Answers to questions: Introduction to Exploratory Data Analysis

Sean Davis*

*seandavi@gmail.com

30 March 2022

Abstract

This is the answer key for questions and exercises in Exploratory Data Analysis Lab.

Contents

Exercises and questions

How do we know if the `tidyverse` package is installed?

1. Try loading the package using `library(tidyverse)`

If you get an error like:

```
Error in library(tidyverse) : there is no package called 'tidyverse'
```

Then you know that the package is not yet installed (or you typed the package name incorrectly).

How do you install the `tidyverse` package if needed?

```
install.packages('tidyverse')
```

How do you load the `tidyverse` package before using it?

```
library(tidyverse)
## -- Attaching packages -------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ----------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Explore the distribution of each of the quantitatve variables in `diamonds` using a histogram.

First, we need to look at which variables are quantitative. How can we do that?

1. We can use `View` on the dataset to look at it in Rstudio and *guess* the column types based on what we see.

```
View(diamonds)
```

2. We can use `summary` on the dataset to have R show us a column-by-column summary of the data.

```
summary(diamonds)
##      carat               cut        color
##  Min.   :0.2000   Fair      : 1610   D: 6775
##  1st Qu.:0.4000   Good      : 4906   E: 9797
##  Median :0.7000   Very Good:12082   F: 9542
##  Mean   :0.7979   Premium   :13791   G:11292
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304
```

```
##  Max.   :5.0100                    I: 5422
##                                    J: 2808
##     clarity          depth           table
##  SI1    :13065   Min.   :43.00   Min.   :43.00
##  VS2    :12258   1st Qu.:61.00   1st Qu.:56.00
##  SI2    : 9194   Median :61.80   Median :57.00
##  VS1    : 8171   Mean   :61.75   Mean   :57.46
##  VVS2   : 5066   3rd Qu.:62.50   3rd Qu.:59.00
##  VVS1   : 3655   Max.   :79.00   Max.   :95.00
##  (Other): 2531
##      price             x               y
##  Min.   :  326   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720
##  Median : 2401   Median : 5.700   Median : 5.710
##  Mean   : 3933   Mean   : 5.731   Mean   : 5.735
##  3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540
##  Max.   :18823   Max.   :10.740   Max.   :58.900
##
##        z
##  Min.   : 0.000
##  1st Qu.: 2.910
##  Median : 3.530
##  Mean   : 3.539
##  3rd Qu.: 4.040
##  Max.   :31.800
##
```

Quantitative variables in the `summary` output include min, max, etc. Therefore, our variables of interest are:

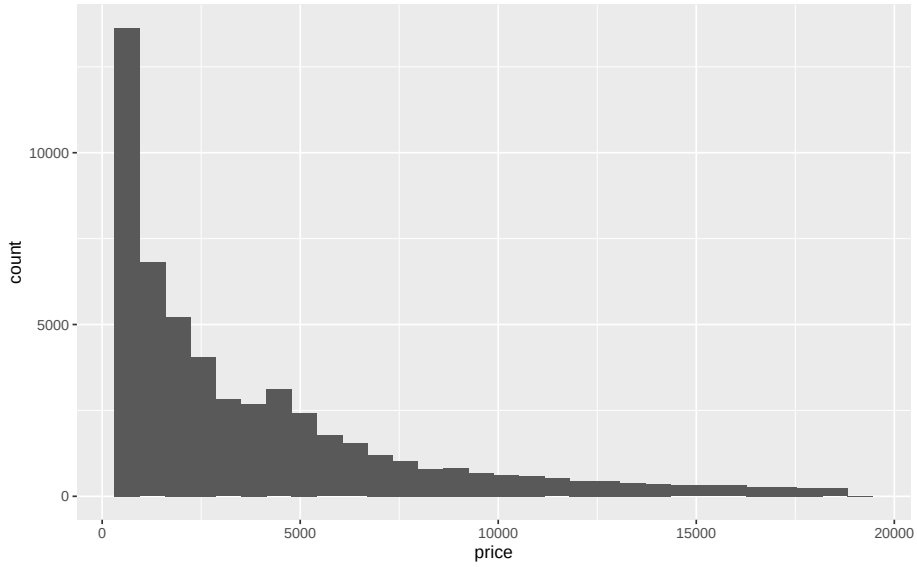
- carat
- depth
- table
- price
- x
- y
- z

To create a histogram using ggplot, we use code like:

```
ggplot(DATASET, aes(x=VARIABLE_NAME)) +
  geom_histogram()
```

For example, to plot the histogram of price:

```
ggplot(diamonds, aes(x=price)) +
  geom_histogram()
## `stat_bin()` using `bins = 30`. Pick better value
## with `binwidth`.
```

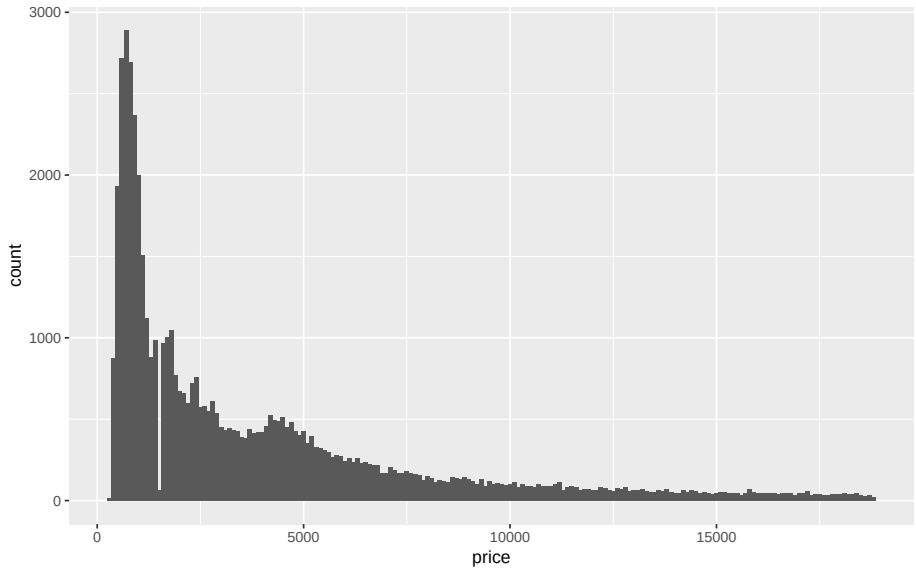


Explore the distribution of `price`

- Do you discover anything unusual or surprising? (Hint: Carefully think about the `binwidth` and make sure you try a wide range of values.)

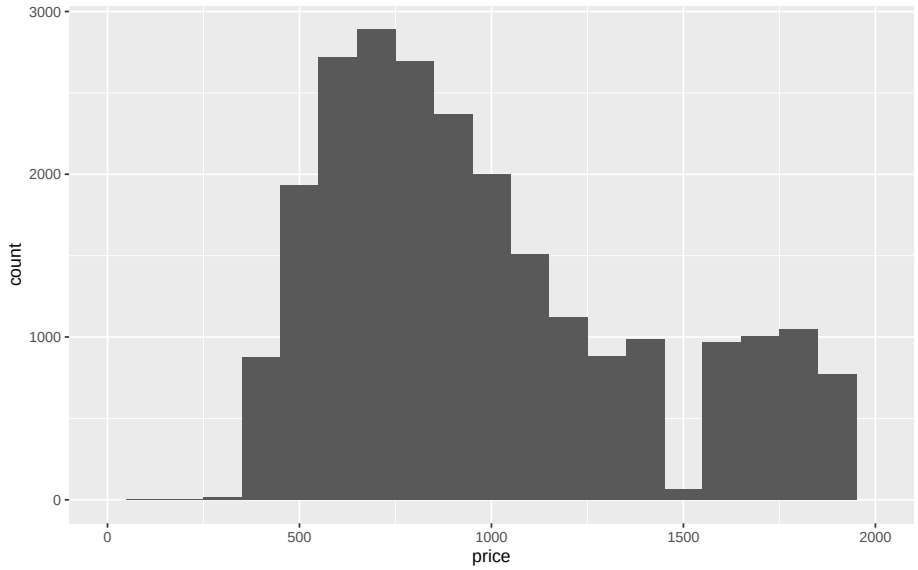
Let's try using a binwidth of 100. The `binwidth` is the size of the x-axis bins in which we will count samples. Therefore, a `binwidth` of 100 for `price` will result in counting all the diamonds in price ranges from \$0-100, \$101-200, . . . , \$1001-1100, etc.

