```yaml
Course: Exploratory_Data_Analysis
Lesson: Clustering_Example


- Class: text
  Output: "Clustering_Example. (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/clusteringExample.)"


- Class: text
  Output:  In this lesson we'll apply some of the analytic techniques we learned in this course to data from the University of California, Irvine. Specifically, the data we'll use is from UCI's Center for Machine Learning and Intelligent Systems. You can find out more about the data at http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones. As this address indicates, the data involves smartphones and recognizing human activity. Cool, right?

- Class: text
  Output: Our goal is to show you how to use exploratory data analysis to point you in fruitful directions of research, that is, towards answerable questions. Exploratory data analysis is a "rough cut" or filter which helps you to find the most beneficial areas of questioning so you can set your priorities accordingly.

- Class: text
  Output: We also hope to show you that "real-world" research isn't always neat and well-defined like textbook questions with clearcut answers.

- Class: cmd_question
  Output:  We've loaded data from this study for you in a matrix called ssd.  Run the R command dim now to see its dimensions.
  CorrectAnswer: dim(ssd)
  AnswerTests: omnitest(correctExpr='dim(ssd)')
  Hint: Type dim(ssd) at the command prompt.

- Class: text
  Output:  Wow - ssd is pretty big, 7352 observations, each of 563 variables. Don't worry we'll only use a small portion of  this "Human Activity Recognition database".

- Class: text
  Output: The study creating this database involved 30 volunteers "performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. ... Each person performed six activities  ... wearing a smartphone (Samsung Galaxy S II) on the waist. ... The experiments have been video-recorded to label the data manually.  The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data."

- Class: cmd_question
  Output:  Use the R command names with just the last two columns (562 and 563) of ssd to see what data they contain.
  CorrectAnswer: names(ssd[562:563])
  AnswerTests: ANY_of_exprs('names(ssd[562:563])','names(ssd[,562:563])','names(ssd[,c(562,563)])','names(ssd[c(562,563)])','names(ssd[c(562:563)])','names(ssd[,c(562:563)])')
  Hint: Type names(ssd[562:563]) at the command prompt.

- Class: cmd_question
  Output: These last 2 columns contain subject and activity information. We saw above that the gathered data had "been  randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data." Run the R command table with ssd$subject as its argument to see if the data in ssd contains training or test data.
  CorrectAnswer: table(ssd$subject)
  AnswerTests: omnitest(correctExpr='table(ssd$subject)')
  Hint: Type table(ssd$subject) at the command prompt.data."
```

```
- Class: mult_question
  Output: From the number of subjects, would you infer that ssd contains training or test data?
  AnswerChoices:  training; test
  CorrectAnswer:  training
  AnswerTests: omnitest(correctVal='training')
  Hint: Count the number of subjects represented here. Does this represent 70% or 30% of the total subject population?

- Class: mult_question
  Output: So ssd contains only training data. If you ran the R command sum with table(ssd$subject) as its argument, what would the number  you get back represent?
  AnswerChoices:  the number of rows in ssd; the number of columns in ssd; the number of rows and columns of ssd; Huh?
  CorrectAnswer:  the number of rows in ssd
  AnswerTests: omnitest(correctVal='the number of rows in ssd')
  Hint: Each row was labeled with one subject and the output from table(ssd$subject) told you how many rows each subject contributed to the study.

- Class: cmd_question
  Output: Try it now (running sum on table(ssd$subject))to see if you get 7352, the number of rows in ssd, as a result.
  CorrectAnswer: sum(table(ssd$subject))
  AnswerTests: omnitest(correctExpr='sum(table(ssd$subject))')
  Hint: Type sum(table(ssd$subject)) at the command prompt.

- Class: cmd_question
  Output: So we're looking at training data from a machine learning repository. We can infer that this data is supposed to train machines to recognize activity collected from the accelerometers and gyroscopes built into the smartphones that the subjects had strapped to their waists. Run the R command table on ssd$activity to see what activities have been characterized by this data.
  CorrectAnswer: table(ssd$activity)
  AnswerTests: omnitest(correctExpr='table(ssd$activity)')
  Hint: Type table(ssd$activity) at the command prompt.

