```
  Course: Exploratory_Data_Analysis
  Lesson: GGPlot2_Extras

- Class: text
  Output: "GGPlot2_Extras. (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 this lesson we'll go through a few more qplot examples using diamond data
  which comes with the ggplot2 package. This data is a little more complicated than the
  mpg data and it contains information on various characteristics of diamonds.


- Class: cmd_question
  Output:  Run the R command str with the argument diamonds to see what the data looks
  like.
  CorrectAnswer: str(diamonds)
  AnswerTests: omnitest(correctExpr='str(diamonds)')
  Hint: Type str(diamonds) at the command prompt.

- Class: mult_question
  Output: From the output, how many characteristics of diamonds do you think this data
  contains?
  AnswerChoices:  10; 53940; 5394; 53950
  CorrectAnswer:  10
  AnswerTests: omnitest(correctVal='10')
  Hint: The output says there are 53940 observations of 10 variables. This is followed
  by a 10-long list of characteristics (carat, cut, color, etc.) that can apply to
  diamonds.

- Class: mult_question
  Output: From the output of str, how many  diamonds are characterized in this dataset?
  AnswerChoices:  10; 53940; 5394; 53950
  CorrectAnswer:  53940
  AnswerTests: omnitest(correctVal='53940')
  Hint: The output says there are 53940 observations of 10 variables. This is followed
  by a 10-long list of characteristics (carat, cut, color, etc.) that can apply to
  diamonds.

- Class: cmd_question
  Output:  Now let's plot a histogram of the price of the 53940 diamonds in this
  dataset. Recall that a histogram requires only one variable of the data, so run the R
  command qplot with the first argument price and the argument data set equal to
  diamonds. This will show the frequency of different diamond prices.
  CorrectAnswer: qplot(price,data=diamonds)
  AnswerTests: omnitest(correctExpr='qplot(price,data=diamonds)')
  Hint: Type qplot(price,data=diamonds) at the command prompt.

- Class: cmd_question
  Output:  Not only do you get a histogram, but you also get a message about the
  binwidth defaulting to range/30. Recall that range refers to the spread or dispersion
  of the data, in this case price of diamonds. Run  the R command range now with
  diamonds$price as its argument.
  CorrectAnswer: range(diamonds$price)
  AnswerTests: omnitest(correctExpr='range(diamonds$price)')
  Hint: Type range(diamonds$price) at the command prompt.

- Class: text
  Output: We see that range returned the minimum and maximum prices, so the diamonds
  vary in price from $326 to $18823. We've done the arithmetic for you, the range
  (difference between these two numbers) is $18497.

- Class: cmd_question
  Output:  Rerun qplot now with 3 arguments. The first is price, the second is data set
  equal to diamonds, and the third is binwidth set equal to 18497/30). (Use the up
  arrow to save yourself some typing.) See if the plot looks familiar.
```

```
    CorrectAnswer: qplot(price,data=diamonds,binwidth=18497/30)
    AnswerTests: omnitest(correctExpr='qplot(price,data=diamonds,binwidth=18497/30)')
    Hint: Type qplot(price,data=diamonds,binwidth=18497/30) at the command prompt.

- Class: text
    Output: No more messages in red, but a histogram almost identical to the previous
    one! If you typed 18497/30 at the command line you would get the result 616.5667.
    This means that the height of each bin tells you how many diamonds have a price
    between x and x+617 where x is the left edge of the bin.

- Class: cmd_question
    Output:  We've created a vector containing integers that are multiples of 617 for
    you. It's called brk. Look at it now.
    CorrectAnswer: brk
    AnswerTests: omnitest(correctExpr='brk')
    Hint: Type brk at the command prompt.

- Class: cmd_question
    Output:  We've also created a vector containing the number of diamonds with  prices
    between each pair of adjacent entries of brk. For instance, the first count is the
    number of diamonds with prices between 0 and $617, and the second is the number of
    diamonds with prices between $617 and $1234. Look at the vector named counts now.
    CorrectAnswer: counts
    AnswerTests: omnitest(correctExpr='counts')
    Hint: Type counts at the command prompt.

- Class: text
    Output: See how it matches the histogram you just plotted? So, qplot really works!

