Michelle_Gomez_midterm.Rmd

Michelle Gomez

7/8/2018

The tidyverse packages

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

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

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

- Plotting - I used `geom_boxplot()` to make box plot graphs of data and `geom_point()` to make scatter plots of the data. Also, functions like `coord_flip()` can be used to flip the axis.
- Data munging/wrangling - The pipe operator `%>%` reads left-to-right, plugging in one input to the next via "pipes". Another useful function is `mutate()` because it allows you to create new columns from existing data.
- Reshaping data - `gather` allows you to gather columns based on key-value pairs while `spread()` lets you make the data wider by separting keys into columns.
- Importing/exporting data - `read_csv()` allows you to load csv files onto R. `write_delim` allows you to export a csv file with a delimeiter of your choosing.

R Basics

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

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

```
## [1] 1 2 3
```

2. Fix this code so it works:

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

```
## [1] "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"
```

The values in a vector can either be characters or integers. No matter where you add the single quotations, the output will always be characters with double quotations because R reads it as you're converting the integers to characters.

Data import/export

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.

```r
library(tidyverse)
file_path <- '/Users/michellegomez/Downloads/rail_trail.txt'
csv_data <- read_delim(file_path, 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()
## )
```

```r
glimpse(csv_data)
```

```
## 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....
## $ 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, 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, 0.0...
## $ 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.

```r
write_delim(csv_data, delim = ',', path = '/Users/michellegomez/Downloads/rail_trail.csv')
file_path2 <- '/Users/michellegomez/Downloads/rail_trail.csv'
csv_data2 <- read_csv(file_path2)
```

