```
  Course: Exploratory_Data_Analysis
  Lesson: Hierarchical_Clustering

- Class: text
  Output: "Hierarchical_Clustering. (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/hierarchicalClustering.)"


- Class: text
  Output:  In this lesson we'll learn about hierarchical clustering, a simple way of
  quickly examining and displaying multi-dimensional data. This technique is usually
  most useful in the early stages of analysis when you're trying to get an
  understanding of the data, e.g., finding some pattern or relationship between
  different factors or variables. As the name suggests hierarchical clustering creates
  a hierarchy of clusters.

- Class: text
  Output:  Clustering organizes data points that are close into groups. So obvious
  questions are "How do we define close?", "How do we group things?", and "How do we
  interpret the grouping?" Cluster analysis is a very important topic in data analysis.

- Class: figure
  Output: To give you an idea of what we're talking about, consider these random points
  we generated. We'll use them to demonstrate hierarchical clustering in this lesson.
  We'll do this in several steps, but first we have to clarify our terms and concepts.
  Figure: ranPoints.R
  FigureType: new

- Class: text
  Output: Hierarchical clustering is an agglomerative, or bottom-up, approach. From
  Wikipedia  (http://en.wikipedia.org/wiki/Hierarchical_clustering), we learn that in
  this method, "each observation starts in its own cluster, and pairs of clusters are
  merged as one moves up the hierarchy." This means that we'll find the closest two
  points and put them together in one cluster, then find the next closest pair in the
  updated picture, and so forth. We'll repeat this process until we reach a reasonable
  stopping place.

- Class: text
  Output: Note the word "reasonable". There's a lot of flexibility in this field and
  how you perform your analysis depends on your problem. Again, Wikipedia tells us,
  "one can decide to stop clustering either when the clusters are too far apart to be
  merged (distance criterion) or when there is a sufficiently small number of clusters
  (number criterion)."

- Class: text
  Output: First, how do we define close? This is the most important step and there are
  several possibilities depending on the questions you're trying to answer and the data
  you have. Distance or similarity are usually the metrics used.

- Class: mult_question
  Output: In the given plot which pair points would you first cluster? Use distance as
  the metric.
  AnswerChoices:  7 and 8; 1 and 4; 5 and 6; 10 and 12
  CorrectAnswer:  5 and 6
  AnswerTests: omnitest(correctVal='5 and 6')
  Hint: The choices aren't very close. Look at the picture for the answer.

- Class: text
  Output: It's pretty obvious that out of the 4 choices, the pair 5 and 6 were the
  closest together. However, there are several ways to measure distance or similarity.
  Euclidean distance and correlation similarity are continuous measures, while
  Manhattan distance is a binary measure. In this lesson we'll just briefly discuss the
  first and last of these. It's important that you use a measure of distance that fits
  your problem.

- Class: figure
  Output: Euclidean distance is what you learned about in high school algebra. Given
```

```
    two points on a plane, (x1,y1) and (x2,y2), the Euclidean distance is the square root
    of the sums of the squares of the distances between the two x-coordinates (x1-x2) and
    the two y-coordinates (y1-y2). You probably recognize this as an application of the
    Pythagorean theorem which yields the length of the hypotenuse of a right triangle.
    Figure: showEuclid.R
    FigureType: new

- Class: text
    Output: It shouldn't be hard to believe that this generalizes to more than two
    dimensions as shown in the formula at the bottom of the picture shown here.

- Class: text
    Output: Euclidean distance is distance "as the crow flies". Many applications,
    however,  can't realistically use crow-flying distance. Cars, for instance, have to
    follow roads.

- Class: figure
    Output:  In this case, we can use Manhattan or city block distance (also known as a
    taxicab metric). This picture, copied from
    http://en.wikipedia.org/wiki/Taxicab_geometry, shows what this means.
    Figure: showTaxi.R
    FigureType: new

- Class: text
    Output: You want to travel from the point at the lower left to the one on the top
    right. The shortest distance is the Euclidean (the green line), but you're limited to
    the grid, so you have to follow a path similar to those shown in red, blue, or
    yellow. These all have the same length (12) which is the number of small gray
    segments covered by their paths.

