Introduction to Exploratory Data Analysis

Sean Davis*

*seandavi@gmail.com

30 March 2022

Abstract

The goal of EDA is to get to know and understand your data and to use the tools at your disposal to visualize, summarize, interpret, and ask meaningful questions of your data. There is no single or "correct" way to perform EDA. In this little lab, we will be exploring the built-in `diamonds` dataset to demonstrate some approaches to EDA.

Contents

1 Learning outcomes . . . 2
1.1 Goals . . . 2
1.2 Objectives . . . 2
2 Setup for R markdown . . . 2
3 Introduction to EDA . . . 4
4 Guide EDA With Questions . . . 4
5 Learning about the Diamonds dataset . . . 5
5.1 Preliminaries . . . 5
6 Variation . . . 5
6.1 Visualizing distributions . . . 6
6.2 Covariation . . . 11

1 Learning outcomes

Note: This lab borrows/steals from Hadley Wickham's R for Data Science Book chapter on the same topic[1].

[1] https://r4ds.had.co.nz/exploratory-data-analysis.html

1.1 Goals

- Know the difference between categorical variables and quantitative (or numeric or continuous variables)
- Use visualization to examing the variation of one variable
- Use visualization to examine the covariation between
  - Two quantitative variables
  - A categorical variable and a quantitative variable
- Gain familiarity with using the ggplot2 library for plotting
- Gain experience with Exploratory Data Analysis

1.2 Objectives

- Be able to install and load the `ggplot2` package
- Use `View` to look at a dataset in Rstudio
- Use `summary`, `glimpse`, and `dim` to examine a dataset
- Use `ggplot2` to generate:
  - histograms
  - barplots
  - boxplots and violin plots
  - scatterplots
- Create and `knit` an Rmarkdown document to capture your work

2 Setup for R markdown

1. Create a new R Markdown document with *File* > *New File* > *R Markdown...* Knit it by clicking the appropriate button. Knit it by using the appropriate keyboard short cut. Verify that you can modify the input and see the output update by re-knitting the document.
2. Replace all the text in your Rmarkdown document with this text:

```
---
title: "EDA on the Diamonds dataset"
author: "YOUR NAME HERE"
date: "`r Sys.Date()`"
output: html_document
---

# Introduction

- use text to describe the Diamonds dataset
- Consider doing some google searching and including urls of background reading

You can include a url by simply including it like so:
```

````
- https://google.com

# Data exploration

## Load data

```{r}
library(tidyverse)
data(diamonds)
```

## Explore data

- `dim`
- `colnames`
- `glimpse`
- `View`

For example:

```{r}
dim(diamonds)
```

### What colors of diamonds are there and in what proportions are they in our dataset?

```{r}
ggplot(diamonds, aes(x=color)) +
  geom_bar()
```

From this plot, the most common color of diamond is ..... The least common color is....

Since diamonds colored .... are rarer than other colors, they might cost more in general.

TODO: Does diamond color affect price? To answer this question, I need to....

### Quesiton #2

### Question #3

...

## Conclusions

## Future directions
````

3 Introduction to EDA

We will often want to use visualisation and transformation to explore data in a systematic way. This is particularly valuable when presented with a new dataset or when the questions to ask of a dataset are unclear. Often refered to as "Exploratory Data Analysis", or EDA for short, it is an iterative cycle. While EDA isn't a formal proces, it can be summarized as:

1. Generate questions about your data.
2. Search for answers by visualising, transforming, and modelling your data.
3. Use what you learn to refine your questions and/or generate new questions.

That said, EDA does not really follow rules nor does it need to follow the steps above strictly. Instead, it is a mindset that relies heavily on curiosity. Like in writing, often the first step is to "brainstorm." Investigate every idea that occurs to you. Some of these ideas will end up in the final product, and some will be dead ends or to be saved for another day. Documenting the experiments from EDA is a useful way to understand your data and capture thoughts for later. Some of your EDA will lead to ideas that you want to communicate to others, but having all your work available for reference can be valuable.

Exploratory data analysis is an integral part of any data analysis, even if the questions are clearly formulated already. Almost all real-world datasets have unique features or quality issues that can only be discovered through exploration. One of the most important questions to ask is: "Are these data appropriate and of high enough quality to answer the questions being asked?"

4 Guide EDA With Questions

The goal of EDA is to get to know and understand your data. Using questions to guide data exploration is a common and effective approach[^1][^2]. In practical terms, writing down the question (in Rmarkdown, for example) and then answering it can help you and your future audience understand your thought process. Asking questions focuses attention to a particular slice of your dataset and helps you decide which graphs, summaries, models, or transformations to make.

> "There are no routine statistical questions, only questionable statistical routines."
> — Sir David Cox

EDA is fundamentally a creative process. Like many creative processes, the key to asking *quality* questions is to generate a large *quantity* of questions. It is difficult to ask insightful questions at the start of an analysis. Start with basic descriptive questions about individual variables (columns), for example. Each new question will expose a new aspect of your data and increase the chances of finding something new and interesting–making a discovery. Unlike experimental science where questions generated by one set of experiments can be costly and time-consuming to answer, questions about data can sometimes be answered quickly. Drill down into the most interesting aspects of data, while avoiding problematic aspects of data. Develop a set of thought-provoking questions (those that interest you) and follow up each question with a new question based findings.

> "Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise."
> — John Tukey

There is no rule about which questions to ask. Questions should guide your research, but do not avoid asking questions that interest you. However, two types of questions will always be useful for making discoveries within your data. Roughly, these questions are:

1. What type of variation occurs within each variable?
2. What type of covariation occurs between my variables?

In this chapter, we will examine what variantion and covariation are. We'll also learn some practical approaches to answer questions 1 and 2. To make the discussion easier, let's define some terms:

- A **variable** is a quantity, quality, or property that you can measure[2].
- A **value** is the state of a variable when you measure it. The value of a variable may change from measurement to measurement[3]
- An **observation** is a set of measurements made under similar conditions (you usually make all of the measurements in an observation at the same time and on the same object)[4]. An observation will contain several values, each associated with a different variable. Some people may call an observation a "data point".
- **Tabular data** is a set of values, each associated with a variable and an observation. Tabular data is *tidy* if each value is placed in its own "cell", each variable in its own column, and each observation in its own row[5].