- Class: mult_question
  Output: We have 6 activities, 3 passive (laying, standing and sitting) and 3 active which involve walking. If you ran the R command sum with table(ssd$activity) as its argument, what would the number  you get back represent?
  AnswerChoices:  the number of rows in ssd; the number of columns in ssd; the number of rows and columns of ssd; Huh?
  CorrectAnswer:  the number of rows in ssd
  AnswerTests: omnitest(correctVal='the number of rows in ssd')
  Hint: Each row was labeled with one activity and the output from table(ssd$activity) told you how many rows were associated with each activity in the study.

- Class: text
  Output: Because it's training data,  each row is labeled with the correct activity (from the 6 possible) and associated with the column measurements (from the accelerometer and gyroscope). We're interested in questions such as, "Is the correlation between the measurements and activities good enough to train a machine?" so that "Given a set of 561 measurements, would a trained machine be able to determine which of the 6 activities the person was doing?"

- Class: cmd_question
  Output: First, let's massage the data a little so it's easier to work with. We've already run the R command transform on the data so that activities are factors. This will let us color code them when we generate plots. Let's look at only the first subject (numbered 1). Create the variable sub1 by assigning to it the output of the R command subset with ssd as the first argument and  the boolean, subject equal to 1, as the second.
  CorrectAnswer: sub1 <- subset(ssd, subject == 1)
  AnswerTests: expr_creates_var("sub1"); omnitest(correctExpr='sub1 <- subset(ssd, subject == 1)')
  Hint: Type sub1 <- subset(ssd, subject == 1) at the command prompt.

- Class: cmd_question
  Output: Look at the dimensions of sub1 now.
```

```
  CorrectAnswer: dim(sub1)
  AnswerTests: omnitest(correctExpr='dim(sub1)')
  Hint: Type dim(sub1) at the command prompt.

- Class: cmd_question
  Output: So sub1 has fewer than 400 rows now, but still a lot of columns which contain
  measurements. Use names on the first 12 columns of sub1 to see what kind of data we
  have.
  CorrectAnswer: names(sub1[1:12])
  AnswerTests:
  ANY_of_exprs('names(sub1[1:12])','names(sub1[,1:12])','names(sub1)[1:12]','names(sub1[c
  (1:12)])','names(sub1[,c(1:12)])')
  Hint: Type names(sub1[1:12]) at the command prompt.

- Class: cmd_question
  Output: We see X, Y, and Z (3 dimensions)  of different aspects of body acceleration
  measurements, such as mean and standard deviation. Let's do some comparisons of
  activities now by looking at plots of mean body acceleration in the X and Y
  directions. Call the function myedit with the string "showXY.R" to see the code
  generating the plots. Make sure your cursor is back in the console window before you
  hit any more buttons.
  CorrectAnswer: myedit("showXY.R")
  AnswerTests: omnitest(correctExpr='myedit("showXY.R")')
  Hint: Type myedit("showXY.R") at the command prompt.

- Class: figure
  Output:  You see both the code and its output! The plots are a little squished, but
  we see that the active activities related to walking (shown in the two blues and
  magenta) show more variability than the passive activities (shown in black, red, and
  green), particularly in the X dimension.
  Figure: showXY.R
  FigureType: new

- Class: cmd_question
  Output: The colors are a little hard to distinguish. Just for fun, call the function
  showMe (we used it in the Working_with_Colors lesson) which displays color vectors.
  Use the vector 1:6 as its argument, and hopefully this will clarify the colors you
  see in the XY comparison plot.
  CorrectAnswer: showMe(1:6)
  AnswerTests: ANY_of_exprs("showMe(1:6)","showMe(c(1:6))")
  Hint: Type showMe(1:6) at the command prompt.

- Class: text
  Output: Nice! We just wanted to show you the beauty and difference in colors. The
  colors at the bottom, black, red and green, mark the passive activities, while the
  true blues and magenta near the top show the walking activities. Let's try clustering
  to see if we can distinguish the activities more.

- Class: cmd_question
  Output: We'll still focus on the 3 dimensions of mean acceleration. (The plot we just
  saw looked at the first 2 dimensions.) Create a distance matrix, mdist, of the first
  3 columns of sub1, by using the R command dist. Use the x[,1:3] notation to specify
  the columns.
