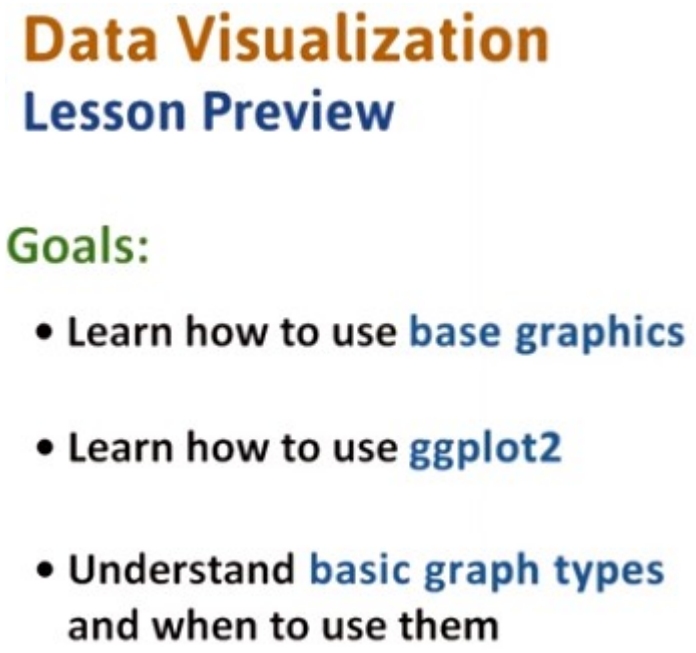
The video time for this lesson is 60 minutes.

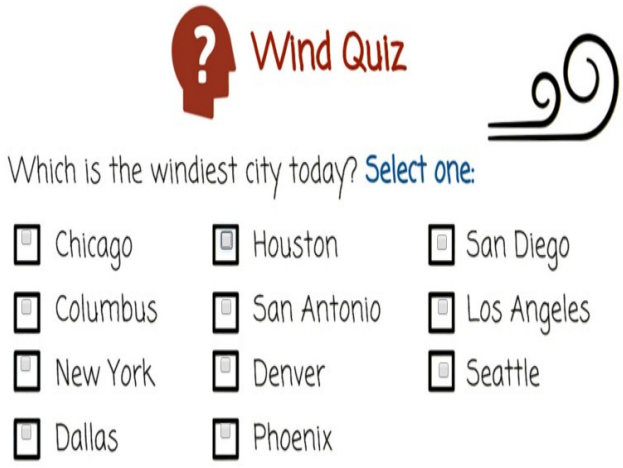
The recommended reading for this lesson is

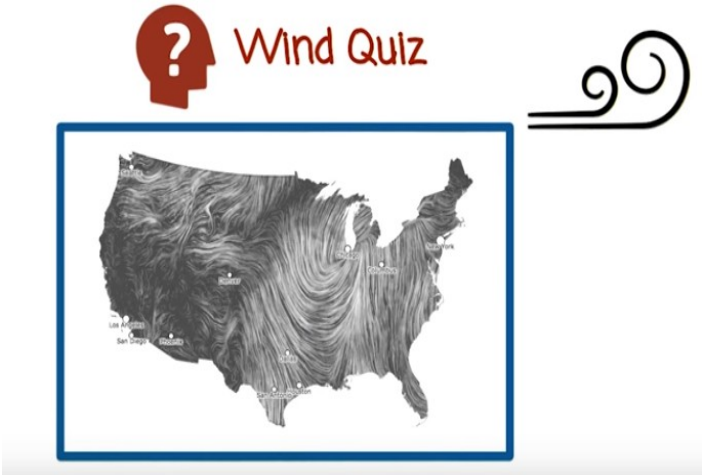
Data Visualization
https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cse6242/recommended+reading/graphics.pdf

ggplot2
https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cse6242/recommended+reading/ggplot2-book.pdf



1. In this lesson we focus on the data visualization. We'll learn how to use base graphics and ggplot2, which are two R packages for data visualization. We will also go over a variety of graph types and see how they can be used to get insights from the data.



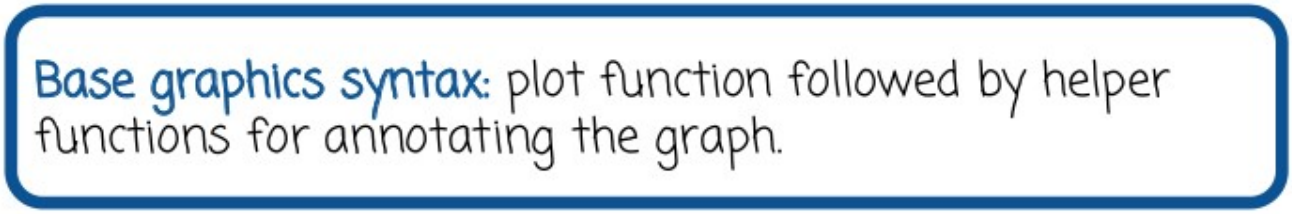


以上 quiz 無答案.

Wind Map
http://hint.fm/wind/

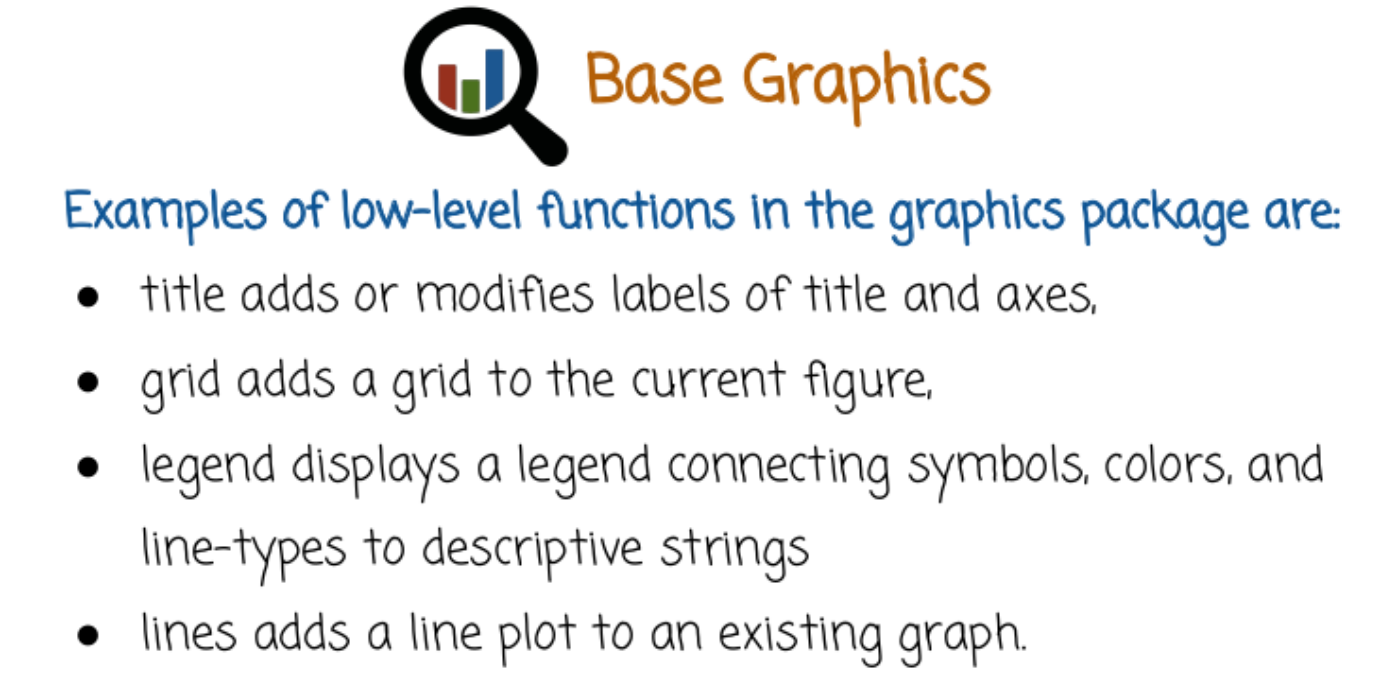
2. Before we start our discussion on data visualization, I have a quiz I want you to try. Using any means at hand, determine which of the listed cities is the windiest today.

3. There are a number of methods you could have used to answer this question. You might have searched the Internet, gathering weather information on each city maybe you just guessed. Actually, I don't know the windiest city today because I don't know what day you're viewing this video, but I do know how you can find out quickly and easily. Click on the link in the instructor's notes and see the answer. This website shows wind data across the US in real time, with this visualization it's easy to get information from the data. In fact, this visualization is very useful. You can find out not only which city is the windiest, but which regions, the pattern of wind, and additional information.

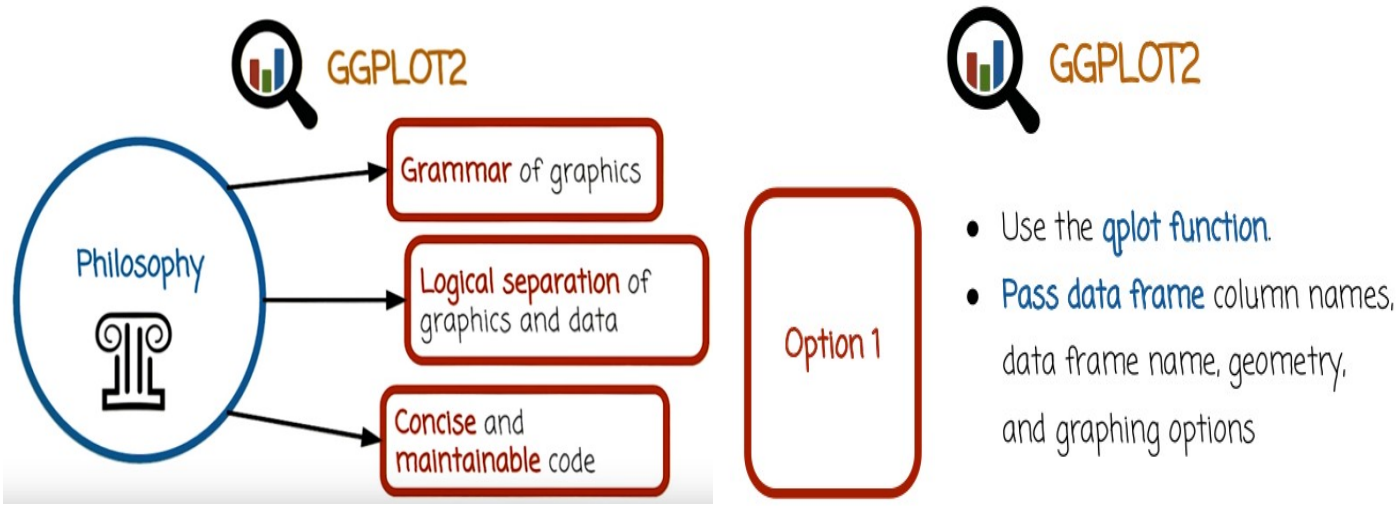



```
plot(x = dataframe$col_1, y = dataframe$col_2)
title(main = "figure title")  # add title
```

4. We will see several ways of visualizing data using r, we will explore specifically two packages. The first one is called Base Graphics, the second one is called gg plot two. Base graphics is a package that is included in R by default, and is already loaded into scope. It features a collection of simple functions that can be used to graph data. Base graphic's syntax usually includes a plot function, followed by helper functions for annotating the graph. We see here an example. The first line shows the plot function. We pass it two arguments, which are columns of a dataframe and assign them to the variables x and y. x is going to correspond to the x-axis, and y's going to correspond to the y-axis. The second function is a helper function. In this case, it adds a title to the figure.

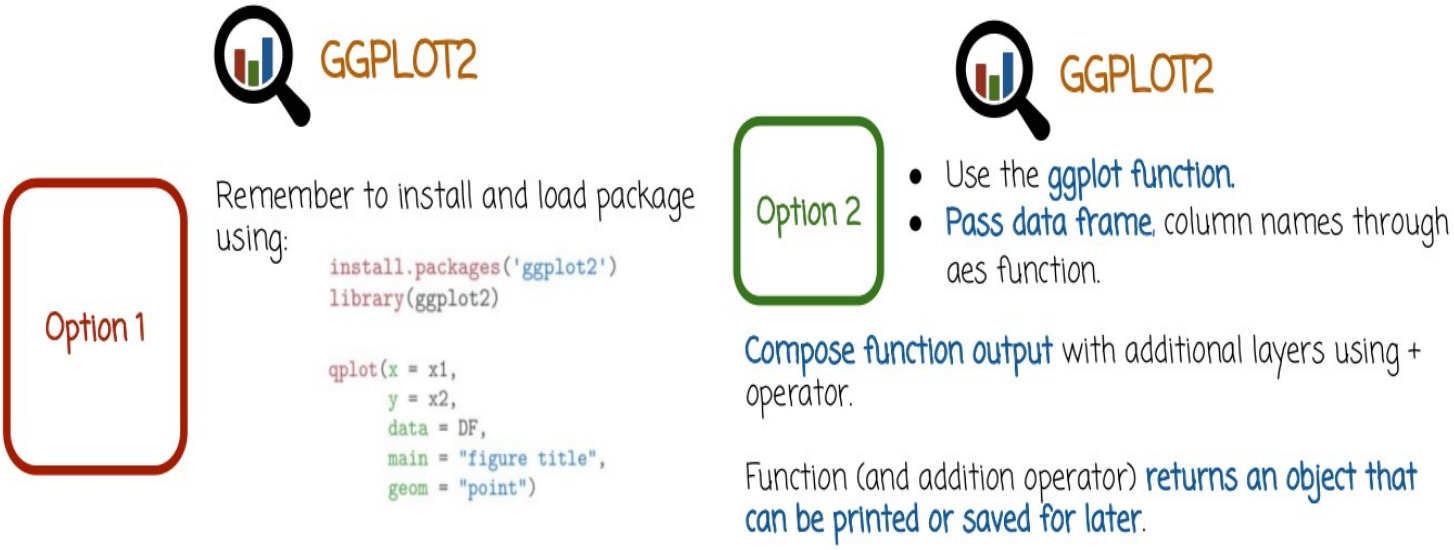


Base graphics has a collection of low-level functions, or helper functions. Here are a few of them. Title function adds or modifies labels of titles and axes. Grid function adds a grid to the current figure. Legend displays a legend connecting symbols, colors, and line-types to descriptive strings. Lines function adds a line plot to an existing graph.



5. GGPLOT2 is a very popular package for visualizing data in R. It has a different philosophy from base graphics. It implements the grammar of graphics, which is a way to structure and define different graphics elements using a basic vocabulary. It enables a logical separation of graphics and data. You can have some variables, keep the data with all the information about the data, including the axis labels, the typos, etc. And then you can execute the graphing part, using a simple function. This is useful because later, you may want to save the data and graphic again later. You can just do that easily by saving and loading the variables. A different way to think of it is that more effort is spend on creating the data frame and less effort to spend on graphing it. The effort is shifted away from the graphing part

into creating an appropriate data frame. As a result of the logical separation of graphics of data and of the grammar of graphics, visualization code that uses GGPLOT2 package is usually concise and maintainable. There are two basic ways to use GGPLOT2. The first option is to use the qplot function. Qplot function is a relatively simple way of using GGPLOT2. You pass to qplot the data frame that you want graphed. The column names, the geometry of the graph and additional graphing options. We'll see an example in a second.



我電腦上已裝 ggplot2. 是通過 sudo apt-get install r-cran-ggplot2 裝的.

But remember, the GGPLOT2 is a separate package. And so you first need to install it. And then you need to load it into a scope using the library function. We see here the install packages function that need to be executed one time in order to install the package from the Internet. The second line needs to be executed every time you want to use GGPLOT2 functions in order to bring these functions into scope. Here is the example. We call the qplot function with five parameters. The third parameter, data, will contain the dataframe. The first two parameters are going to hold the vectors to be graphed. In this case, they are dataframe columns, x1 assigned to the variable x and x2 assigned to the variable y. Similar to base graphics, qplot will then map the variable x to graphics that should be tied to the X-axis. And the variable y to graphics that should be tied to the Y-axis the last two parameters are going to be used to annotate the graph, the 4th parameter, main, is going to hold the title of the figure. The 5th parameter, geom, is going to hold the geometry of the point. In this case, the point value corresponds to a scattered plot using point geometry. Notice that since we actually assigned the values to a specific parameter name, we can use whichever order we want. We don't have to stick to the order that you see here. For example, we can assign data equals DF in the first position. And main = figure title in the second position and then x = x1, y = x2 and so on. The second option in GGPLOT2 is to use the GGPLOT function. GGPLOT function is slightly more complicated than qplot function. When into pass date frame and column names through an AES function, we'll see an example of that shortly. In GGPLOT2 whether you use GGPLOT function or qplot function you can compose the outputs of these functions with addition layers using the plus operator. In other words GGPLOT and qplot function return an object. If you have multiple such objects you can compose them using the plus operator to create a figure with multiple objects in it. We'll see several examples of this later on. Another nice property is that the object returned by GGPLOT or qplot can be saved. Later we can load it and print it or modify it. We don't have to execute the same code again for creating that object every time we want to recreate the graph.

The next portion of this lesson uses a dataset called 'faithful'.

To use this data execute the following steps:

install.packages(ggplot2) ... You only need to do this once, so if you have already done it, you do not need to do it again. ← 由前面圖知, 應該是 install.packages('ggplot2')
library(ggplot2)
data(faithful)

6. We're going to look at different datasets and plot types. The first plot type is called strip plot. Strip plot is a very simple plot that displays a vector of values with the x-axis corresponding to the index of the value in the vector and the y value corresponding to the value of that component. We're going to demonstrate it using the Old Faithful dataset, specifically named faithful data frame in R, that holds the eruption time and waiting time to the next eruption, both in minutes, of the Old Faithful geyser in Yellowstone National Park, Wyoming, US. Notice here, we call the function names so that we can see the names of the two columns. This is a data frame with two columns, eruptions and waiting. Eruptions, again, is the eruption time, and waiting is the waiting time to the next eruption.

7. Will demonstrate how to create strip plots with base graphics in this strip plots quiz. Remember that strip plots graph one-dimensional numeric data as points in a two-dimensional space with one coordinate corresponding to the index of the data point and the other coordinate corresponding to its value. In this quiz fill in the blanks to create a strip plot given the following information, specifically fill in these two blanks to create a strip plot that corresponds to the descriptions that you see here of the xlabel and of the ylabel.