[2] A variable is often *represented* as a column in a `data.frame`.

[3] A value is often *represented* as a specific row/column in a `data.frame`.

[4] An observation is often *represented* as a row in a `data.frame`

[5] Note that not all `data.frame`s are *tidy* but a tidy dataset is generally *represented* as a data.frame

In real-life, most data are not tidy. In the process of EDA, it may be beneficial to ask questions that help define a (set of) tidy data to facilitate further analysis. However, EDA and data analysis do not *require* tidy data. Furthermore, one person's definition of tidy data may differ a bit from another person's.

5 Learning about the Diamonds dataset

We are going to be examining the `diamonds` dataset. This dataset is included with the R `ggplot2` package.

5.1 Preliminaries

We will use the `tidyverse` packages throughout this section.

- How do we see if the `tidyverse` package is installed?
- How do you install the `tidyverse` package if needed?
- How do we load the `tidyverse` package before use?

6 Variation

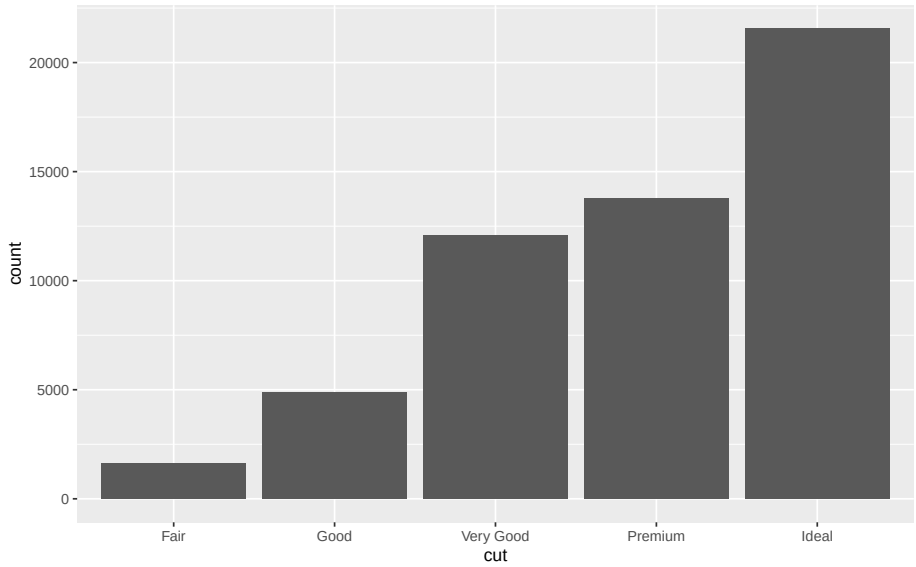
Variation is the tendency of measurements to change from one measurement to the next. You can see variation easily in real life, either when measuring the same quantity between individuals or even when measuring the exact same quantity twice. This is true even if you measure quantities that are considered constant, such as the weight of a coin. Each measurement will include a small amount of error that varies from measurement to measurement. Categorical variables can also vary if you measure across different subjects (e.g. the eye colors of different people), or different times (e.g. the energy levels of an electron at different moments). Every

variable has its own pattern of variation, which can reveal interesting information. Some variation can inform about the system that we are studying (heights of people), while other aspects of variation can highlight problems or unusual features of a variable's values (a person with a height of 67 feet–an "unexpected" value). One of the best ways to understand patterns of variation is to visualise the distribution of the variable's values.

6.1 Visualizing distributions

The choice of how to visualize the distribution of a variable will depend on whether the variable is *categorical* or *continuous* (or quantitative or numeric). A variable is **categorical** if it can only take one of a small set of values. In R, categorical variables are usually saved as factors or character vectors. To examine the distribution of a categorical variable, use a bar chart:

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```

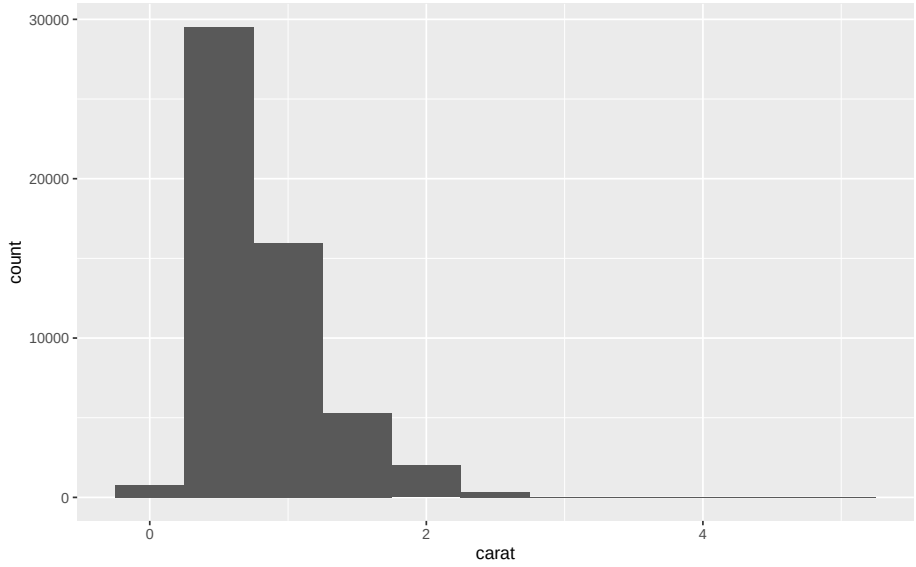


The height of the bars displays how many observations occurred with each x value. You can compute these values manually with `dplyr::count()`:

```
diamonds |>
  count(cut)
## # A tibble: 5 x 2
##   cut           n
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

A variable is **continuous** if it can take any of an infinite set of ordered values. Numbers and date-times are two examples of continuous variables. To examine the distribution of a continuous variable, use a histogram:

```
ggplot(data = diamonds) +
  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



You can compute this by hand by combining `dplyr::count()` and `ggplot2::cut_width()`:

```
diamonds |>
  count(cut_width(carat, 0.5))
## # A tibble: 11 x 2
##    `cut_width(carat, 0.5)`     n
##    <fct>                   <int>
##  1 [-0.25,0.25]              785
##  2 (0.25,0.75]             29498
##  3 (0.75,1.25]             15977
##  4 (1.25,1.75]              5313
##  5 (1.75,2.25]              2002
##  6 (2.25,2.75]               322
##  7 (2.75,3.25]                32
##  8 (3.25,3.75]                 5
##  9 (3.75,4.25]                 4
## 10 (4.25,4.75]                 1
## 11 (4.75,5.25]                 1
```

A histogram divides the x-axis into equally spaced bins and then uses the height of a bar to display the number of observations that fall in each bin. In the graph above, the tallest bar shows that almost 30,000 observations have a `carat` value between 0.25 and 0.75, which are the left and right edges of the bar.

