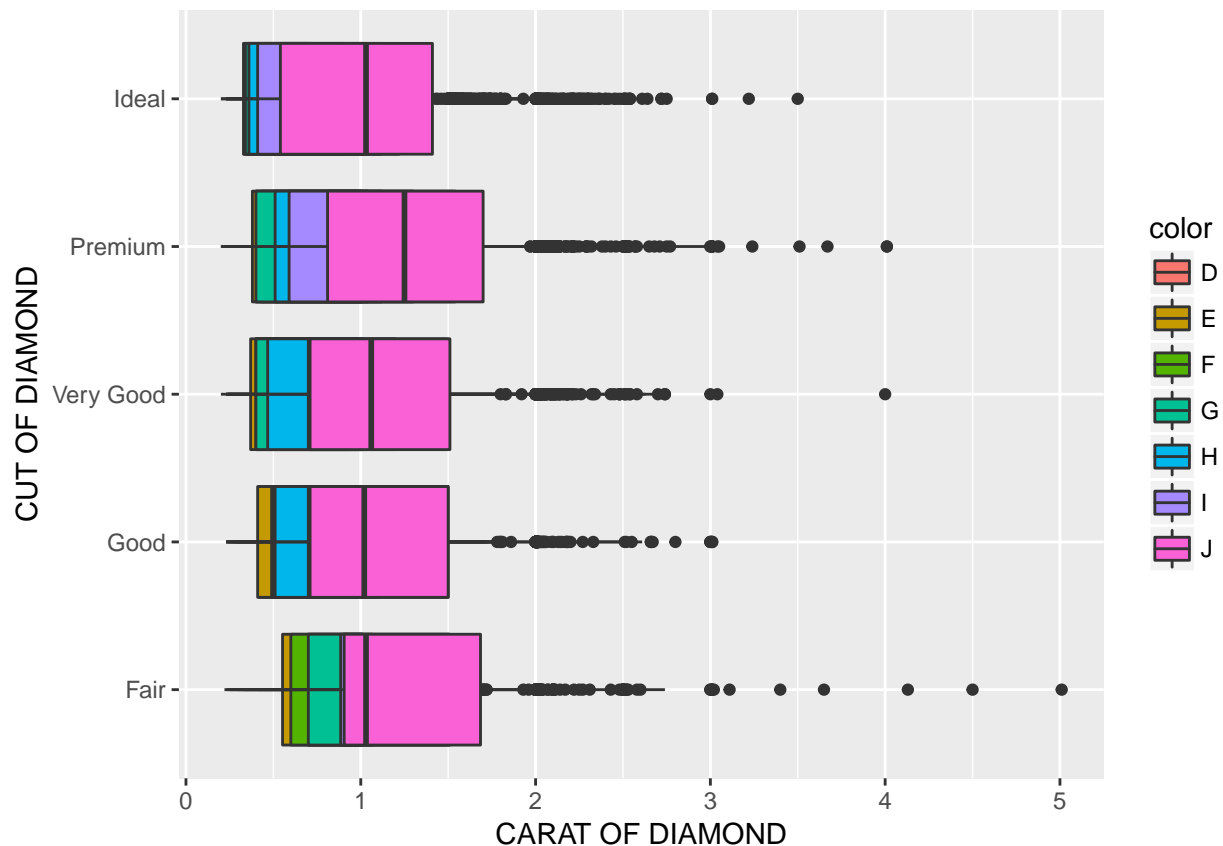
There are a few features of this visualization which I disagree with.

- a. **Area based Visualization** - I do not generally agree with bubble charts because they are area based representations of values which makes them difficult to interpret and compare.
- b. **Multiple Dimensions on same axis** - It seems like the author represented multiple dimensions - Age Group and Sex on the same y-axis in the graphic. It seems confusing and definitely a bad idea.
- c. **Percentages don't add up** - The sum of percentages between "Yes" and "No" responses on each row do not add up to 100%. In fact they are substantially lower than 100%. I believe there was another category in the survey form - "Don't Know" or equivalent NA. If this response is not important, I would probably remove NAs before computing percentages.