- Class: text
    Output: More formally, Manhattan distance is the sum of the absolute values of the
    distances between each coordinate, so the distance between the points (x1,y1) and
    (x2,y2) is |x1-x2|+|y1-y2|. As with Euclidean distance, this too generalizes to more
    than 2 dimensions.

- Class: figure
    Output:  Now we'll go back to our random points. You might have noticed that these
    points don't really look randomly positioned, and in fact, they're not. They were
    actually generated as 3 distinct clusters. We've put the coordinates of these points
    in a data frame for you, called dataFrame.
    Figure: ranPoints.R
    FigureType: new

- Class: text
    Output:  We'll use this dataFrame to demonstrate an agglomerative (bottom-up)
    technique of hierarchical clustering and create a dendrogram. This is an abstract
    picture (or graph) which shows how the 12 points in our dataset cluster together. Two
    clusters (initially, these are points) that are close are connected with a line,
    We'll use Euclidean distance as our metric of closeness.

- Class: cmd_question
    Output:  Run the R command dist with the argument dataFrame to compute the distances
    between all pairs of these points. By default dist uses Euclidean distance as its
    metric, but other metrics such as Manhattan, are available. Just use the default.
    CorrectAnswer: dist(dataFrame)
    AnswerTests: omnitest(correctExpr='dist(dataFrame)')
    Hint: Type dist(dataFrame) at the command prompt.

- Class: text
    Output: You see that the output is a lower triangular matrix with rows numbered from
    2 to 12 and columns numbered from 1 to 11. Entry (i,j) indicates the distance between
    points i and j. Clearly you need only a lower triangular matrix since the distance
    between points i and j equals that between j and i.

- Class: mult_question
    Output: From the output of dist, what is the minimum distance between two points?
    AnswerChoices:  0.0815; 0.08317; -0.0700; 0.1085
    CorrectAnswer:   0.0815
```

```
    AnswerTests: omnitest(correctVal='0.0815')
    Hint: Recall a previous question where points 5 and 6 looked close.

  - Class: figure
    Output: So 0.0815 (units are unspecified) between points 5 and 6 is the shortest
    distance. We can put these points in a single cluster and look for another close pair
    of points.
    Figure: cluster56.R
    FigureType: new

  - Class: mult_question
    Output: Looking at the picture, what would be another good pair of points to put in
    another cluster given that 5 and 6 are already clustered?
    AnswerChoices:  7 and the cluster containing 5 ad 6; 10 and 11; 1 and 4; 7 and 8
    CorrectAnswer:   10 and 11
    AnswerTests: omnitest(correctVal='10 and 11')
    Hint: Which of the choices looks closest on the picture?

  - Class: figure
    Output: So 10 and 11 are another pair of points that would be in a second cluster.
    We'll start creating our dendrogram now. Here're the original plot and two beginning
    pieces of the dendrogram.
    Figure: startDendro.R
    FigureType: new

  - Class: cmd_question
    Output:  We can keep going like this in the obvious way and pair up individual
    points, but as luck would have it, R provides a simple function which you can call
    which creates a dendrogram for you. It's called hclust() and takes as an argument the
    pairwise distance matrix which we looked at before. We've stored this matrix for you
    in a variable called distxy. Run hclust now with distxy as its argument and put the
    result in the variable hc.
    CorrectAnswer: hc <- hclust(distxy)
    AnswerTests: expr_creates_var("hc"); omnitest(correctExpr='hc <- hclust(distxy)')
    Hint: Type hc <- hclust(distxy) at the command prompt.

  - Class: figure
    Output: You're probably curious and want to see hc.
    Figure: clearPlot.R
    FigureType: new

  - Class: cmd_question
    Output: Call the R function plot with one argument, hc.
    CorrectAnswer: plot(hc)
    AnswerTests: omnitest(correctExpr='plot(hc)')
    Hint: Type plot(hc) at the command prompt.

