David Kinney

Sejal Jaiswal
March 14th, 2018

MUST READ MACHINE LEARNING +1

K-Means Clustering in R Tutorial

Learn all about clustering and, more specifically, k-means in this R Tutorial, where you'll focus on a case study with Uber data.

Clustering is an unsupervised learning technique. It is the task of grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters. Similarity is an amount that reflects the strength of relationship between two data objects. Clustering is mainly used for exploratory data mining. It is used in many fields such as machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression, and computer graphics.

In this tutorial, you will see:

- You'll first take a look at the different types of clustering: hard and soft clustering
- Next, you'll study the types of clustering methods, such as connectivity-, centroid-, distribution- and density-based clustring.
- You will then learn about the k-means clustering algorithm, an example of centroid-based clustering. You will work on a case study to see the working of k-means on the Uber dataset using R. The dataset is freely available and contains raw data on Uber pickups with information such as the date, time of the trip along with the longitude-latitude

Want to leave a comment?

hierarchical clustering in depth. You will also learn about Principal Component Analysis (PCA), a common approach to dimensionality reduction in Machine Learning.

Clustering: Types

Clustering can be broadly divided into two subgroups:

- Hard clustering: in hard clustering, each data object or point either belongs to a cluster completely or not. For example in the Uber dataset, each location belongs to either one borough or the other.
- Soft clustering: in soft clustering, a data point can belong to more than one cluster with some probability or likelihood value. For example, you could identify some locations as the border points belonging to two or more boroughs.

Clustering Algorithms

Clustering algorithms can be categorized based on their cluster model, that is based on how they form clusters or groups. This tutorial only highlights some of the prominent clustering algorithms.

- Connectivity-based clustering: the main idea behind this clustering is that data points that are closer in the data space are more related (similar) than to data points farther away. The clusters are formed by connecting data points according to their distance. At different distances, different clusters will form and can be represented using a dendrogram, which gives away why they are also commonly called "hierarchical clustering". These methods do not produce a unique partitioning of the dataset, rather a hierarchy from which the user still needs to choose appropriate clusters by choosing the level where they want to cluster. They are also not very robust towards outliers, which might show up as additional clusters or even cause other clusters to merge.

Want to leave a comment?

- Distribution-based clustering: this clustering is very closely related to statistics: distributional modeling. Clustering is based on the notion of how probable is it for a data point to belong to a certain distribution, such as the Gaussian distribution, for example. Data points in a cluster belong to the same distribution. These models have a strong theoritical foundation, however they often suffer from overfitting. Gaussian mixture models, using the expectation-maximization algorithm is a famous distribution based clustering method.

- Density-based methods search the data space for areas of varied density of data points. Clusters are defined as areas of higher density within the data space compared to other regions. Data points in the sparse areas are usually considered to be noise and/or border points. The drawback with these methods is that they expect some kind of density guide or parameters to detect cluster borders. DBSCAN and OPTICS are some prominent density based clustering.

One algorithm to rule them all

Now that you have seen various types of clustering algorithms, the big question is: "how can you identify the correct algorithm to use?"

Well, sorry but there is no ONE algorithm to rule them all. False alarm!!

"Clustering is in the eye of the beholder!"

Clustering is an subjective task and there can be more than one correct clustering algorithm. Every algorithm follows a different set of rules for defining the 'similarity' among data points. The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one clustering algorithm over another. An algorithm might work well on a particular dataset but fail for a different kind of dataset.

Want to leave a comment?

Don't worry if you don't know too much about Uber, all you need to know is that the Uber platform connects you with (cab)drivers who can drive you to your destiny. The data is freely available on Kaggle. The dataset contains raw data on Uber pickups with information such as the date, time of the trip along with the longitude-latitude information.

New York city has five boroughs: Brooklyn, Queens, Manhattan, Bronx, and Staten Island. At the end of this mini-project, you will apply k-means clustering on the dataset to explore the dataset better and identify the different boroughs within New York. All along, you will also learn the various steps that you should take when working on a data science project in general.