2. Reproduce this graphic using the diamonds data set.

Here is the code which reproduces the plot from the question:

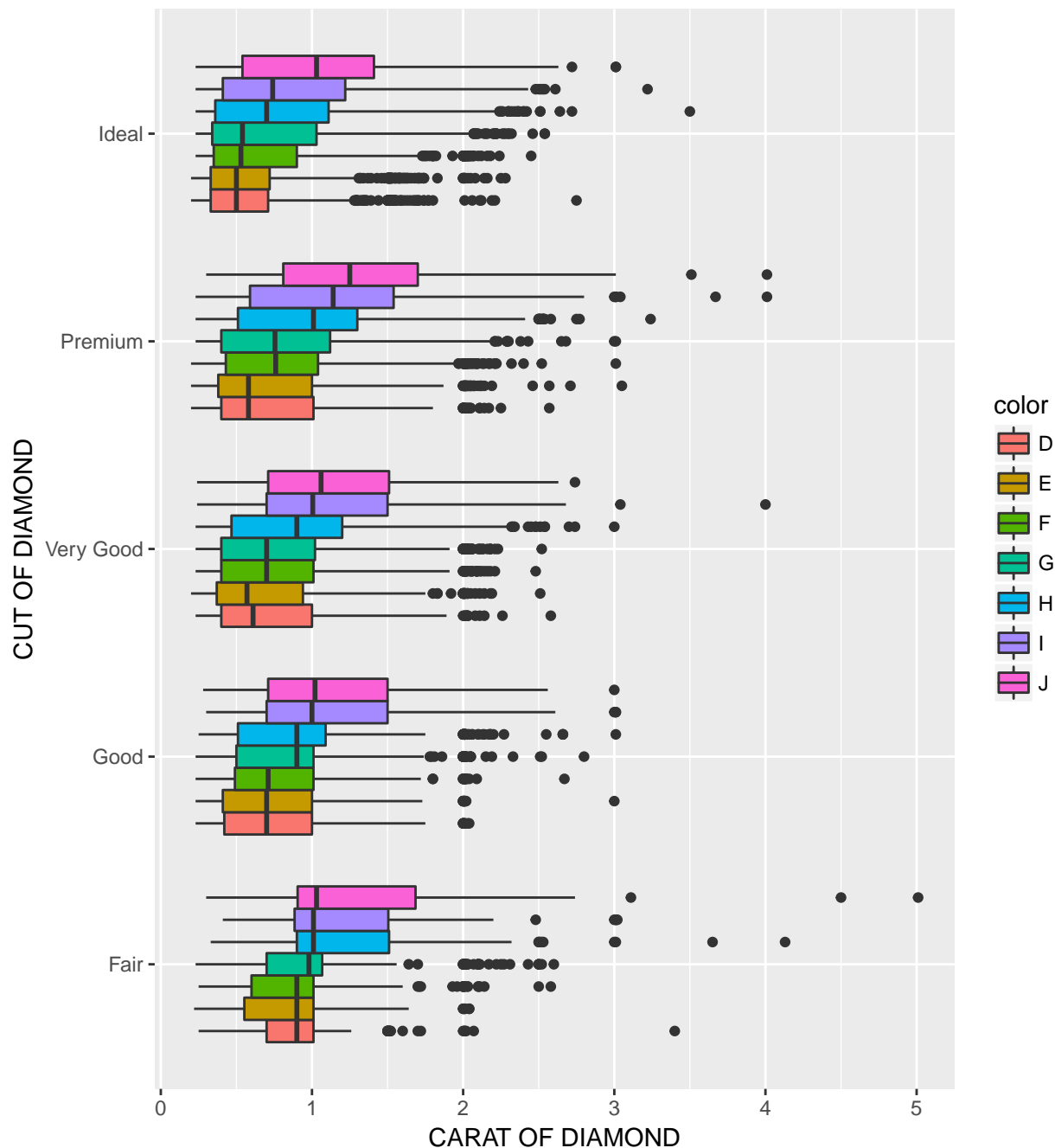
```
ggplot(data = diamonds, aes(x = cut, y = carat, fill = color)) +  
  geom_boxplot(position = "identity") +  
  xlab("CUT OF DIAMOND") +  
  ylab("CARAT OF DIAMOND") +  
  coord_flip()
```



3. The previous graphic is not very useful. We can make it much more useful by changing one thing about it. Make the change and plot it again.

