```
  Course: Exploratory_Data_Analysis
  Lesson: GGPlot2_Part1

- Class: text
  Output: "GGPlot2_Part1. (Slides for this and other Data Science courses may be found at github https://github.com/DataScienceSpecialization/courses/. If you care to use them, they must be downloaded as a zip file and viewed locally. This lesson corresponds to 04_ExploratoryAnalysis/ggplot2.)"


- Class: text
  Output:  In another lesson, we gave you an overview of the three plotting systems in R. In this lesson we'll focus on the third and newest plotting system in R, ggplot2. As we did with the other two systems, we'll focus on creating graphics on the screen device rather than another graphics device.

- Class: text
  Output:  The ggplot2 package is an add-on package available from CRAN via install.packages(). (Don't worry, we've installed it for you already.) It is an implementation of The Grammar of Graphics, an abstract concept (as well as book) authored and invented by Leland Wilkinson and implemented by Hadley Wickham while he was a graduate student at Iowa State. The web site http://ggplot2.org provides complete documentation.

- Class: text
  Output: A grammar of graphics represents an abstraction of graphics, that is,  a theory of graphics which conceptualizes basic pieces from which you can build new graphics and graphical objects. The  goal of the grammar is to “Shorten the distance from mind to page”. From Hadley Wickham's book  we learn that

- Class: text
  Output:  The ggplot2 package "is composed of a set of independent components that can be composed in many different ways. ... you can create new graphics that are precisely tailored for your problem." These components include aesthetics which are attributes such as colour, shape, and  size, and geometric objects or geoms such as points, lines, and bars.

- Class: text
  Output: Before we delve into details, let's review the other 2 plotting systems.

- Class: mult_question
  Output: Recall what you know about R's base plotting system. Which of the following does NOT apply to it?
  AnswerChoices: Start with plot (or similar) function; Use annotation functions to add/modify (text, lines, points, axis); It is convenient and mirrors how we think of building plots and analyzing data; Can easily go back once the plot has started (e.g., to adjust margins or correct a typo)
  CorrectAnswer: Can easily go back once the plot has started (e.g., to adjust margins or correct a typo)
  AnswerTests: omnitest(correctVal='Can easily go back once the plot has started (e.g., to adjust margins or correct a typo)')
  Hint: Which choice is the only one which looks backward?


- Class: mult_question
  Output: Recall what you know about R's lattice plotting system. Which of the following does NOT apply to it?
  AnswerChoices:  Plots are created with a single function call (xyplot, bwplot, etc.); Most useful for conditioning types of plots and putting many panels on one plot; Margins and spacing are set automatically because entire plot is specified at once; Can always add to the plot once it is created
  CorrectAnswer: Can always add to the plot once it is created
  AnswerTests: omnitest(correctVal='Can always add to the plot once it is created')
  Hint: Which choice is the only one which is inconsistent with the other three?

- Class: mult_question
  Output: If we told you that ggplot2 combines the best of base and lattice, that would mean it ...?
  AnswerChoices: Automatically deals with spacings, text, titles but also allows you to
```

```
    annotate; Like lattice it allows for multipanels but more easily and intuitively; Its default mode makes many choices for you (but you can customize!); All of the others
    CorrectAnswer: All of the others
    AnswerTests: omnitest(correctVal='All of the others')
    Hint: Which choice is the only one that encompasses the other three?

  - Class: text
    Output: Yes, ggplot2 combines the best of base and lattice. It allows for multipanel (conditioning) plots (as lattice does) but also post facto annotation (as base does), so you can add titles and labels. It uses the low-level grid package (which comes with R) to draw the graphics. As part of its grammar philosophy, ggplot2 plots are composed of aesthetics (attributes such as size, shape, color) and geoms (points, lines, and bars), the geometric objects you see on the plot.

  - Class: text
    Output: The ggplot2 package has 2 workhorse functions. The more basic workhorse function is qplot, (think quick plot), which works like the plot function in the base graphics system. It can produce many types of plots (scatter, histograms, box and whisker) while hiding tedious details from the user. Similar to lattice functions, it looks for data in a data frame or parent environment.