Strip Plots Quiz

Strip plots graph one-dimensional numeric data as points in a two-dimensional space, with one coordinate corresponding to the index of the data point, and the other coordinate corresponding to its value. **Fill in the blanks to create a strip plot** given the following information:

```
names(faithful)
## [1] "eruptions" "waiting"
```

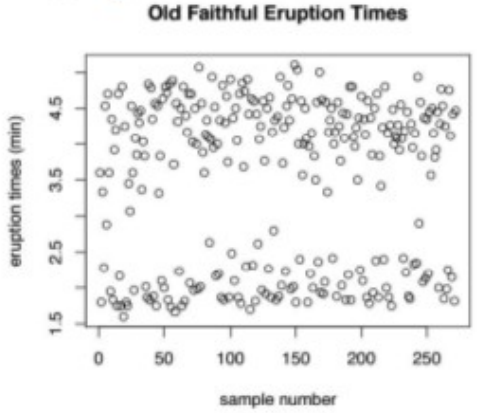
```
plot( faithful $ eruptions , xlab = "sample number", ylab
= "eruption times (min)", main = "Old Faithful Eruption
Times")
```

8. In the first blank, we write the name of the data frame. Notice that there is a dollar symbol afterwards, which is a hint that we're going to have to use a data frame in the first blank, and then refer to one of its columns in the second blank. In the second blank, we write eruptions. The reason we write eruptions here is, because the y label variable has eruption times which gives us a hint that we want to plot eruption times on the y axis.

9. The code at the top here graphs the strip plot of the Old Faithful Eruptions time taken from the quiz before. At the bottom left corner here, you see the graphics that this code produces. The x-axis is the sample number or the component of the value within the vector of values of eruption times. There is a column in the data frame faithful. The y-axis is the eruption time. What can we conclude from the plot? Select the true statements. Old Faithful has two typical eruption times. The order in which the data frame rows are stored is related to the eruption variable.

Strip Plots Analysis Quiz

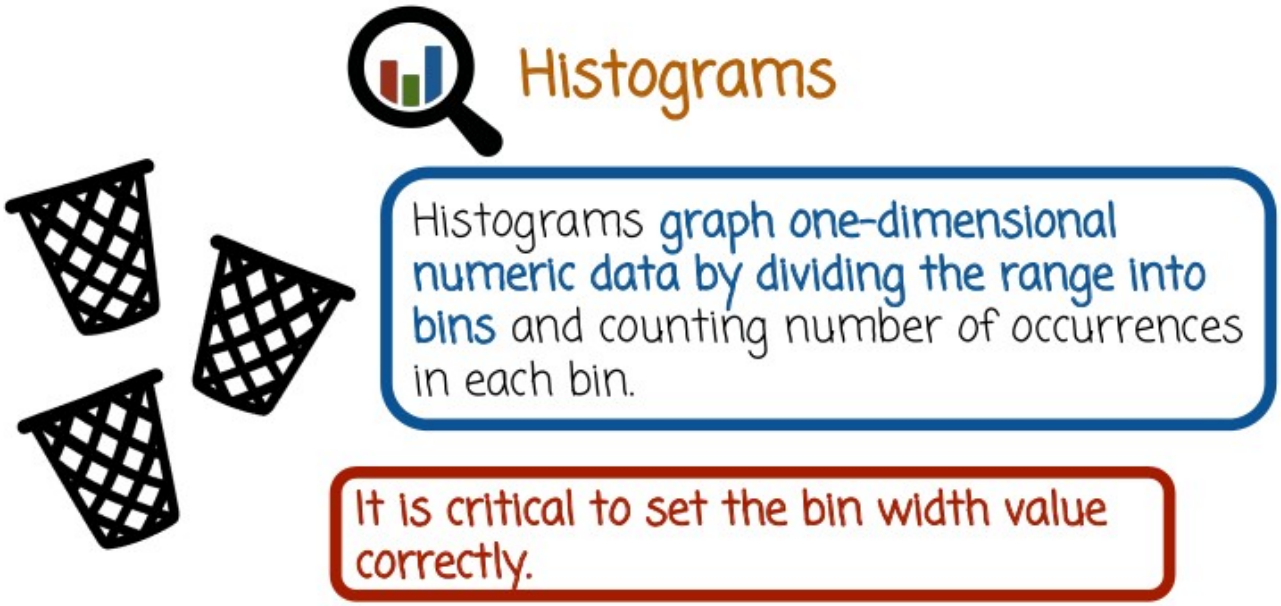
```
plot(faithful$eruptions, xlab = "sample number", ylab =
"eruption times (min)", main = "Old Faithful Eruption
Times")
```



What can we conclude from the plot? Select the true statements.

- [x] Old Faithful has two typical eruption times.
- [ ] The order in which the data frame rows are stored is related to the eruption variable.

10. The first statement is true, Old Faithful has two typical eruption times. One long eruption time around 4.5 minutes and a short eruption time around 1.5 minutes. The strip plots very nicely displays these two specific eruption times. And we see that there is very clear cluster around the longer time and a clear cluster around the shorter time, with fewer points falling in the middle. The second statement apparently is false, although we can not prove it from the data. The reason it's apparently false is that the components of the vector do not seem to relate in any clear way to the eruption time. Eruption time alternating between short and long, and short and long, and long and long, and short and short. There's no clear relationship that shows the first samples corresponding to the short eruption time and the later samples corresponding to the longer eruption time.

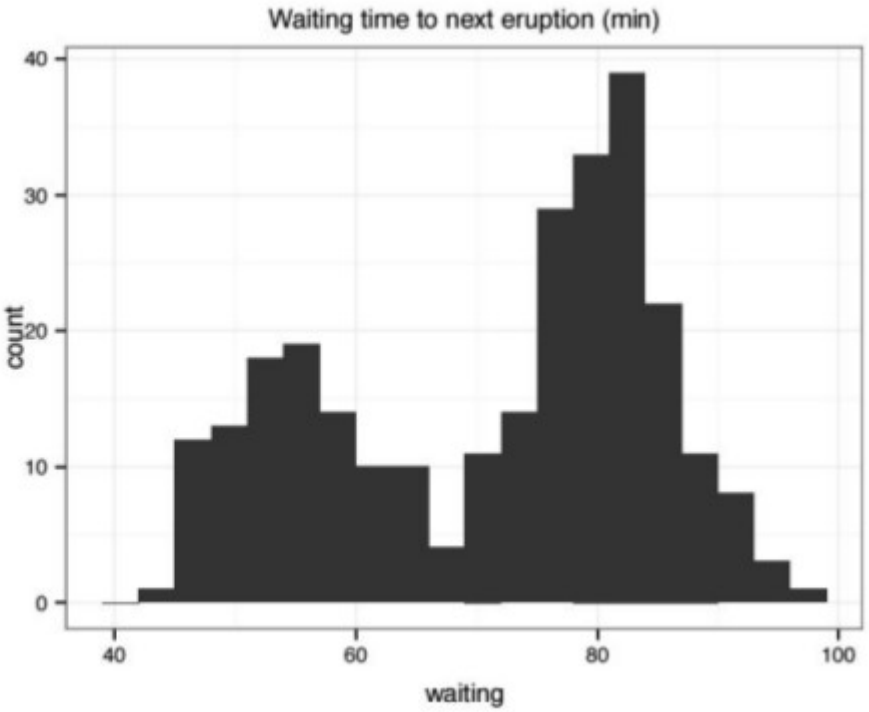


11. Histograms graph one-dimensional data by dividing the range into bins and counting the number of occurrences in each bin. It is critical to set the bin width value correctly. If you set the bin width value too small or too big, the graph will not be very useful.

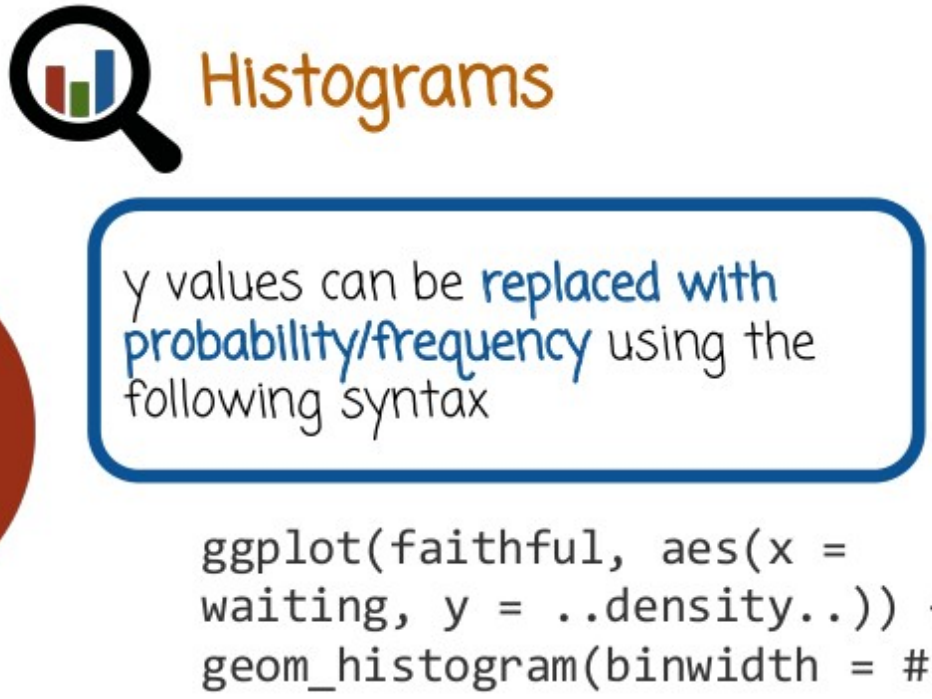

```
qplot(x = waiting,
      data = faithful,
      binwidth = 3,
      main = "Waiting time to next eruption (min)")
ggplot(faithful ,aes(x = waiting)) +
  geom_histogram(binwidth = 1)
```

This is an example in R of creating a histogram of the waiting column in the faithful data frame. We do that two times, the first time with the qplot function and the second time with the ggplot function. Both of these functions are inside the ggplot2 package. In the first example we called qplot with four arguments. The first one, waiting, is the name of the data frame column that we wish to bind to the x variable that will be tied to the x axis. The second line in the name of the data frame, faithful. That will be tied to the data variable. The third is the binwidth and the fourth is the title of the graph. As before, we can use which ever order we want to specify these variables. The reason we can change the order is that we tie specific variables to variables within the function that are going to be tied. As before, we can use whichever order we want when we pass these arguments to the function. The reason we can do that is that we include the variable names, such as x, data, binwidth, and main rather

than just pass to the function waiting, faithful or 3 in the title string. How does qplot know to display histogram based on these commands? Well, a histogram in qplot is the default graphics if you pass to qplot just one vector of numeric values. In the second example, we used ggplot function which is a little bit more complex. We passed the data frame name and then we passed the corresponding column that we want to graph. We assign it to the value x and we pass it through the aes function. Ggplot function does not return a graph just yet because we haven't told it which geometry we want to use. So we use the plus operator to compose the data return from ggplot function with a geometry object. Specifically a histogram object with binwidth of, in this case, 1. Although we could have used 3 as we did in the Q plus function. Both of these functions will display similar histograms, one with a binwidth of three, one with a binwidth of one. In this case we have a title here, we could also add the title.

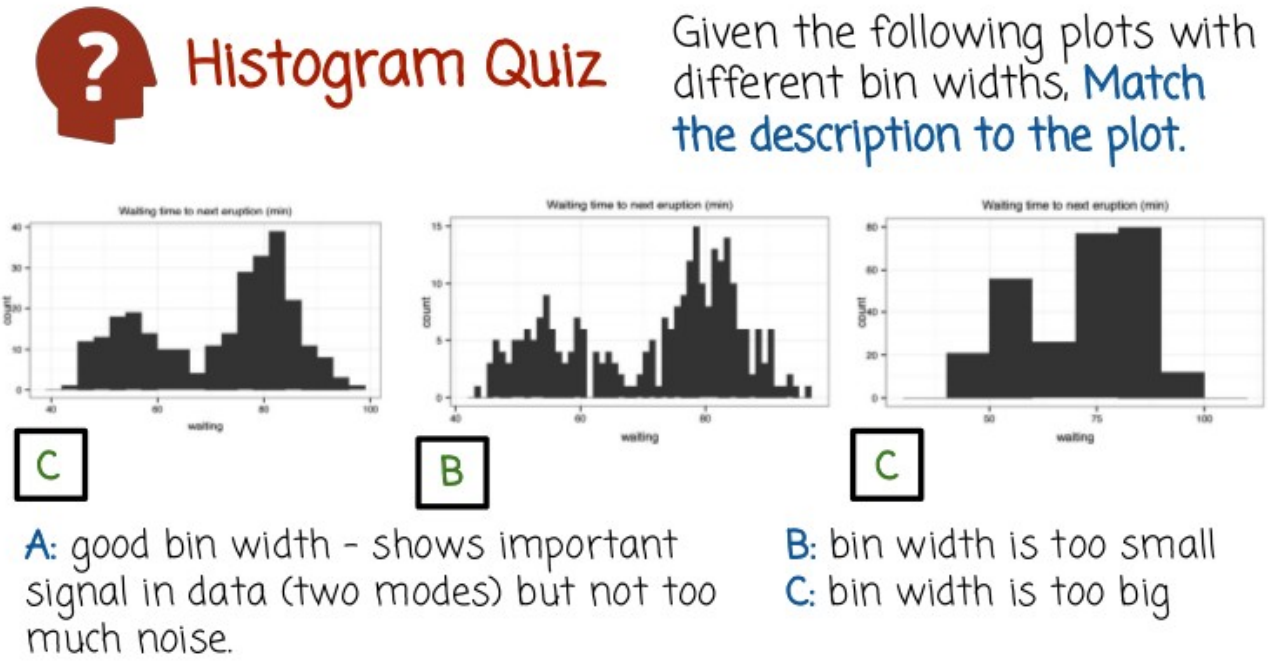


Here is the output of the first function in the previous slide, the histogram of waiting time to next eruption. We see clearly from these graphics that there are two typical waiting time, one waiting time around 50 minutes and one around 80 minutes.

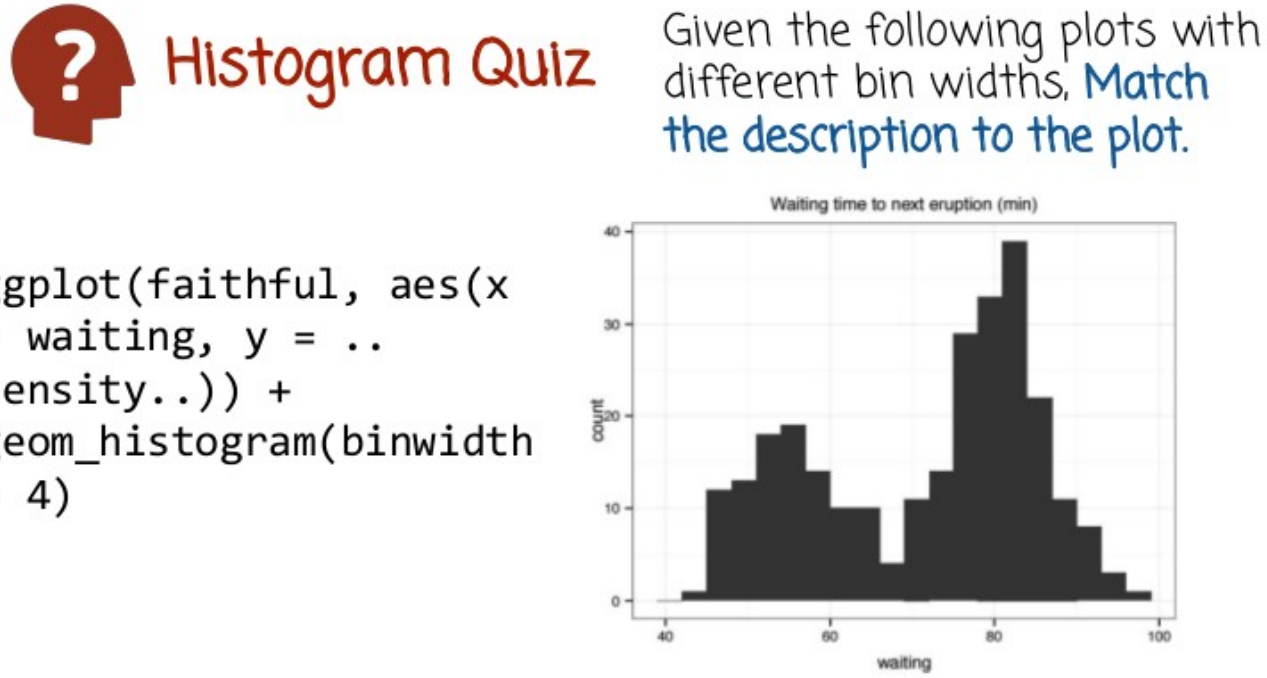


The y-axis in histogram is usually the count, or how many times a measurement is counted is this corresponding bin. But in some cases it may be useful to have the y values show the probability or the frequency. This converts the histogram into a probability mass function and the area under that function is going to be one (這就是我研究中畫的 normalized histogram 啊). Or in other words, the height of the bin times the width of the bin is the probability of getting values within that range. Let's see how we do that with ggplot too. We see here the code similar to the previous code with one difference. We assigned the value dot dot density dot dot to the variable y. This tells GGPlot to show the y axis not with count values, but rather with density values.

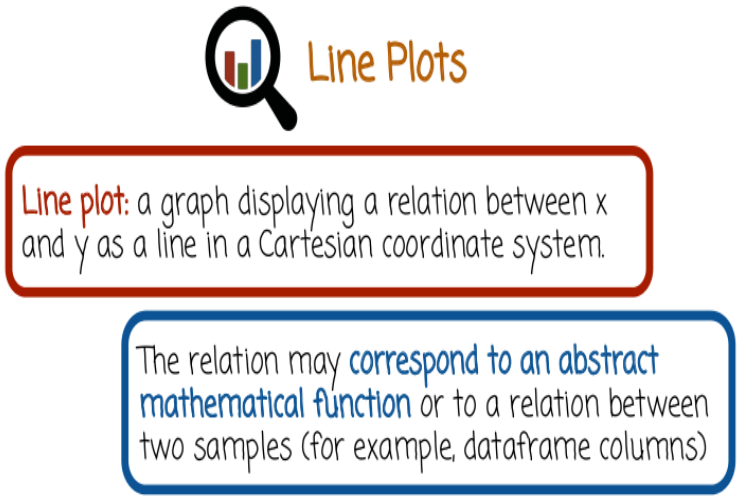
12. Selecting the best thing to use when graphing a specific data set with a histogram is difficult. You may need to try a few different values, and then select the value or values that provide the most insight about the data. In this quiz, given the following plots with different bin widths, match the description to the plot. The three descriptions are A good bin width, shows important signal in data, two modes, but not too much noise. B, bin width is too small. C, bin width is too large.