You can set the width of the intervals in a histogram with the `binwidth` argument, which is measured in the units of the `x` variable. You should always explore a variety of binwidths when working with histograms, as different binwidths can reveal different patterns. For example, here is how the graph above looks when we zoom into just the diamonds with a size of less than three carats and choose a diamonds binwidth.

- As an exercise, try changing the binwidth setting in the plot above.

Now that you can visualise variation, what should you look for in your plots? And what type of follow-up questions should you ask? I've put together a list below of the most useful types of information that you will find in your graphs, along with some follow-up questions for each type of information. The key to asking good follow-up questions will be to rely on your curiosity (What do you want to learn more about?) as well as your skepticism (How could this be misleading?).

6.1.1 Typical values

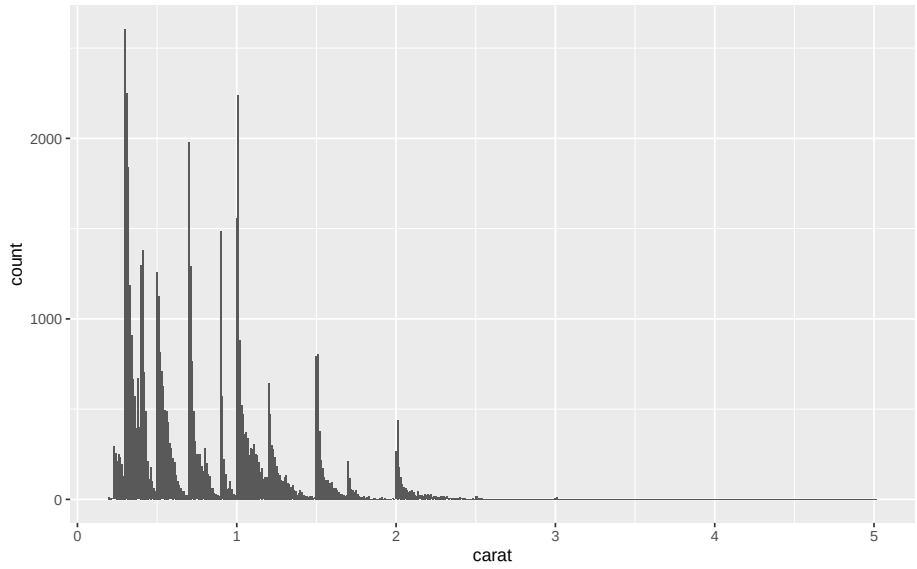
In both bar charts and histograms, tall bars show the common values of a variable, and shorter bars show less-common values. Places that do not have bars reveal values that were not seen in your data. To turn this information into useful questions, look for anything unexpected:

- Which values are the most common? Why?
- Which values are rare? Why? Does that match your expectations?
- Can you see any unusual patterns? What might explain them?

As an example, the histogram below suggests several interesting questions:

- Why are there more diamonds at whole carats and common fractions of carats?
- Why are there more diamonds slightly to the right of each peak than there are slightly to the left of each peak?

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

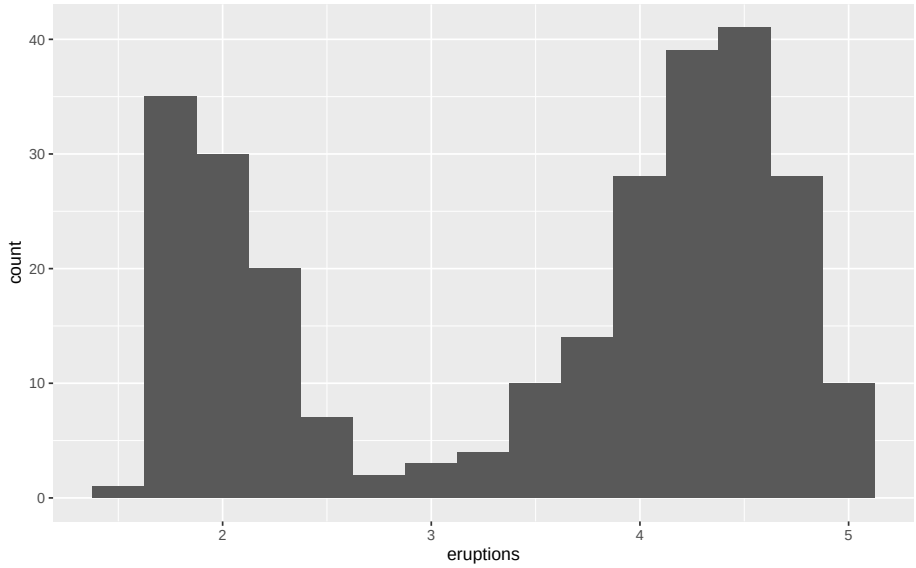


Clusters of similar values suggest that subgroups exist in your data. To understand the subgroups, ask questions like (not for this assignment):

- How are the observations within each cluster similar to each other?
- How are the observations in separate clusters different from each other?
- How can you explain or describe the clusters?
- Why might the appearance of clusters be misleading?