- Class: cmd_question
    Output:  You're probably sick of it  but rerun qplot again, this time with 4
    arguments. The first 3 are the same as the last qplot command you just ran  (price,
    data set equal to diamonds, and binwidth set equal to 18497/30). (Use the up arrow to
    save yourself some typing.) The fourth argument is fill set equal to cut. The shape
    of the histogram will be familiar, but it will be more colorful.
    CorrectAnswer: qplot(price,data=diamonds,binwidth=18497/30,fill=cut)
    AnswerTests:
    omnitest(correctExpr='qplot(price,data=diamonds,binwidth=18497/30,fill=cut)')
    Hint: Type qplot(price,data=diamonds,binwidth=18497/30,fill=cut) at the command prompt.

- Class: text
    Output: This shows how the counts within each price grouping (bin) are distributed
    among the different cuts of diamonds. Notice how qplot displays these distributions
    relative to the cut legend on the right. The fair cut diamonds are at the bottom of
    each bin, the good cuts are above them, then the very good above them, until the
    ideal cuts are at the top of each bin. You can quickly see from this display that
    there are very few fair cut diamonds priced above $5000.

- Class: cmd_question
    Output:  Now we'll replot the histogram as a density function which will show the
    proportion of diamonds in each bin. This means that the shape will be similar but the
    scale on the y-axis will be different since, by definition, the density function is
    nonnegative everywhere, and the area under the curve is one. To do this, simply call
    qplot with 3 arguments. The first 2 are  price and data (set equal to diamonds). The
    third is geom which should be set equal to the string "density". Try this now.
    CorrectAnswer: qplot(price,data=diamonds,geom="density")
    AnswerTests: omnitest(correctExpr='qplot(price,data=diamonds,geom="density")')
    Hint: Type qplot(price,data=diamonds,geom="density") at the command prompt.

- Class: text
    Output:  Notice that the shape is similar to that of the histogram we saw previously.
    The highest peak is close to 0 on the x-axis meaning that most of the diamonds in the
    dataset were inexpensive. In general, as prices increase (move right along the
    x-axis) the number of diamonds (at those prices) decrease. The exception to this is
    when the price is around $4000; there's a slight increase in frequency. Let's see if
    cut is responsible for this increase.

- Class: cmd_question
    Output: Rerun qplot, this time with 4 arguments. The first 2 are the usual, and the
```

```
    third is geom set equal to "density". The fourth is color set equal to cut. Try this
    now.
91    CorrectAnswer: qplot(price,data=diamonds,geom="density",color=cut)
92    AnswerTests:
    omnitest(correctExpr='qplot(price,data=diamonds,geom="density",color=cut)')
93    Hint: Type qplot(price,data=diamonds,geom="density",color=cut) at the command prompt.
94
95  - Class: text
96    Output: See how easily qplot did this? Four of the five cuts have 2 peaks, one at
    price $1000 and the other between $4000 and $5000. The exception is the Fair cut
    which has a single peak at $2500. This gives us a little more understanding of the
    histogram we saw before.
97
98  - Class: text
99    Output: Let's move on to scatterplots. For these we'll need to specify two variables
    from the diamond dataset.
100
101 - Class: cmd_question
102   Output:  Let's start with carat and price. Use these as the first 2 arguments of
    qplot. The third should be data set equal to the dataset. Try this now.
103   CorrectAnswer: qplot(carat,price,data=diamonds)
104   AnswerTests: omnitest(correctExpr=' qplot(carat,price,data=diamonds)')
105   Hint: Type  qplot(carat,price,data=diamonds) at the command prompt.
106
107 - Class: text
108   Output: We see the positive trend here, as the number of carats increases the price
    also goes up.
109
110 - Class: cmd_question
111   Output:  Now rerun the same command, except add a fourth parameter, shape, set equal
    to cut.
112   CorrectAnswer: qplot(carat,price,data=diamonds, shape=cut)
113   AnswerTests: omnitest(correctExpr='qplot(carat,price,data=diamonds, shape=cut)')
114   Hint: Type qplot(carat,price,data=diamonds, shape=cut) at the command prompt.
115
116 - Class: cmd_question
117   Output:  The same scatterplot appears, except the cuts of the diamonds are
    distinguished by different symbols. The legend at the right tells you which symbol is
    associated with each cut. These are small and hard to read, so rerun the same
    command, except this time instead of setting the argument shape equal to cut, set the
    argument color equal to cut.
