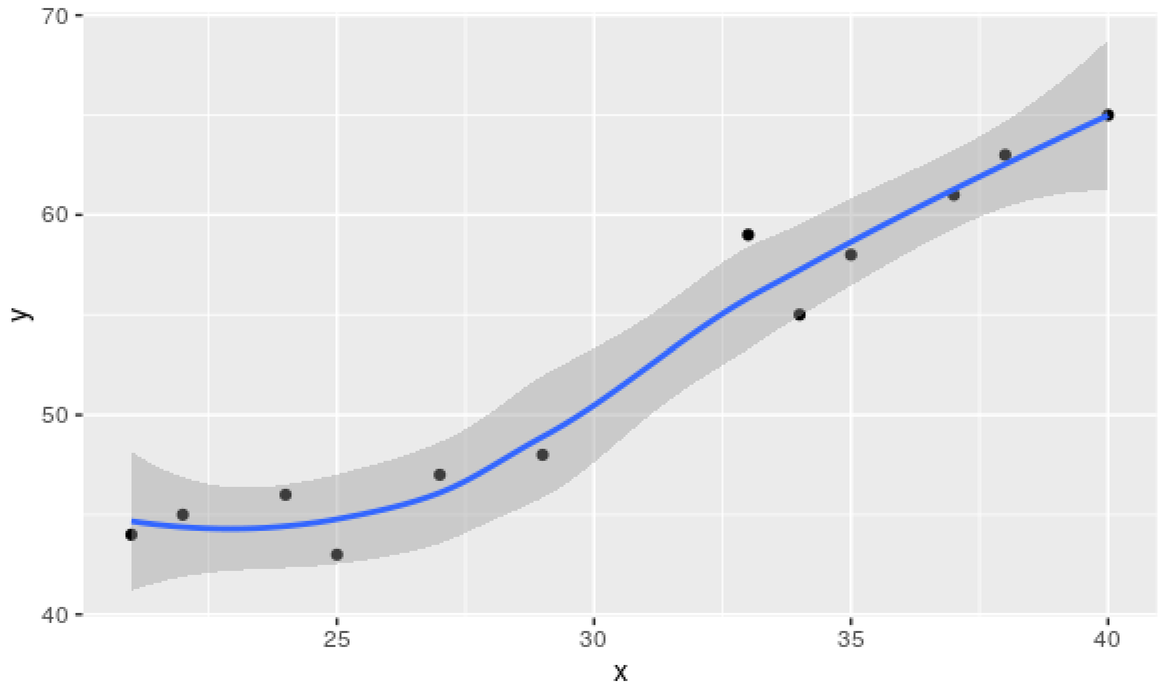
Smoothing

In this reading, you will learn about smoothing in ggplot2 and how it can be used to make your data visualizations in R clearer and easier to follow. Sometimes it can be hard to understand trends in your data from scatter plots alone. **Smoothing** enables the detection of a data trend even when you can't easily notice a trend from the plotted data points. Ggplot2's smoothing functionality is helpful because it adds a **smoothing line** as another layer to a plot; the smoothing line helps the data to make sense to a casual observer.

Example code

```
ggplot(data, aes(x=distance,
y= dep_delay)) +
    geom_point() +
    geom_smooth()
```

The example code creates a plot with a trend line similar to the blue line below.