  CorrectAnswer: mdist <- dist(sub1[,1:3])
  AnswerTests: expr_creates_var("mdist"); ANY_of_exprs('mdist <-
  dist(sub1[,1:3])','mdist <- dist(sub1[,c(1:3)])')
  Hint: Type mdist <- dist(sub1[,1:3]) the command prompt.

- Class: cmd_question
  Output: Now create the variable hclustering by calling the R command hclust and
  passing it mdist as an argument. This will use the Euclidean distance as its default
  metric.
  CorrectAnswer: hclustering <- hclust(mdist)
  AnswerTests: expr_creates_var("hclustering"); omnitest(correctExpr='hclustering <-
  hclust(mdist)')
  Hint: Type hclustering <- hclust(mdist) the command prompt.

- Class: cmd_question
  Output: Now call the pretty plotting function (which we've already sourced) myplclust
```

```
    with 2 arguments. The first is hclustering, and the second is the argument lab.col set equal to unclass(sub1$activity).
    CorrectAnswer: myplclust(hclustering, lab.col = unclass(sub1$activity))
    AnswerTests: omnitest(correctExpr='myplclust(hclustering, lab.col = unclass(sub1$activity))')
    Hint: Type myplclust(hclustering, lab.col = unclass(sub1$activity)) the command prompt.

  - Class: text
    Output: Well that dendrogram doesn't look too helpful, does it? There's no clear grouping of colors, except that active colors (blues and magenta) are near each other as are the  passive (black, red, and green). So average acceleration doesn't tell us much. How about maximum acceleration? Let's look at that for the first subject (in our array sub1) for the X and Y dimensions. These are in column 10 and 11.

  - Class: figure
    Output:  Here they are plotted side by side, X dimension on the left and Y on the right. The x-axis of each show the 300+ observations and the y-axis indicates the maximum acceleration.
    Figure: showMax.R
    FigureType: new

  - Class: mult_question
    Output: From the 2 plots, what separation, if any, do you see?
    AnswerChoices:  passive activities mostly fall below the walking activities; laying generates the most acceleration in the X dimension; passive activities generate the most acceleration; there is no pattern
    CorrectAnswer:  passive activities mostly fall below the walking activities
    AnswerTests: omnitest(correctVal='passive activities mostly fall below the walking activities')
    Hint: There is a pattern. Which choice makes the most obvious sense?

  - Class: cmd_question
    Output: Finally we're seeing something vaguely interesting! Let's focus then on the 3 dimensions of maximum acceleration, stored in columns 10 through 12 of sub1. Create a new distance matrix, mdist, of these 3 columns of sub1, by using the R command dist. Again, use the x[,10:12] notation to catch the columns.
    CorrectAnswer: mdist <- dist(sub1[,10:12])
    AnswerTests: expr_creates_var("mdist"); ANY_of_exprs('mdist <- dist(sub1[,10:12])','mdist <- dist(sub1[,c(10:12)])')
    Hint: Type mdist <- dist(sub1[,10:12]) the command prompt.

  - Class: cmd_question
    Output: Now create the variable hclustering by calling  hclust with mdist as the argument.
    CorrectAnswer: hclustering <- hclust(mdist)
    AnswerTests: expr_creates_var("hclustering"); omnitest(correctExpr='hclustering <- hclust(mdist)')
    Hint: Type hclustering <- hclust(mdist) the command prompt.

  - Class: cmd_question
    Output: Again, call the  myplclust with 2 arguments. The first is hclustering, and the second is the argument lab.col set equal to unclass(sub1$activity).
    CorrectAnswer: myplclust(hclustering, lab.col = unclass(sub1$activity))
    AnswerTests: omnitest(correctExpr='myplclust(hclustering, lab.col = unclass(sub1$activity))')
    Hint: Type myplclust(hclustering, lab.col = unclass(sub1$activity)) the command prompt.

  - Class: text
    Output: Now we see clearly that the data splits into 2 clusters, active and passive activities. Moreover, the light blue (walking down)  is clearly distinct from the other  walking activities. The dark blue (walking level) also seems to be somewhat clustered. The passive activities, however, seem all jumbled together with no clear pattern visible.