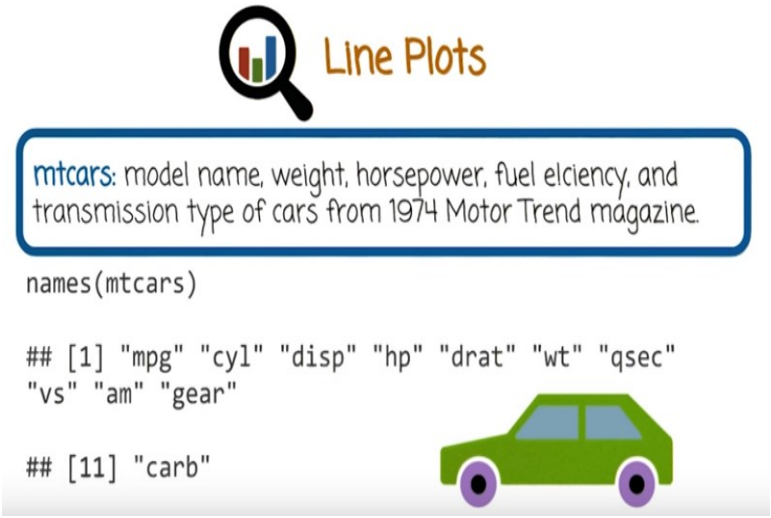


13. The plot on the left shows a very nice histogram. It's very clear, two modes. And at the same time, we don't see a lot of noise in the data that does not correspond to a specific pattern. The second plot shows a case where the bin width is too small. As a result, we see a lot of noise. We may be able to detect that there are two modes, but we may not. For example, in this case, it seems as if the second mode is actually divided into two other modes with a clear value in-between. This is very likely the result of noise rather than a specific pattern. And the third plot, we see that the bin width is too big. It's difficult to see the shape of the graph, the shape of the modes, their heights and whether or not we are missing some important detail. Among these three graphs, the first one provides the best balance between capturing important patterns and not capturing too much noise.



Here is the code that can be used to generate that graph.





要先 data(mtcars)

14. A line plot is a plot that shows y-values as a function of x-values. This is one of the most common plots, and I'm sure you've seen it many times before. We're going to explore line plots using a data set called mtcars. The data frame mtcars has the following columns, model name, weight, horsepower, fuel efficiency, and transmission type. These columns describe cars from the 1974 Motor Trend magazine. Here we call the function names in order to display the column names. Specifically, a line plot is a graph that displays a relation between x and y as a combination of lines in a Cartesian coordinate system. The relation may correspond to the abstract mathematical function or to a relation between two samples.


```
x = seq(-2, 2, length.out = 30)
y = x^2
qplot(x, y, geom = "line")  # line plot
qplot(x, y, geom = c("point", "line"))  # line and point plot
dataframe = data.frame(x = x, y = y)
ggplot(dataframe, aes(x = x, y = y)) +
  geom_line() + geom_point()  # same as above but with ggplot
```

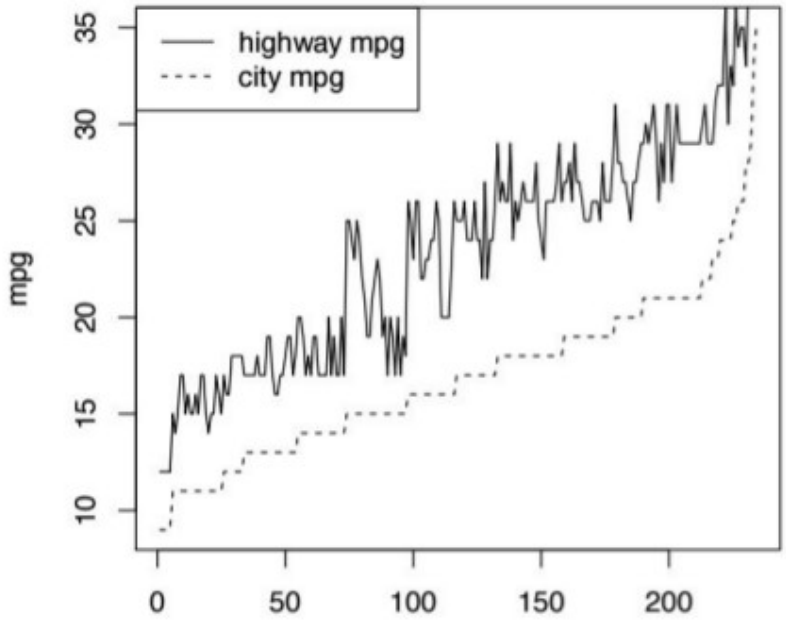
Let's see a few examples using ggplot2 package. First we create a vector of values x. This vector contains values between -2 and 2, and we going to have 30 values that are equally spaced within that range. The second vector that will hold the y-values is going to be the square of the x-values. The first function here qplot(x, y, geom = "line") will create a line plot and plug the y-values against the x-values and will connect the corresponding points with lines. We're not necessarily going to see these lines, because they're going to be very small. Instead, what we're going to see is a quadratic curve. Notice here that we need to pass to qplot, the geom = "line" argument. In qplot, we can also create multiple geometries in a single plot by concatenating the geometries and assigning them to the value geom. In this case, we concatenate the geom "point" and "line," which will produce a line plot with the points shown as specific circles. The last two rows show a way of doing the same thing using ggplot function. First, we need to create a data frame that hold data. Then we pass the data frame and the column names through the aes function. And then we add those additional layers using the + operator, line geometry, which is the object returned from geom_line() function, and point geometry, which is the object returned from geom_point() function.


```
S = sort.int(mpg$cty, index.return = T)
#  x: city mpg
#  ix: indices of sorted values of city mpg
plot(S$x,  #  plot sorted city mpg values with a line plot
     type = "l",
     lty = 2,
     xlab = "sample number (sorted by city mpg)",
     ylab = "mpg")
lines(mpg$hwy[S$ix] ,lty = 1)  # add dashed line of hwy mpg
legend("topleft", c("highway mpg", "city mpg"),
  lty = c(1, 2))
```

mpg 是另一個 dataframe, 是跟 mtcars 不同的一個 dataframe (後面已確認)

This is a slightly more complicated example. It shows the line plot using the base graphics package. What we want to show are two different line plots. The first line plot will need to be the highway miles per gallon. The second line plot is going to the city miles per gallon. These will be the y-values. The x-values will correspond to different cars in the mtcars data frame. In this plot function, we're going to create the sorted city mpg line plot. The y-axis will correspond to the city mpg, and the x-axis will correspond to different car models. The first argument is the data. The second one is going to tell plot function to show a line plot. The third one is going to tell the plot function what is going to be the style of the line, in this case, whether it's going to be solid or dashed. The last two arguments are going to be description of the xlab and the ylab. After we created the first line plot, we add another line plot, in this case of highway mpg. Then we add the legend that will show the viewer which graph corresponds to

which variable.



Let's see what graph this piece of code generates. We see here the two line plots. The top line plot is the highway mpg, and the bottom line plot is city mpg. The x-axis corresponds to different car models and is sorted by the bottom line, city mpg. Interestingly, as the city mpg increases, highway mpg tend to increase as well. But notice that the difference between them actually becomes larger for large values of mpg. In the low range, the difference is not that significant between the city and the highway mpg.

Denoting n values by $x^{(1)}, \ldots, x^{(n)}$, the smoothed histogram is the following function $f_h: \mathbb{R} \to \mathbb{R}_+$

$$f_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x^{(i)})$$

意思是說：
p=0 時 , Kh (p) = max,
p 增加時 , Kh (p) 減小 .

這不是 Kh 乘以 (x – x^(i)) 的意思 ,
而是 (x – x^(i)) 是函數 Kh() 的 argument.

Where the kernel function $K_h: \mathbb{R} \to \mathbb{R}$ typically achieves it maximum at 0, and decreases as $|x-x^{(i)}|$ increases. We also assume that the kernel function integrates to one $\int K_h(x)dx=1$ and satisfies the relation

$$K_h(r) = h^{-1} K_1(r/h).$$

We refer to K_1 as the base form of the kernel and denote it as K.

15. An alternative to the histogram is the smoothed histogram. The smoothed histogram is similar to a

histogram, but instead of showing bins of specific width and height, it shows a curve. Let's see first the mathematical definition. And then we'll see how to create such a graphing practice. We assume that our measurements are denoted by x (1) to x (n). In this case, the smoothed histogram is a function that maps a specific value on the real line to a non-negative value denoted here using the symbol R+. That function is defined as follows. 1/n, sum of i goes from 1 to n of k sub h, (x- xi). But remember xi are the measurements that we have. And the index i here goes over all n measurements. The k sub h notation here corresponds to a kernel function. A kernel function is a function that typically achieves its maximum at 0 and decreases as the distance between x and xi increases. We also assume that it integrates to 1. In that it satisfies the relationship expressed in this equation right here. The h value is also called the bandwidth, which is similar to the binwidth in histogram. When h equals 1, we have K1, as a special case of K sub h function. We refer to K1 as the base form of the kernel. And we sometimes denote it using just K without the subscript 1. This mathematical definition is not easy to understand right away. But what happens is that we take each of the data point, x1 through xn. We replace the data point with the kernel function that is centered at that specific data point. We'll see in a bit a few examples of kernel functions. So we take each data point. We replace it by kernel function centered at that data point, and then we take the average of all of these functions. This gives us the smoothed histogram function. Smoothed histogram is a replacement to histogram and sometimes is more useful since it's a better approximation of the underlying density.

Smoothed Histograms

Four popular kernel choices are the **tricube, triangular, uniform, and Gaussian kernels**, defines as $K_h(r)=x^{-1}K(r/h)$ where the $K(\cdot)$ functions are respectively

$$K(r) = (1 - |r|^3)^3 \cdot 1_{\{|r|<1\}} \qquad \text{(Tricube)}$$

$$K(r) = (1 - |r|) \cdot 1_{\{|r|<1\}} \qquad \text{(Triangular)}$$

$$K(r) = 2^{-1} \cdot 1_{\{|r|<1\}} \qquad \text{(Uniform)}$$

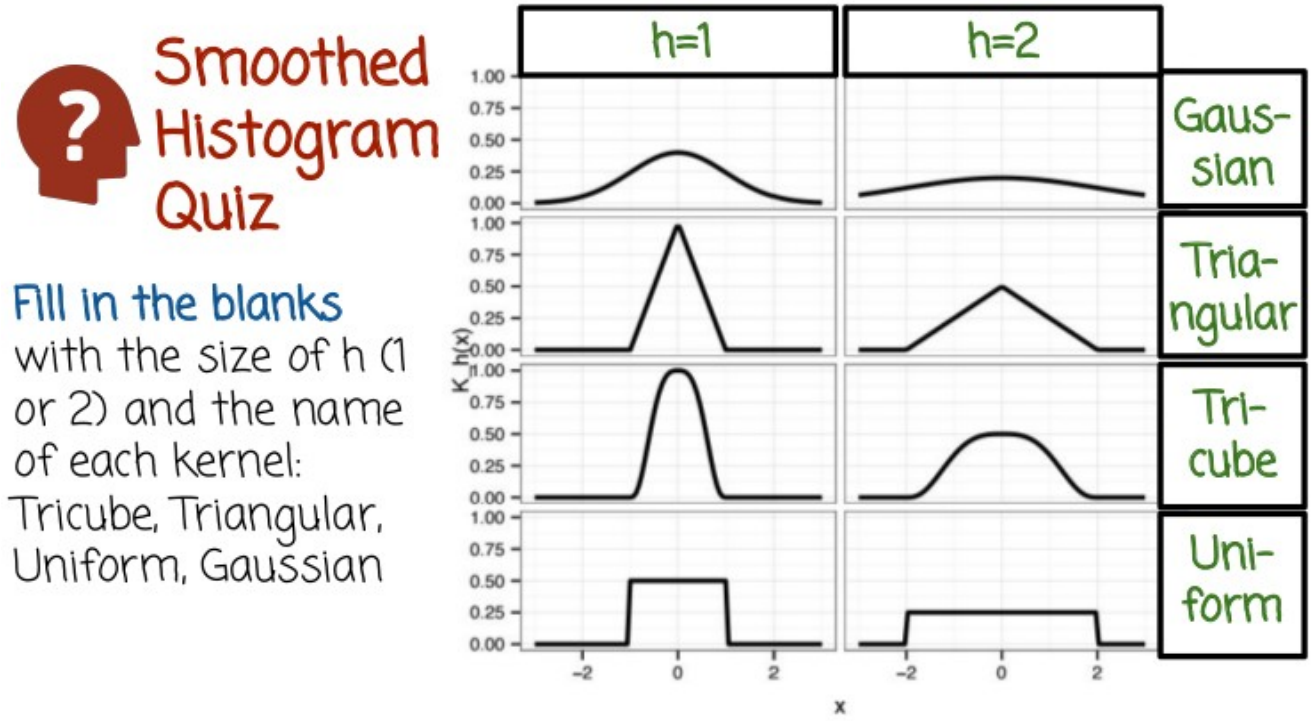
$$K(r) = \exp(-x^2/2)/\sqrt{2\pi} \qquad \text{(Gaussian).}$$

As h increases the kernel functions K_h become wider.

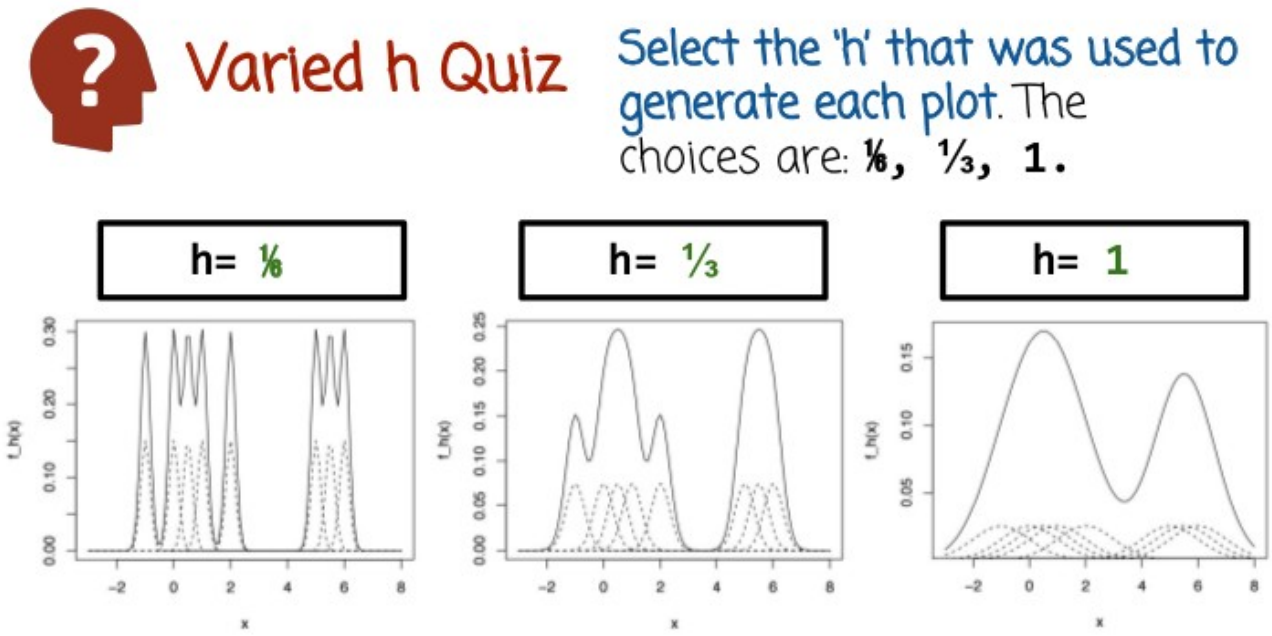
Let's see a few possible kernel choices. Here are some popular choices, tricube, triangular, uniform, and Gaussian. Remember, the kernel definition needs to satisfy this relationship, which allows us to define the kernel function using just the base kernel form, or K sub 1, or just K. Here are the four definitions for these four kernels, and in a little bit, we'll see how they look like. The Gaussian kernel may be familiar to those of you that studied probability before. It is also the Gaussian density. A uniform kernel looks like a bump. It equals 0 except for an area round 0 where it equals 1. A triangular kernel looks like a triangle. The tricube looks like a smooth bump. The notation here corresponds to a function that is 1, if the condition inside the curly braces holds, and 0 otherwise. So for example, the

uniform kernel equals 2 to the power -1, which is one-half times 1 if the absolute value of r is between 1 and -1 and 0 otherwise. In other words, the uniform kernel is everywhere is 0 except for between -1 and 1 where it equals one-half. In all of these cases, as h increases, the kernel functions Kh become wider. This explains why we call h the bandwidth. Smaller h will create a smooth histogram that has a lot of noise and may be too wiggly. A large h may create a smoothed histogram that is very smooth, perhaps overly smooth, hiding important details in the data.

16. Remember in the previous example, we saw four different kernels. Tricube, triangular, uniform, and Gaussian. And each of these kernels was parametrized by a bandwidth H. In this quiz, fill in the missing blanks where the columns correspond to values of h being 1 or 2, and the rows corresponding to the four possible kernel choices among triangular, uniform, tricube, and Gaussian.



17. The left column corresponds to h = 1. The reason we know that is that the only other option is h = 2. And as we saw before, as you increase h, the bandwidth becomes wider. Therefore the narrower kernel functions, which correspond to the left column, have a smaller bandwidth than the wider kernel functions, which will get bandwidth of h = 2. We can also find that out by comparing exactly the graph that you see here to the mathematical definition or for example, the kernel functions in previous slides. The top row corresponds to the Gaussian kernel. The second row corresponds to the triangular kernel. The third row corresponds to the tricube kernel. The fourth row corresponds to the uniform kernel. Remember that each one of these eight graphs can be used as a kernel in smooth histogram. A smooth histogram places any one of these kernels in place of each of the data points and then adds up all of these functions and divides by the number of data points.



18. In this quiz, we see three different smooth histograms corresponding to the same data set but different kernel functions. Select the h or bandwidth that was used to generate each plot where the three possible choices are 1/6, 1/3, and 1. Notice how each of the data points is replaced by kernel function denoted using the dashed lines. These different functions are then added up and the result is divided by the number of data points.