  - Class: cmd_question
    Output:  Nice plot, right? R's plot conveniently labeled everything for you. The
    points we saw are the leaves at the bottom of the graph, 5 and 6 are connected, as
    are 10 and 11. Moreover, we see that the original 3 groupings of points are closest
    together as leaves on the picture. That's reassuring.  Now call plot again, this time
    with the argument as.dendrogram(hc).
    CorrectAnswer: plot(as.dendrogram(hc))
    AnswerTests: omnitest(correctExpr='plot(as.dendrogram(hc))')
    Hint: Type plot(as.dendrogram(hc)) at the command prompt.

  - Class: cmd_question
    Output: The essentials are the same, but the labels are missing and the leaves
    (original points) are all printed at the same level. Notice that the vertical heights
    of the lines and labeling of the scale on the left edge give some indication of
    distance. Use the R command abline to draw a horizontal blue line at 1.5 on this
    plot. Recall that this requires 2 arguments, h=1.5 and col="blue".
    CorrectAnswer: abline(h=1.5,col="blue")
    AnswerTests: omnitest(correctExpr='abline(h=1.5,col="blue")')
    Hint: Type abline(h=1.5,col="blue") at the command prompt.

  - Class: cmd_question
    Output: We see that this blue line intersects 3 vertical lines and this tells us that
```

```
    using the distance 1.5 (unspecified units) gives us 3 clusters (1 through 4), (9
    through 12), and (5 through 8). We call this a "cut" of our dendrogram. Now cut the
    dendrogam by drawing a red horizontal line at .4.
    CorrectAnswer: abline(h=.4,col="red")
    AnswerTests: omnitest(correctExpr='abline(h=.4,col="red")')
    Hint: Type abline(h=.4,col="red") at the command prompt.

- Class: cmd_question
    Output: How many clusters are there with a cut at this distance?
    CorrectAnswer: 5
    AnswerTests: equiv_val(5)
    Hint: How many vertical lines does this red line cross?

- Class: cmd_question
    Output: We see that by cutting at .4 we have 5 clusters, indicating that this
    distance is small enough to break up our original grouping of points. If we drew a
    horizontal line at .05, how many clusters would we get
    CorrectAnswer: 12
    AnswerTests: equiv_val(12)
    Hint: Recall that our shortest distance was around .08, so a distance smaller than
    that would make all the points their own private clusters.

- Class: cmd_question
    Output: Try it now (draw a horizontal line at .05) and make the line green.
    CorrectAnswer: abline(h=.05,col="green")
    AnswerTests: abline(h=.05,col="green")
    Hint: Type abline(h=.05,col="green") at the command prompt.

- Class: text
    Output: So the number of clusters in your data depends on where you draw the line!
    (We said there's a lot of flexibility here.) Now that we've seen the practice, let's
    go back to some "theory". Notice that the two original groupings, 5 through 8, and 9
    through 12, are connected with a horizontal line near the top of the display. You're
    probably wondering how distances between clusters of points are measured.

- Class: text
    Output: There are several ways to do this. We'll just mention two. The first is
    called complete linkage and it says that if you're trying to measure a distance
    between two clusters, take the greatest distance between the pairs of points in those
    two clusters. Obviously such pairs contain one point from each cluster.

- Class: figure
    Output: So if we were measuring the distance between the two clusters of points (1
    through 4) and (5 through 8), using complete linkage as the metric we would use the
    distance between points 4 and 8 as the measure since this is the largest distance
    between the pairs of those groups.
    Figure: complete.R
    FigureType: new

- Class: figure
    Output: The distance between the two clusters of points (9 through 12) and (5 through
    8), using complete linkage as the metric, is the distance between points 11 and 8
    since this is the largest distance between the pairs of those groups.
    Figure: complete2.R
    FigureType: new

- Class: figure
    Output: As luck would have it, the distance between the two clusters of points (9
    through 12) and (1 through 4), using complete linkage as the metric, is the distance
    between points 11 and 4.
    Figure: complete3.R
    FigureType: new

- Class: cmd_question
    Output:  We've created the dataframe dFsm for you containing these 3 points, 4, 8,
    and 11. Run dist on dFsm to see what the smallest distance between these 3 points is.