The histogram below shows the length (in minutes) of 272 eruptions of the Old Faithful Geyser in Yellowstone National Park. Eruption times appear to be clustered into two groups: there are short eruptions (of around 2 minutes) and long eruptions (4-5 minutes), but little in between.

```
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_histogram(binwidth = 0.25)
```

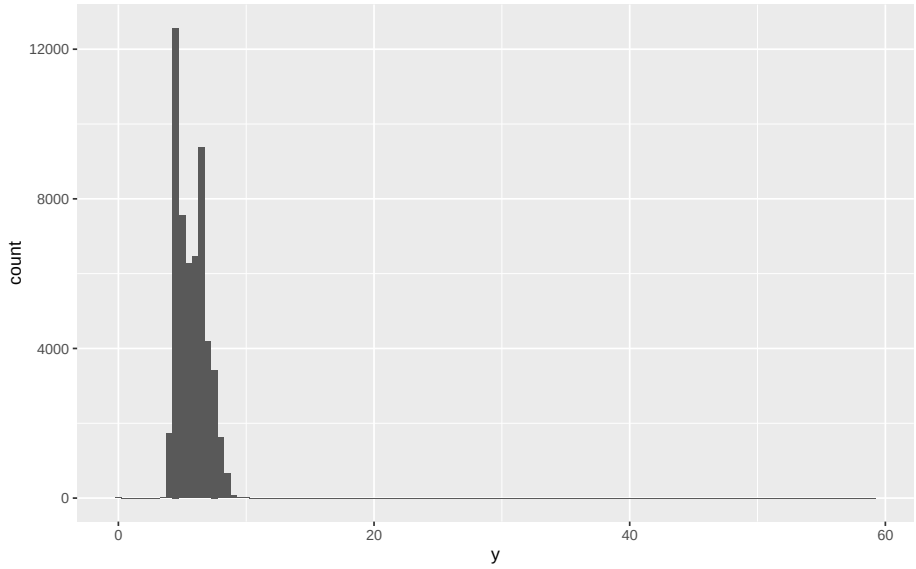


Many of the questions above will prompt you to explore a relationship *between* variables, for example, to see if the values of one variable can explain the behavior of another variable. We'll get to that shortly.

6.1.2 Unusual values

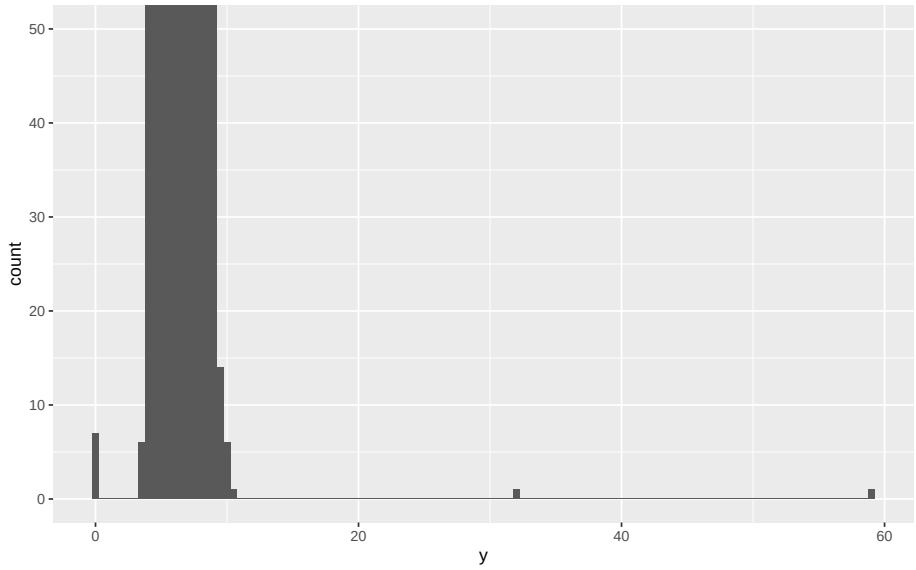
Outliers are observations that are unusual; data points that don't seem to fit the pattern. Sometimes outliers are data entry errors; other times outliers suggest important new science. When you have a lot of data, outliers are sometimes difficult to see in a histogram. For example, take the distribution of the `y` variable from the diamonds dataset. The only evidence of outliers is the unusually wide limits on the x-axis.

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5)
```



There are so many observations in the common bins that the rare bins are so short that you can't see them (although maybe if you stare intently at 0 you'll spot something). To make it easy to see the unusual values, we need to zoom to small values of the y-axis with `coord_cartesian()`:

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```



(`coord_cartesian()` also has an `xlim()` argument for when you need to zoom into the x-axis. ggplot2 also has `xlim()` and `ylim()` functions that work slightly differently: they throw away the data outside the limits.)

This allows us to see that there are three unusual values: 0, ~30, and ~60. We pluck them out with dplyr (if needed, consider reading up on `dplyr` outside this lab).