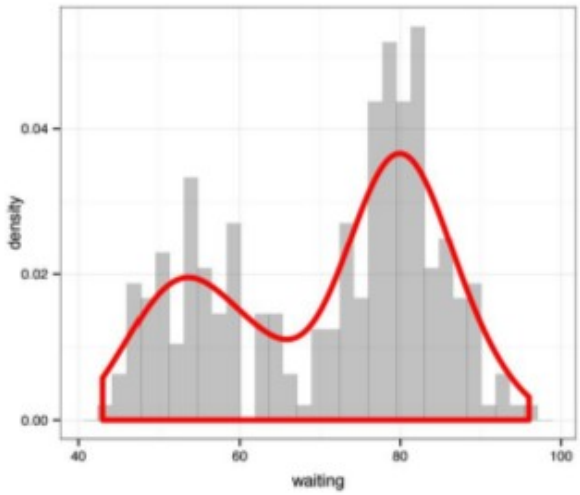
19. The left graph corresponds to the narrowest kernel functions. As a result, we assign it to bandwidth six, which is the smallest possible bandwidth among our three choices. The middle graph has a bandwidth that's slightly larger. But it's not the largest. You can see that the bandwidth of the kernels are slightly larger. The right graph has the widest kernel, and so we pick the largest bandwidth 1

20. In this quiz, we'll see how to create a small histogram in ggplot. Two, using the ggplot function. And we'll overlay it on a scatter plot of the old faithful waiting times. Fill in the blanks with the values that will create the graph shown on the right. You may need to try different values to find out which are the correct ones.

Old Faithful Histogram Quiz

```
ggplot( faithful , aes
(x = waiting, y = ..
density..)) +
geom_histogram(alpha =
  0.3 ) + geom_density
(size = 1.5 , color
= " red ")
```

Using the Old Faithful dataset, write the command the will create the histogram.



21. The first blank should hold the name of the data frame, faithful. The first function ggplot creates the histogram that you see in gray. We use the plus operator to add to it the histogram geometry and then, on top of that, we want to overlay the smooth histogram. The second blank, 0.3, assigned to the variable alpha, tells R to make the histogram slightly transparent. We do that since the histogram will be in the background and we don't want it to be too dark, so that the smooth histogram will not show properly. The third blank size equals 1.5 determines the width of the smooth histogram line. And the fourth blank determines the color of the smooth histogram line

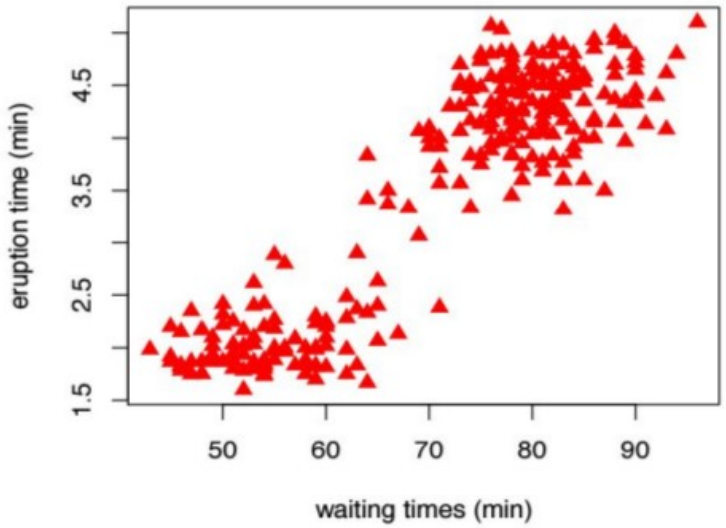
Scatter Plot

A scatter plot graphs the relationships between two numeric variables. It graphs each pair of variables as a point in a two dimensional space whose coordinates are the corresponding x, y values.

22. A scatter plot graphs the relationships between two numeric variables. It graphs each pair of

variables as a point in a two dimensional space whose coordinates are the corresponding x, y values.


```
plot(faithful$waiting,
     faithful$eruptions,
     pch = 17,
     col = 2,
     cex = 1.2,
     xlab = "waiting times (min)",
     ylab = "eruption time (min)")
```



Here is an example of how to graph a scatter plot using base graphics plot function. We pass it two vectors. The first, in this case, is the waiting column of the faithful data frame, and the second one is the eruptions column of the faithful data frame. Then a sequence of values that describe how the different pairs should be graphed. pch corresponds to the type of marker, col correspond to color, and cex correspond to the size. And we labeled the x label and the y label. Here is one of the graph in the previous slide produces. We see, again, an interesting situation where we have two clear clusters. First cluster has a short wait and a short eruption time, whereas the second cluster has a long wait time and a long eruption time. This makes sense, because if the gazer takes longer time before it erupts, meaning the waiting time is long, more pressure builds up and the eruption time is longer

23. Based on the scatter plot for Old Faithful, answer the following questions. Number of distinct cases of eruption duration. Short wait times generally result in blank eruptions. Long wait times generally result in blank eruptions.

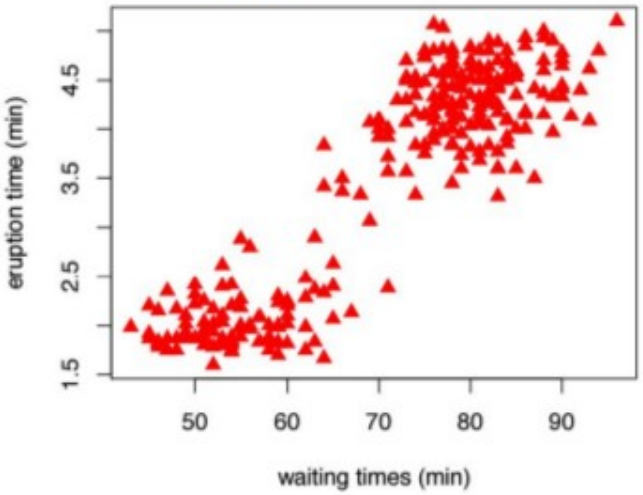
Scatter Plot Quiz

Based on the scatter plot for Old Faithful answer the following questions.

Number of distinct cases of eruption duration = 2.

Short wait times generally result in short eruptions.

Long wait times generally result in long eruptions.



24. Number of distinct cases of eruption duration are 2. If you look at the y axis, we see two clear clusters. The second blank is short, because short wait times generally result in short eruptions. The third blank is long, because long wait times, corresponding to this cluster right here, generally result in long eruptions.

The next portion of this lesson uses the datafames mtcars and mpg.
Load it into R using data(mtcars) and data(mpg).

Variable Relationships and Scatter Plots

The relationship between two numeric variables and a categorical variable can be graphed using a scatter plot where the categorical variable controls the size, color, or shape of the markers.

Line Plots

mtcars: model name, weight, horsepower, fuel elciency, and transmission type of cars from 1974 Motor Trend magazine.

```
names(mtcars)

## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec"
"vs" "am" "gear"

## [11] "carb"
```

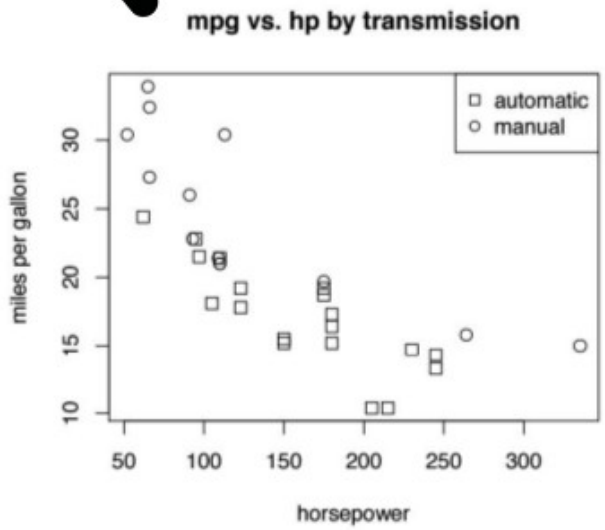
25. Thus far we saw how to create a graph to describe one dimensional numeric data or two dimensional numeric data. One dimensional numeric data was visualized using the script plot or the histogram. Two dimensional numeric data was visualized using scatter plot or a line plot. The relationship between two numeric variables in a third categorical variable can be graphed using a scatter plot where the categorical variable controls the size, color, or shape of the markers. The categorical variable, what I mean here is, the variable that's not numeric like two or three and a half or

7.1, but rather yes, no, or color, or shape. A variable that cannot be ordered using the normal ordering relation. We're going to use next the data frame mtcars, which has model name, weight, horse power, fuel efficiency, and transmission type of cars from 1974 Motor Trend Magazine. Here are the column names of the mtcars dataframe.


```
plot(mtcars$hp,
     mtcars$mpg,
     pch = mtcars$am,
     xlab = "horsepower",
     cex = 1.2,
     ylab = "miles per gallon",
     main = "mpg vs. hp by transmission")
legend("topright", c("automatic", "manual"), pch = c(0, 1))
```

The goal of this graph here, is to show the relationship between two numeric variables horsepower and miles per gallon. But also to show the relationship between these two variables in the transmission (即是 automatic 還是 manual), marked here by the am column in the dataframe. The first vector is going to be tied to the X-axis. The second is going to be tied to the Y-axis. These are the two numeric variables, horsepower and miles per gallon. pch is going to determine the shape of the marker in the scatter plot that the function plot will produce. So we're going to tell the plot function to determine the marker type based on the value of the transmission column, or AM column, of the dataframe. We're going to tell it to label x-axis using horsepower, the y-axis using miles per gallon. Gives us a general title to the graph of mpg vs horsepower by transmission. cex, the term is the size of the marker. We're going to add the legend as well.

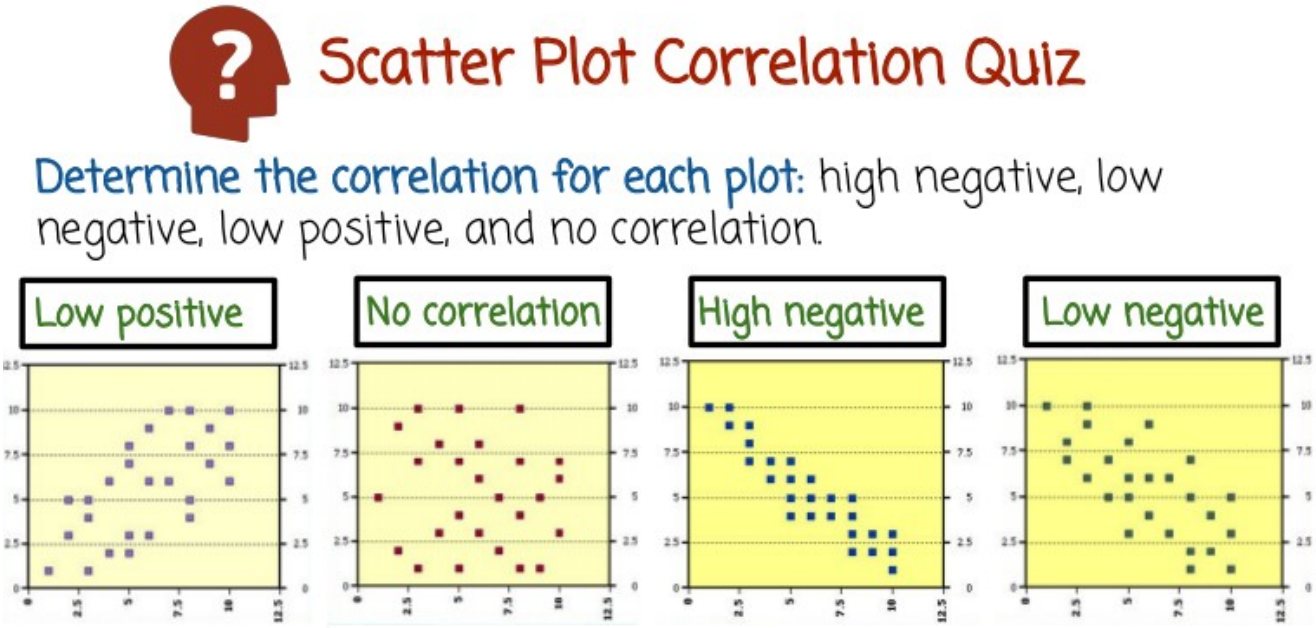
Variable Relationships and Scatter Plots



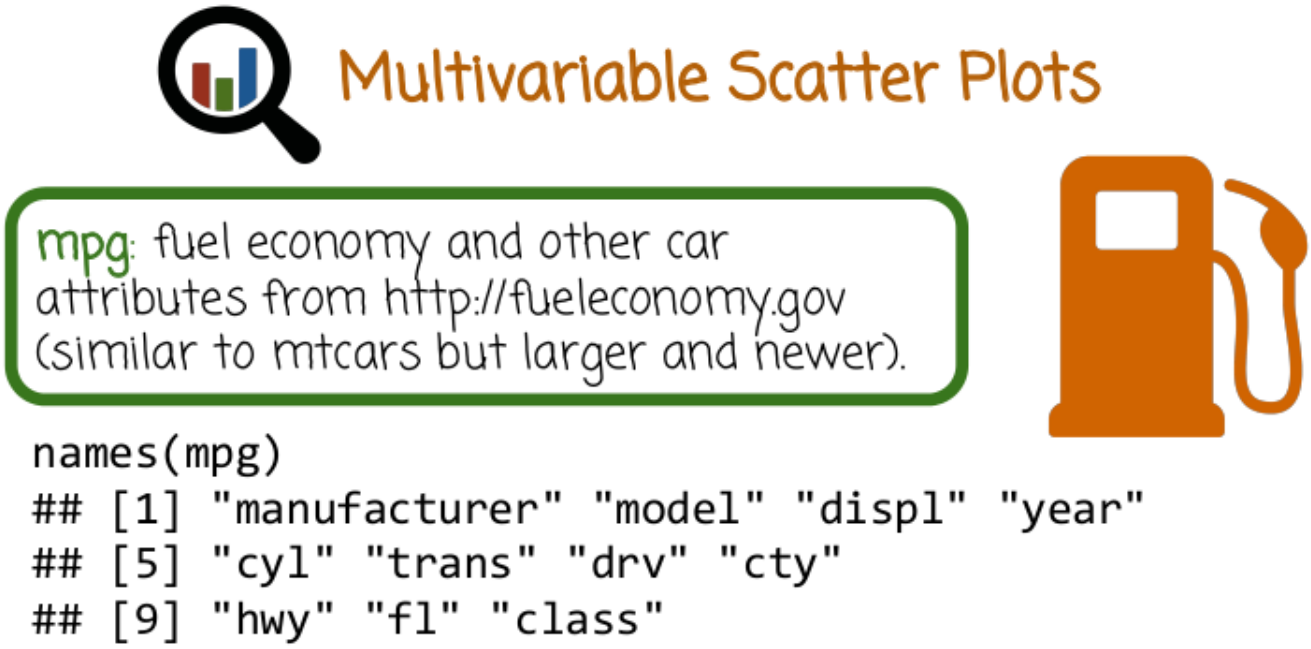
- There is an **inverse relationship** between horsepower and mpg.
- For a given horsepower amount, **manual transmission cars are generally more fuel efficient.**
- Cars with the **highest horsepower tend to be manual.**

This is the graph that is produced by the code segment in the previous slide. We see the miles per gallon versus horsepower, but you can also distinguish between different transmission types determined by the size or the type of the marker, whether it's a circle or a square. We see an inverse relationship between horsepower and miles per gallon, the bigger the horsepower the lower the miles per gallon. We also see for a given horsepower amount, manual transmission cars are generally more fuel efficient than the automatic transmission cars. We also see that cars with the highest horsepower, tend to be manual, indicated by these two cars here, on the right. Looking at the dataframe, and finding these two cars corresponding to the two right-most values, we find that they're actually sports cars, Maserati Borine, Ford Tentera, both of them, cars with manual transmission.

26. Scatter plot often reveals whether there is correlation between the two variables. In this quiz determine the correlation for each scatter plot. The four options are high-negative, low-negative, low-positive, and no correlation.



27. The first plot shows a low positive correlation. The second plot shows no correlation. The third plot shows high negative correlation. The last plot shows low negative correlation. If you'll remember from previous courses, a positive correlation shows a tendency to increase one variable as the other variable increases. For example, we see it in this graph. A negative correlation shows a tendency of one variable to decrease when another variable increases. Low versus high, either positive or negative correlation, correspond to whether that tendency is very clear, or strong, such as in this graph. Or is it not very strong, or more noisy for example, in this graph?

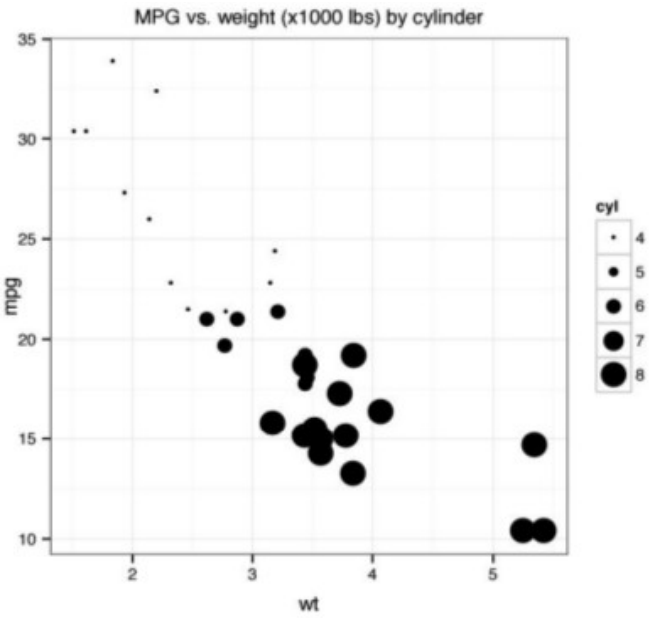


28. We're going to look next at the mpg data frame which we already looked at before. MPG data frame is similar to mtcars. It's slightly larger and newer. It holds the fuel economy and other car attributes from fueleconomy.gov website. Here are the columns' names, manufacturer, model, displacement, which corresponds to the size of the engine, year, cylinders, transmission and so on.