Problem Understanding

There is a lot of information stored in the traffic flow of any city. This data when mined over location can provide information about the major attractions of the city, it can help us understand the various zones of the city such as residential areas, office/school zones, highways, etc. This can help governments and other institutes plan the city better and enforce suitable rules and regulations accordingly. For example, a different speed limit in school and residential zone than compared to highway zones.

The data when monitored over time can help us identify rush hours, holiday season, impact of weather, etc. This knowledge can be applied for better planning and traffic management. This can at a large, impact the efficiency of the city and can also help avoid disasters, or at least faster redirection of traffic flow after accidents.

However, this is all looking at the bigger problem. This tutorial will only concentrate on trying to solve the problem of identifying the five boroughs of New York city using k-means algorithm, so as to get a better understanding of the algorithms, all along learning to tackle a data science problem.

Understanding The Data

Want to leave a comment?

- `uber-raw-data-may14.csv`
- `uber-raw-data-jun14.csv`
- `uber-raw-data-jul14.csv`
- `uber-raw-data-aug14.csv`
- `uber-raw-data-sep14.csv`

This tutorial makes use of various libraries. Remember that when you work locally, you might have to install them. You can easily do so, using `install.packages()` .

Let's now load up the data:

```
# Load the .csv files
apr14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
may14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
jun14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
jul14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
aug14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
sep14 <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/ma
```

Let's bind all the data files into one. For this, you can use the `bind_rows()` function under the `dplyr` library in R.

```
library(dplyr)
data14 <- bind_rows(apr14, may14, jun14, jul14, aug14, sep14)
```

So far, so good! Let's get a summary of the data to get an idea of what you are dealing with.

```
summary(data14)
```

Want to leave a comment?

```
 Length:4534327     Min.   :39.66   Min.   :-74.93   B02512: 205673
 Class :character   1st Qu.:40.72   1st Qu.:-74.00   B02598:1393113
 Mode  :character   Median :40.74   Median :-73.98   B02617:1458853
                    Mean   :40.74   Mean   :-73.97   B02682:1212789
                    3rd Qu.:40.76   3rd Qu.:-73.97   B02764: 263899
                    Max.   :42.12   Max.   :-72.07
```

The dataset contains the following columns:

- `Date.Time` : the date and time of the Uber pickup;
- `Lat` : the latitude of the Uber pickup;
- `Lon` : the longitude of the Uber pickup;
- `Base` : the TLC base company code affiliated with the Uber pickup.