```
ggplot(diamonds, aes(x=price)) +
  geom_histogram(binwidth=100) # change binwidth
```



We can "zoom in" on what I see as an anomoly using `xlim` which limits the x-axis on the plot:

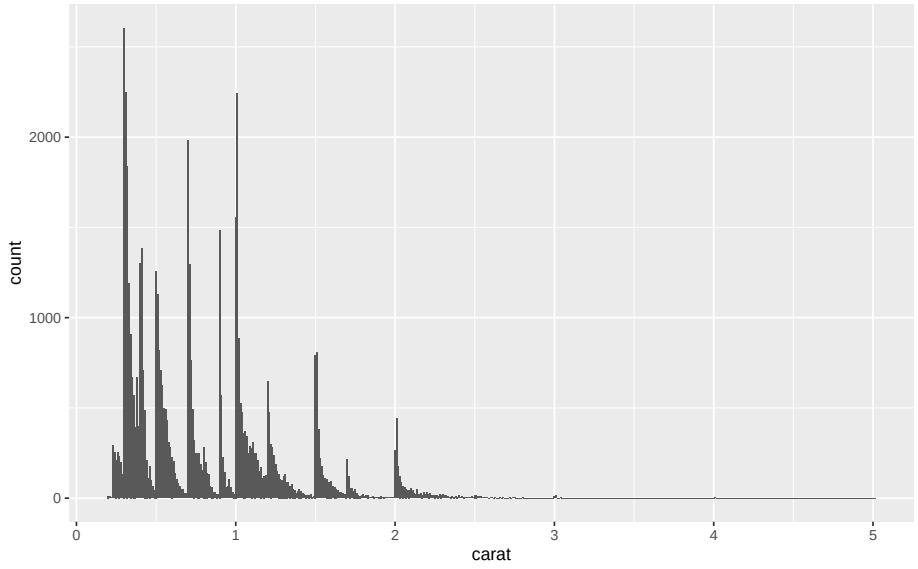
```
ggplot(diamonds, aes(x=price)) +
  geom_histogram(binwidth=100) +
  xlim(0, 2000) # zoom in
## Warning: Removed 29733 rows containing non-finite values
## (stat_bin).
## Warning: Removed 2 rows containing missing values
## (geom_bar).
```



How many diamonds are 0.99 carats and how many are 1 carat?

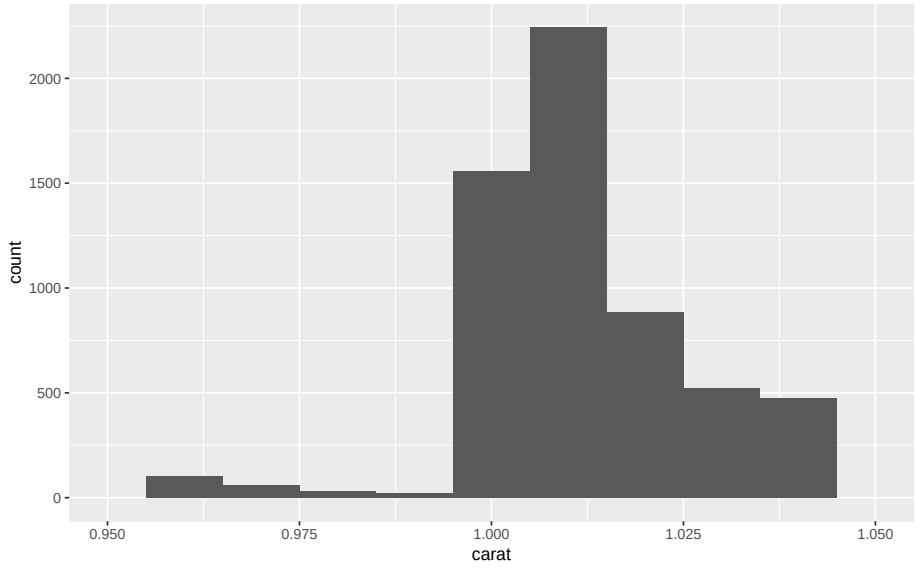
Again, we can turn to a histogram to take a look at our `carat` distribution.