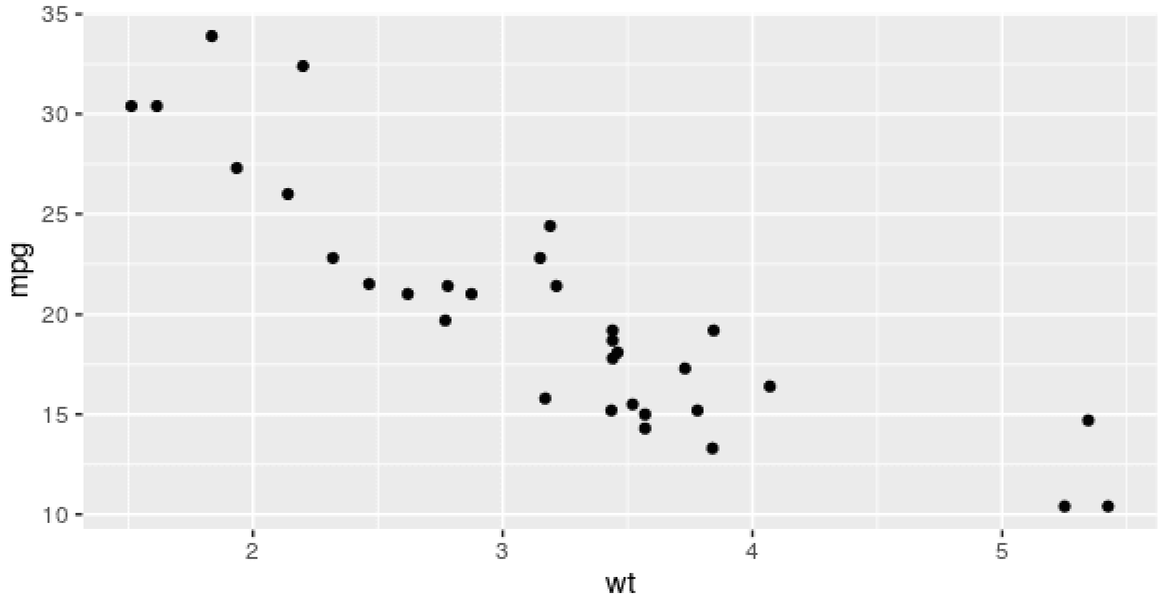
Two types of smoothing

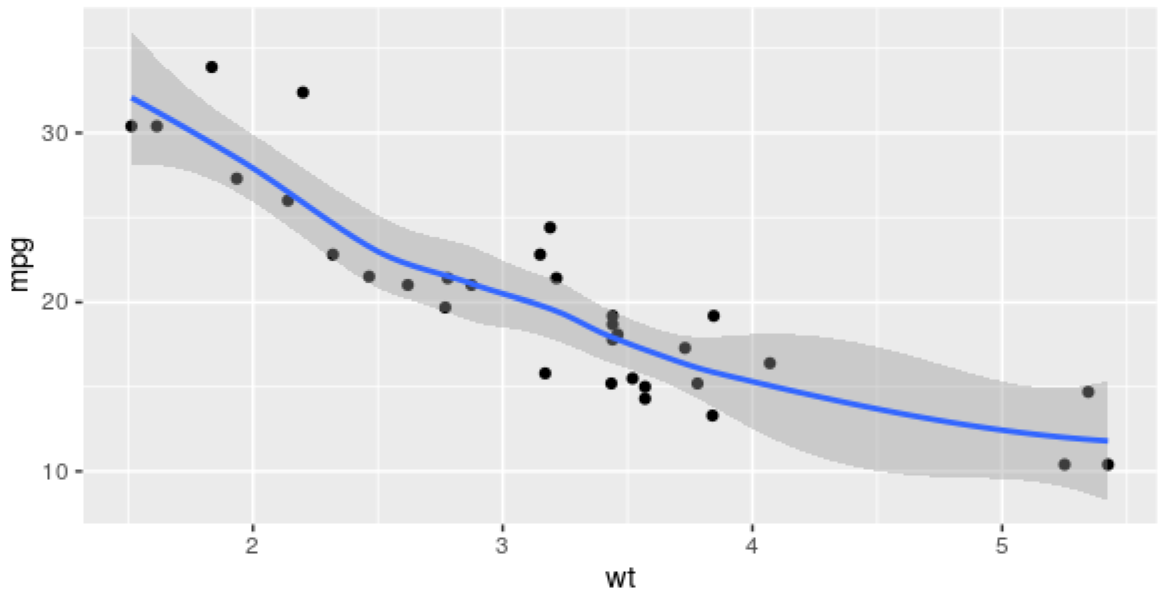
Type of smoothing	**Description**	**Example code**

Loess smoothing	The loess smoothing process is best for smoothing plots with less than 1000 points.	`ggplot(data, aes(x=, y=))+`
		`  geom_point() +`
		`  geom_smooth(method="loess")`

Gam smoothing	Gam smoothing, or generalized additive model smoothing, is useful for smoothing plots with a large number of points.	`ggplot(data, aes(x=, y=)) +`
		`  geom_point() +`
		`  geom_smooth(method="gam",`
		`formula = y ~s(x))`

The smoothing functionality in ggplot2 helps make data plots more readable, so you are better able to recognize data trends and make key insights. The first plot below is the data before smoothing, and the second plot below is the same data after smoothing.





Filtering and plots

By this point you have likely downloaded at least a few packages into your R library. The tools in some of these packages can actually be combined and used together to become even more useful. This reading will share a few resources that will teach you how to use the filter function from **dplyr** to make the plots you create with **ggplot2** easier to read.

Example of filtering data for plotting

Filtering your data before you plot it allows you to focus on specific subsets of your data and gain more targeted insights. To do this, just include the dplyr filter() function in your ggplot syntax.

Example code

```
data %>%
    filter(variable1 == "DS") %>%
    ggplot(aes(x = weight, y = variable2, colour = variable1)) +
    geom_point(alpha = 0.3,  position = position_jitter()) +
stat_smooth(method = "lm")
```