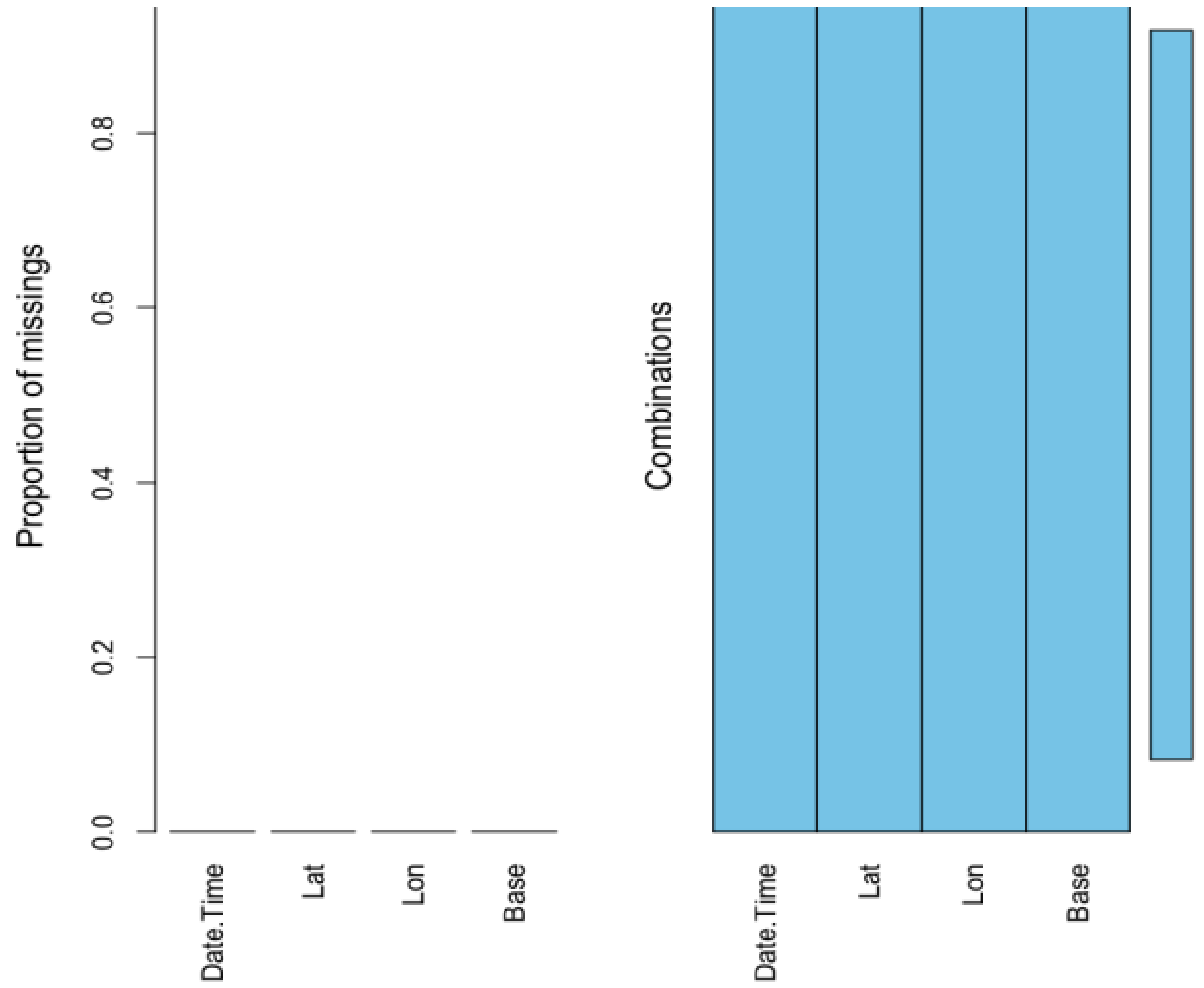
Data Preparation

This step consists of cleaning and rearranging your data so that you can work on it more easily. It's a good idea to first think of the sparsity of the dataset and check the amount of missing data.

```
# VIM library for using 'aggr'
library(VIM)

# 'aggr' plots the amount of missing/imputed values in each column
aggr(data14)
```

Want to leave a comment?



As you can see, the dataset has no missing values. However, this might not always be the case with real datasets and you will have to decide how you want to deal with these values. Some popular methods include either deleting the particular row/column or replacing with a mean of the value.

You can see that the first column is `Date.Time`. To be able to use these values, you need to separate them. So let's do that, you can use the `lubridate` library for this. Lubridate makes it simple for you to identify the order in which the year, month, and day appears in your dates and manipulate them.

```
library(lubridate)
```

Want to leave a comment?

```
data14$Weekday <- factor(wday(data14$Date.Time))
data14$Hour <- factor(hour(data14$Date.Time))
data14$Minute <- factor(minute(data14$Date.Time))
data14$Second <- factor(second(data14$Date.Time))


#data14$date_time
data14$Month
```

Let's check out the first few rows to see what our data looks like now....

```
head(data14, n=10)
```

Date.Time	Lat	Lon	Base	Year	Month	Day	Weekday	Hour
2014-04-01 00:11:00	40.7690	-73.9549	B02512	2014	4	1	3	0
2014-04-01 00:17:00	40.7267	-74.0345	B02512	2014	4	1	3	0
2014-04-01 00:21:00	40.7316	-73.9873	B02512	2014	4	1	3	0
2014-04-01 00:28:00	40.7588	-73.9776	B02512	2014	4	1	3	0
2014-04-01 00:33:00	40.7594	-73.9722	B02512	2014	4	1	3	0
2014-04-								

Want to leave a comment?