```
ggplot(diamonds, aes(x=carat)) +
  geom_histogram(binwidth=0.01)
```



We can zoom in using `xlim` again:

```
ggplot(diamonds, aes(x=carat)) +
  geom_histogram(binwidth=0.01) +
  xlim(0.95,1.05)
## Warning: Removed 47617 rows containing non-finite values
## (stat_bin).
## Warning: Removed 2 rows containing missing values
## (geom_bar).
```



There is an obvious jump in `carat` from 0.99 to 1.0. Why?. While our data don't tell us the *why* directly, a little googling turns up this explanation[1].

> The size difference between a 0.98 ct diamond and a 1.01 ct diamond is difficult to distinguish, but some people prefer symbolic numbers, like round ones. The trade calls such diamonds weighing 0.25 ct, 0.50 ct, 0.75 ct, 1.00 ct, etc., "magic sizes." Although you may not see many diamonds that are magic sizes, you still should know about them. Here's why: Diamonds that are at or just above a magic size are generally more expensive per carat than diamonds that weigh a little less. So if you can live without the symbolic "weight" of these round numbers, you could save some money by choosing a diamond that weighs a few points less.

[1] see https://4cs.gia.edu/en-us/blog/nine-things-about-diamond-carat-w

With this explanation, take a look at the full histogram of `carat` again. Can you pick out the "magic sizes" now?

Create a boxplot for each of the categorical variables and price. What categorical variables influence price and how?

The categorical variables in our dataset (the ones that are not quantitative variables, but have a relatively small number of repeated values) are:

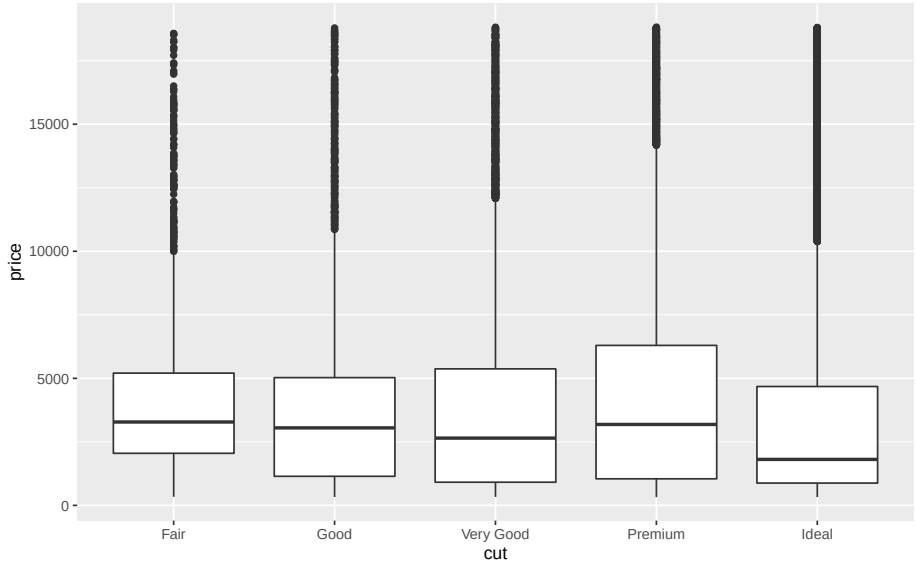
- cut
- color
- clarity

To create a boxplot, we use the ggplot code that looks like:

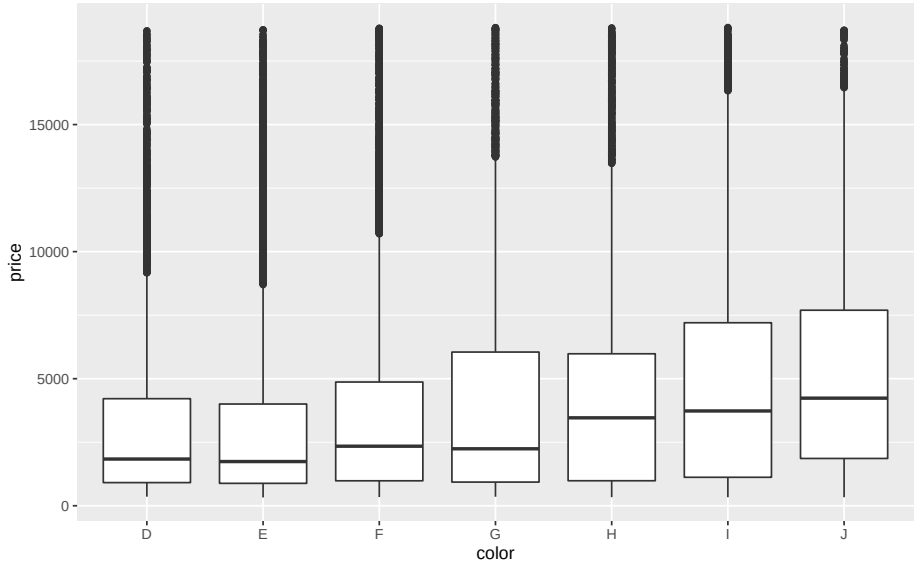
```
ggplot(diamonds, aes(x=CATEGORICAL_VARIABLE, y=QUANTITATIVE_VARIABLE)) +
  geom_boxplot()
```

For example, for cut vs. price:

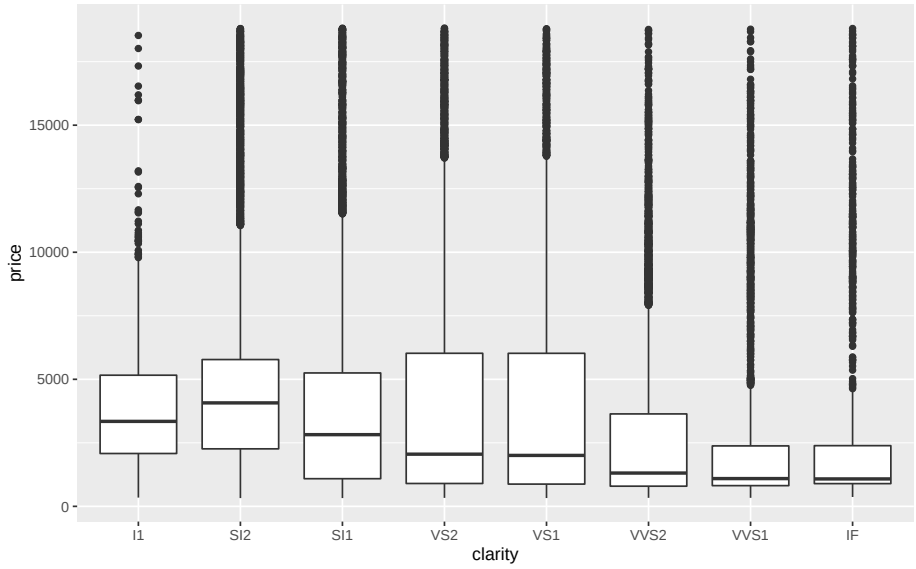
```
ggplot(diamonds, aes(x=cut, y=price)) +
  geom_boxplot()
```




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

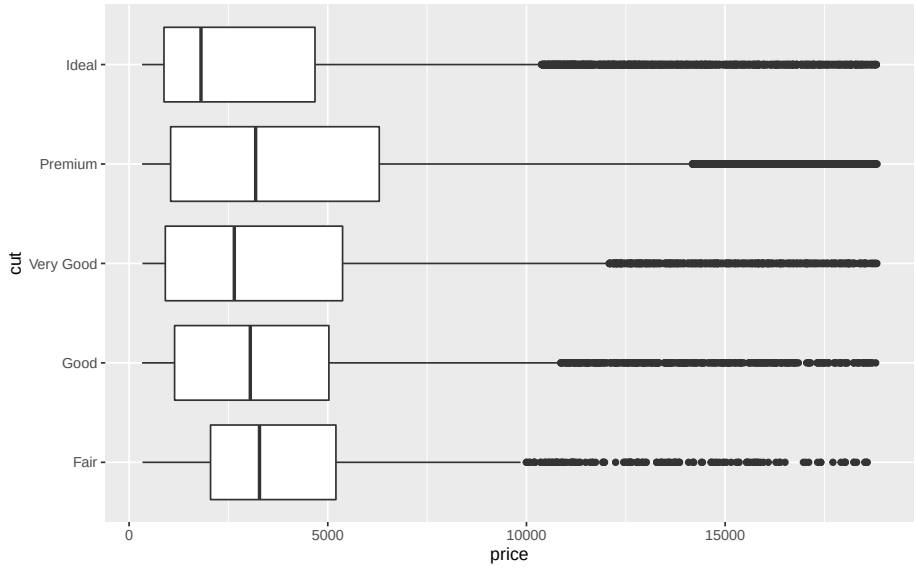