  - Class: cmd_question
    Output: Let's try some SVD now. Create the variable svd1 by assigning to it the output of a call to the R command svd. The argument to svd should be scale(sub1[,-c(562,563)]). This will remove the last 2 columns from sub1 and scale the data. Recall that the last 2 columns  contain activity and subject information
```

```
  which we won't need.
  CorrectAnswer: svd1 <- svd(scale(sub1[,-c(562,563)]))
  AnswerTests: expr_creates_var("svd1"); ANY_of_exprs('svd1 <- svd(scale(sub1[,-c(562,563)]))','svd1 <- svd(scale(sub1[,-c(562:563)]))')
  Hint: Type svd1 <- svd(scale(sub1[,-c(562,563)])) the command prompt.

- Class: mult_question
  Output: To see LEFT singular vectors of sub1, which component of svd1 would we examine?
  AnswerChoices:  u; v; d; x
  CorrectAnswer:  u
  AnswerTests: omnitest(correctVal='u')
  Hint: One of the choices isn't even part of the svd output. Recall that singular value decomposition expresses the matrix X as the product of three other matrices, X=UDV. Which of these is leftmost?

- Class: cmd_question
  Output: Call the R command dim with svd1$u as an argument.
  CorrectAnswer: dim(svd1$u)
  AnswerTests:  omnitest(correctExpr='dim(svd1$u)')
  Hint: Type dim(svd1$u) at the command prompt.

- Class: text
  Output: We see that the u matrix is a 347 by 347 matrix. Each row in u corresponds to a row in the matrix sub1. Recall that in sub1 each row has an associated activity.

- Class: figure
  Output:   Here we're looking at the 2 left singular vectors of svd1 (the first 2 columns of svd1$u). Each entry of the columns belongs to a particular row with one of the 6  activities assigned to it. We see the activities distinguished by color. Moving from left to right, the first section of rows are green (standing), the second red (sitting), the third black (laying), etc.  The first column of u shows separation of the nonmoving (black, red, and green) from the walking activities. The second column is harder to interpret. However, the magenta cluster, which represents walking up, seems separate from the others.
  Figure: showU2.R
  FigureType: new

- Class: text
  Output: We'll try to figure out why that is. To do that we'll have to find which of the 500+ measurements (represented by the columns of sub1) contributes to the variation of that component. Since we're interested in sub1 columns, we'll look at the RIGHT singular vectors (the columns of svd1$v), and in particular, the second one since the separation of the magenta cluster stood out in the second column of svd1$u.

- Class: figure
  Output:   Here's a plot of the second column of svd1$v. We used transparency in our plotting but nothing clearly stands out here. Let's use clustering to find the feature (out of the 500+) which contributes the most to the variation of this second column of svd1$v.
  Figure: showV2.R
  FigureType: new

- Class: cmd_question
  Output: Create the variable maxCon by assigning to it the output of the R command which.max using the second column of svd1$v as an argument.
  CorrectAnswer: maxCon <- which.max(svd1$v[,2])
  AnswerTests:  expr_creates_var("maxCon"); omnitest(correctExpr='maxCon <- which.max(svd1$v[,2])')
  Hint: Type maxCon <- which.max(svd1$v[,2]) at the command prompt.

- Class: cmd_question
  Output: Now create a distance matrix mdist by assigning to it the output of the R command dist using 4 columns of sub1  as the arguments. These 4 columns are 10 through 12 (10:12) and maxCon. Recall that you'll have to concatenate these 2 column expressions when specifying them.
  CorrectAnswer: mdist <- dist(sub1[,c(10:12,maxCon)])
  AnswerTests:  expr_creates_var("mdist"); omnitest(correctExpr='mdist <- dist(sub1[,c(10:12,maxCon)])')
  Hint: Type mdist <- dist(sub1[,c(10:12,maxCon)]) at the command prompt.
```

```
- Class: cmd_question
  Output: Now create hclustering, the output of the R command hclust using mdist as the argument.
  CorrectAnswer: hclustering <- hclust(mdist)
  AnswerTests:  expr_creates_var("hclustering"); omnitest(correctExpr='hclustering <- hclust(mdist)')
  Hint: Type hclustering <- hclust(mdist) at the command prompt.