```
## 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(csv_data2)
```

```
## 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....
## $ 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, 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, 0.0...
## $ 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, ...
```

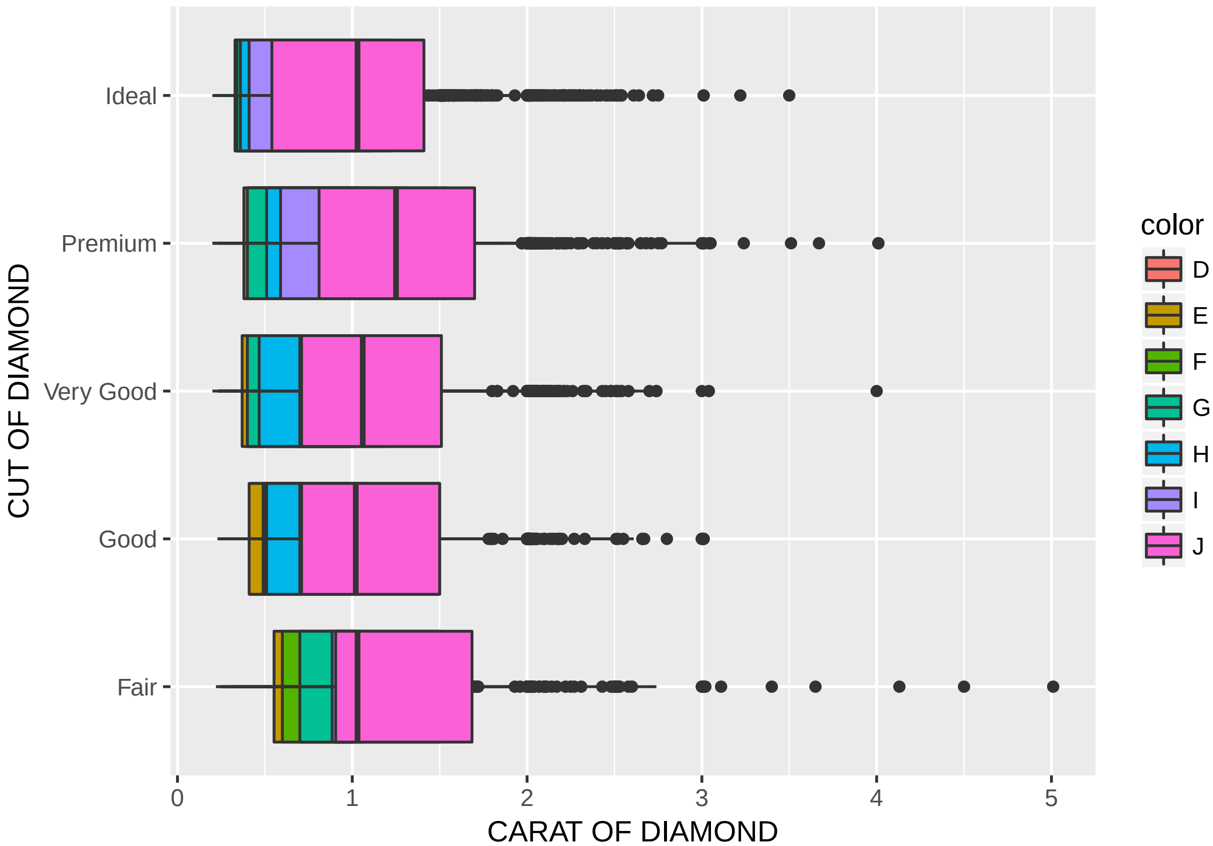
Visualization

1. Critique this graphic: give only three examples of what is wrong with this graphic. Be concise.
   A few things the graphic did wrong:

- Used bubbles for comparison of categorical variables instead of a bar graph.
- Did not incorporate male and female breakdown within each age group category.
- Because the breakdown is in percentages, they should have done a better job of showing the breakdown as part of a whole like in a stacked bar graph.

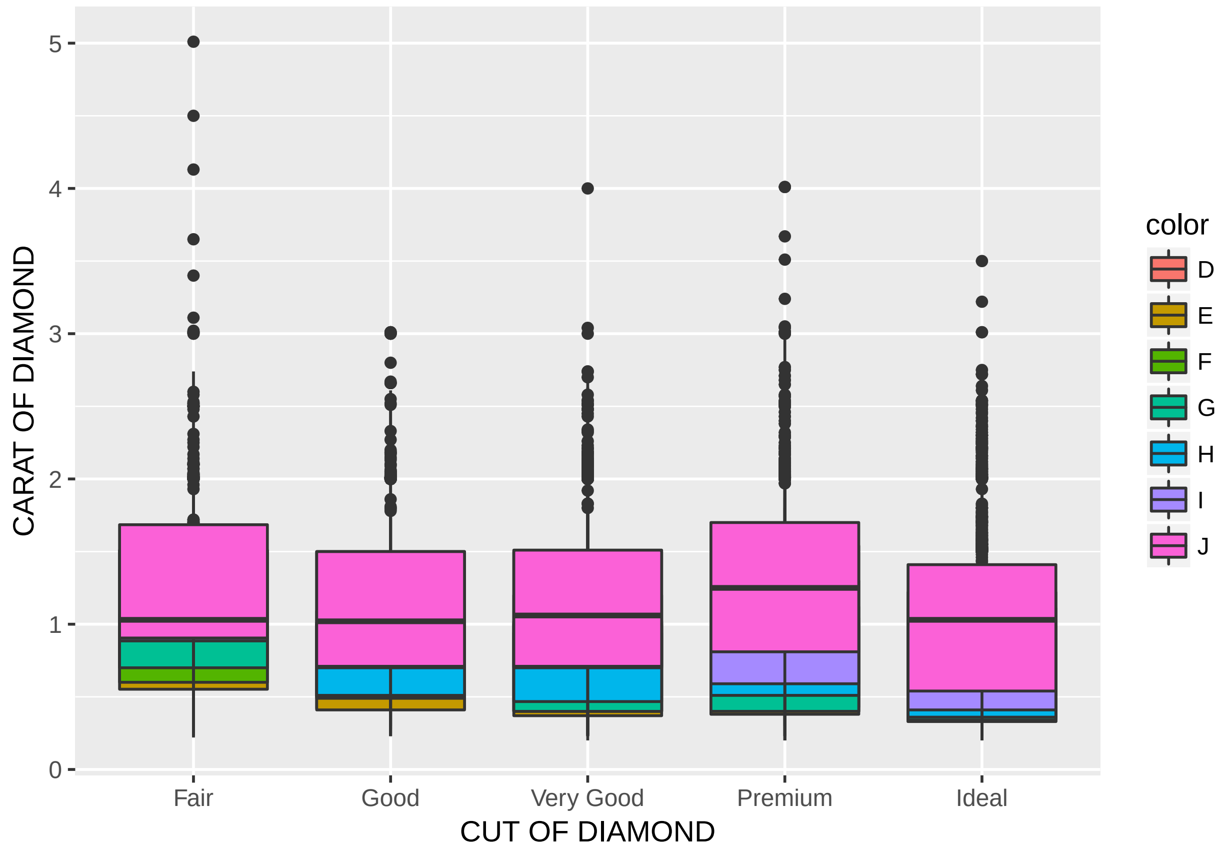
2. Reproduce this graphic using the diamonds data set.