  - Class: text
    Output: The more advanced workhorse function in the package is ggplot, which is  more flexible and can be customized for doing things qplot cannot do. In this lesson we'll focus on qplot.

  - Class: cmd_question
    Output: We'll start by showing how easy and versatile qplot is. First, let's look at some data which comes with the ggplot2 package. The mpg data frame contains fuel economy data for 38 models of cars manufactured in 1999 and 2008. Run the R command str with the argument mpg. This will give you an idea of what mpg contains.
    CorrectAnswer: str(mpg)
    AnswerTests: omnitest(correctExpr='str(mpg)')
    Hint: Type str(mpg) at the command prompt.

  - Class: cmd_question
    Output: We see that there are 234 points in the dataset concerning 11 different characteristics of the cars. Suppose we want to see if there's a correlation between engine displacement (displ) and highway miles per gallon (hwy). As we did with the plot function of the base system we could simply call  qplot with 3 arguments, the first two are the variables we want to examine and the third argument data is set equal to the name of the dataset which contains them (in this case, mpg). Try this now.
    CorrectAnswer: qplot(displ, hwy, data = mpg)
    AnswerTests: omnitest(correctExpr='qplot(displ, hwy, data = mpg)')
    Hint: Type qplot(displ, hwy, data = mpg) at the command prompt.

  - Class: cmd_question
    Output: A nice scatterplot done simply, right? All the labels are provided. The first argument is shown along the x-axis and the second along the y-axis. The negative trend (increasing displacement and lower gas mileage) is pretty clear. Now suppose we want to do the same plot but this time use different colors to distinguish between the 3 factors (subsets) of different types of drive (drv) in  the data (front-wheel, rear-wheel, and 4-wheel). Again, qplot makes this very easy. We'll just add what ggplot2 calls an aesthetic, a fourth argument, color, and set it equal to drv. Try this now. (Use the up arrow key to save some typing.)
    CorrectAnswer: qplot(displ, hwy, data = mpg, color = drv)
    AnswerTests: omnitest(correctExpr='qplot(displ, hwy, data = mpg, color = drv)')
    Hint: Type qplot(displ, hwy, data = mpg, color = drv) at the command prompt.

  - Class: text
    Output: Pretty cool, right? See the legend to the right which qplot helpfully supplied? The colors were automatically assigned by qplot so the legend decodes the colors for you. Notice that qplot automatically used dots or points to indicate the data. These points are geoms (geometric objects). We could have used a different aesthetic, for instance shape instead of color, to distinguish between the drive types.

  - Class: cmd_question
    Output: Now let's add a second geom to the default points. How about some smoothing function to produce trend lines, one for each color? Just add a fifth argument, geom,
    and using the R function c(), set it equal to the concatenation of the two strings "point" and "smooth". The first refers to the data points and second to the trend lines we want plotted. Try this now.
    CorrectAnswer: qplot(displ, hwy, data = mpg, color=drv, geom = c("point", "smooth"))
    AnswerTests: omnitest(correctExpr='qplot(displ, hwy, data = mpg, color=drv, geom = c("point", "smooth"))')
    Hint: Type qplot(displ, hwy, data = mpg, color=drv, geom = c("point", "smooth")) at the command prompt.

  - Class: text
    Output: Notice the gray areas surrounding each trend lines. These indicate the 95% confidence intervals for the lines.

  - Class: cmd_question
    Output: Before we leave qplot's scatterplotting ability, call qplot again, this time with 3 arguments. The first is y set equal to hwy, the second is data set equal to mpg, and the third is color set equal to drv. Try this now.
    CorrectAnswer: qplot(y=hwy, data = mpg, color = drv)
    AnswerTests: omnitest(correctExpr='qplot(y=hwy, data = mpg, color = drv)')
    Hint: Type qplot(y=hwy, data = mpg, color = drv) at the command prompt.

  - Class: cmd_question
    Output: What's this plot showing? We see the x-axis ranges from 0 to 250 and we remember that we had 234 data points in our set, so we can infer that each point in the plot represents one of the hwy values (indicated by the y-axis). We've created the vector myhigh for you which contains the hwy data from the mpg dataset. Look at myhigh now.