Multivariable Scatter Plots

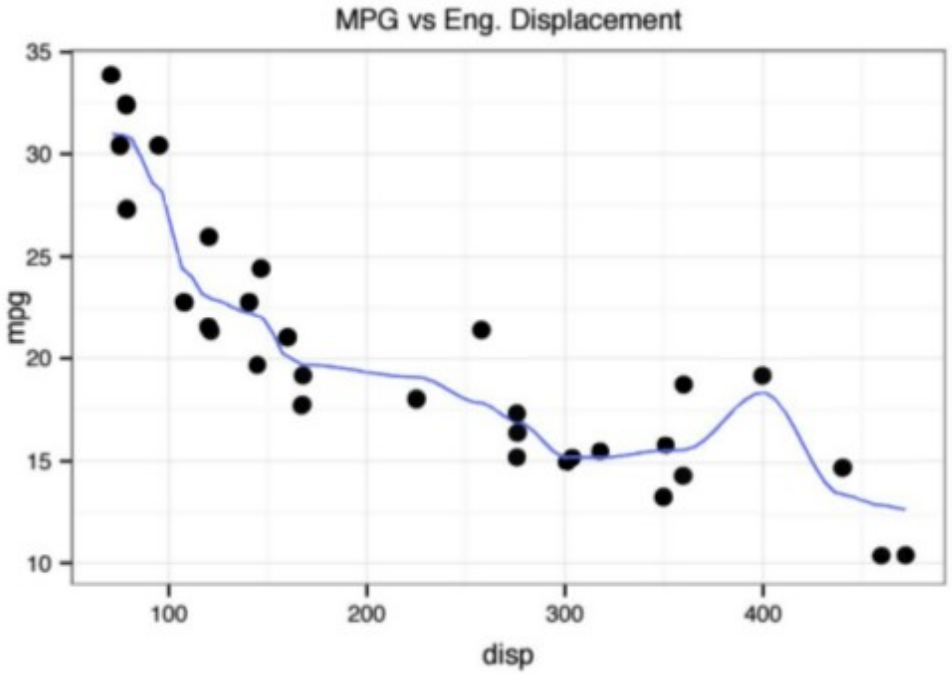
Changing marker size in a scatter plot

```
qplot(x = wt,
      y = mpg,
      data = mtcars,
      size = cyl,
      main = "MPG vs. weight (x1000 lbs) by cylinder")
```



In this example we're going to change the marker size to reveal a relationship between three different numeric variables in a scatter plot. Because scatter plots are two-dimensional and we want to examine relationship between three different numeric variables, we need to use the marker size to reflect the third variable. Here's how we do it with the qplot function. The weight column is going to be tied to the variable x, or the x axis, the mpg to the y. The cylinder data frames, or number of cylinders, is going to be tied to the size variable that controls the size of the markers. Here is the graph that is created by the previous slide. As we see, there is an inverse relationship between the weight in the miles per gallon, heavier cars tend to have lower miles per gallon. We also see, the heavier cars tend to have more cylinders.

Noisy Data



29. When data is noisy, it is useful to add a smooth line curve to visualize median trends. For example, when we plot the miles per gallon against displacement or size of engine, it is a little difficult to see trends beyond just a general trend of inverse relationship. One technique to address this issue is to add a smooth line curve right here that shows a finer relationship. Specifically, a possible increase around 400 which maybe noisy or may not be. It's impossible to tell given the dataset that we have here, but that's a pattern that is revealed using the smooth line.

Noisy Data

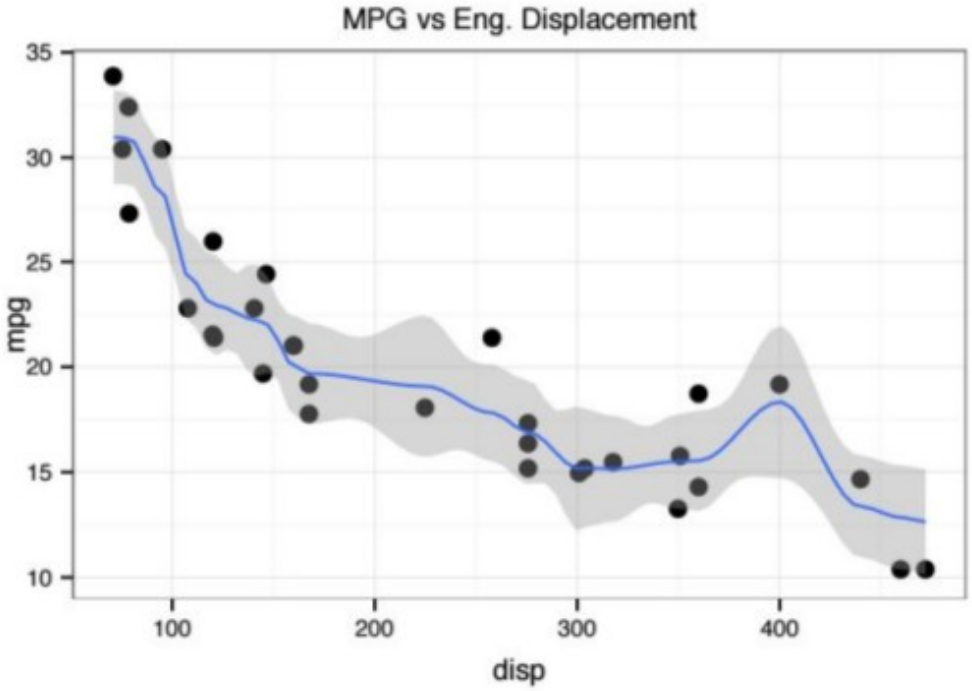
Adding a smooth line curve y_s, which a weighted average of the original data. $(y^{(i)}, x^{(i)})$ $i = 1, \ldots, n$:

$$ys(x) = \sum_{i=1}^{n} \frac{K_h(x - x^{(i)})}{\sum_{i=1}^{n} K_h(x - x^{(i)})} y^{(i)}.$$

Where the K_h functions above are the kernel functions described earlier.

- $y_s(x)$ = a weighted average of $y^{(i)}$ values
- The denominator ensures the weights sum to 1.

Smooth lines are similar to smooth histograms. Let's see the mathematical definition. We assume that we have n pairs of values, xi and yi. We want to add the smooth line to these n pairs. Here's the definition of the smooth line. It's a function that maps every value on the real line to another value on the real line. And it's defined as a weighted average of the y values, where the weights are given by the ratio right here. The ratio here has a numerator and a denominator. The numerator has kernel functions just like we saw in the smooth histograms. In the denominator is a normalization coefficient that makes sure the numerator's all summed to one. If we did not have the denominator or normalization coefficients, the weights corresponding to the fractions would not sum to one. And thus the smooth line would become inflated or deflated making sure that the weights sum to one calibrates, the smooth line to have values in the same range as the original measurements. Intuitively again, the smooth line is the weighted average of the y values, where the weights are higher for y values that are closer to the current position that you are evaluating. The weight will go down for y values that are far away from the point at which you are looking at, or evaluating. That's why this is called local average, another name for this specific model is local regression, or locally constant regression, or another Io Watson regression.



Here is again the graph of the smooth line, tying it back to the mathematical definition in the previous slide. Remember that the value of the line, or the y coordinate corresponding to a specific x coordinate is determined by the average of the y coordinate of the points that are nearby to where you are evaluating it. So for example, the value of the line around 400 will be a weighted average with a weight of this point right here being particularly high in the weighted average. And the weights of the neighboring points being lower, where even further points will have even lower or 0 weight

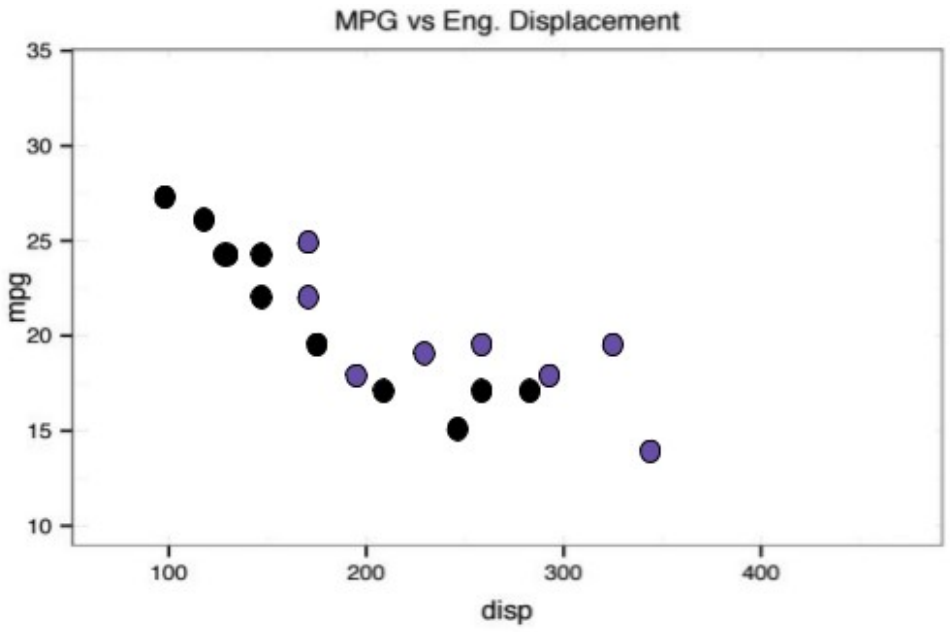
30. Fill in the blanks to add a line and standard error to the MPG versus Displacement plot.

Noisy Data Quiz

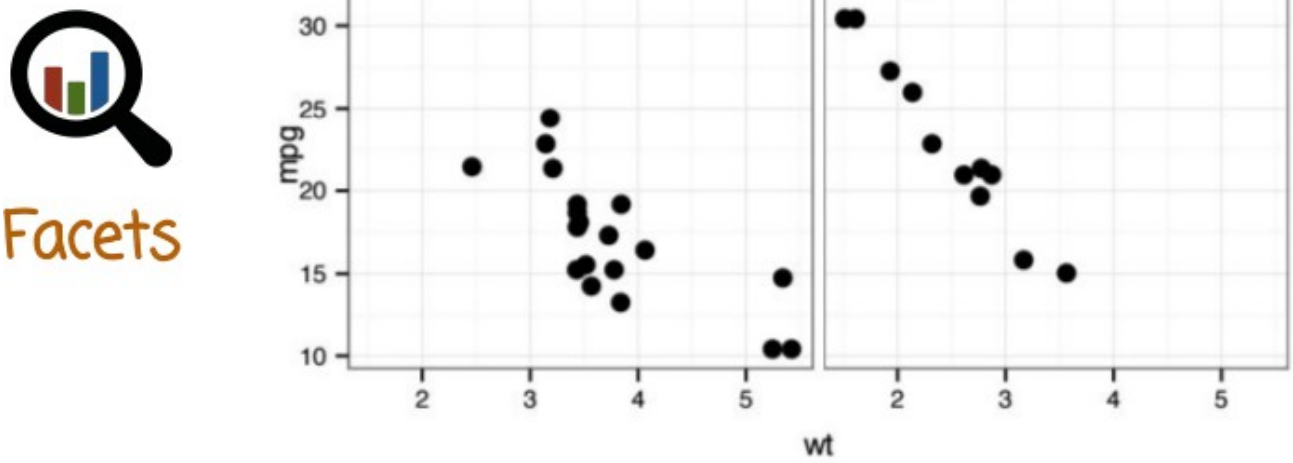
Fill in the blanks to add a line and standard errors to the MPG vs Displacement plot.

```
qplot( disp , mpg , data = mtcars , main =
" MPG vs. Eng. Displacement ") + stat_smooth(method =
" loess ", degree = 0 , span = 0.2 , se =
 TRUE )
```

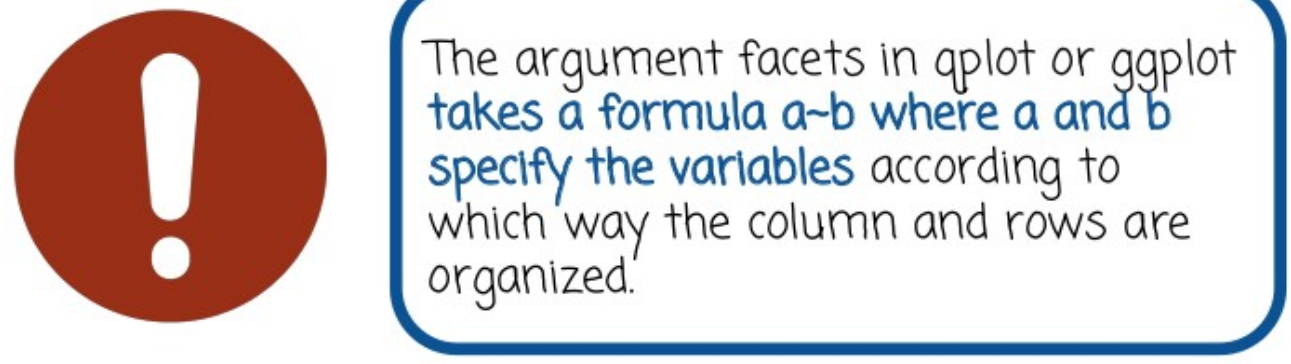
31. The default order of the arguments past the qplot means that the first variable or vector will be tied to the x-axis. In this case it's going to be displacement. The second variable will be tied to the y-axis, in this case mpg. In the third blank we need to specify the data frame name. Notice it's a sign to the data variable which holds the data frame name. In the fourth blank, we need to write the title of the figure, in this case, MPG vs Eng Displacement. After we finish the qplot function, which draws the scatter plot, we want to add a line smoother or smooth line. So we call the stat_smooth function that adds another layer of smooth line. We need to pass it the specific method by which it can compute the smooth line. In this case, we used a technique called loess. The degree is a parameter associated with the local smoothing operation, in this case, we choose degree 0. This corresponds relatively closely to the mathematical definition we saw in the previous slide of locally constant regression. The span variable is related to the bandwidth that we saw in kernel smoothing. se stands for standard errors and passing it the value true will graph not only the line smoother but also standard error or confidence bounds around that line. Notice that other values are possible as well.



32. We already saw a few techniques of graphing a relationships between three variables. For example, using scatter plots where either the type of the marker or the size of the marker corresponds to the paired variable. We're next going to see a different technique called facets(分面), that allows us to visualize more than two dimensions.



Here's an example of a facet graph, we have two separate panels. Each panel is showing the mpg or miles per gallon on the y axis and the weight on the x axis. The difference between the two panels is labeled at the top, in this case, based on the transmission. The left panel corresponds to automatic transmission cars and the right column corresponds to manual transmission cars. An important feature is that the x axis on both panels is identical and the y axis on both panels is also identical. And the two panels are lined up perfectly against each other, so that you can compare the y values in one panel to the y values in another panel.



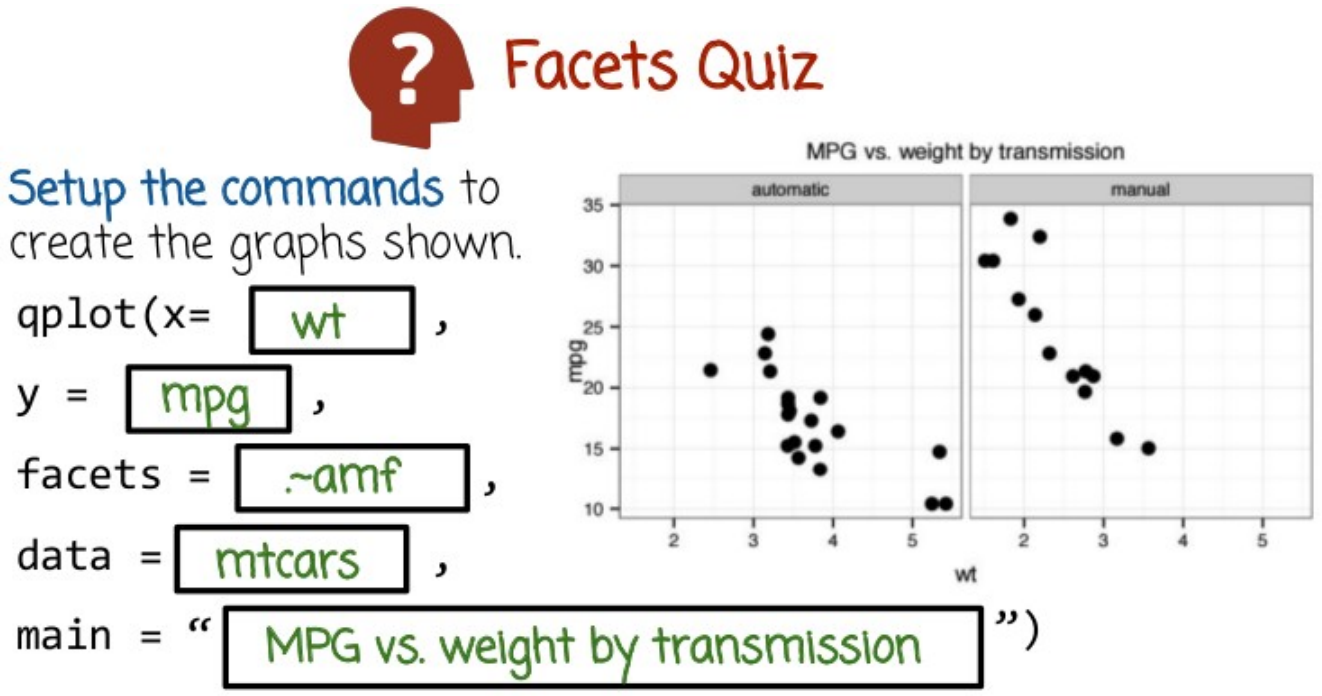
Qplot or a ggplot functions allow us to create facets. We need to pass a formula of the following form a~b where a and b specify the variables according to which way the column and rows of the panels are organized. We're going to show several examples of how to create facets based on the mtcars dataframe.

Before the following quizzes we need to modify the mtcars dataframe to have new columns with more appropriate names for better axes labeling:

```
mtcars$amf[mtcars$am==0] = 'automatic'
mtcars$amf[mtcars$am==1] = 'manual'
mtcars$vsf[mtcars$vs==0] = 'flat'
mtcars$vsf[mtcars$vs==1] = 'V-shape'
```

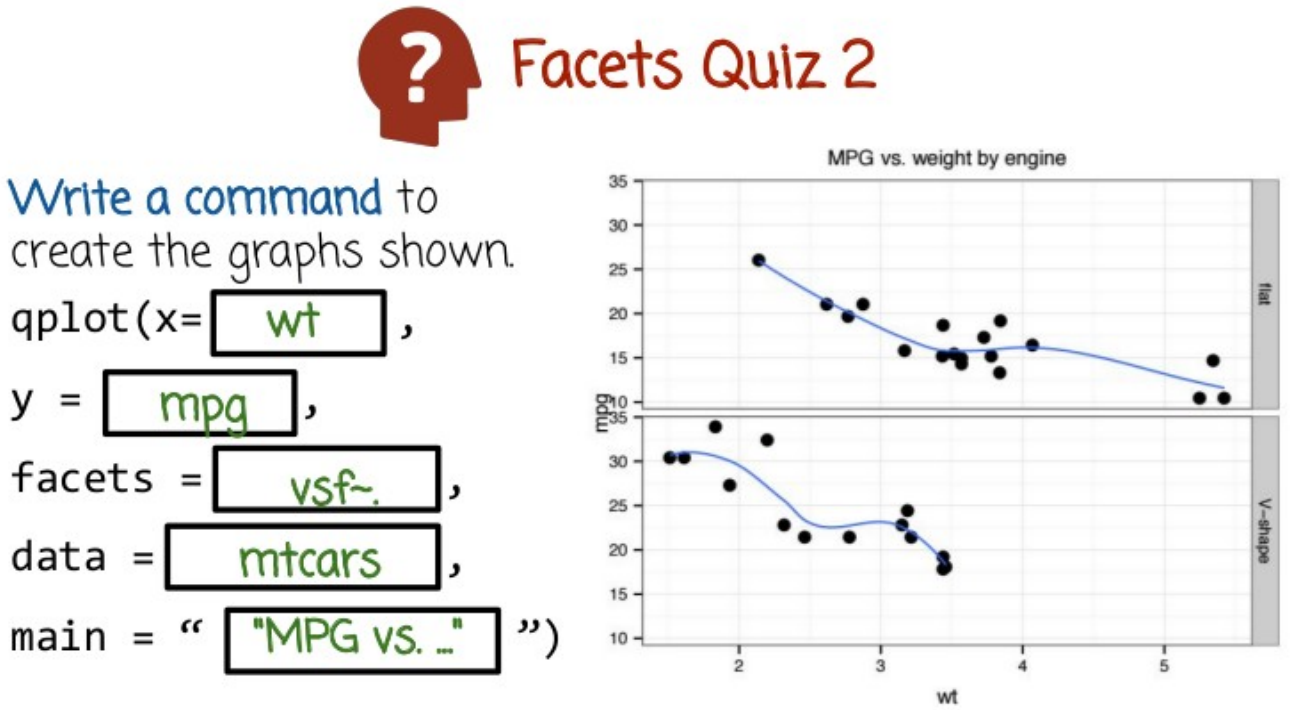
But before we show the examples, we need to modify the dataframe a little bit. To have new columns with more appropriate names, so that we can have better axie labeling. Specifically, we're going to create two new columns, amf and vsf. Amf corresponds to the transmission type, automatic or manual. And vsf will correspond to the shape of the engine, whether it's a flat engine or a V-shape engine. Notice that we label these columns using strings that are descriptive. These strings will get automatically extracted using the ggplot2 package. And the corresponding axes labelings and legends will be created without us having to do any additional efforts. This demonstrates the principle I discussed earlier of ggplot2 package. Separating between the variables or the dataframes and the graphing. We spend a little bit of extra work creating a dataframe with descriptive names and then the graphing becomes much easier.