    CorrectAnswer: dist(dFsm)
    AnswerTests: omnitest(correctExpr='dist(dFsm)')
    Hint: Type dist(dFsm) at the command prompt.
```

```
- Class: figure
  Output: We see that the smallest distance is between points 2 and 3 in this reduced
  set, (these are actually points 8 and 11 in the original set), indicating that the
  two clusters these points represent ((5 through 8) and (9 through 12) respectively)
  would be joined (at a distance of 1.869) before being connected with the third
  cluster (1 through 4). This is consistent with the dendrogram we plotted.
  Figure: dendro.R
  FigureType: new

- Class: figure
  Output: The second way to measure a distance between two clusters that we'll just
  mention is called average linkage. First you compute an "average" point in each
  cluster (think of it as the cluster's center of gravity). You do this by computing
  the mean (average) x and y coordinates of the points in the cluster.
  Figure: average.R
  FigureType: new

- Class: figure
  Output: Then you compute the distances between each cluster average to compute the
  intercluster distance.
  Figure: average2.R
  FigureType: new

- Class: cmd_question
  Output:  Now look at the hierarchical cluster we created before, hc.
  CorrectAnswer: hc
  AnswerTests: omnitest(correctExpr='hc')
  Hint: Type hc at the command prompt.

- Class: mult_question
  Output: Which type of linkage did hclust() use to agglomerate clusters?
  AnswerChoices:  average; complete
  CorrectAnswer:  complete
  AnswerTests: omnitest(correctVal='complete')
  Hint: Look at the output when you looked at hc. What was the cluster method?

- Class: text
  Output: In our simple set of data, the average and complete linkages aren't that
  different, but in more complicated datasets the type of linkage  you use could affect
  how your data clusters. It is a good idea to experiment with different methods of
  linkage to see the varying ways your data groups. This will help you determine the
  best way to continue with your analysis.

- Class: text
  Output: The last method of visualizing data we'll mention in this lesson concerns
  heat maps. Wikipedia (http://en.wikipedia.org/wiki/Heat_map) tells us a heat map is
  "a graphical representation of data where the individual values contained in a matrix
  are represented as colors. ... Heat maps originated in 2D displays of the values in a
  data matrix. Larger values were represented by small dark gray or black squares
  (pixels) and smaller values by lighter squares."

- Class: text
  Output:  You've probably seen many examples of heat maps, for instance weather radar
  and displays of ocean salinity. From Wikipedia
  (http://en.wikipedia.org/wiki/Heat_map) we learn that heat maps are often used in
  molecular biology "to represent the level of expression of many genes across a number
  of comparable samples  (e.g. cells in different states, samples from different
  patients) as they are obtained from DNA microarrays."

- Class: figure
  Output: We won't say too much on this topic, but a very nice concise tutorial on
  creating heatmaps in R exists at
  http://sebastianraschka.com/Articles/heatmaps_in_r.html#clustering. Here's an image
  from the tutorial to start you thinking about the topic. It shows a sample heat map
  with a dendrogram on the left edge mapping the relationship between the rows. The
  legend at the top shows how colors relate to values.
  Figure: showheat.R
  FigureType: new
```

```
222
223
224 - Class: cmd_question
225   Output:  R provides a handy function to produce heat maps. It's called heatmap. We've
      put the point data we've been using throughout this lesson in a matrix. Call heatmap
      now with 2 arguments. The first is dataMatrix and the second is col set equal to
      cm.colors(25). This last is optional, but we like the colors better than the default
      ones.
226   CorrectAnswer: heatmap(dataMatrix,col=cm.colors(25))
227   AnswerTests: omnitest(correctExpr='heatmap(dataMatrix,col=cm.colors(25))')
228   Hint: Type heatmap(dataMatrix,col=cm.colors(25)) at the command prompt.
229
230 - Class: text
231   Output: We see an interesting display of sorts. This is a very simple heat map -
      simple because the data isn't very complex. The rows and columns are grouped together
      as shown by colors. The top rows (labeled 5, 6, and 7) seem to be in the same group
      (same colors) while 8 is next to them but colored differently. This matches the
      dendrogram shown on the left edge. Similarly, 9, 12, 11, and 10 are grouped together
      (row-wise) along with 3 and 2. These are followed by 1 and 4 which are in a separate
      group. Column data is treated independently of rows but is also grouped.