- Class: cmd_question
  Output: Call the  myplclust with 2 arguments, hclustering, and  lab.col set equal to unclass(sub1$activity).
  CorrectAnswer: myplclust(hclustering, lab.col = unclass(sub1$activity))
  AnswerTests: omnitest(correctExpr='myplclust(hclustering, lab.col = unclass(sub1$activity))')
  Hint: Type myplclust(hclustering, lab.col = unclass(sub1$activity)) at the command prompt.

- Class: text
  Output: Now we see some real separation. Magenta (walking up) is on the far left, and the two other walking activities, the two blues, are on the far right, but in separate clusters from one another. The nonmoving activities still are jumbled together.

- Class: cmd_question
  Output: Run the R command names with the argument sub1[maxCon] to see what measurement is associated with this maximum contributor.
  CorrectAnswer: names(sub1[maxCon])
  AnswerTests: ANY_of_exprs('names(sub1[maxCon])','names(sub1)[maxCon]')
  Hint: Type names(sub1[maxCon]) or names(sub1)[maxCon] at the command prompt.

- Class: text
  Output: So the mean body acceleration in the frequency domain in the Z direction is the main contributor to this clustering phenomenon we're seeing. Let's move on to k-means clustering to see if this technique can distinguish between the activities.

- Class: cmd_question
  Output: Create the variable kClust by assigning to it the output of the R command kmeans with 2 arguments. The first is sub1 with the last 2 columns removed. (Recall these don't have pertinent information for clustering analysis.) The second argument to kmeans is centers set equal to 6, the number of activities we know we have.
  CorrectAnswer: kClust <- kmeans(sub1[, -c(562, 563)], centers = 6)
  AnswerTests: expr_creates_var("kClust"); ANY_of_exprs('kClust <- kmeans(sub1[, -c(562, 563)], centers = 6)','kClust <- kmeans(sub1[, -c(562:563)], centers = 6)')
  Hint: Type kClust <- kmeans(sub1[, -c(562, 563)], centers = 6) the command prompt.

- Class: cmd_question
  Output: Recall that without specifying coordinates for the cluster centroids (as we did), kmeans will generate starting points randomly. Here we did only 1 random start (the default). To see the output, run the R command table with 2 arguments. The first is kClust$cluster (part of the output from kmeans), and the second is sub1$activity.
  CorrectAnswer: table(kClust$cluster, sub1$activity)
  AnswerTests: omnitest(correctExpr='table(kClust$cluster, sub1$activity)')
  Hint: Type table(kClust$cluster, sub1$activity) the command prompt.

- Class: text
  Output: Your exact output will depend on the state of your random number generator. We notice that when we just run with 1 random start, the clusters tend to group the nonmoving activities together in one cluster. The walking activities seem to cluster individually by themselves. You could run the call to kmeans with one random start again and you'll probably get a slightly different result, but....

- Class: cmd_question
  Output: ... instead call  kmeans with 3 arguments, the last of which will tell it to try more random starts and return the best one. The first 2 arguments should be the same as before (sub1 with the last 2 columns removed and centers set equal to 6). The third is nstart set equal to 100. Put the result in kClust again.
  CorrectAnswer: kClust <- kmeans(sub1[, -c(562, 563)], centers = 6, nstart=100)
  AnswerTests: expr_creates_var("kClust"); ANY_of_exprs('kClust <- kmeans(sub1[,
```

```
  -c(562, 563)], centers = 6, nstart=100)','kClust <- kmeans(sub1[, -c(562:563)], centers = 6, nstart=100)')
  Hint: Type kClust <- kmeans(sub1[, -c(562, 563)], centers = 6, nstart=100) the command prompt.

- Class: cmd_question
  Output:  Again, run the R command table with 2 arguments. The first is kClust$cluster (part of the output from kmeans), and the second is sub1$activity.
  CorrectAnswer: table(kClust$cluster, sub1$activity)
  AnswerTests: omnitest(correctExpr='table(kClust$cluster, sub1$activity)')
  Hint: Type table(kClust$cluster, sub1$activity) the command prompt.