33. Fill in the missing blanks to create the graph on the right side.



34. Notice that the x axis of both panels is the weight so we pass the variable weight or WT, which is the column name corresponding to weight. Then we tie to the variable x which corresponds to the x axis values. Similarly the y axis, MPG or miles per gallon. So we pass the mpg column name and assign it the variable y inside the qplot function. Remember that the argument that we need to pass to create a facet plot is a formula. This formula has a tilde, and then an argument to the left of the tilde and to the right of the tilde. This specific formula is telling qplot to have only one row because we have a period to the left of the tilde. When you have multiple columns, where each column corresponds to different values of the amf column in the data frame, amf being the transmission type column. Assign that formula to the facets variable. Data will hold the name of the data frame and main is the overall title of the figure.

35. Fill in the missing blanks to create a graph on the right.



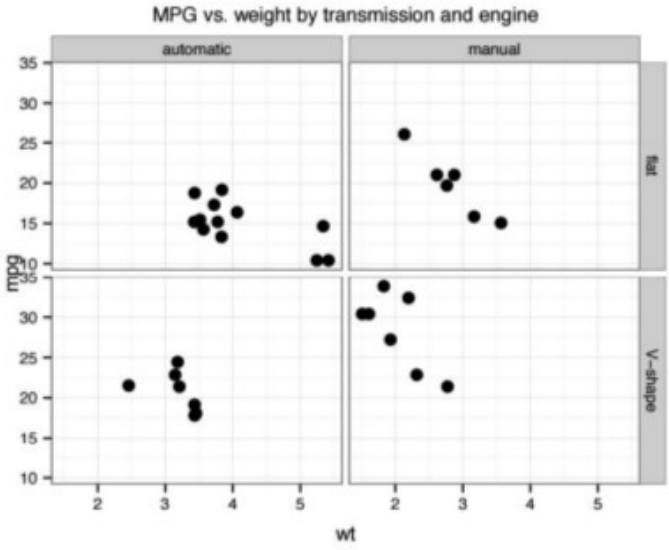
36. The x-axis is the weight. So we're going to assign the column name wt to the x variable. The y-axis in both panels is miles per gallon. So we're going to assign the column name MPG in the data frame to the y variable. The facets in this case are showing two different rows and one column. So we need to write the variable that's going to dictate how the rows will appear to the left of the tilde symbol. In this case, it's vsf, corresponding to the shape of the engine, flat or v-shape. To the right of the tilde we're going to put a period or a dot to indicate that there's only one column. Data would hold the name of the data frame, and main would hold the overall title of the figure.

37. Fill in the missing blanks to create the graphics on the right.

Facets Quiz 3

Write a command to create the graphs shown.

```
qplot(x = wt ,
y= mpg ,
facets= vsf~amf ,
data= mtcars ,
main = " "MPG vs. ..." ")
```



38. Notice that in each of the four panels, the x-axis holds the weight, so we assign the data frame column named wt to the x variable. The y-axis in each of the panels corresponds to miles per gallon. So we assign the column name mpg to the variable y. There are four facets, or two rows and two columns, where the rows correspond to the shape or vsf column name. And the columns correspond to amf or transmission column. So we pass the formula vsf tilde amf. The data frame name, mtcars, is assigned to the variable data. And the title of the overall figure is assigned to the variable main.

39. From graphs created previously from the mtcars data, which type of car has the following characteristics? Fill in the missing blanks with M for manual, V for v-shape, A for automatic, and F for flat. Each blank should have two letters. One for the transmission, manual or automatic and one for the shape, v-shape or flat.

Facets Data Inference Quiz

From graphs created from the mtcars data, which type of car has the following characteristics?

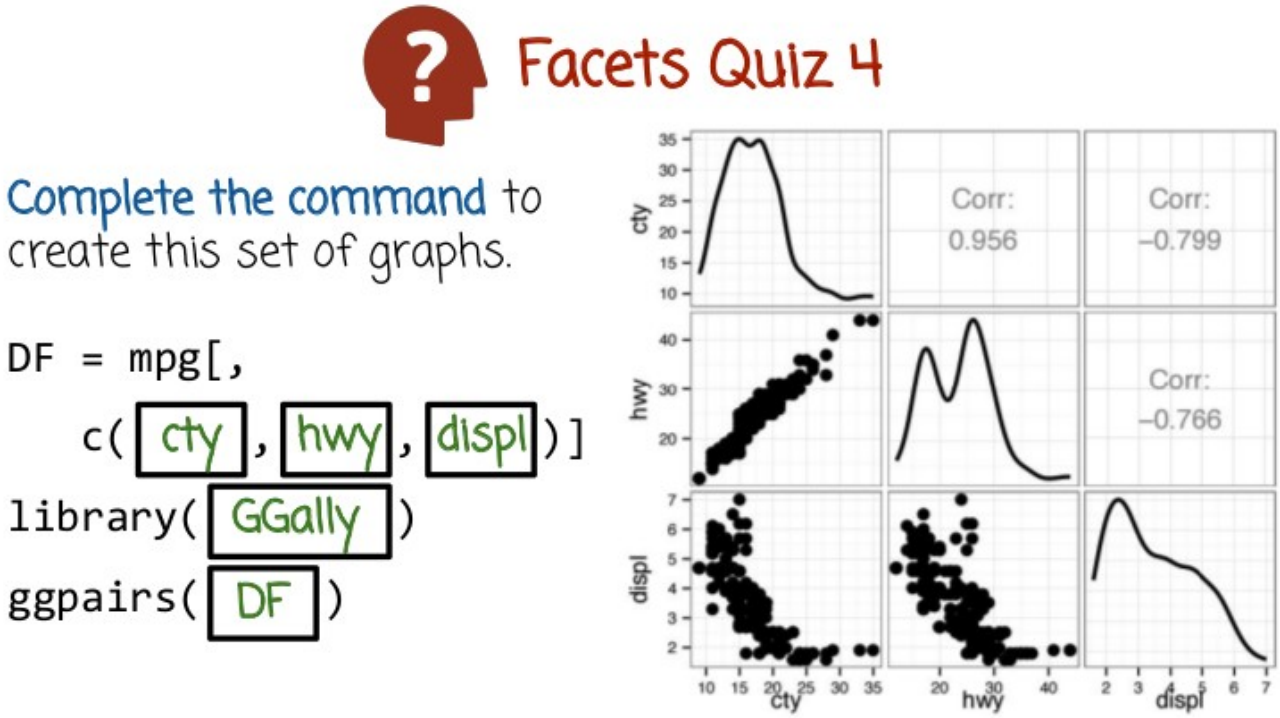
Type M for manual, V for v-shape, A for automatic, F for flat.

M,V Tend to have lower weights and be more fuel efficient

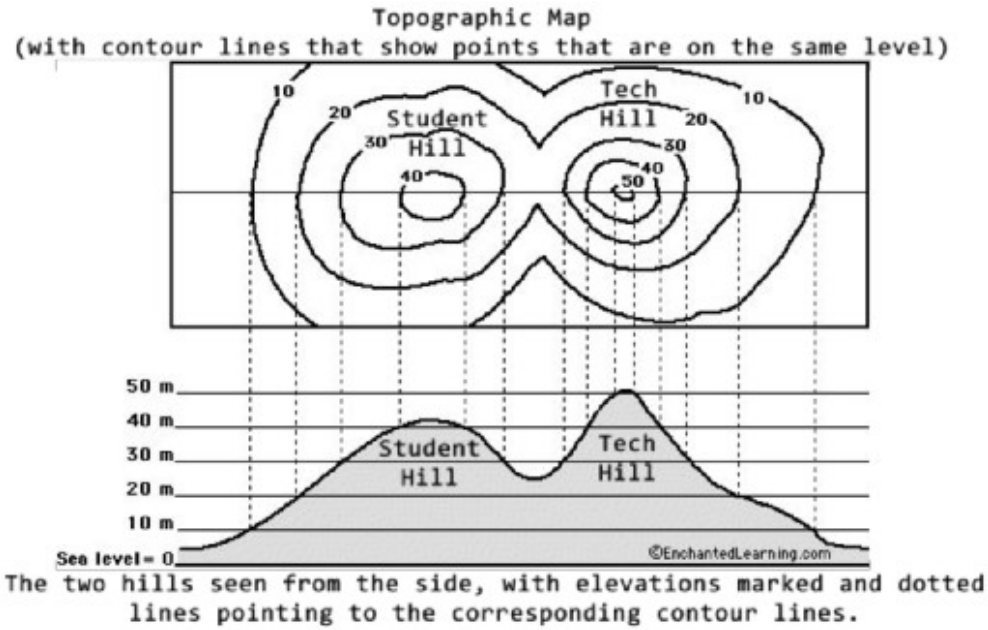
A,F Tend to be heavier and less fuel efficient

40. From the previous graphs, we can deduce that manual cars with V-shaped engine tend to have lower weight and be more fuel efficient. We can also see that automatic cars with flat engine tend to be heavier and less fuel efficient. This particular analysis where we can analyze miles per gallon, weight, transmission, and engine shape actually shows relationship between four variables. If we didn't use panels, we would have to create much more scatter plots, and we will not be able to directly interpret the relationship between these four variables as easily. Creating the four panels side-by-side where each panel is a two-dimensional graph or scatter plot allows us to analyze relationship between four variables efficiently.

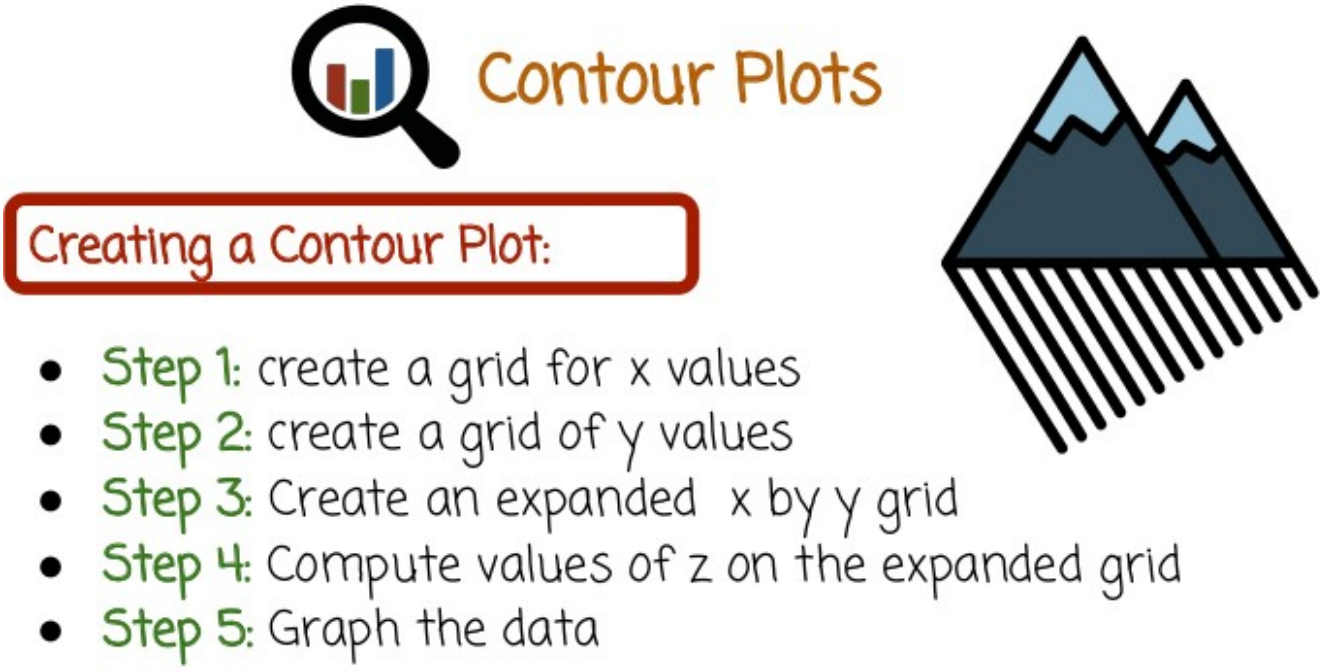
41. In this quiz, we'll see how to use the function ggpairs in the ggplot2 package to illustrate relationships between multiple variables using an array of scatter plots, smooth histograms, and correlation values. Fill in the missing blanks to create the graph on the right.



42. The first line, we create a data frame that is going to have three columns, and the three columns are going to be extracted from the mpg data frame. Notice here we do not have any name to the left of the coma, which means we select all rows, but we only going to select three columns from the mpg data frame. Specifically, the city miles per gallon column, the highway miles per gallon column, and the displacement. We can see that these are the three columns that we need, because these are the three different rows and columns in this facet array that we need. After we created our new data frame that is extracted from MPG but contains only three columns, the columns that we want to explore, we need to call the package GGally that has the function ggpairs in it. Next we call the ggpairs function with the data frame passed as an argument to it. Notice here(左下角) we have facets describing a scatter plot relationship between city miles per gallon, displacement, city miles per gallon highway. On the diagonal we have the smooth histogram of displacement, highway miles per gallon, city miles per gallon. On the top right panels, we just see the correlation values. There is no need for additional graphs to be displayed here, because these graphs would be identical to these graphs. The ggpairs function can be used on more than three columns. You can use it on larger data frames with four, five, six or more columns. That's an effective way to visualize relationships between higher dimensional data.



43. Contour plots graph a relationship between three numeric variables, z as a function of x and y. An example of a contour plot is a topographic map. We want to know the location of a hill, as well as its height. In this graph, we can see the height. In a topographic map, we can also know its location. The different contours correspond to points that have equal elevation. In this case, the topographic map shows two different hills. We know that they are hills and not valleys based on these numbers that show the elevation value, 10, 20, 30, 40.



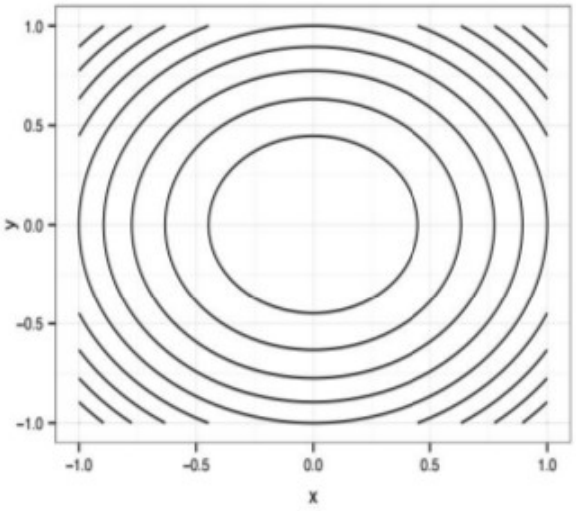
The steps to create the contour plot are, create a grid for x values, create a grid for y values, create an expanded x by y grid, compute values of z on the expanded grid, and then graph the data.

44. Fill in the missing blanks to create the graph on the right.

Contour Plots Quiz

Complete the command to generate the plot shown.