232
233 - Class: cmd_question
234   Output:  We've subsetted some vehicle data from mtcars, the Motor Trend Car Road
      Tests which is part of the package datasets. The data is in the matrix mt and
      contains 6 factors of 11 cars. Run heatmap now with mt as its only argument.
235   CorrectAnswer: heatmap(mt)
236   AnswerTests: omnitest(correctExpr='heatmap(mt)')
237   Hint: Type heatmap(mt) at the command prompt.
238
239 - Class: cmd_question
240   Output:  This looks slightly more interesting than the heatmap for the point data. It
      shows a little better how the rows and columns are treated (clustered and colored)
      independently of one another. To understand the disparity in color (between the left
      4 columns and the right 2) look at mt now.
241   CorrectAnswer: mt
242   AnswerTests: omnitest(correctExpr='mt')
243   Hint: Type mt at the command prompt.
244
245 - Class: cmd_question
246   Output: See how four of the columns are all relatively small numbers and only two
      (disp and hp) are large? That explains the big difference in color columns. Now to
      understand the grouping of the rows call plot with one argument, the dendrogram
      object denmt we've created for you.
247   CorrectAnswer: plot(denmt)
248   AnswerTests: omnitest(correctExpr='plot(denmt)')
249   Hint: Type plot(denmt) at the command prompt.
250
251 - Class: cmd_question
252   Output:  We see that this dendrogram is the one displayed at the side of the heat
      map. How was this created? Recall that we generalized the distance formula for more
      than 2 dimensions. We've created a distance matrix for you, distmt. Look at it now.
253   CorrectAnswer: distmt
254   AnswerTests: omnitest(correctExpr='distmt')
255   Hint: Type distmt at the command prompt.
256
257 - Class: text
258   Output: See how these distances match those in the dendrogram?  So hclust really
      works! Let's review now.
259
260 - Class: mult_question
261   Output:  What is the purpose of hierarchical clustering?
262   AnswerChoices:  Give an idea of the relationships between variables or observations;
      Present a finished picture; Inspire other researchers; None of the others
263   CorrectAnswer: Give an idea of the relationships between variables or observations
264   AnswerTests: omnitest(correctVal='Give an idea of the relationships between variables
      or observations')
265   Hint: Recall that this is a technique for EXPLORING data when you're first starting
      to research a problem.
266
```

```
- Class: mult_question
  Output:  True or False? When you're doing hierarchical clustering there are strict
  rules that you MUST follow.
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: There are always choices to be made in this area.

- Class: mult_question
  Output:  True or False? There's only one way to measure distance.
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: There are always choices to be made in this area.

- Class: mult_question
  Output:  True or False? Complete linkage is a method of computing distances between
  clusters.
  AnswerChoices:  True; False
  CorrectAnswer: True
  AnswerTests: omnitest(correctVal='True')
  Hint: Once a cluster contains more than one point you need a method of defining a
  distance between it and other clusters.

- Class: mult_question
  Output:  True or False? Average linkage uses the maximum distance between points of
  two clusters as the distance between those clusters.
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: Average linkage uses the average or mean point as the representative of a
  cluster. The distance between these average points is the distance between the
  clusters.

- Class: mult_question
  Output:  True or False? The number of clusters you derive from your data depends on
  the distance at which you choose to cut it.
  AnswerChoices:  True; False
  CorrectAnswer: True
  AnswerTests: omnitest(correctVal='True')
  Hint: Recall our example where we drew horizontal cut lines through the dendrogram.

- Class: mult_question
  Output:  True or False? Once you decide basics, such as defining a distance metric
  and linkage method, hierarchical clustering is deterministic.
  AnswerChoices:  True; False
  CorrectAnswer: True
  AnswerTests: omnitest(correctVal='True')
  Hint: Once you pick your algorithm, all you have to do is apply it.

- Class: text
  Output: Congratulations! We hope this lesson didn't fluster you or get you too heated!

- 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: ""
```