118   CorrectAnswer: qplot(carat,price,data=diamonds, color=cut)
119   AnswerTests: omnitest(correctExpr='qplot(carat,price,data=diamonds, color=cut)')
120   Hint: Type qplot(carat,price,data=diamonds, color=cut) at the command prompt.
121
122 - Class: text
123   Output: That's easier to see! Now we'll close with two, more complicated scatterplot
    examples.
124
125 - Class: cmd_question
126   Output:  We'll rerun the plot you just did (carat,price,data=diamonds and color=cut)
    but add an additional parameter. Use geom_smooth with the method set equal to the
    string "lm".
127   CorrectAnswer: qplot(carat,price,data=diamonds, color=cut) + geom_smooth(method="lm")
128   AnswerTests: omnitest(correctExpr='qplot(carat,price,data=diamonds, color=cut) +
    geom_smooth(method="lm")')
129   Hint: Type qplot(carat,price,data=diamonds, color=cut) + geom_smooth(method="lm") at
    the command prompt.
130
131
132 - Class: text
133   Output: Again, we see the same scatterplot, but slightly more compressed and showing
    5 regression lines, one for each cut of diamonds. It might be hard to see, but around
    each line is a shadow showing the 95% confidence interval. We see, unsurprisingly,
    that the better the cut, the steeper (more positive) the slope of the lines.
134
135 - Class: cmd_question
136   Output:  Finally, let's rerun that plot you just did qplot(carat,price,data=diamonds,
    color=cut) + geom_smooth(method="lm") but add one (just one) more argument to qplot.
```

```
    The new argument is facets and it should be set equal to the formula .~cut. Recall
    that the facets argument indicates we want a multi-panel plot. The symbol to the left
    of the tilde indicates rows (in this case just one) and the symbol to the right of
    the tilde indicates columns (in this five, the number of cuts). Try this now.
137   CorrectAnswer: qplot(carat,price,data=diamonds, color=cut, facets=.~cut) +
    geom_smooth(method="lm")
138   AnswerTests: omnitest(correctExpr='qplot(carat,price,data=diamonds, color=cut,
    facets=.~cut) + geom_smooth(method="lm")')
139   Hint: Type qplot(carat,price,data=diamonds, color=cut, facets=.~cut) +
    geom_smooth(method="lm") at the command prompt.
140
141 - Class: text
142   Output:  Pretty good, right? Not too difficult either. Let's review what we learned!
143
144
145 - Class: mult_question
146   Output:  Which types of plot does qplot plot?
147   AnswerChoices:  histograms; scatterplots; box and whisker plots; all of the others
148   CorrectAnswer: all of the others
149   AnswerTests: omnitest(correctVal='all of the others')
150   Hint: That qplot is amazing! It seems to do everything!
151
152 - Class: mult_question
153   Output:  Any and all of the above choices work; qplot is just that good. What does
    the gg in ggplot2 stand for?
154   AnswerChoices:  good grief; grammar of graphics; goto graphics; good graphics
155   CorrectAnswer: grammar of graphics
156   AnswerTests: omnitest(correctVal='grammar of graphics')
157   Hint: Think of building blocks and components, also nouns, verbs, and diagramming
    sentences.
158
159 - Class: mult_question
160   Output:  True or False? The geom argument takes a string for a value.
161   AnswerChoices:  True; False
162   CorrectAnswer: True
163   AnswerTests: omnitest(correctVal='True')
164   Hint: Recall our examples, for instance, geom="density".
165
166 - Class: mult_question
167   Output:  True or False? The method argument takes a string for a value.
168   AnswerChoices:  True; False
169   CorrectAnswer: True
170   AnswerTests: omnitest(correctVal='True')
171   Hint: Recall our examples, for instance, method="lm".
172
173 - Class: mult_question
174   Output:  True or False? The binwidth argument takes a string for a value.
175   AnswerChoices:  True; False
176   CorrectAnswer: False
177   AnswerTests: omnitest(correctVal='False')
178   Hint: Recall our examples, for instance, binwidth=18497/30.
179
180 - Class: mult_question
181   Output:  True or False? The user must specify x- and y-axis labels when using qplot.
182   AnswerChoices:  True; False
183   CorrectAnswer: False
184   AnswerTests: omnitest(correctVal='False')
185   Hint: Recall our examples when we saw labels that we didn't specify.
186
187 - Class: text
188   Output: Now for some ggplots.
189
190 - Class: cmd_question
191   Output:  First create a graphical object g by assigning to it the output of a call to
    the function ggplot with 2 arguments. The first is the dataset diamonds and the
    second is a call to the function aes with 2 arguments, depth and price. Remember you
    won't see any result.