```
library(ggplot2)
ggplot(data = diamonds) +
  geom_boxplot(mapping = aes(x = cut, y = carat, fill = color), position = "identity") +
  coord_flip() +
  xlab("CUT OF DIAMOND") +
  ylab("CARAT OF DIAMOND")
```



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.

```
library(ggplot2)
ggplot(data = diamonds) +
  geom_boxplot(mapping = aes(x = cut, y = carat, fill = color), position = "identity") +
  xlab("CUT OF DIAMOND") +
  ylab("CARAT OF DIAMOND")
```



Data munging and wrangling

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.

```
library(dplyr)
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.

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 53,940 x 11
##    carat cut       color clarity depth table price     x     y     z
```

```
##     <dbl> <ord>     <ord> <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##   1 0.230 Ideal     E     SI2       61.5  55.0   326  3.95  3.98  2.43
##   2 0.210 Premium   E     SI1       59.8  61.0   326  3.89  3.84  2.31
##   3 0.230 Good      E     VS1       56.9  65.0   327  4.05  4.07  2.31
##   4 0.290 Premium   I     VS2       62.4  58.0   334  4.20  4.23  2.63
##   5 0.310 Good      J     SI2       63.3  58.0   335  4.34  4.35  2.75
##   6 0.240 Very Good J     VVS2      62.8  57.0   336  3.94  3.96  2.48
##   7 0.240 Very Good I     VVS1      62.3  57.0   336  3.95  3.98  2.47
##   8 0.260 Very Good H     SI1       61.9  55.0   337  4.07  4.11  2.53
##   9 0.220 Fair      E     VS2       65.1  61.0   337  3.87  3.78  2.49
##  10 0.230 Very Good H     VS1       59.4  61.0   338  4.00  4.05  2.39
## # ... with 53,930 more rows, and 1 more variable: price_per_carat <dbl>
```

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 %>%
  filter(price > 10000 & carat < 1.5)%>%
  group_by(price, carat) %>% summarize(count=n())
```

```
## # A tibble: 802 x 3
## # Groups:   price [?]
##     price carat count
##     <int> <dbl> <int>
##   1 10003  1.03     1
##   2 10004  1.23     1
##   3 10006  1.25     1
##   4 10009  1.21     1
##   5 10011  1.01     1
##   6 10011  1.05     1
##   7 10012  1.35     1
##   8 10016  1.13     1
##   9 10018  1.21     1
##  10 10019  1.01     1
## # ... with 792 more rows
```

Do the results make sense? Why? Do we need to be wary of any of these numbers? Why? Yes because carata may not be accurate.