```
ggplot(diamonds, aes(x=clarity, y=price)) +
  geom_boxplot()
```



You can also create a "horizontal" boxplot by exchanging the `x` and `y` values.

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



Install the lvplot package, and try using `geom_lv()` to display the distribution of price vs cut.

One problem with boxplots is that they were developed in an era of much diamonds datasets and tend to display a prohibitively large number of "outlying values". One approach to remedy this problem is the letter value plot. What do you learn? How do you interpret the plots?

```
install.packages("lvplot")
```

Once installed, we load the library.

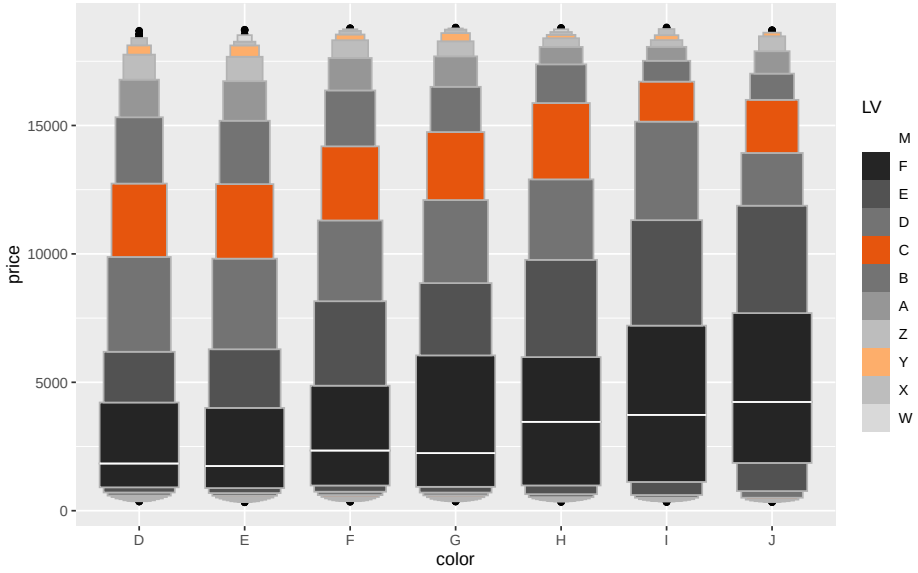
```
library(lvplot)
```

The suggestion is to use the `geom_lv()` function. We can get help on that function by typing:

```
help(geom_lv)
```

The help is pretty extensive, but we can start with the idea that we are going to replace our boxplots with lvplots. Take my word for the following and use the help page to play with the code to try different plots if you like.

```
ggplot(diamonds, aes(x = color, y = price)) +
  geom_lv(alpha=1, aes(fill=..LV..)) + scale_fill_lv()
```



3. Compare `geom_violin()` with a facetted `geom_boxplot()`. What are the pros and cons of each method?