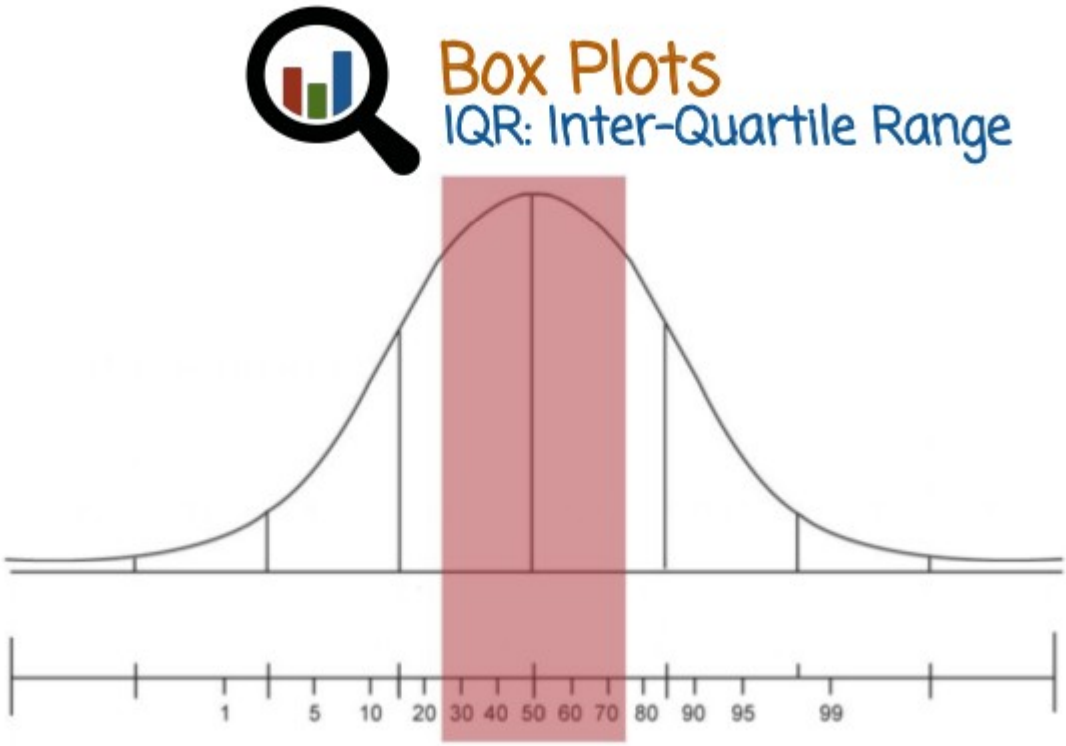
```
x_grid = seq(-1,1,length.out
= 100)
y_grid = x_grid
R = expand.grid(x_grid, y_grid)
names(R) = c('x','y')
R$z = R$x^2 + R$y^2
ggplot(R, aes(x=x, y=y,
z=z)) + stat_contour()
```



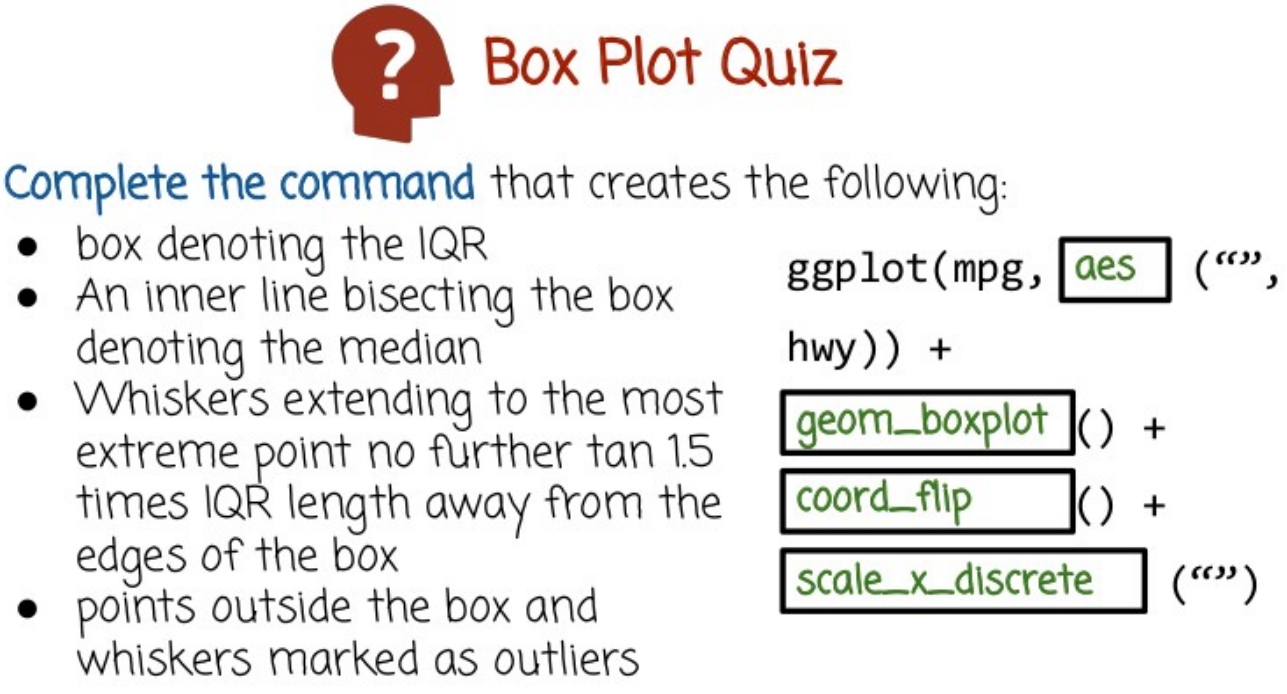
45. Remember the sequence from the previous slide of steps we need to create the contour plot. The first step is to create a grid of x-values. In this case, we do it using the seq function. We tell it to create 100 points equally spaced between -1 and 1. The x-value is between -1 and 1. We can create the y_grid in the same way, but because it's identical, we can just copy it from the x_grid. Next, we need to create the expanded grid. We do it using the function expand.grid, and we pass to it the x_grid and the y_grid. Next, we need to give it the proper names. The first name is going to be x. The second is going to be y. Giving it the proper names would help us label the different axes correctly later when we call the graphics function. The next line we create or we compute the z-values. In this case, since the coutur lines are circular, the z function or the elevation function is going to be the square of the x-axis plus the square of the y-axis. To recap, we have a data frame R that has three columns, an x column, a y column and a z column. In other words, each row in the data frame R has a triplet of values x, y, and z. We pass this data frame, next, to a graphing function. We use the ggplot function rather than the qplot function, because as we see here, we have a blank missing that requires the aes function, which is in the next blank. This creates in the first element of the graph, but we still need to create another layer and add it using the + operator. That additional layer is an object that is returned from the function stat_contour.

注意上圖不是 box plot. 真正的 box plot 是 47 段上面那個.

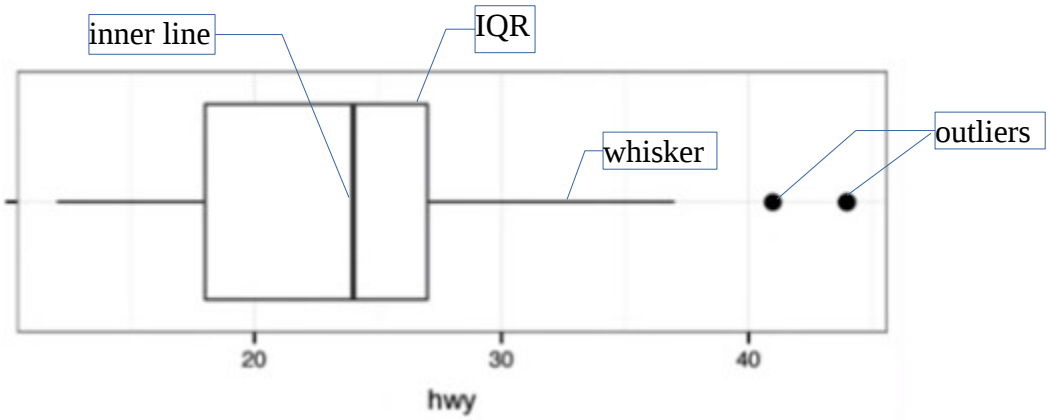
46. Box plots are alternatives to histograms that are usually more lossy, in the sense that they lose more data. But they emphasize quantiles and outliers in a way that a histogram cannot. Sometimes a histogram is more useful, but in other cases, box plots are more useful, and they reveal information that a histogram does not have. Recall that the R (此 R 是一個變量名, 不是指 R 語言) percentile of a numeric dataset is the point at which approximately R percent of the data lies underneath and approximately 100 minus R percent of the data lie above. Another name for the R percentile is the zero point R quantile. Let's see a few example of percentiles or quantiles. The median is the point at which 50% of the data lies under and 50% of the data lies over. Assuming that the curve here is the histogram of the data or a smooth histogram, in this case, a distribution function generating the data. The median is exactly halfway or in the middle point. But that's not always the case. In some cases, the left tail is heavier or lighter than the right tail. And then the median will not be in the middle. The median can be off to the side. But in any case, it's the point at which half of the data lies under and half of the data lies over. This shaded part here shows the half of the data that lies above or over the median. The 25 percentile is the point at which 25% of the data lies to the left of or under. The 75 percentile is the point at which 75% of the data lies to the left of or smaller than.



The IQR or inter-quartile range is the range between the 25 percentile and the 75 percentile. This is also the central 50% of the data.



上圖中的 mpg 是 dataframe 的名字, hwy 是 mpg 的一列.

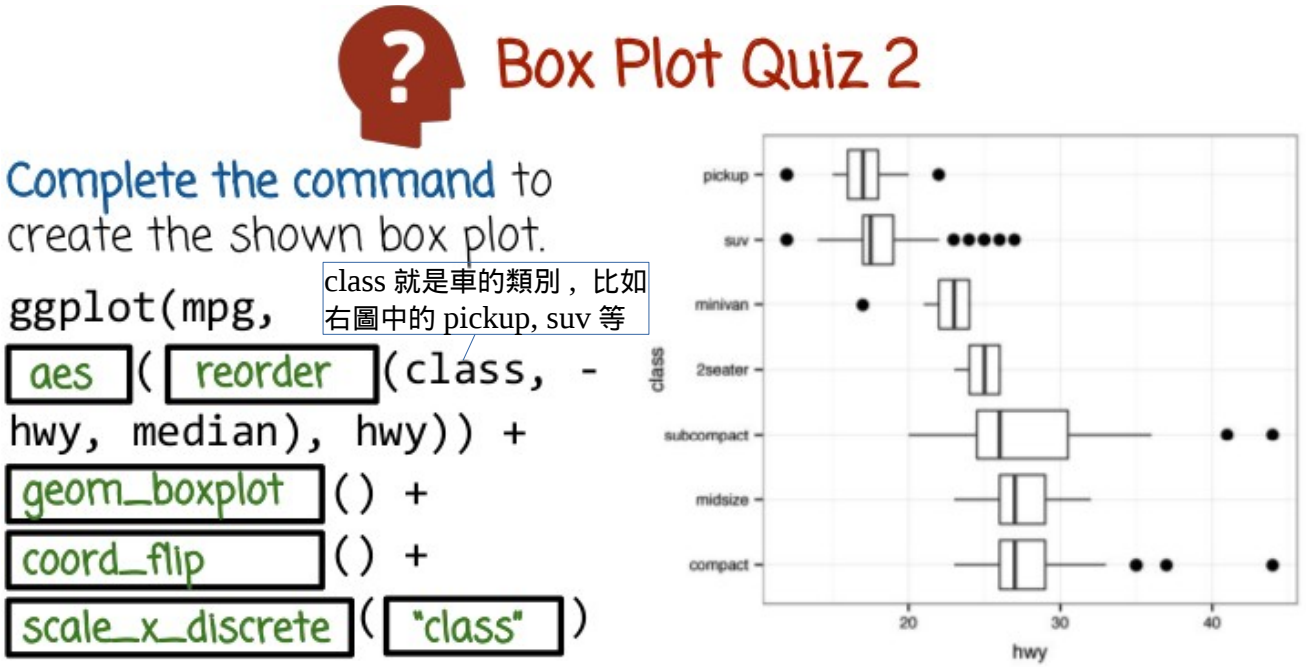


47. A box plot is a plot that shows the following element. The IQR, or interquartile range, is the range between the 25 percentile and the 75 percentile, which is denoted using a box. An inner line bisecting the box denotes the median or the 50th percentile. The whiskers(络腮胡子) indicate the range of the data, while we exclude outliers in order to show where the majority or the primary part of the data reside in. The outliers are marked separately outside of the whiskers. In this quiz, we'll see how to create a box plot in R using the ggplot package. Complete the command that creates a box plot. Notice that we have here the data frame name mpg and the column hwy, which corresponds with the highway miles per gallon in the mpg package.

(圖見上面)

48. In the first blank, we need to call the AES function, because the function that we are using is ggplot rather than qplot. If you recall, ggplot requires us to use this AES function. In the second space, after the plus operator, we need to add the box plot geometry. We get that from the corresponding function geom_boxplot. In the third space, this example we execute the function coord_flip because we want the box plot (整個 box plot) to be lying on its side as opposed to standing up. We can also omit that function, if we want the boxplot to be standing up. The last space, we put scale_x_discrete, which is also required, if we want to make the box lying down instead of standing up. Here is the boxplot. You see it's lying down instead of standing up. The X axis is labeled by the name of the column, hwy or highway. The box of the box plot denotes the IQR, interquartile range, between little bit less than 20 and little bit less than 30. The line bisecting the box is the median. These lines right here are the whiskers, showing where most of the data lie. And these two points here are outliers. As you can see the box plot provides different information then the histogram. If you have a histogram, you can see the shape of the data, whether there is one mode, or two modes, or three modes, but it's hard to read from a histogram where the median is, where the IQR is, etc. In the box plot, we have in some sense less information. It is a simpler graph. Emphasize percentile information such as the median, the IQR and outliers.

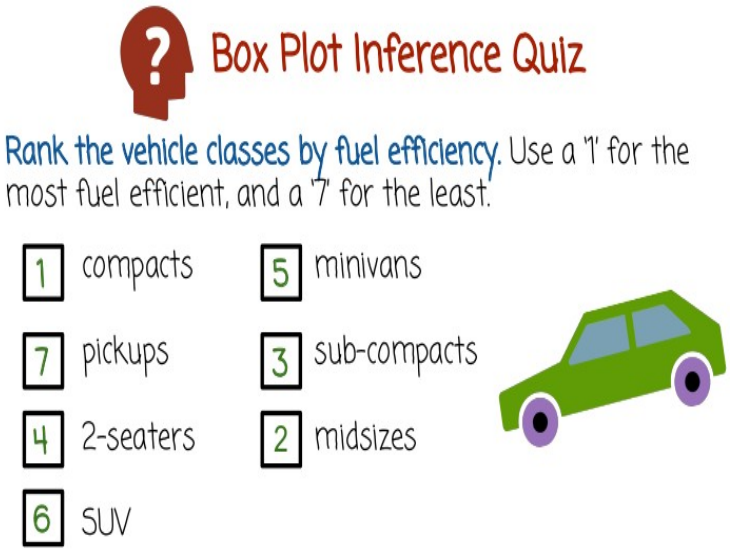
49. Complete the command to create the box plot on the right. The box plot on the right is actually multiple box plots, one for each row. Each box plot describes the highway miles per gallon for a different class of cars within the MPG data frame. What we have pick up trucks, SUV, mini vans, two seater sub-compacts, mid size, and a compact. We want to graph for each class a separate box plot. We want the box plots lying on the side instead of standing up. And we want them all neatly arranged in rows. So that we can compare the different box plots to each other.

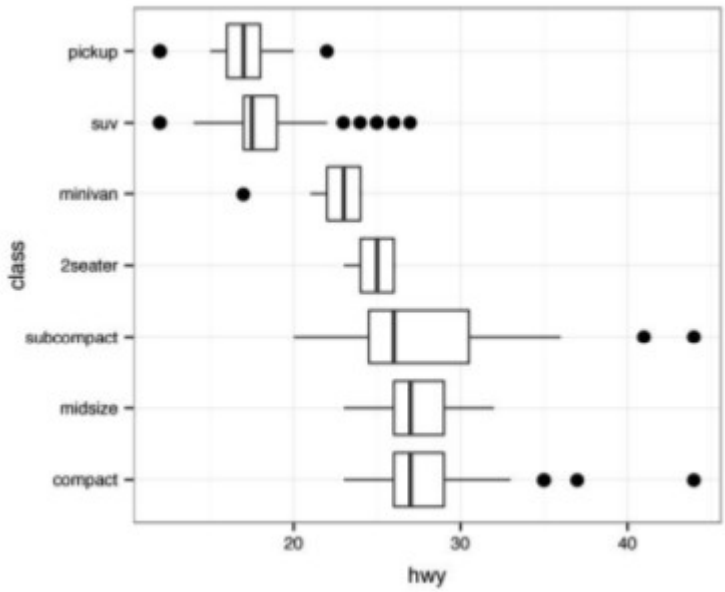


上圖中的 mpg 是 dataframe 的名字, class 是 mpg 的一列, hwy 是 mpg 的另一列.

50. The first case here we call the AES function because we use here the GG plot function rather than the Q plot function. In this space here we used the reorder function and we tell the reorder function to reorder the classes based on minus highway, which means we want to arrange it in order of increasing highway miles per gallon (即 hwy). For the highway miles per gallon is measured using the median function. In other words, compute the highway miles per gallons for each class, look at the median of that class. And order the classes based on that median. In the third space, this is after the plus operator, we need to add the appropriate geometry, in this case a boxplot geometry. The fourth place we call the function coord flip because we want the box plot lying on the side instead of standing up. This pace here we call the function scale x discrete and the last pace we use “class” because we want to make sure that the corresponding axis is described properly using the class name.

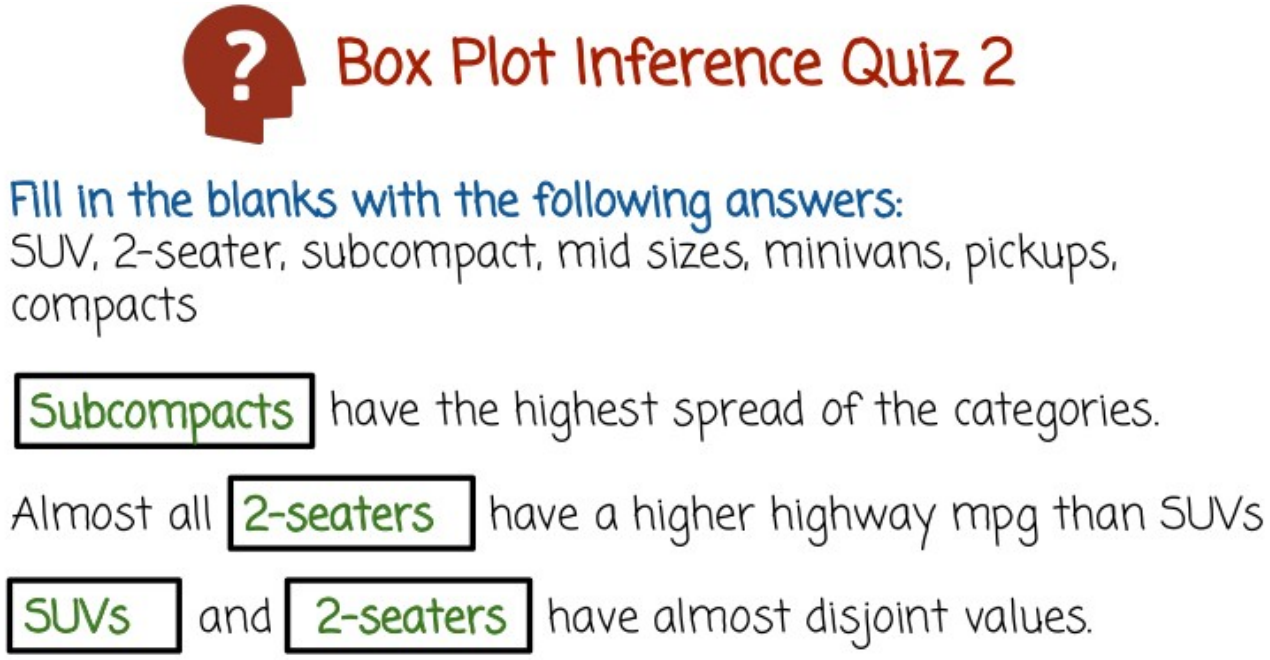
51. Based on the box plots from the previous quiz, rank the vehicle classes by fuel efficiency. Use a 1 for the most fuel efficient class and a 7 for the least.