    CorrectAnswer: myhigh
    AnswerTests: omnitest(correctExpr='myhigh')
    Hint: Type myhigh at the command prompt.

  - Class: text
    Output: Comparing the values of myhigh with the plot, we see the first entries in the vector (29, 29, 31, 30,...) correspond to the leftmost points in the the plot (in order), and the last entries in myhigh (28, 29, 26, 26, 26) correspond to the rightmost plotted points. So, specifying the y parameter only, without an x argument, plots the values of the y argument in the order in which they occur in the data.

  - Class: cmd_question
    Output: The all-purpose qplot can also create box and whisker plots.  Call qplot now with 4 arguments. First specify the variable by which you'll split the data, in this case drv, then specify the variable which you want to examine, in this case hwy. The third argument is data (set equal to mpg), and the fourth, the geom, set equal to the string "boxplot"
    CorrectAnswer: qplot(drv,hwy,data=mpg,geom="boxplot")
    AnswerTests: omnitest(correctExpr='qplot(drv,hwy,data=mpg,geom="boxplot")')
    Hint: Type qplot(drv,hwy,data=mpg,geom="boxplot") at the command prompt.

  - Class: cmd_question
    Output: We see 3 boxes, one for each drive. Now to impress you, call qplot with 5 arguments. The first 4 are just as you used previously, (drv, hwy, data set equal to mpg, and geom set equal to the string "boxplot"). Now add a fifth argument, color, equal to manufacturer.
    CorrectAnswer: qplot(drv,hwy,data=mpg,geom="boxplot",color=manufacturer)
    AnswerTests: omnitest(correctExpr='qplot(drv,hwy,data=mpg,geom="boxplot",color=manufacturer)')
    Hint: Type qplot(drv,hwy,data=mpg,geom="boxplot",color=manufacturer) at the command prompt.

  - Class: text
    Output: It's a little squished but we just wanted to illustrate qplot's capabilities. Notice that there are still 3 regions of the plot (determined by the factor drv). Each is subdivided into several boxes depicting different manufacturers.

  - Class: cmd_question
    Output: Now, on to histograms. These display frequency counts for a single variable. Let's start with an easy one. Call qplot with 3 arguments. First specify the variable for which you want the frequency count, in this case hwy, then specify the data (set equal to mpg), and finally, the aesthetic, fill, set equal to drv. Instead of a plain
    old histogram, this will again use colors to distinguish the 3 different drive factors.
    CorrectAnswer: qplot(hwy, data = mpg, fill = drv)
    AnswerTests: omnitest(correctExpr='qplot(hwy, data = mpg, fill = drv)')
    Hint: Type qplot(hwy, data = mpg, fill = drv) at the command prompt.

  - Class: text
    Output: See how qplot consistently uses the colors. Red (if 4-wheel drv is in the
    bin) is at the bottom of the bin, then green on top of it (if present), followed by
    blue (rear wheel drv). The color lets us see right away that 4-wheel drive vehicles
    in this dataset don't have gas mileages exceeding 30 miles per gallon.

  - Class: text
    Output: It's cool that qplot can do this  so easily, but some people may find this
    multi-color histogram hard to interpret. Instead of using colors to distinguish
    between the drive factors let's use facets or panels. (That's what lattice called
    them.) This just means we'll split the data into 3 subsets (according to drive) and
    make 3 smaller individual plots of each subset in one plot (and with one call to
    qplot).

  - Class: text
    Output: Remember that with base plot we had to do each subplot individually. The
    lattice system made plotting  conditioning plots  easier. Let's see how easy it is
    with qplot.

  - Class: cmd_question
    Output:  We'll do two plots, a scatterplot and then a histogram, each with 3 facets.
    For the scatterplot, call qplot with 4 arguments. The first two are displ and hwy and
    the third is the argument data set equal to mpg. The fourth is the argument facets
    which will be set equal to the expression . ~ drv which is ggplot2's shorthand for
    number of rows (to the left of the ~) and number of columns (to the right of the ~).