2014-04-01 00:39:00	40.7223	-73.9887	B02512	2014	4	1	3	0
2014-04-01 00:45:00	40.7620	-73.9790	B02512	2014	4	1	3	0
2014-04-01 00:55:00	40.7524	-73.9960	B02512	2014	4	1	3	0
2014-04-01 01:01:00	40.7575	-73.9846	B02512	2014	4	1	3	1

Awesome!

For this case study, this is the only data manipulation you will require for a good data understanding as well as to work with k-means clustering.

Now would be a good time to divide your data into training and test set. This is an important step in every data science project, it is done to train the model on the training set, determine the values of the parameters required and to finally test the model on the testing set. For example, when working with clustering algorithms, this division is done so that you can identify the parameters such as `k`, which is the number of clusters in k-means clustering. However, for this case study, you already know the number of clusters expected, which is 5 - the number of boroughs in NYC. Hence, you shall not be working the traditional way but rather, keep it primarily about learning about k-means clustering.

Have a look at DataCamp's Python Machine Learning: Scikit-Learn Tutorial for a project that guides you through all the steps for a data science (machine learning) project using Python. You will also work with k-means algorithm in this tutorial.

Want to leave a comment?

K-Means Clustering with R

K-means clustering is the most commonly used unsupervised machine learning algorithm for dividing a given dataset into k clusters. Here, `k` represents the number of clusters and must be provided by the user. You already know `k` in case of the Uber dataset, which is 5 or the number of boroughs. k-means is a good algorithm choice for the Uber 2014 dataset since you do not know the target labels making the problem unsupervised and there is a pre-specified `k` value.

Here you are using clustering for classifying the pickup points into various boroughs. The general scenario where you would use clustering is when you want to learn more about your dataset. So you can run clustering several times, investigate the interesting clusters and note down some of the insights you get. Clustering is more of a tool to help you explore a dataset, and should not always be used as an automatic method to classify data. Hence, you may not always deploy a clustering algorithm for real-world production scenario. They are often too unreliable, and a single clustering alone will not be able to give you all the information you can extract from a dataset.

The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized. There are several k-means algorithms available. However, the standard algorithm defines the total within-cluster variation as the sum of squared distances Euclidean distances between items and the corresponding centroid:

$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2$

where:

- x_i is a data point belonging to the cluster C_k
- μ_k is the mean value of the points assigned to the cluster C_k

Want to leave a comment?

1. Specify `k` - the number of clusters to be created.
2. Select randomly `k` objects from the dataset as the initial cluster centers.
3. Assign each observation to their closest centroid, based on the Euclidean distance between the object and the centroid.
4. For each of the k clusters recompute the cluster centroid by calculating the new mean value of all the data points in the cluster.
5. Iteratively minimize the total within sum of square. Repeat Step 3 and Step 4, until the centroids do not change or the maximum number of iterations is reached (R uses 10 as the default value for the maximum number of iterations).

The total within sum of square or the total within-cluster variation is defined as:

$$\sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

This is the summation of all the clusters over the sum of squared Euclidean distances between items and their corresponding centroid.

Now that you have seen the theory, let's implement the algorithm and see the results!

You can use the `kmeans()` function in R. `k` value will be set as 5. Also, there is a `nstart` option that attempts multiple initial configurations and reports on the best one within the kmeans function. Seeds allow you to create a starting point for randomly generated numbers, so that each time your code is run, the same answer is generated.

```
set.seed(20)
clusters <- kmeans(data14[,2:3], 5)

# Save the cluster number in the dataset as column 'Borough'
data14$Borough <- as.factor(clusters$cluster)
```