The previous graphic is not very useful because individual color boxplots overlap each other which makes is unreadable. One simple way to fix this is by using the `dodge` position adjustment. Here is the code which generates the fixed plot:

```
ggplot(data = diamonds, aes(x = cut, y = carat, fill = color)) +  
  geom_boxplot(position = "dodge") +  
  xlab("CUT OF DIAMOND") +  
  ylab("CARAT OF DIAMOND") +  
  coord_flip()
```



Data munging and wrangling (6 points)

1. Is this data “tidy”? If yes, leave it alone and go to the next problem. If no, make it tidy.
Note: this data set is called table2 and is available in the tidyverse package. It should be ready for you to use after you’ve loaded the tidyverse package.

```
table2
```

```
## # A tibble: 12 x 4
##   country    year type      count
##   <chr>      <int> <chr>    <int>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases      2666
## 4 Afghanistan 2000 population 20595360
## 5 Brazil      1999 cases      37737
## 6 Brazil      1999 population 172006362
## 7 Brazil      2000 cases      80488
## 8 Brazil      2000 population 174504898
## 9 China       1999 cases      212258
## 10 China      1999 population 1272915272
## 11 China      2000 cases      213766
## 12 China      2000 population 1280428583
```

No, this data is not tidy because observations for `country` and `year` variables are split over multiple rows for different values of `type` variable. A tidy data should have one column per variable and one row for each observation. Here is the code to make it tidy:

```
table2 %>%
  spread(key = type, value = count)
```

```
## # A tibble: 6 x 4
##   country    year cases population
##   <chr>      <int> <int>    <int>
## 1 Afghanistan 1999     745  19987071
## 2 Afghanistan 2000    2666  20595360
## 3 Brazil      1999   37737  172006362
## 4 Brazil      2000   80488  174504898
## 5 China       1999  212258  1272915272
## 6 China       2000  213766  1280428583
```

2. Create a new column in the `diamonds` data set called `price_per_carat` that shows the price of each diamond per carat (hint: divide). Only show me the code, not the output.

```
diamonds %>%
  mutate(price_per_carat = price / carat)
```

3. For each cut of diamond in the diamonds data set, how many diamonds, and what proportion, have a price > 10000 and a carat < 1.5? There are several ways to get to an answer, but your solution must use the data wrangling verbs from the tidyverse in order to get credit.

```
diamonds %>%
  group_by(cut) %>%
  summarise(
    count = sum(price > 10000 & carat < 1.5, na.rm = TRUE),
    proportion = mean(price > 10000 & carat < 1.5, na.rm = TRUE)
  )
```

```
## # A tibble: 5 x 3
##   cut      count proportion
##   <ord>    <int>      <dbl>
## 1 Fair         4    0.00248
## 2 Good        17    0.00347
## 3 Very Good   155    0.0128
## 4 Premium     173    0.0125
## 5 Ideal       485    0.0225
```

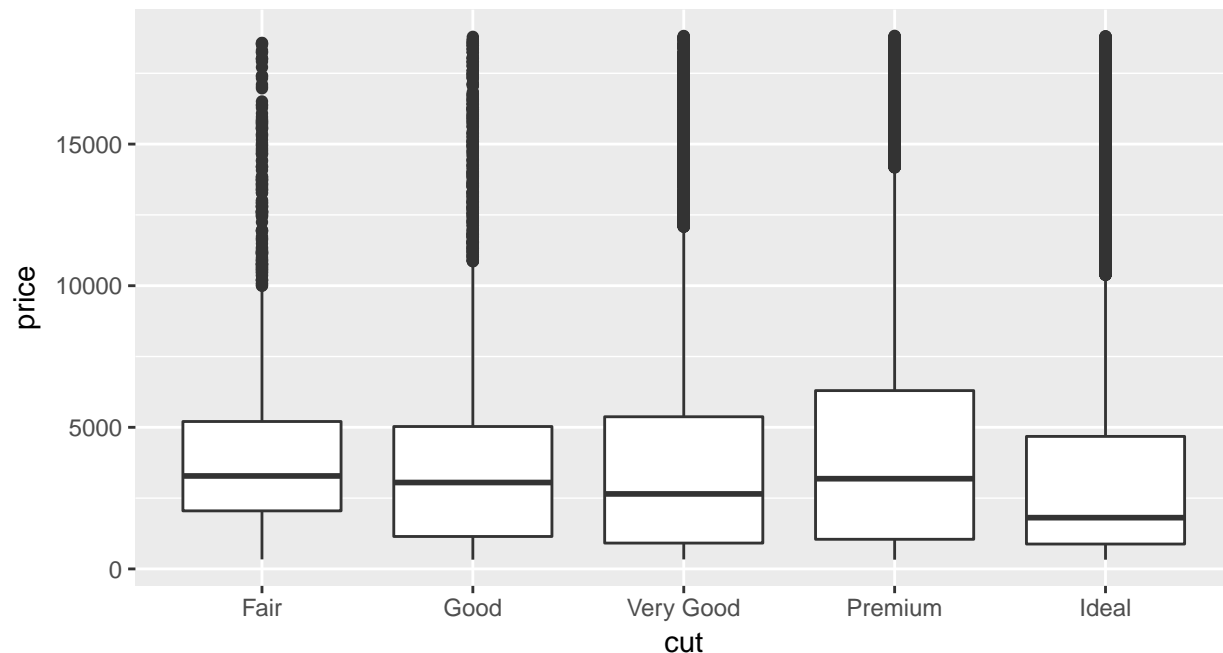
- Do the results make sense? Why?

