# Unofficial Kahn Academy R Supplement: Stem and Leaf Plots / Line Graphs

You may notice that we are skipping over the "comparing features of distributions" lesson. This is because that lesson is very conceptual and it is arguable whether an R supplement would be helpful, so we'll go ahead to stem and leaf plots and line graphs.

At this level, many of the commands remain relatively basic. For example, for a stem-and-leaf graph you simply use the *stem* command.

```
stem(mtcars$mpg)
```

```
##
##   The decimal point is at the |
##
##   10 | 44
##   12 | 3
##   14 | 3702258
##   16 | 438
##   18 | 17227
##   20 | 00445
##   22 | 88
##   24 | 4
##   26 | 03
##   28 |
##   30 | 44
##   32 | 49
```

As you may notice, the graphic is a little different in R than it was in KA. In KA the | symbol represents the tens place, whereas in R the | symbol represents the decimal point (so, 10.44 mpg instead of 104 mpg, etc.). Unfortunately, I do not believe there's a way to easily modify the stem command to make it look like the KA example, but it's interpretable either way.

Line charts in R can be either very basic or they can have lots of bells and whistles. We'll discuss a basic line chart here, but if you ever want to use a line chart in, say, a professional presentation or a published paper, you'll want the bells and whistles. In R there are two main ways to create graphics: with the built-in code or code that you need to install the *ggplot2* package to use (as a reminder, to do so you simply need to type install.package("ggplot2") followed by library(ggplot2). The *ggplot2* package has its own online fora and instructional books, and makes it easier to make really stunning graphics. There's a temptation to just use the built-in code, but doing so can be very onerous and require a lot of lines of code; ggplot2 drastically simplifies the making of sharp looking charts and graphs.

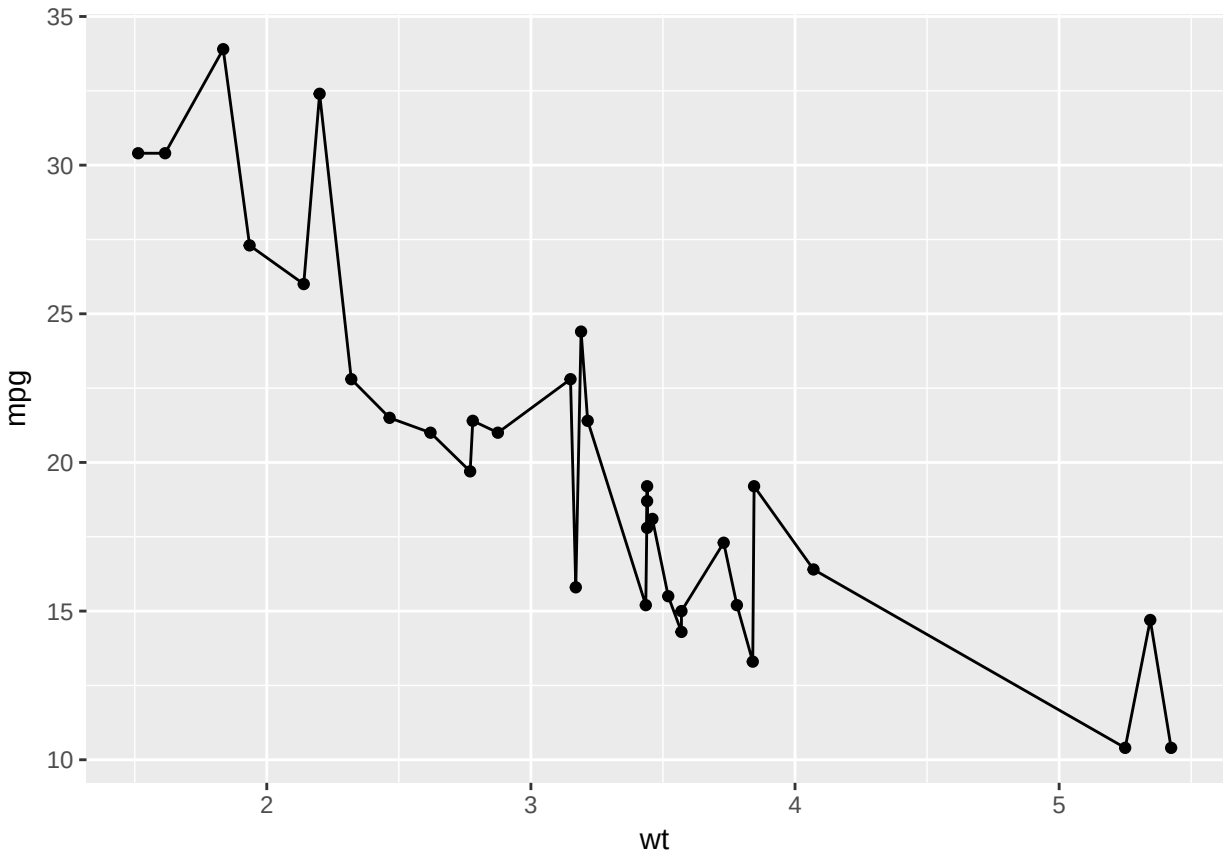
For these reasons, here we will introduce basic ggplot line graphs, but once again will not go too far. Further instruction about ggplot2 is widely available online.