EDA

1. During what time period is this data from?

```
txhousing %>%
  select(year, month, date) %>%
  arrange(desc(date))
```

```
## # A tibble: 8,602 x 3
##      year month  date
##     <int> <int> <dbl>
##   1  2015     7  2016
##   2  2015     7  2016
##   3  2015     7  2016
##   4  2015     7  2016
```

```
## 5 2015 7 2016
## 6 2015 7 2016
## 7 2015 7 2016
## 8 2015 7 2016
## 9 2015 7 2016
## 10 2015 7 2016
## # ... with 8,592 more rows
```

The date is from October 2013 through August 2015.
2. How many cities are represented?

```r
txhousing %>% group_by(city) %>% summarize(count=n())
```

```
## # A tibble: 46 x 2
##    city                   count
##    <chr>                  <int>
##  1 Abilene                  187
##  2 Amarillo                 187
##  3 Arlington                187
##  4 Austin                   187
##  5 Bay Area                 187
##  6 Beaumont                 187
##  7 Brazoria County          187
##  8 Brownsville              187
##  9 Bryan-College Station    187
## 10 Collin County            187
## # ... with 36 more rows
```

46 cities are represented.
3. Which city, month and year had the highest number of sales?

```r
txhousing %>%
  select(sales, city, month, year) %>%
  arrange(desc(sales))
```

```
## # A tibble: 8,602 x 4
##    sales city    month  year
##    <dbl> <chr>   <int> <int>
##  1  8945 Houston     7  2015
##  2  8628 Houston     6  2006
##  3  8468 Houston     7  2013
##  4  8449 Houston     6  2015
##  5  8439 Houston     5  2013
##  6  8391 Houston     6  2014
##  7  8391 Houston     7  2014
##  8  8167 Houston     8  2014
##  9  8155 Houston     8  2013
## 10  8040 Houston     5  2006
## # ... with 8,592 more rows
```

The highest sales were for Houston on August 2015.
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.

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

```r
txhousing %>%
  filter(is.na(sales)) %>%
```

```r
  group_by(city) %>% summarize(count=n()) %>% mutate(proportion = count/187)
```

```
## # A tibble: 20 x 3
##    city                count proportion
##    <chr>               <int>      <dbl>
##  1 Brazoria County        14    0.0749
##  2 Brownsville             2    0.0107
##  3 Corpus Christi          1    0.00535
##  4 Galveston               1    0.00535
##  5 Harlingen              25    0.134
##  6 Kerrville             104    0.556
##  7 Killeen-Fort Hood       1    0.00535
##  8 Laredo                 36    0.193
##  9 Longview-Marshall      12    0.0642
## 10 Lubbock                 1    0.00535
## 11 McAllen                 2    0.0107
## 12 Midland                75    0.401
## 13 Nacogdoches            11    0.0588
## 14 Odessa                 72    0.385
## 15 Port Arthur             2    0.0107
## 16 San Marcos             46    0.246
## 17 South Padre Island    116    0.620
## 18 Temple-Belton          11    0.0588
## 19 Texarkana              17    0.0909
## 20 Waco                   19    0.102
```

Above you see only cities with missing sales and the proportions of those missing sales to total number of sales. All other cities have a proportion of 0.