Want to leave a comment?

```
str(clusters)
```

```
List of 9
 $ cluster     : int [1:4534327] 3 4 4 3 3 4 4 3 4 3 ...
 $ centers     : num [1:5, 1:2] 40.7 40.8 40.8 40.7 40.7 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "Lat" "Lon"
 $ totss       : num 22107
 $ withinss    : num [1:5] 1386 1264 948 2787 1029
 $ tot.withinss: num 7414
 $ betweenss   : num 14692
 $ size        : int [1:5] 145109 217566 1797598 1802301 571753
 $ iter        : int 4
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

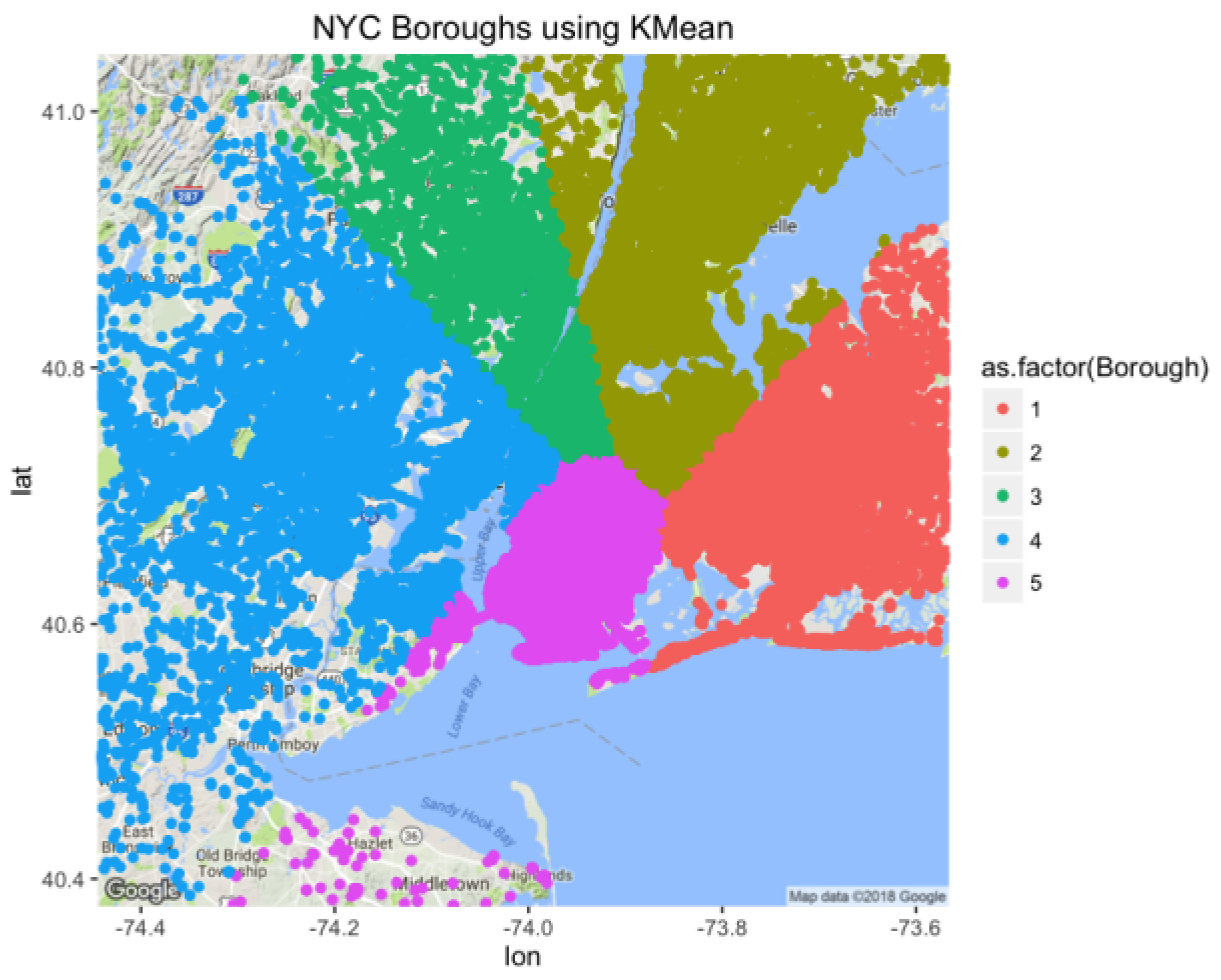
The above list is an output of the `kmeans()` function. Let's see some of the important ones closely:

- `cluster` : a vector of integers (from 1:k) indicating the cluster to which each point is allocated.
- `centers` : a matrix of cluster centers.
- `withinss` : vector of within-cluster sum of squares, one component per cluster.
- `tot.withinss` : total within-cluster sum of squares. That is, `sum(withinss)` .
- `size` : the number of points in each cluster.

Let's plot some graphs to visualize the data as well as the results of the k-means clustering well.

Want to leave a comment?