Yes the results make sense to some extent. As the cut gets better from Fair to Ideal, the proportion of expensive diamonds, even though with lesser weight, increases. This shows that the price of diamonds also depend on the cut in addition to its weight in carats.

- Do we need to be wary of any of these numbers? Why?

Yes, we should be careful while making any inferences based on these numbers. Look at the box plot below of cut v/s price, we can clearly see that the points we are using lie in the area normally considered as outliers.

```
ggplot(data = diamonds, aes(x = cut, y = price)) +
  geom_boxplot()
```



EDA (6 points)

Take a look at the txhousing data set that is included with the ggplot2 package and answer these questions:

1. During what time period is this data from?

```
# TODO: Be careful with the mutate below. It is working fine for this
# dataset, but min/max may break due to months being 1 or 2 digit vals.
txhousing %>%
  mutate(yr_mon = paste0(year, "-", month)) %>%
  summarize(
    from = min(yr_mon),
    to = max(yr_mon)
  )
```

```
## # A tibble: 1 x 2
##   from   to
##   <chr> <chr>
## 1 2000-1 2015-7
```

The data in txhousing data set is from January 2000 to July 2015.

2. How many cities are represented?

```
txhousing %>%
  select(city) %>%
  n_distinct()
```

```
## [1] 46
```

There are 46 cities represented in txhousing data set.