52. In the previous box plot we see that compact cars are the most fuel efficient, pickups are seventh place, two-seaters are fourth place, SUV are sixth place, minivans fifth place, sub compacts third place, and midsize second place. In this case the particular ordering was chosen based on the median, or the middle of the box plot. The box plot gives however additional information beyond just the median and we can see whether the outliers, for example, of the SUV are intersecting with the IQR of subcompact. In this case, the answer is yes. Or whether the outliers of the pickup class intersects with the IQR of midsize. In this case, the answer is no.

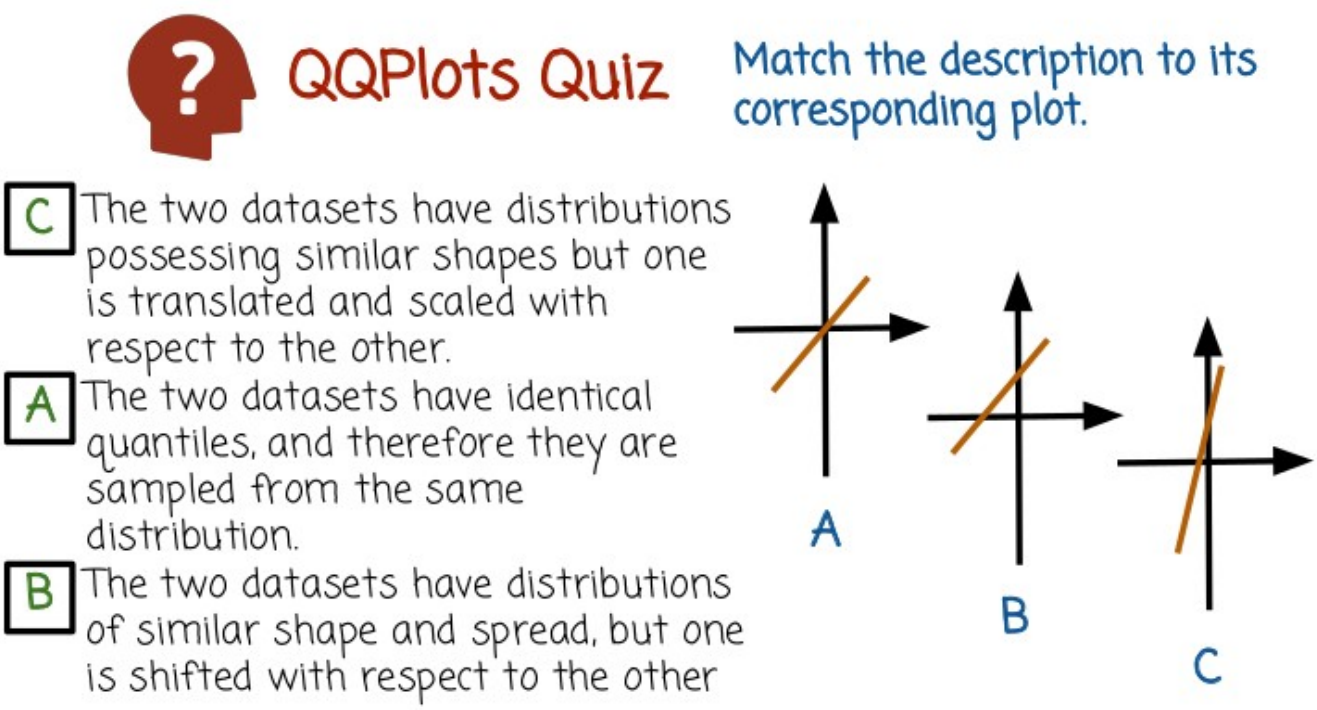
53. Fill in the blanks with the following answers based on the previous quiz and the box plot graph in it



54. In the first case, the answer is subcompacts, because they're the category with the highest spread of values. We can see that almost all two-seaters have a higher highway miles per gallon than SUVs, and we can see that the SUVs and two-seaters have almost disjoint values.


```
ggplot(R, aes(sample = samples)) +
  stat_qq(distribution = qt, dparams = pm)
```

55. A QQPlot is a scatter plot of quantiles of one data set on the x axis versus the quantiles of the other data set on the y axis. In some cases the x or the y axis describes quantiles coming from a theoretical distribution rather than specific data set. Let's see at first an example of how to compute a QQPlot in R using the GGPlot tool package. In this case we call the function GGPlot, we need to use the aes function. We add stat_qq layer and we pass it, in this case for example, we want to use a theoretical distribution called t-distribution, so we need to pass a parameter to that. We're going to plot the quantiles from the theoretical t-distribution on one axis, and the samples from a given data set, samples column in the R data frame, on the other axis.



56. Here are three examples of QQplots, A, B, and C. They all show a straight line shape. Match the descriptions to the corresponding plot.

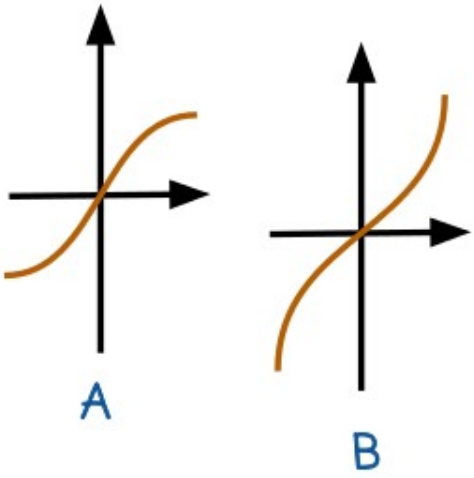
57. In the first description the two datasets have distributions possessing similar shapes but one distribution or one dataset's distribution is translated and scaled with respect to the other. In this case, we see a line with a slope different than one that does not necessarily pass thought the origin. The second description, we have two datasets that have identical quantiles, and, therefore, they are sampled from the same distribution. The corresponding graph is the first one, or A, where we have a straight line with a slope of 1 or 45 degrees passing through the origin. This is the identity function which shows quantiles of the first data set are exactly identical to the quantiles of the second data set. In the last description, the two datasets have distributions of similar shape and spread, but one is shifted with respect to the other. This description corresponds to the second graph, or B, because the two datasets have distributions of similar shape and spread, but one is shifted. We're going to have a straight line of slope 1 of 45 degrees but the straight line with 45 degrees is going to be shifted up or down rather than passing through the origin. It may not be immediately clear why the description here corresponds to the qualities of the straight line described in the description or shown in the graphs. For example, why should a graph be translated upward and keep the same slope in the last description? Why should the slope be different in the first description? But you can prove these qualities mathematically. Also, these qualities are not too hard to see if you think very carefully about the interpretation of quantiles and the definition of the qqplot.

QQPlots Quiz 2

Match the plots the corresponding description.

B The dataset whose quantiles correspond to the y-axis is drawn from a distribution having heavier tails than the other dataset.

A The dataset corresponding to the x-axis is sampled from a distribution with heavier tails than the other dataset.

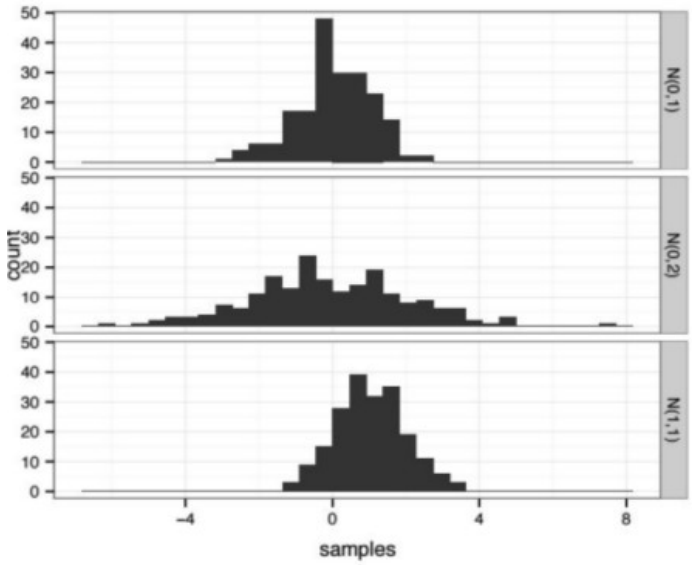


58. In this case, we have two Q-Q plots that are not straight lines. One has a shape similar to the letter S, and the other one is an inverted S. Match the corresponding description to the plot.

59. In the first description, the dataset whose quantiles correspond to the y-axis is drawn from a distribution having heavier tails than the other dataset. The correct graph is B. We see that the specific value of the X coordinate Is being mapped to a higher value on the Y coordinate because of the shape of the graph. In the second description the dataset corresponding to the x-axis is sampled from a distribution with heavier tails than the other dataset. The corresponding graph is A, and the shape is an S shape. We see here, that the specific value on the x-axis will be mapped to a lower value on the y-axis. In some sense, the quantiles on the y-axis are shrunk corresponding to the quantiles on the x-axis.

qqplot Example

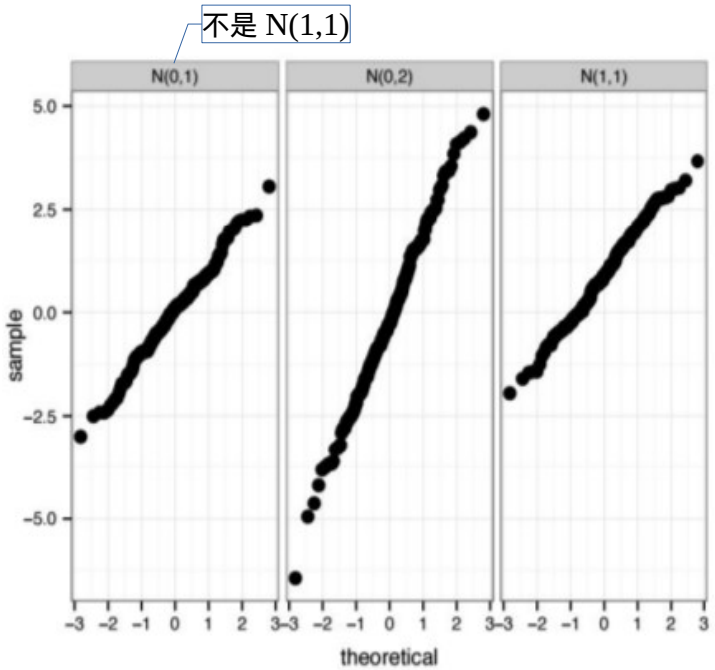
```
D = data.frame(samples = c(rnorm(200, 1, 1),
                           rnorm(200, 0, 1),
                           rnorm(200, 0, 2)))
D$parameter[1:200]   = 'N(1,1)';
D$parameter[201:400] = 'N(0,1)';
D$parameter[401:600] = 'N(0,2)';
qplot(samples,
      facets = parameter~.,
      geom = 'histogram',
      data = D)
```



上圖不是 qqplot, 而是普通的 histogram.

60. We're going to see next a few examples illustrate the qqplot graph. We create here a data frame which samples from three different normal distributions. Each sample is going to have 200 points and we're going to have three different sets of parameters. A mean 1, and standard deviation 1. A mean 0, and standard deviation 1. And a mean 0, and standard deviation 2. The mean, corresponding to the location of the samples, or the average. The standard deviation corresponds to the spread. In order for the qqplot to have nice names annotating the graph, we need to give descriptive names to these three different samples. The notation N(1,1) corresponds to the normal distribution with means 1 and standard deviation 1. Then we call the q plot function with the corresponding data, we want to create a facet plot where the rows describe the different values of the parameter or variable. We use the histogram geometry, and we use the data frame b. Here's the facet plot of the three histograms. As you see, all three histograms have a bell shape, which is the correct trace of a Gaussian or a normal distribution in the first case It's centered around zero. In the second case, it's also centered around zero. So zero being the first two parameters describing the mean, but the second case, the distribution is more spread out. This is indicated by a larger standard deviation, two instead of one. In the third case, the distribution Is a normal distribution but centered around one rather than around zero. So we see the third graph is similar to the first graph, but is shifted to the right.

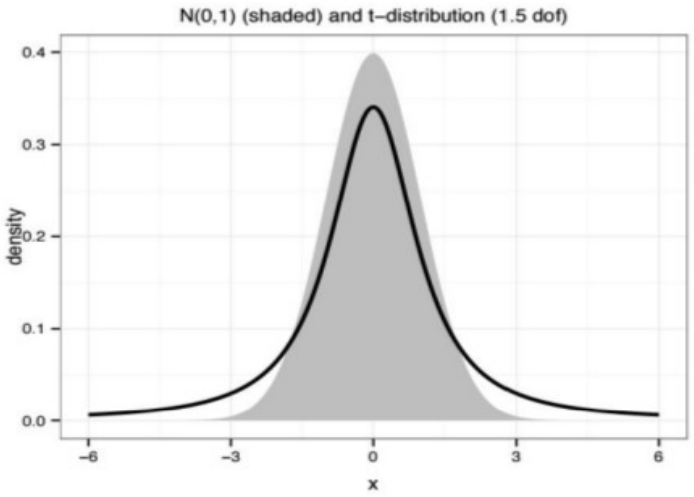

```
D = data.frame(samples = c(rnorm(200, 1, 1),
                           rnorm(200, 0, 1),
                           rnorm(200, 0, 2)));
D$parameter[1:200]   = 'N(1,1)';
D$parameter[201:400] = 'N(0,1)';
D$parameter[401:600] = 'N(0,2)';
ggplot(D, aes(sample = samples)) +
  stat_qq() +
  facet_grid(.~parameter)
```



Now let's see how we create qqplots with these data sets. This is the dataset from the previous example. In the last row, we call the ggplot function. And we tell it to create qqplots of the given samples against the theoretical Gaussian distribution with parameters 0 and 1. And put the results in a facet graph, where the columns, the value of the parameter, corresponding to the different data set. So this will give us a facet graph with three panels arranged along columns. In each panel, we're going to have a qqplot of each of these datasets against the theoretical normal distribution with parameters 0 and

1. Here is the result. As you can see, all three graphs show a linear shape or shape similar to a straight line. If you go back to the quiz, you'll see why we have a straight line. In y in the first case, the straight line passes through the origin and has a slope of one. And in the second case, we have a straight line that does not pass though the origin, and so on.

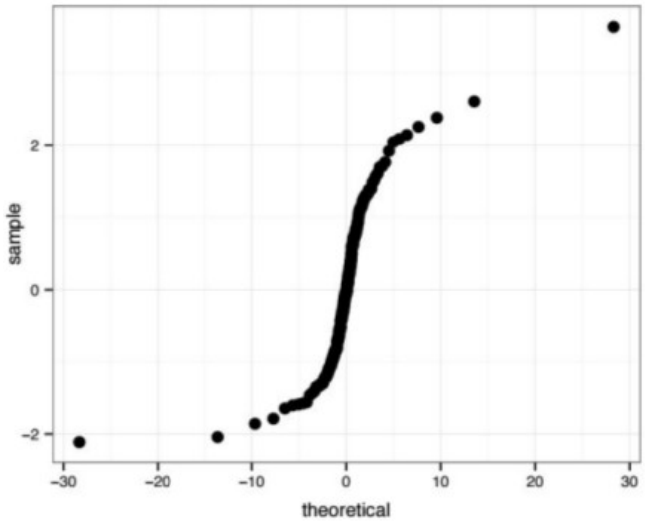

```
x_grid = seq(-6, 6, length.out = 200)
R = data.frame(density = dnorm(x_grid, 0, 1))
R$tdensity = dt(x_grid, 1.5)
R$x = x_grid
ggplot(R, aes(x = x, y = density)) +
  geom_area(fill = I('grey')) +
  geom_line(aes(x = x, y = tdensity)) +
  labs(title = "N(0,1) (shaded) and t-distribution (1.5 dof)")
```



上圖也不是 qqplot. 上圖只是普通的 Gaussian 分佈函數和 t 分佈函數.

We'll see now another example of the qqplot. Where we are going to show the quantiles of the Gaussian distribution on one axis and the t-distribution on another axis. The t-distribution has much heavier tails than the normal distribution. It has a fundamentally different shape in that the tails decay polynomially, rather than exponentially. This will give us a non-linear qqplot shape. So first we're going to create a grid between the values -6 and 6 having 200 equally spaced points. Then we're going to computer the Gaussian density or the density function of the normal distributions with parameters 0 and 1 at these grid points. Next we're going to call ggplot function and we're going to tell it to draw two different graphs. The two graphs are supposed to illustrate the Gaussian distribution or the normal distribution and the t-distribution and overlay them one on top of the other. In the case of the Gaussian distribution, we want it to be displayed using filled area, so we call the fill geometry. We're going to tell it to use the color grey. In the case of the t-density, we want to use a line graph, so we call the line geometry. Finally, we use the label annotation here to describe the graph. Let's see the graph that we get as a result. We see here the two distributions, the t-distribution using the solid line, the Gaussian or normal distribution using the shaded area. Both seem similar at first glance, having a bell-shaped curve. But if you zoom out, you'll see that the tails of the t-distribution decay much slower than the tails of the Gaussian distribution no matter what is the variance of the Gaussian.


```
x_grid = seq(-6, 6, length.out = 200)
R = data.frame(density = dnorm(x_grid, 0, 1))
R$samples = rnorm(200, 0, 1)
pm = list(df = 1.5)
ggplot(R, aes(sample = samples)) +
  stat_qq(distribution = qt, dparams = pm)
```



We're going to move next to display the qqplot of samples from the normal or Gaussian distribution versus the theoretical quantiles of the t-distribution. We create 200 samples, this is going to be our data set from the normal distributions with parameters 0 and 1. And we're going to call the function ggplot with the data set that we just sampled from the normal distribution, telling it to use a qqplot and drawing the quantiles of the dataset against the quantiles from the distribution, denoted here by qt which is the t-distribution with the specific parameter. In this case, df corresponds to degrees of freedom 1.5. That's the parameter of the t-distribution. Let's see what graph we get as a result of this code. We see here, a non-linear qqplot, the x-axis describes the quantiles of the theoretical t-distribution. The y-axis corresponds to the quantiles of the dataset drawn from a normal distribution. The shape of the graph is an s shape. If you'll go back to the quiz, you will see that the presence of the s shape is consistent with the case of the t-distribution and the Gaussian distribution having fundamentally different shape with the t-distribution having much heavier tails.

Data Visualization
Lesson Summary

We learned a variety of different ways to visualize data.

Next:

- How to bring a dataset to a form that is easy to visualize

61. You should now be comfortable with graphs and plots, as well as some of the data sets available in r. In the next lesson, we will learn how to process data and bring it to a form that is easy to visualize using the techniques that we just learned.