```
ggtitle("NYC Boroughs using KMean")
```



The boroughs (clusters) formed is matched against the real boroughs. The cluster number corresponds to the following boroughs:

1. Bronx

Want to leave a comment?

4. Staten Island

5. Queens

As you can see, the results are pretty impressive. And now, that you have used k-mean to categorize the pickup point and have additional knowledge added to the dataset. Let's try to do something with this new found knowledge. You can use the borough information to check out Uber's growth within the boroughs for each month. Here's how...

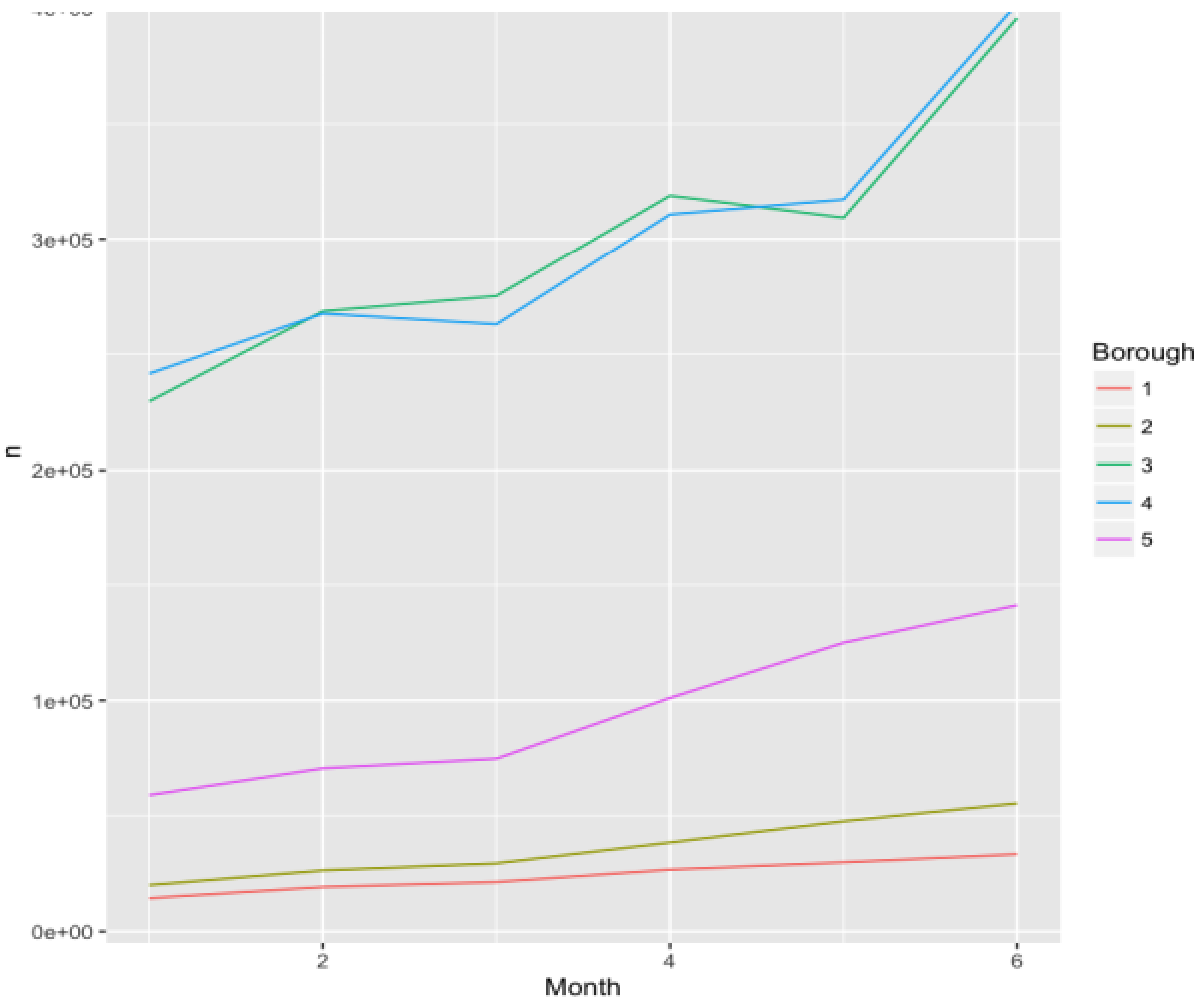
```
library(DT)