192   CorrectAnswer: g <- ggplot(diamonds,aes(depth,price))
193   AnswerTests: expr_creates_var("g"); omnitest(correctExpr='g <-
```

```
    ggplot(diamonds,aes(depth,price))')
194   Hint: Type g <- ggplot(diamonds,aes(depth,price)) at the command prompt.
195
196 - Class: cmd_question
197   Output:  Does g exist? Yes! Type summary with g as an argument to see what it holds.
198   CorrectAnswer: summary(g)
199   AnswerTests: omnitest(correctExpr='summary(g)')
200   Hint: Type summary(g) at the command prompt.
201
202 - Class: cmd_question
203   Output:  We see that g holds the entire dataset. Now suppose we want to see a
      scatterplot of the relationship. Add to g a call to the function geom_point with 1
      argument, alpha set equal to 1/3.
204   CorrectAnswer: g+geom_point(alpha=1/3)
205   AnswerTests: omnitest(correctExpr='g+geom_point(alpha=1/3)')
206   Hint: Type g+geom_point(alpha=1/3) at the command prompt.
207
208 - Class: text
209   Output:  That's somewhat interesting. We see that depth ranges from 43 to 79, but the
      densest distribution is around 60 to 65. Suppose we want to see if this relationship
      (between depth and price) is affected by cut or carat. We know cut is a factor with 5
      levels (Fair, Good, Very Good, Premium, and Ideal). But carat is  numeric and not a
      discrete factor. Can we do this?
210
211 - Class: text
212   Output: Of course! That's why we asked. R has a handy command, cut, which allows you
      to divide your data into sets and label each entry as belonging to one of the sets,
      in effect creating a new factor. First, we'll have to decide where to cut the data.
213
214 - Class: cmd_question
215   Output: Let's divide the data into 3 pockets, so 1/3 of the data falls into each.
      We'll use the R command quantile to do this. Create the variable cutpoints and assign
      to it the output of a call to the function quantile with 3 arguments. The first is
      the data to cut, namely diamonds$carat; the second is a call to the R function seq.
      This is also called with 3 arguments, (0, 1, and length set equal to 4). The third
      argument to the call to quantile is the boolean na.rm set equal to TRUE.
216   CorrectAnswer: cutpoints <- quantile(diamonds$carat,seq(0,1,length=4),na.rm=TRUE)
217   AnswerTests: expr_creates_var("cutpoints"); omnitest(correctExpr='cutpoints <-
      quantile(diamonds$carat,seq(0,1,length=4),na.rm=TRUE)')
218   Hint: Type cutpoints <- quantile(diamonds$carat,seq(0,1,length=4),na.rm=TRUE) at the
      command prompt.
219
220 - Class: cmd_question
221   Output: Look at cutpoints now to understand what it is.
222   CorrectAnswer: cutpoints
223   AnswerTests:  omnitest(correctExpr='cutpoints')
224   Hint: Type cutpoints  at the command prompt.
225
226 - Class: cmd_question
227   Output: We see a 4-long vector (explaining why length was set equal to 4). We also
      see that .2 is the smallest carat size in the dataset and 5.01 is the largest. One
      third of the diamonds are between .2 and .5 carats and another third are between .5
      and 1 carat in size. The remaining third are between 1 and 5.01 carats. Now we can
      use the R command cut to label each of the 53940 diamonds in the dataset as belonging
      to one of these 3 factors. Create a new name in diamonds, diamonds$car2 by assigning
      it the output of the call to cut. This command takes 2 arguments, diamonds$carat,
      which is what we want to cut, and cutpoints, the places where we'll cut.
228   CorrectAnswer: diamonds$car2 <- cut(diamonds$carat,cutpoints);
      stageVariable("diamonds$car2",diamonds$car2)
229   AnswerTests:   omnitest(correctExpr='diamonds$car2 <- cut(diamonds$carat,cutpoints)')
230   Hint: Type diamonds$car2 <- cut(diamonds$carat,cutpoints)  at the command prompt.
231
232 - Class: cmd_question
233   Output: Now we can continue with our multi-facet plot. First we have to reset g since
      we changed the dataset (diamonds) it contained (by adding a new column). Assign to g
      the output of a call to ggplot with 2 arguments. The dataset diamonds is the first,
      and a call to the function aes with 2 arguments (depth,price) is the second.