    Here the . indicates a single row and drv implies 3, since there are 3 distinct drive
    factors. Try this now.
    CorrectAnswer: qplot(displ, hwy, data = mpg, facets = . ~ drv)
    AnswerTests: omnitest(correctExpr='qplot(displ, hwy, data = mpg, facets = . ~ drv)')
    Hint: Type qplot(displ, hwy, data = mpg, facets = . ~ drv) at the command prompt.

  - Class: text
    Output: The result is a 1 by 3 array of plots. Note how each is labeled at the top
    with the factor label (4,f, or r). This shows us more detailed information than the
    histogram. We see the relationship between displacement and highway mileage for each
    of the 3 drive factors.

  - Class: cmd_question
    Output:  Now we'll do a histogram, again calling qplot with 4 arguments. This time,
    since we need only one variable for a histogram, the first is hwy and the second is
    the argument data set equal to mpg. The third is the argument facets which we'll set
    equal to the expression drv ~ . . This will give us a different arrangement of the
    facets. The fourth argument is binwidth. Set this equal to 2. Try this now.
    CorrectAnswer: qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2)
    AnswerTests: omnitest(correctExpr='qplot(hwy, data = mpg, facets = drv ~ ., binwidth
    = 2)')
    Hint: Type qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2) at the command
    prompt.

  - Class: mult_question
    Output: The facets argument, drv ~ ., resulted in what arrangement of facets?
    AnswerChoices:  1 by 3; 3 by 1; 2 by 2; huh?
    CorrectAnswer:  3 by 1
    AnswerTests: omnitest(correctVal='3 by 1')
    Hint: How many row? How many columns?


  - Class: text
    Output:  Pretty good, right? Not too difficult either. Let's review what we learned!

  - Class: mult_question
    Output: Which of the following is a basic workhorse function of ggplot2?
    AnswerChoices:  hist; xyplot; scatterplot; gplot; qplot
    CorrectAnswer:  qplot
    AnswerTests: omnitest(correctVal='qplot')
    Hint: Which function did we invoke the most in this lesson?

  - Class: mult_question
    Output:  Which types of plot does qplot plot?
    AnswerChoices:  histograms; scatterplots; box and whisker plots; all of the others
    CorrectAnswer: all of the others
    AnswerTests: omnitest(correctVal='all of the others')
    Hint: That qplot is amazing! It seems to do everything!

  - Class: mult_question
    Output:  What does the gg in ggplot2 stand for?
    AnswerChoices:  good grief; grammar of graphics; goto graphics; good graphics
    CorrectAnswer: grammar of graphics
    AnswerTests: omnitest(correctVal='grammar of graphics')
    Hint: Think of building blocks and components.

  - Class: mult_question
    Output:  True or False? The geom argument takes a string for a value.
    AnswerChoices:  True; False
    CorrectAnswer: True
    AnswerTests: omnitest(correctVal='True')
    Hint: Recall our examples, for instance, geom="density".

  - Class: mult_question
    Output:  True or False? The data argument takes a string for a value.
    AnswerChoices:  True; False
    CorrectAnswer: True
    AnswerTests: omnitest(correctVal='False')
    Hint: Recall our examples. Did we ever put the dataset name in quotation marks?

  - Class: mult_question
    Output:  True or False? The binwidth argument takes a string for a value.
    AnswerChoices:  True; False
    CorrectAnswer: False
    AnswerTests: omnitest(correctVal='False')
    Hint: Recall our examples, for instance, binwidth=18497/30.

  - Class: mult_question
    Output:  True or False? The user must specify x- and y-axis labels when using qplot.
    AnswerChoices:  True; False
    CorrectAnswer: False
    AnswerTests: omnitest(correctVal='False')
    Hint: Recall our examples when we saw labels that we didn't specify.

  - Class: text
    Output: Congrats! You've finished plot 1 of  ggplot2. In the next lesson the plot thickens.

  - Class: mult_question
    Output: "Would you like to receive credit for completing this course on
      Coursera.org?"
    CorrectAnswer: NULL
    AnswerChoices: Yes;No
    AnswerTests: coursera_on_demand()
    Hint: ""
```