data14$Month <- as.double(data14$Month)
month_borough_14 <- count_(data14, vars = c('Month', 'Borough'), sort = TRUE) %>%
  arrange(Month, Borough)
datatable(month_borough_14)
```

Let's get a graphical view of the same...

```
library(dplyr)
monthly_growth <- month_borough_14 %>%
  mutate(Date = paste("04", Month)) %>%
  ggplot(aes(Month, n, colour = Borough)) + geom_line() +
  ggtitle("Uber Monthly Growth - 2014")
monthly_growth
```

Want to leave a comment?



As you have seen, k-means is a pretty good clustering algorithm. But, it has some drawbacks. The biggest disadvantage is that it requires us to pre-specify the number of clusters (`k`). However, for the Uber dataset, you had some domain knowledge available that told you the number of boroughs in New York City. This might not always be the case with real world datasets. Hierarchical clustering is an alternative approach that does not require a particular choice of clusters. An additional disadvantage of k-means is that it is sensitive to outliers and different results can occur if you change the ordering of the data.

k-means is a lazy learner where generalization of the training data is delayed until a query is made to the system. Which means k-means starts working only when you trigger it to, thus lazy learning methods can construct a different approximation or result to the target function for each encountered query. It is a good method for online learning but it requires a

Want to leave a comment?

You learned a basic outline of how to attack a data science project while using k-means to categorize the pickup points to identify the various boroughs of New York City. You have also used the new found knowledge to further visualize the growth of Uber in the various boroughs.

You got a glimpse of clustering in context of unsupervised learning. Classification is a similar task but in the context of supervised learning. To learn more about this, head over to DataCamp's Supervised Learning in R: Classification course. The course covers four of the most common classification algorithms.

▲ 72

38

COMMENTS

kannandeepak1992
01/04/2018 09:49 AM

Hi

There is no month column in data14

Do we need to create them ? Can you clarify this please ?