```
unusual <- diamonds |>
  filter(y < 3 | y > 20) |>
  select(price, x, y, z) |>
  arrange(y)
unusual
## # A tibble: 9 x 4
##   price     x     y     z
##   <int> <dbl> <dbl> <dbl>
## 1  5139  0      0    0
## 2  6381  0      0    0
## 3 12800  0      0    0
## 4 15686  0      0    0
## 5 18034  0      0    0
## 6  2130  0      0    0
## 7  2130  0      0    0
## 8  2075  5.15  31.8  5.12
## 9 12210  8.09  58.9  8.06
```

The `y` variable measures one of the three dimensions of these diamonds, in mm. We know that diamonds can't have a width of 0mm, so these values must be incorrect. We might also suspect that measurements of 32mm and 59mm are implausible: those diamonds are over an inch long, but don't cost hundreds of thousands of dollars!

It's good practice to repeat your analysis with and without the outliers. If they have minimal effect on the results, and you can't figure out why they're there, it's reasonable to replace them with missing values, and move on. However, if they have a substantial effect on your results, you shouldn't drop them without justification. You'll need to figure out what caused them (e.g. a data entry error) and disclose that you removed them in your write-up.

6.1.3 Exercises

1. Read the help page for the diamonds dataset. `help(diamonds)` Keep in mind that a little googling about diamonds (or any dataset) can be helpful to understanding the context for the data.
2. Explore the distribution of each of the quantitatve variables in `diamonds` using a histogram. What do you learn? Think about a diamond and how you might decide which dimension is the length, width, and depth.
3. 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.)
4. How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference?

6.2 Covariation

If variation describes the behavior *within* a variable, covariation describes the behavior *between* variables. **Covariation** is the tendency for the values of two or more variables to vary together in a related way. The best way to spot covariation is to visualize the relationship between two or more variables. How you do that should again depend on the type of variables involved.

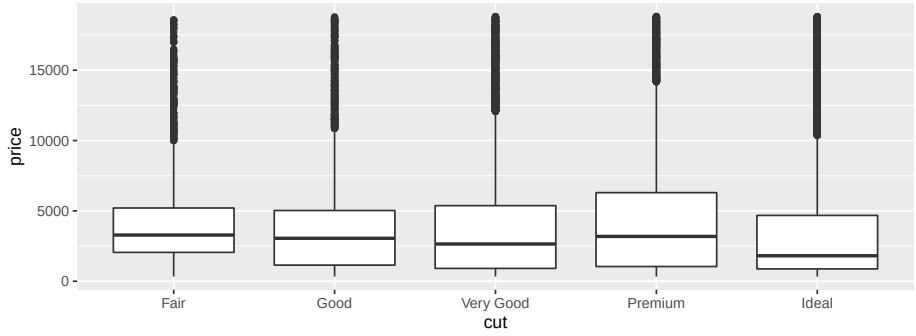
6.2.1 A categorical and continuous variable

It's common to want to explore the distribution of a continuous variable broken down by a categorical variable, as in the previous frequency polygon. A common approacy to display the distribution of a continuous variable broken down by a categorical variable is the boxplot. A **boxplot** is a type of visual shorthand for a distribution of values that is popular among statisticians. Each boxplot consists of:

- A box that stretches from the 25th percentile of the distribution to the 75th percentile, a distance known as the interquartile range (IQR). In the middle of the box is a line that displays the median, i.e. 50th percentile, of the distribution. These three lines give you a sense of the spread of the distribution and whether or not the distribution is symmetric about the median or skewed to one side.
- Visual points that display observations that fall more than 1.5 times the IQR from either edge of the box. These outlying points are unusual so are plotted individually.
- A line (or whisker) that extends from each end of the box and goes to the farthest non-outlier point in the distribution.

Let's take a look at the distribution of price by cut using `geom_boxplot()`:

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

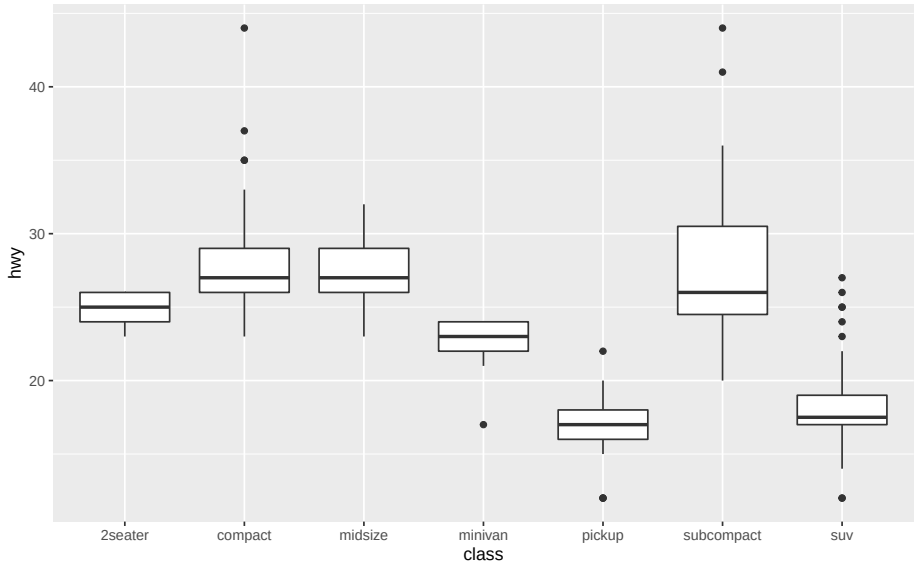


Boxplots are compact so we can more easily compare them (and fit more on one plot). It supports the counterintuitive finding that better quality diamonds are cheaper on average! In the exercises, you'll be challenged to figure out why.

`cut` is an ordered factor: fair is worse than good, which is worse than very good and so on. Many categorical variables don't have such an intrinsic order, so you might want to reorder them to make a more informative display. One way to do that is with the `reorder()` function.

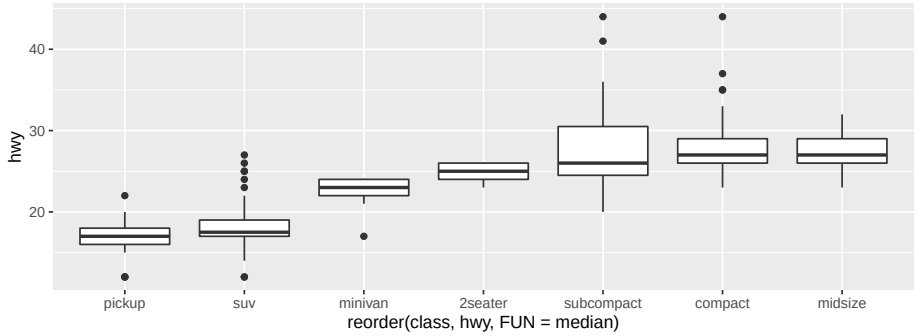
For example, take the `class` variable in the `mpg` dataset. You might be interested to know how highway mileage varies across classes:

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```



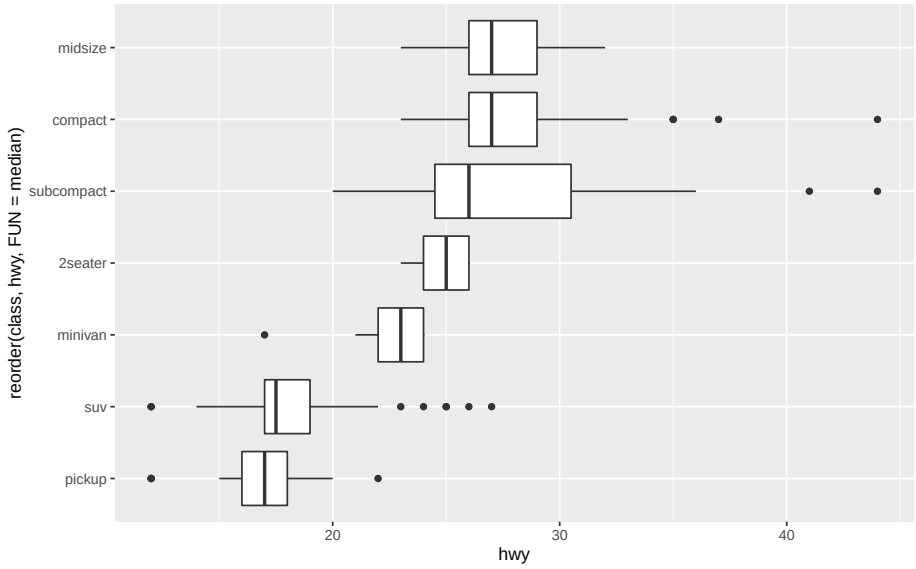
To make the trend easier to see, we can reorder `class` based on the median value of `hwy`:

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy))
```



If you have long variable names, `geom_boxplot()` will work better if you flip it 90°. You can do that with `coord_flip()`.