In the case of a line graph; it seems reasonable that the miles per gallon that a car gets has a relationship with the weight of the car, with heavier cars requiring more gas and therefore having fewer miles per gallon. Let's plot this relationship on a line chart using ggplot2.

ggplot2 has its own "grammar," or logic of doing things that simplifies the making of graphs. On a very basic level, ggplot2 lets you input the basic graphs and relationships that you want to do, then you can add the bells and whistles one step at a time. For example, the code below shows a basic line graph. You start by specifying which dataset you are drawing from (in this case, **mtcars**), then you insert a comma, space, then "aes"" (which stands for "aesthetic"). It's within this "aes" command within a command that you put your x and y axes. After closing that subcommand you simply use an addition sign to continue to add bells and

whistles. For a basic graph, you need to add *geom_line()* and *geom_point* to have the line and points in the chart.

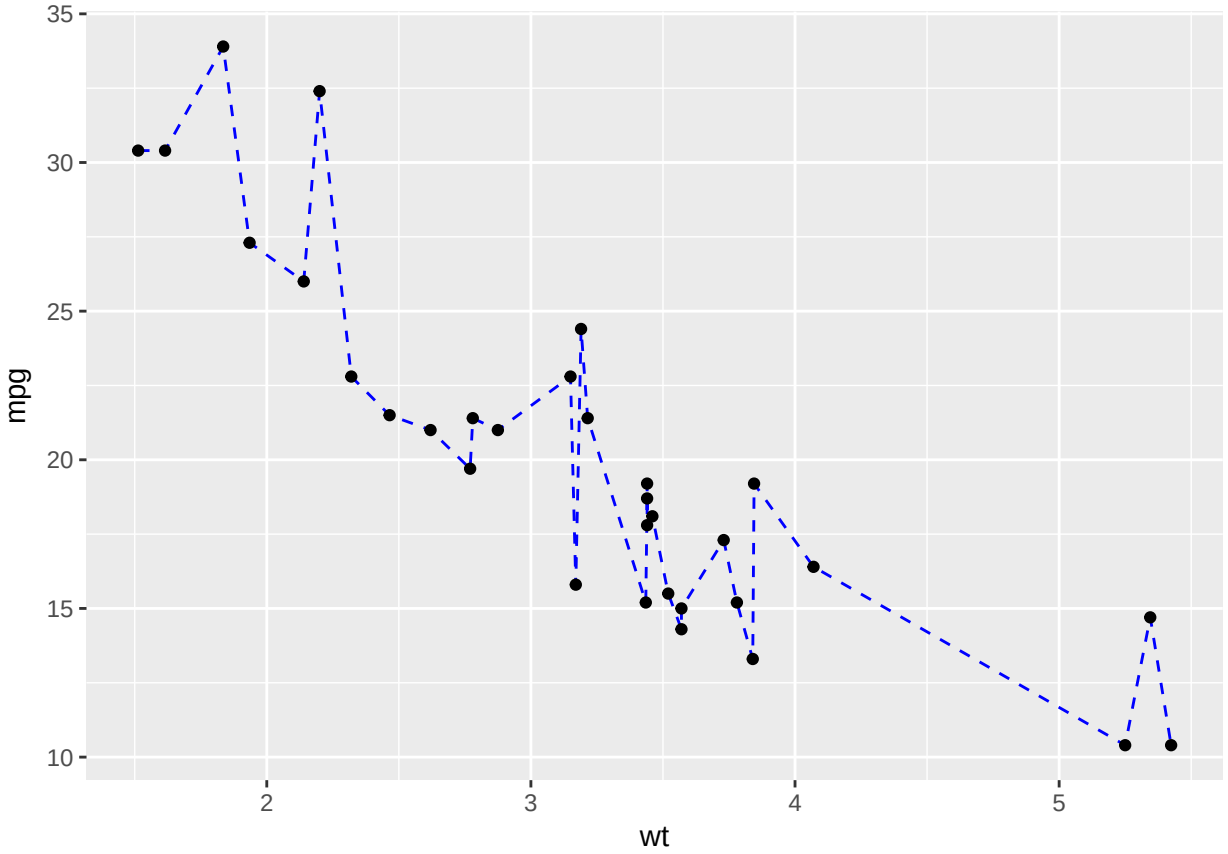
```
#install.packages("ggplot2")
library(ggplot2)
ggplot(data=mtcars, aes(x=wt, y=mpg)) +
  geom_line()+
  geom_point()
```




As you can see, there is a general downward trend, even though across small lengths the relationship may be reversed. In more advanced statistics there are ways to "smooth" this line to look at a general trend that gets past the bumpy noise.

Graphically, you can customize your lines by simply adding information in the geom_line and geom_point subcommands. For example, if you want your line to be dashed, you just have do the same command and type *linetype = "dashed"* in the geom_line() subcommand. If you want dashed and blue lines, you simply add a comma within the geom_line subcommand and add *color="blue"*.

```
ggplot(data=mtcars, aes(x=wt, y=mpg)) +
  geom_line(linetype= "dashed", color= "blue")+
  geom_point()
```




Again, this tutorial will not go through all the different ways you can spruce up your graph, but hopefully in this example you have a sense of how you can use ggplot to easily create good line (and other) charts.