3. Which city, month and year had the highest number of sales?

```
txhousing %>%
  arrange(desc(sales))
```

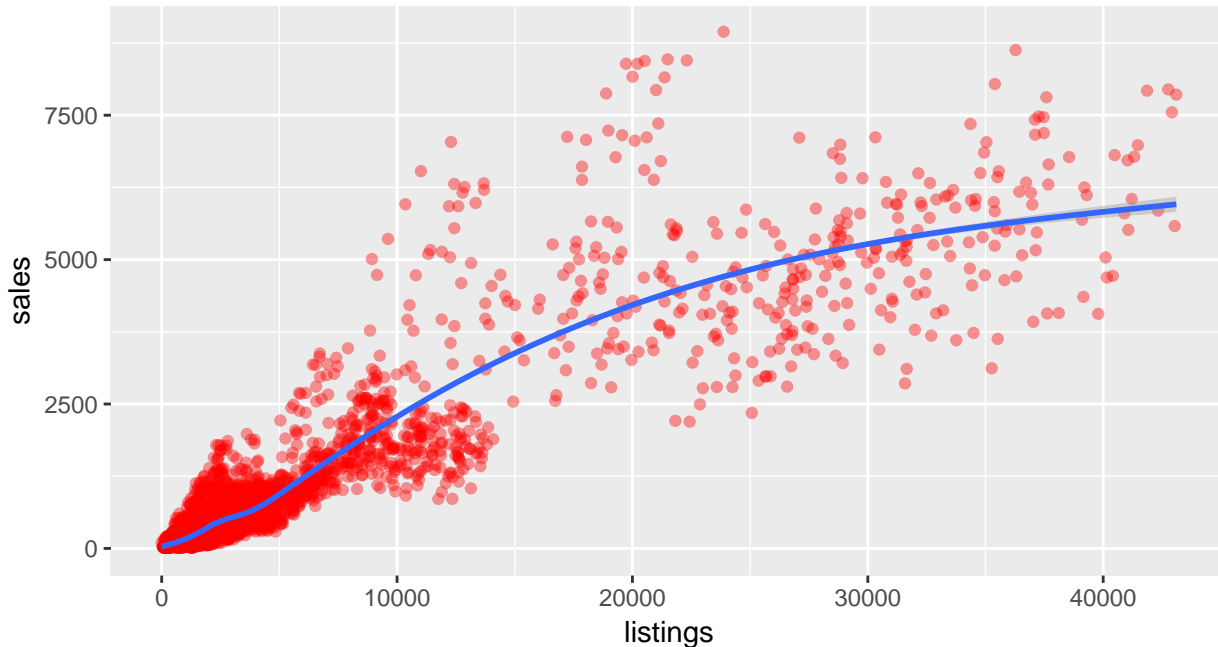
```
## # A tibble: 8,602 x 9
##   city      year month sales      volume median listings inventory date
##   <chr>    <int> <int> <dbl>      <dbl>    <dbl>    <dbl>    <dbl> <dbl>
## 1 Houston  2015      7 8945. 2568156780. 217600.   23875.    3.40 2016.
## 2 Houston  2006      6 8628. 1795898108. 155200.   36281.    5.60 2006.
## 3 Houston  2013      7 8468. 2168720825. 187800.   21497.    3.30 2014.
## 4 Houston  2015      6 8449. 2490238594. 222400.   22311.    3.20 2015.
## 5 Houston  2013      5 8439. 2121508529. 186100.   20526.    3.30 2013.
## 6 Houston  2014      6 8391. 2342443127. 211200.   19725.    2.90 2014.
## 7 Houston  2014      7 8391. 2278932511. 199700.   20214.    3.00 2014.
## 8 Houston  2014      8 8167. 2195184825. 202400.   20007.    2.90 2015.
## 9 Houston  2013      8 8155. 2083377894. 186700.   21366.    3.30 2014.
## 10 Houston 2006      5 8040. 1602621368. 151200.   35398.    5.50 2006.
## # ... with 8,592 more rows
```

Houston, in the month of July of 2015 had the highest number of sales.

4. What kind of relationship do you think exists between the number of listings and the number of sales? Check your assumption and show your work.

I think the number of listings should have a strong positive correlation with the number of sales. The more the listings, the more chances of making a sale.

```
ggplot(data = txhousing, aes(x = listings, y = sales)) +  
  geom_point(alpha = 0.4, color = "red") +  
  geom_smooth()
```



The scatter plot above confirms my assumption.

5. What proportion of sales is missing for each city?

Here is a list of proportion of sales missing by each city sorted in descending order of missing proportion.