234   CorrectAnswer: g <- ggplot(diamonds,aes(depth,price))
235   AnswerTests: expr_creates_var("g"); omnitest(correctExpr='g <-
```

```
    ggplot(diamonds,aes(depth,price))')
    Hint: Type g <- ggplot(diamonds,aes(depth,price)) at the command prompt.

  - Class: cmd_question
    Output: Now add to g  calls to 2 functions. This first is a call to geom_point with
    the argument alpha set equal to 1/3. The second is a call to the function facet_grid
    using the formula cut ~ car2 as its argument.
    CorrectAnswer: g+geom_point(alpha=1/3)+facet_grid(cut~car2)
    AnswerTests:  omnitest(correctExpr='g+geom_point(alpha=1/3)+facet_grid(cut~car2)')
    Hint: Type g+geom_point(alpha=1/3)+facet_grid(cut~car2)  at the command prompt.

  - Class: text
    Output: We see a multi-facet plot with 5 rows, each corresponding to a cut factor.
    Not surprising. What is surprising is the number of columns. We were expecting 3 and
    got 4. Why?

  - Class: cmd_question
    Output: The first 3 columns are labeled with the cutpoint boundaries. The fourth is
    labeled NA and shows us where the data points with missing data (NA or Not Available)
    occurred. We see that there were only a handful (12 in fact) and they occurred in
    Very Good, Premium, and Ideal cuts. We created a vector, myd,  containing the indices
    of these datapoints. Look at these entries in diamonds by typing the expression
    diamonds[myd,]. The myd tells R what rows to show and the empty column entry says to
    print all the columns.
    CorrectAnswer: diamonds[myd,]
    AnswerTests:  omnitest(correctExpr='diamonds[myd,]')
    Hint: Type diamonds[myd,]  at the command prompt.

  - Class: text
    Output: We see these entries match the plots. Whew - that's a relief. The car2 field
    is, in fact, NA for these entries, but the carat field shows they each had a carat
    size of .2. What's going on here?

  - Class: text
    Output: Actually our plot answers this question. The boundaries for each column
    appear in the gray labels at the top of each column, and we see that the first column
    is labeled (0.2,0.5]. This indicates that this column contains data greater than .2
    and less than or equal to .5. So diamonds with carat size .2 were excluded from the
    car2 field.

  - Class: cmd_question
    Output: Finally, recall the last plotting command
    (g+geom_point(alpha=1/3)+facet_grid(cut~car2)) or retype it if you like and add
    another call. This one to the function geom_smooth. Pass it 3 arguments, method set
    equal to the string "lm", size set equal to 3, and color equal to the string "pink".
    CorrectAnswer:
    g+geom_point(alpha=1/3)+facet_grid(cut~car2)+geom_smooth(method="lm",size=3,color="pink
    ")
    AnswerTests:
    omnitest(correctExpr='g+geom_point(alpha=1/3)+facet_grid(cut~car2)+geom_smooth(method="
    lm",size=3,color="pink")')
    Hint: Type
    g+geom_point(alpha=1/3)+facet_grid(cut~car2)+geom_smooth(method="lm",size=3,color="pink
    ")  at the command prompt.

  - Class: text
    Output: Nice thick regression lines which are somewhat interesting. You can add
    labels to the plot if you want but we'll let you experiment on your own.

  - Class: cmd_question
    Output: Lastly, ggplot2 can, of course, produce boxplots. This final exercise is the
    sum of 3 function calls. The first call is to ggplot with 2 arguments, diamonds and a
    call to aes with carat and price as arguments. The second call is to geom_boxplot
    with no arguments. The third is to facet_grid with one argument, the formula . ~ cut.
    Try this now.
    CorrectAnswer: ggplot(diamonds,aes(carat,price))+geom_boxplot()+facet_grid(.~cut)
    AnswerTests:
    omnitest(correctExpr='ggplot(diamonds,aes(carat,price))+geom_boxplot()+facet_grid(.~cut
    )')
```

```
  Hint: Type ggplot(diamonds,aes(carat,price))+geom_boxplot()+facet_grid(.~cut) at the
  command prompt.

- Class: text
  Output: Yes! A boxplot looking like marshmallows about to be roasted. Well done and
  congratulations! You've finished this jewel of a lesson. Hope it paid off!

- 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: ""

```