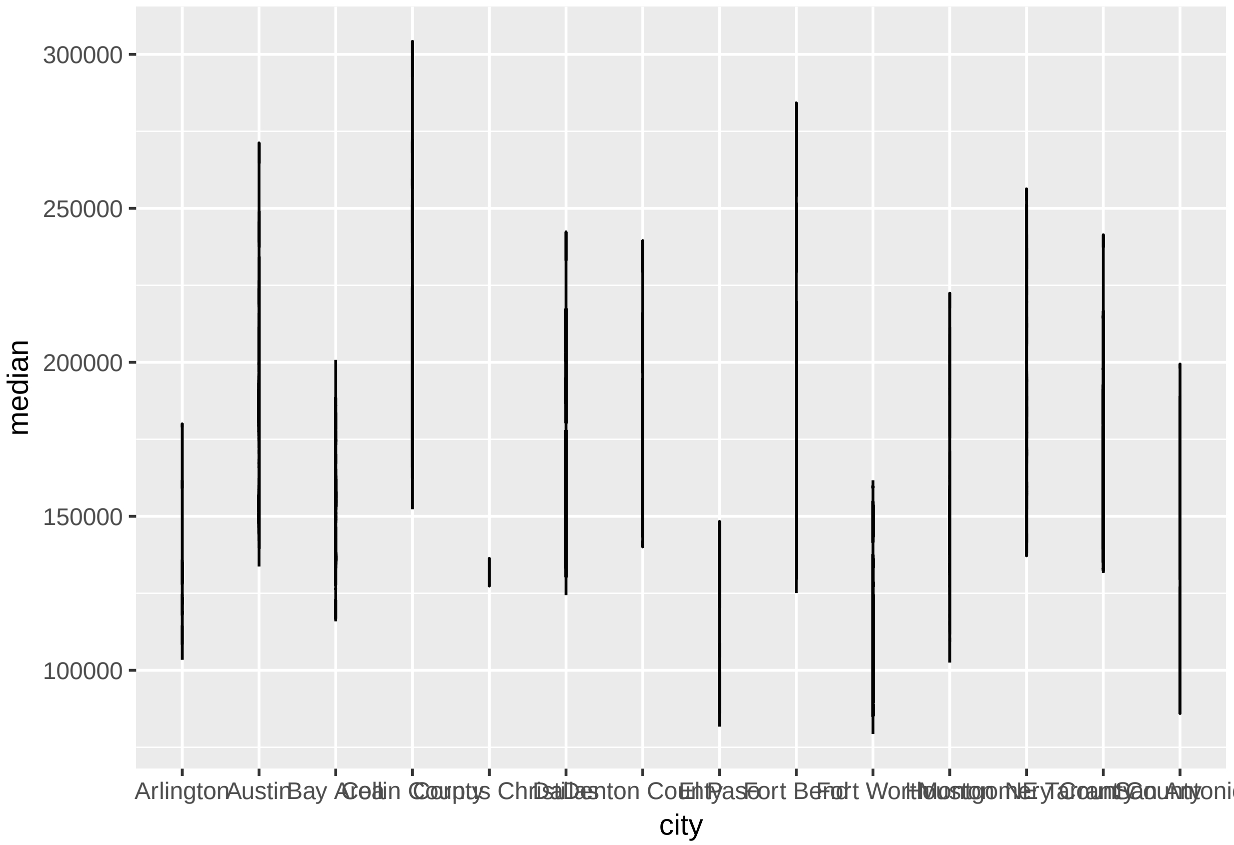
6. Looking at only the cities and months with greater than 500 sales:

```r
txhousing %>%
filter(sales > 500)
```

```
## # A tibble: 1,883 x 9
##    city       year month sales   volume median listings inventory  date
##    <chr>     <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
##  1 Arlington  2000     8   507 60875199 103400     1417      3.50  2001
##  2 Arlington  2001     5   536 69878959 114400     1592      3.70  2001
##  3 Arlington  2001     6   534 67744182 108500     1627      3.80  2001
##  4 Arlington  2001     8   505 65080743 113600     1616      3.70  2002
##  5 Arlington  2002     5   503 67240236 116100     1741      3.90  2002
##  6 Arlington  2002     7   509 66954143 119100     1925      4.40  2002
##  7 Arlington  2003     5   502 67131982 118000     2544      5.90  2003
##  8 Arlington  2003     7   524 73194692 123500     2799      6.50  2004
##  9 Arlington  2003     8   531 72397143 123900     2801      6.40  2004
## 10 Arlington  2004     5   527 72401436 118300     2922      6.50  2004
## # ... with 1,873 more rows
```

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

```r
txhousing %>%
  filter(sales > 500) %>%
  select(median, city) %>%
  ggplot(aes(x = city, y = median)) +
  geom_line()
```



The distributions are clearly different as you see in graph above.
- Any cities that stand out that you'd want to investigate further?
I would like to investigate Collin County (highest median) and Corpus Christi (smallest distribution). - Why might we want to filter out all cities and months with sales less than 500?
There are many reasons, but I think it's better to filter this way to narrow down on those that sell more to understand why.

Git and Github