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy)) +
  coord_flip()
```



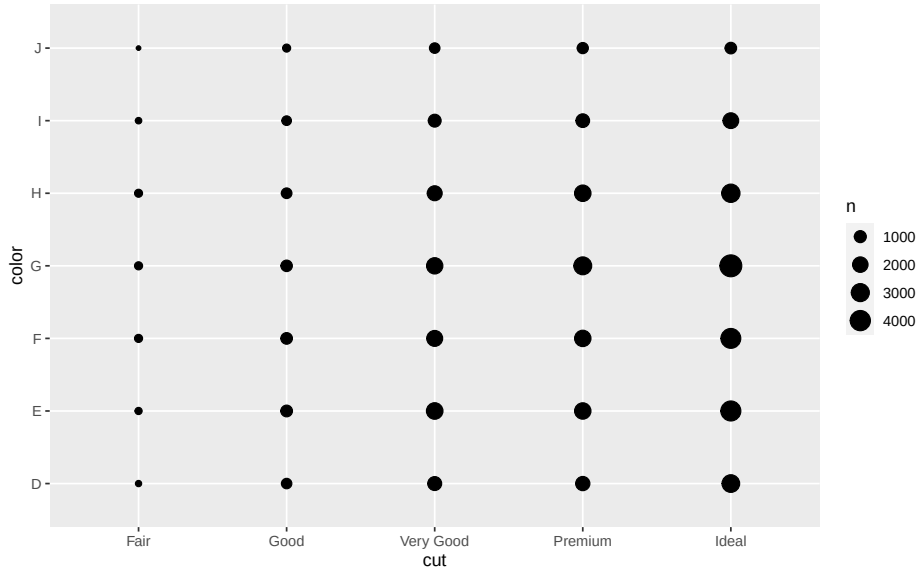
6.2.1.1 Exercises

1. Create a boxplot for each of the categorical variables and price. What categorical variables influence price and how?
2. Exchange x variable and y variable in a vertical boxplot, and create a horizontal boxplot. How does this compare to using `coord_flip()`?
3. 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?
4. Compare `geom_violin()` with a facetted `geom_boxplot()`. What are the pros and cons of each method?

6.2.2 Two categorical variables

To visualise the covariation between categorical variables, you'll need to count the number of observations for each combination. One way to do that is to rely on the built-in `geom_count()`:

```
ggplot(data = diamonds) +
  geom_count(mapping = aes(x = cut, y = color))
```



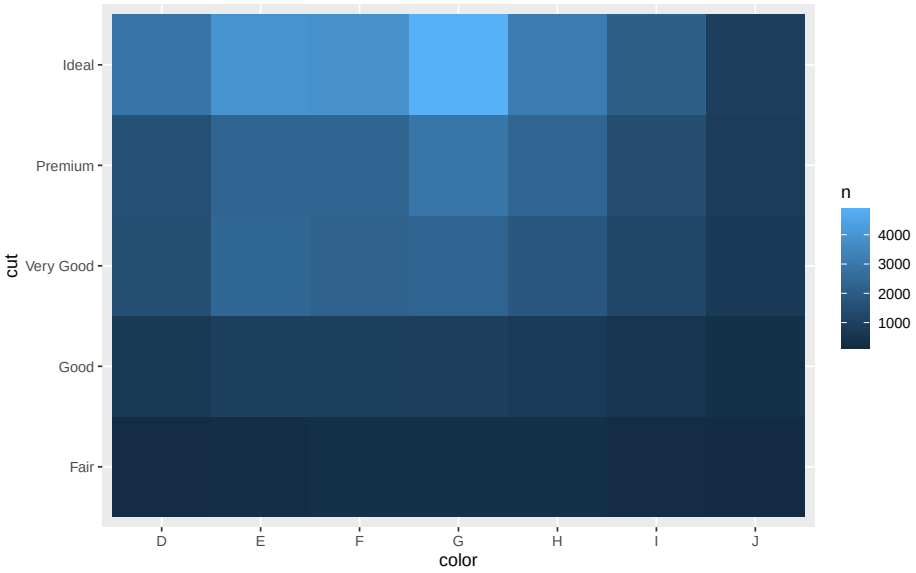
The size of each circle in the plot displays how many observations occurred at each combination of values. Covariation will appear as a strong correlation between specific x values and specific y values.

Another approach is to compute the count with dplyr:

```
diamonds |>
  count(color, cut)
## # A tibble: 35 x 3
##    color cut           n
##    <ord> <ord>     <int>
##  1 D     Fair        163
##  2 D     Good        662
##  3 D     Very Good  1513
##  4 D     Premium    1603
##  5 D     Ideal      2834
##  6 E     Fair        224
##  7 E     Good        933
##  8 E     Very Good  2400
##  9 E     Premium    2337
## 10 E     Ideal      3903
## # ... with 25 more rows
```

Then visualise with `geom_tile()` and the fill aesthetic:

```
diamonds |>
  count(color, cut) |>
  ggplot(mapping = aes(x = color, y = cut)) +
    geom_tile(mapping = aes(fill = n))
```



If the categorical variables are unordered, you might want to use the seriation package to simultaneously reorder the rows and columns in order to more clearly reveal interesting patterns. For larger plots, you might want to try the heatmaply package, which creates interactive plots.

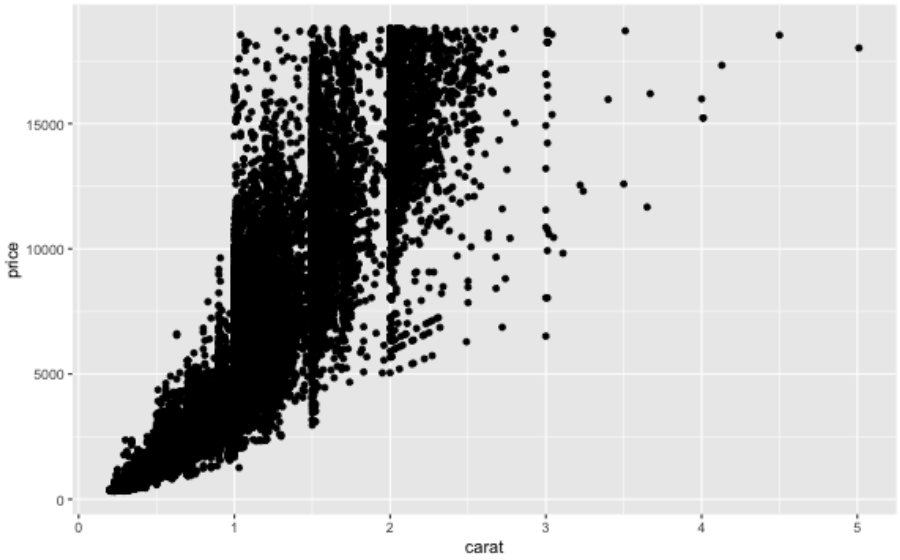
6.2.2.1 Exercises

1. How could you rescale the count dataset above to more clearly show the distribution of cut within colour, or colour within cut?
2. Use `geom_tile()` together with dplyr to explore how average flight delays vary by destination and month of year. What makes the plot difficult to read? How could you improve it?
3. Why is it slightly better to use `aes(x = color, y = cut)` rather than `aes(x = cut, y = color)` in the example above?

6.2.3 Two continuous or quantitative variables

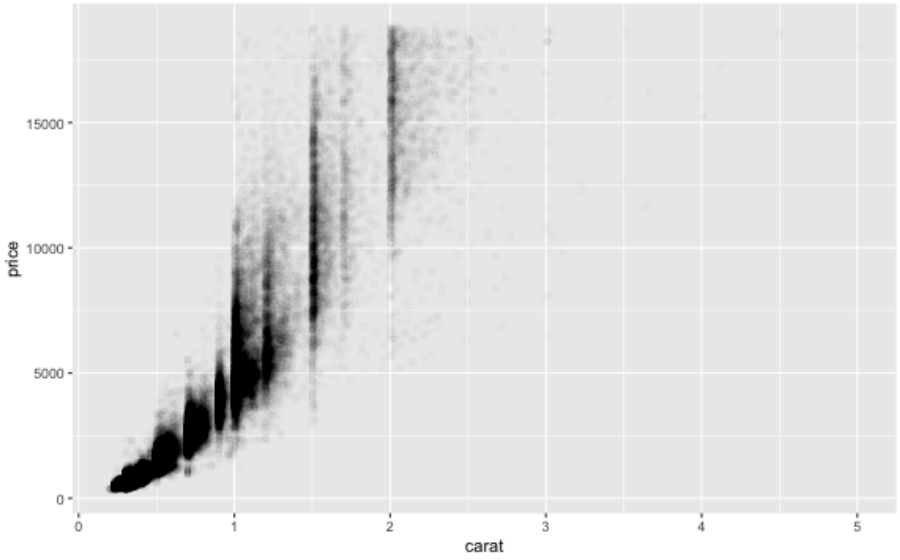
You've already seen one great way to visualise the covariation between two continuous variables: draw a scatterplot with `geom_point()`. You can see covariation as a pattern in the points. For example, you can see an exponential relationship between the carat size and price of a diamond.

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```



Scatterplots become less useful as the size of your dataset grows, because points begin to overplot, and pile up into areas of uniform black (as above). You've already seen one way to fix the problem: using the `alpha` aesthetic to add transparency.

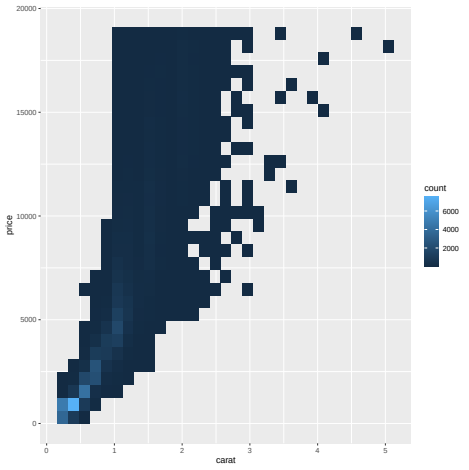
```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price), alpha = 1 / 100)
```



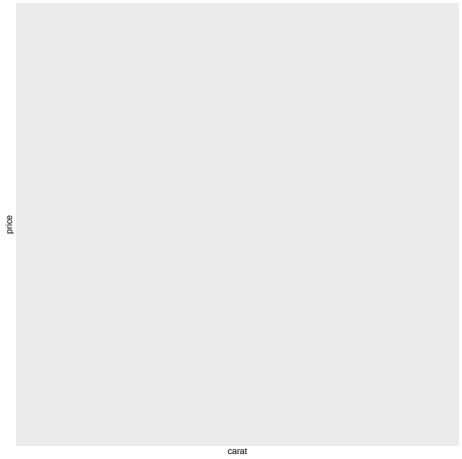
But using transparency can be challenging for very large datasets. Another solution is to use bin. Previously you used `geom_histogram()` and `geom_freqpoly()` to bin in one dimension. Now you'll learn how to use `geom_bin2d()` and `geom_hex()` to bin in two dimensions.

`geom_bin2d()` and `geom_hex()` divide the coordinate plane into 2d bins and then use a fill color to display how many points fall into each bin. `geom_bin2d()` creates rectangular bins. `geom_hex()` creates hexagonal bins. You will need to install the hexbin package to use `geom_hex()`.

