In this video, we'll create a basic scatterplot using ggplot.

Let's start by reading in our data.

We'll be using the same data set we used during week one, WHO.csv.

So let's call it WHO and use the read.csv function to read in the data file WHO.csv.

Make sure you're in the directory containing this file first.

Now, let's take a look at the structure of the data using the str function.

We can see that we have 194 observations, or countries, and 13 different variables-- the name of the country, the region the country's in, the population in thousands, the percentage of the population under 15 or over 60, the fertility rate or average number of children per woman, the life expectancy in years, the child mortality rate, which is the number of children who die by age five per 1,000 births, the number of cellular subscribers per 100 population, the literacy rate among adults older than 15, the gross national income per capita, the percentage of male children enrolled in primary school, and the percentage of female children enrolled in primary school.

In week one, the very first plot we made in R was a scatterplot of fertility rate versus gross national income.

Let's make this plot again, just like we did in week one.

So we'll use the plot function and give as the first variable WHO$GNI, and then give as the second variable, WHO$FertilityRate.

This plot shows us that a higher fertility rate is correlated with a lower income.

Now, let's redo this scatterplot, but this time using ggplot.

We'll see how ggplot can be used to make more visually appealing and complex scatterplots.

First, we need to install and load the ggplot2 package.

So first type install.packages("ggplot2").

When the CRAN mirror window pops up, make sure to pick a location near you.

Then, as soon as the package is done installing and you're back at the blinking cursor, load the package with the library function.

Now, remember we need at least three things to create a plot using ggplot-- data, an aesthetic mapping of variables in the data frame to visual output, and a geometric object.

So first, let's create the ggplot object with the data and the aesthetic mapping.

We'll save it to the variable scatterplot, and then use the ggplot function, where the first argument is the name of our data set, WHO, which specifies the data to use, and the second argument is the aesthetic mapping, aes.

In parentheses, we have to decide what we want on the x-axis and what we want on the y-axis.

We want the x-axis to be GNI, and we want the y-axis to be FertilityRate.

Go ahead and close both sets of parentheses, and hit Enter.

Now, we need to tell ggplot what geometric objects to put in the plot.

We could use bars, lines, points, or something else.

This is a big difference between ggplot and regular plotting in R. You can build different types of graphs by using the same ggplot object.

There's no need to learn one function for bar graphs, a completely different function for line graphs, etc.

So first, let's just create a straightforward scatterplot.

So the geometry we want to add is points.

We can do this by typing the name of our ggplot object, scatterplot, and then adding the function, geom_point().

If you hit Enter, you should see a new plot in the Graphics window that looks similar to our original plot, but there are already a few nice improvements.

One is that we don't have the data set name with a dollar sign in front of the label on each axis, just the variable name.

Another is that we have these nice grid lines in the background and solid points that pop out from the background.

We could have made a line graph just as easily by changing point to line.

So in your R console, hit the up arrow, and then just delete "point" and type "line" and hit Enter.

Now, you can see a line graph in the Graphics window.

However, a line doesn't really make sense for this particular plot, so let's switch back to our points, just by hitting the up arrow twice and hitting Enter.

In addition to specifying that the geometry we want is points, we can add other options, like the color, shape, and size of the points.

Let's redo our plot with blue triangles instead of circles.

To do that, go ahead and hit the up arrow in your R console, and then in the empty parentheses for geom_point, we're going to specify some properties of the points.

We want the color to be equal to "blue", the size to equal 3-- we'll make the points a little bigger -- and the shape equals 17.

This is the shape number corresponding to triangles.

If you hit Enter, you should now see in your plot blue triangles instead of black dots.

Let's try another option.

Hit the up arrow again, and change "blue" to "darkred", and change shape to 8.

Now, you should see dark red stars.

There are many different colors and shapes that you can specify.

We've provided some information in the text below this video.

Now, let's add a title to the plot.

You can do that by hitting the up arrow, and then at the very end of everything, add ggtitle, and then in parentheses in quotes, the title you want to give your plot.

In our case, we'll call it "Fertility Rate vs. Gross National Income".

If you look at your plot again, you should now see that it has a nice title at the top.

Now, let's save our plot to a file.

We can do this by first saving our plot to a variable.

So in your R console, hit the up arrow, and scroll to the beginning of the line.

Before scatterplot, type fertilityGNIplot = and then everything else.

This will save our scatterplot to the variable, fertilityGNIplot.

Now, let's create a file we want to save our plot to.

We can do that with the PDF function.

And then in parentheses and quotes, type the name you want your file to have.

We'll call it MyPlot.pdf.

Now, let's just print our plot to that file with the print function -- so print(fertilityGNIplot).

And lastly, we just have to type dev.off() to close the file.

Now, if you look at the folder where WHO.csv is, you should see another file called MyPlot.pdf, containing the plot we made.

In the next video, we'll see how to create more advanced scatterplots using ggplot.