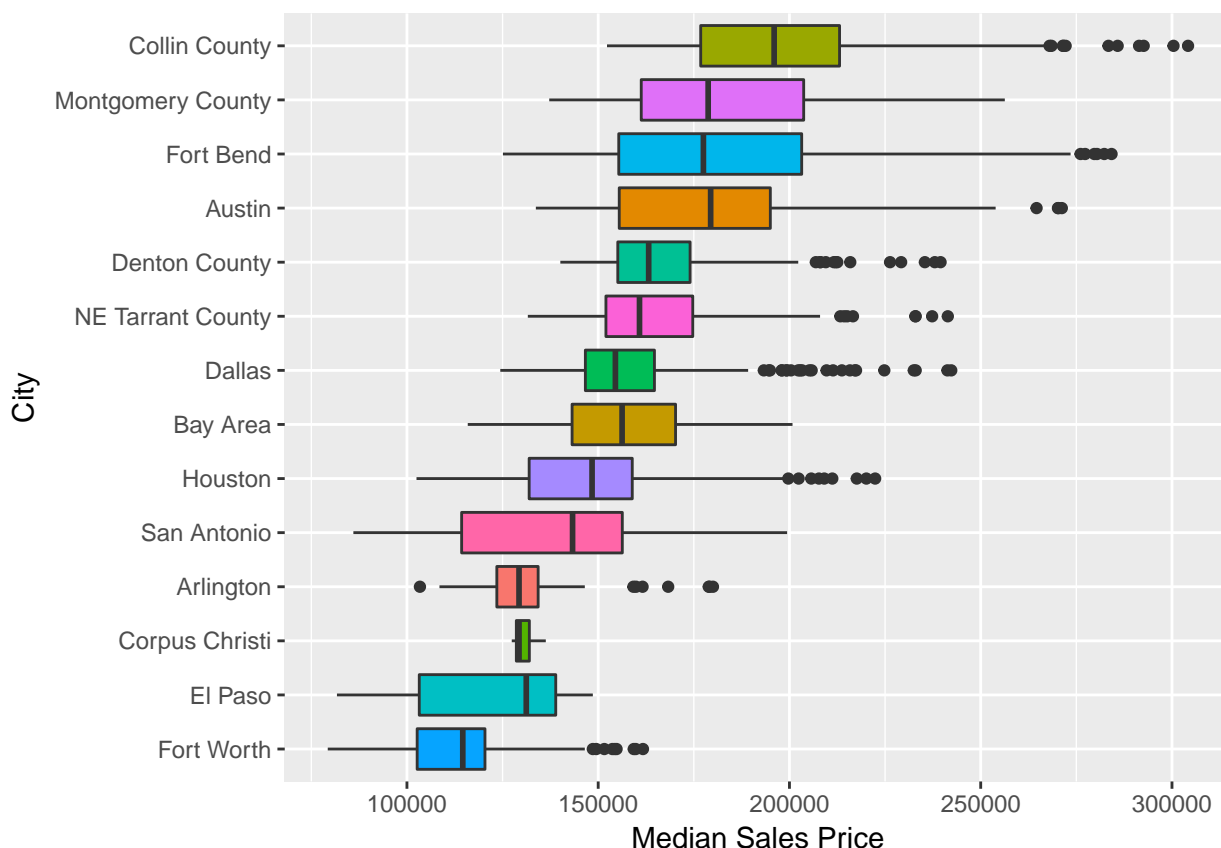
```
txhousing %>%  
  group_by(city) %>%  
  summarize(prop_missing = mean(is.na(sales))) %>%  
  arrange(desc(prop_missing))
```

```
## # A tibble: 46 x 2  
##   city                prop_missing  
##   <chr>                <dbl>  
## 1 South Padre Island    0.620  
## 2 Kerrville             0.556  
## 3 Midland              0.401  
## 4 Odessa               0.385  
## 5 San Marcos           0.246  
## 6 Laredo               0.193  
## 7 Harlingen            0.134  
## 8 Waco                 0.102  
## 9 Texarkana            0.0909  
## 10 Brazoria County      0.0749  
## # ... with 36 more rows
```

6. Looking at only the cities and months with greater than 500 sales:

- Are the distributions of the median sales price (column name median), when grouped by city, different? The same? Show your work.

```
txhousing %>%  
  filter(sales > 500) %>%  
  ggplot(aes(x = reorder(city, median, FUN = mean), y = median, fill = city)) +  
  geom_boxplot(show.legend = FALSE) +  
  xlab("City") + ylab("Median Sales Price") +  
  coord_flip()
```



The distribution of median sales prices certainly differs by city. Some cities like “Colin Country” have higher median sales price compared to others. The distribution around median varies a lot as well. For example, cities like “El Paso”, “San Antonio”, “Fort Bend” etc. have higher variance around median, whereas cities like “Corpus Christi” and “Arlington” have very less variance.

- Any cities that stand out that you’d want to investigate further?

Yes. We should further investigate the data for the cities of “Dallas”, “Colin County” and “Houston” because they seem to have too many outliers.

- Why might we want to filter out all cities and months with sales less than 500?

We want to filter out all cities and months with sales less than 500 because having them all on our plot will make it unreadable. Also these observations will not be useful in the EDA as they represent the sparse sales and not the majority of sales.

Git and Github (1.5 points)

To demonstrate your use of git and Github, at the top of your document put a hyperlink to your Github repository.

Once you are finished with your midterm, commit your final changes with the comment “finished the midterm - woohoo” and push your R Markdown file and your html or pdf file to Github.

See the Section My Git Repository above.