```
ggplot(data = diamonds) +
  geom_bin2d(mapping = aes(x = carat, y = price))
```

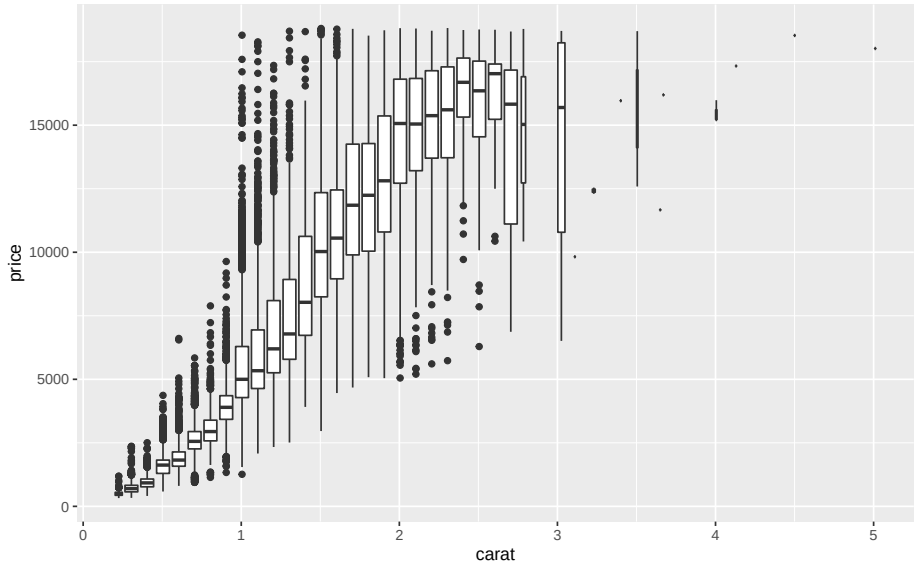



```
# install.packages("hexbin")
ggplot(data = diamonds) +
  geom_hex(mapping = aes(x = carat, y = price))
## Warning: Computation failed in `stat_binhex()`:
```



Another option is to bin one continuous variable so it acts like a categorical variable. Then you can use one of the techniques for visualising the combination of a categorical and a continuous variable that you learned about. For example, you could bin `carat` and then for each group, display a boxplot:

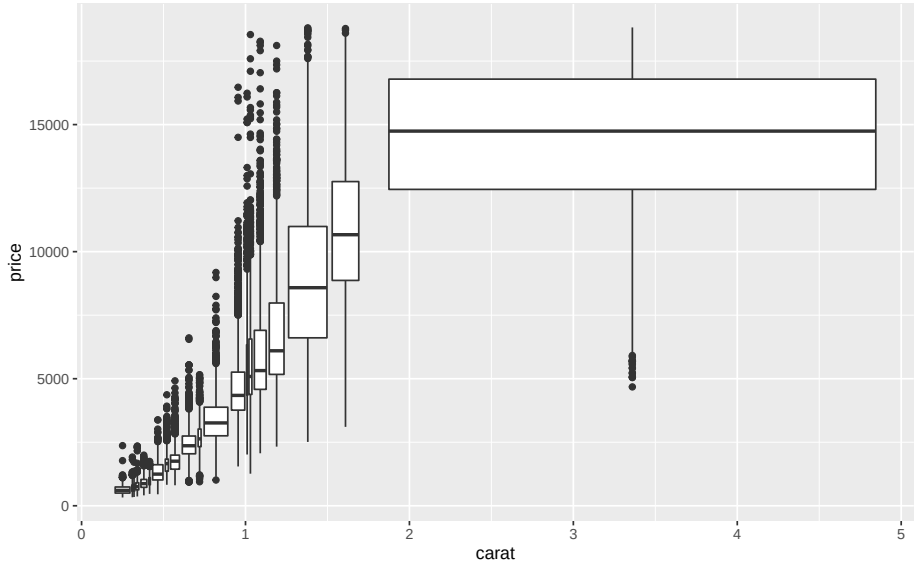
```
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```



`cut_width(x, width)`, as used above, divides `x` into bins of width `width`. By default, boxplots look roughly the same (apart from number of outliers) regardless of how many observations there are, so it's difficult to tell that each boxplot summarises a different number of points. One way to show that is to make the width of the boxplot proportional to the number of points with `varwidth = TRUE`.

Another approach is to display approximately the same number of points in each bin. That's the job of `cut_number()`:

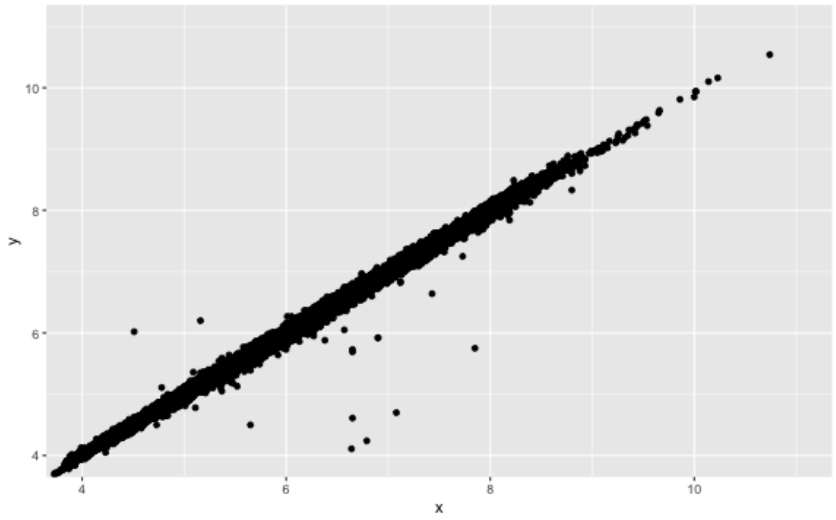
```
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_number(carat, 20)))
```



6.2.3.1 Exercises

1. Instead of summarising the conditional distribution with a boxplot, you could use a frequency polygon. What do you need to consider when using `cut_width()` vs `cut_number()`? How does that impact a visualisation of the 2d distribution of `carat` and `price`?
2. Visualise the distribution of carat, partitioned by price.
3. How does the price distribution of very large diamonds compare to small diamonds? Is it as you expect, or does it surprise you?
4. Combine two of the techniques you've learned to visualise the combined distribution of cut, carat, and price.
5. Two dimensional plots reveal outliers that are not visible in one dimensional plots. For example, some points in the plot below have an unusual combination of `x` and `y` values, which makes the points outliers even though their `x` and `y` values appear normal when examined separately.

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = x, y = y)) +
  coord_cartesian(xlim = c(4, 11), ylim = c(4, 11))
```



Why is a scatterplot a better display than a binned plot for this case?