- Class: text
  Output: We see that even with 100 random starts, the passive activities tend to cluster together. One of the clusters contains only laying, but in another cluster, standing and sitting group together.

- Class: cmd_question
  Output:  Use dim to find the dimensions of kClust's centers. Use the x$y notation to access them.
  CorrectAnswer: dim(kClust$centers)
  AnswerTests: omnitest(correctExpr='dim(kClust$centers)')
  Hint: Type dim(kClust$centers) the command prompt.

- Class: text
  Output: So the centers are a 6 by 561 array. Sometimes it's a good idea to look at the features (columns) of these centers to see if any dominate.

- Class: cmd_question
  Output:  Create the variable laying and assign to it the output of the call to the R command which with the argument kClust$size==29.
  CorrectAnswer: laying <- which(kClust$size==29)
  AnswerTests: expr_creates_var("laying"); omnitest(correctExpr='laying <- which(kClust$size==29)')
  Hint: Type laying <- which(kClust$size==29) the command prompt.

- Class: cmd_question
  Output:  Now call plot with 3 arguments. The first is kClust$centers[laying,1:12], and the second is pch set to 19. The third is ylab set equal to "Laying Cluster"
  CorrectAnswer: plot(kClust$centers[laying, 1:12],pch=19,ylab="Laying Cluster")
  AnswerTests:  omnitest(correctExpr='plot(kClust$centers[laying, 1:12],pch=19,ylab="Laying Cluster")')
  Hint: Type plot(kClust$centers[laying, 1:12],pch=19,ylab="Laying Cluster") the command prompt.

- Class: cmd_question
  Output: We see the first 3 columns dominate this cluster center. Run names with the first 3 columns of sub1 as the argument to remind yourself of what these columns contain.
  CorrectAnswer: names(sub1[,1:3])
  AnswerTests: ANY_of_exprs('names(sub1[,1:3])','names(sub1[,c(1:3)])','names(sub1[,c(1,2,3)])','names(sub1[c(1,2,3)])','names(sub1[1:3])','names(sub1[c(1:3)])','names(sub1)[c(1:3)]','names(sub1)[c(1,2,3)]','names(sub1)[1:3]')
  Hint: Type names(sub1[,1:3]) the command prompt.

- Class: text
  Output: So the 3 directions of mean body acceleration seem to have the biggest effect on laying.

- Class: cmd_question
  Output:  Create the variable walkdown and assign to it the output of the call to the R command which with the argument kClust$size==49.
  CorrectAnswer: walkdown <- which(kClust$size==49)
  AnswerTests: expr_creates_var("walkdown"); omnitest(correctExpr='walkdown <- which(kClust$size==49)')
  Hint: Type walkdown <- which(kClust$size==49) the command prompt.

- Class: cmd_question
```

```yaml
    Output:  Now call plot with 3 arguments. The first is kClust$centers[walkdown,1:12], and the second is pch set to 19. The third is ylab set equal to "Walkdown Cluster"
    CorrectAnswer: plot(kClust$centers[walkdown, 1:12],pch=19,ylab="Walkdown Cluster")
    AnswerTests:  omnitest(correctExpr='plot(kClust$centers[walkdown, 1:12],pch=19,ylab="Walkdown Cluster")')
    Hint: Type plot(kClust$centers[walkdown, 1:12],pch=19,ylab="Walkdown Cluster") the command prompt.

- Class: text
    Output: We see an interesting pattern here. From left to right, looking at the 12 acceleration measurements in groups of 3, the points decrease in value. The X direction dominates, followed by Y then Z. This might tell us something more about the walking down activity.

- Class: text
    Output: We'll wrap up here and hope this example convinced you that real world analysis can be frustrating sometimes and not always obvious. You might have to try several techniques of exploratory data analysis before you hit one that pays off and leads you to the questioms that will be the most promising to explore.

- Class: text
    Output: We saw  here that the sensor measurements were pretty good at discriminating between the 3 walking activities, but the passive activities were harder to distinguish from one another. These might require more analysis or an entirely different set of sensory measurements.

- Class: text
    Output: Congratulations! We hope you enjoyed the 6 activities and 500+ features of this lesson.

- 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: ""
```