```
data14$Month <- as.double(data14$Month)
```

▲ 4 REPLY

Sejal Jaiswal
10/04/2018 03:49 AM

Hi kannandeepak1992,

Want to leave a comment?

▲ 3 REPLY

kushal tandon
09/04/2018 04:00 PM

We can benefit more from having open courses like these!!

▲ 2 REPLY

Khushboo Khandelwal
10/04/2018 05:27 PM

Hi Sejal!

Great Tutorial:)

I need some guidance, I am working on a project for a client, well not really working but it's part of my master's curriculum. The data set are real world data. I have bunch of ordinal categorial variables (~20) as inputs and now I would need to cluster them to create market segmentation. These variables are actually scaled variables i.e. on a scale from 1 to 5, where 1 being the lowest and 5 the highest range on that scale. While designing the survey questionnaire, we had ensured to maintained the uniformity i.e. all these ordinal categorical variables have fixed scale i.e. 1 to 5 to avoid any false positives and less data transformation. In short, each question is framed to address different aspects of consumer buying behaviors in short.

Now, I am trying to explore ways where I can form clusters using these variables. I also have their demographics data(Age,Gender, Income, Education, etc.,). I have never worked on ordinal categorical variables to create cluster as my understanding is limited to numerical/continuous variables only as k-means works really well based on euclidean distance principle.

Could you please suggest a way to work on this?

Appreciate your help!

Thanks!

Khushboo

Want to leave a comment?

You could read in more detail about the different kind of clustering methods possible.

Maybe it could help checking out 'Association Rule Mining' techniques, which are very famous for market basket analysis. This might be more relevant for your problem perhaps.

▲ 1 REPLY

Khushboo Khandelwal
25/05/2018 02:18 PM

Hi Sejal,

Well, I think it's too late to respond.

Any way , I submitted the project. But thanks for your help! I appreciated it.

Regards,

Khushboo

▲ 1

Nikhil Gupta
04/06/2018 09:23 AM

Hi Sejal,

Can we also use K-modes clustering in such a scenario?

Thanks,

Nikhil

▲ 1

Clifford DMello
04/05/2018 09:10 AM

Hi Khusboo/Sejal,

Did you get any insights on the type of clustering algo to be used for categorical data. I too have a similar problem which needs clustering. I plan to use k means clustering, not too sure if would work.

Any help appreciated. Thanks in advance.

▲ 1 REPLY

Want to leave a comment?

categorical data you can use K modes algo and if your data is mix of continuous and

categorical var try K prototype algo.

▲ 2

Khushboo Khandelwal
25/05/2018 02:19 PM

Hi Clifford,

I think I might like to try out K modes/K prototype algorithm to begin with as pointed by Ashutosh Pandey.

Regards,

Khushboo

▲ 1

Ashutosh Pandey
22/05/2018 02:48 AM

Try K modes

▲ 2 REPLY

Sidd Gupta
10/04/2018 07:47 PM

Anyone else having an issue with the following warning " Removed 10062 rows containing missing values (geom_point). " ? I'm guessing the zoom = 10 parameter cut off some of the long/lat?

▲ 4 REPLY

Marta Jornet
29/04/2018 01:06 PM

Hi,

Want to leave a comment?

groups. But this groups calculated by k-means, really don't match with the boroughs, do they?

I guess you are assuming that they are similar enough to the boroughs to assume each one of this calculated groups represent a borough? For instance Queens, in the map representation it is identified with the purple dots, but looking at a NYC map, this area looks more like Brooklyn than Queens.

▲ 1 REPLY

Sejal Jaiswal
01/05/2018 12:18 PM
Hi Marta,

You are right, the groups are calculated by k-means and they don't necessarily match the real borough outline....infact, some of them are overlapping and others are simply not precise.

This is partly because of the dataset we are using, the dataset contains many longitudes and latitudes that do not constitute to fall under NYC geographically (or according to the US jurisdiction - not sure what the correct term here would be) but are classified as NYC rides by Uber (maybe because the drop points were in NYC?). This brings in many more data points in the outskirts of NYC that push the clusters further from the real clustering that should have been. Hence, it is hard to find good clusters that match the real geographical outline using the k-means algorithm alone.

▲ 1 REPLY

michaelalwill
12/06/2018 10:01 AM
Towards that end, it might have been interesting to include a step of comparing NYC boundaries according to the Census (county boundaries, as each borough is a county) and removing data points not included. The general topic of working with varying geography files in R is one I'd like to see explored, particularly the "point-in-polygon" problem that would come up here.

▲ 1

Want to leave a comment?

Do we have a similar tutorial on hierarchical clustering?

▲ 3 REPLY

Antoine Soetewey
25/07/2018 06:07 AM

Hi Abhay,

Here is a tutorial on hierarchical clustering in R: https://www.datacamp.com/community/tutorials/hierarchical-clustering-R.

Best,

Antoine

▲ 1 REPLY

Julio Cezar Nogueira
17/05/2018 11:01 AM

Muito boa a aplicação !!! Foi muito útil, vou replica-la dentro do R usando o modelo de K-means do STAN usando o pacote do Rstan

▲ 1 REPLY

Amha Georgis
18/05/2018 03:11 PM

I like the real world example demonstrated well. I installed VIM package and loaded the library but aggr() function cannot be found.

"Version:1.0 StartHTML:0000000107 EndHTML:0000000867 StartFragment:0000000127 EndFragment:0000000849

```
Error in aggr(data14) : could not find function "aggr"
```

I don't know what steps did I miss?

▲ 1 REPLY

Want to leave a comment?

install.packages("VIM")

library(VIM)

aggr(data14)

Best,

Antoine

▲ 1 REPLY

John Efechaobor
08/06/2018 03:40 PM

Hi Sejal,

Great article and very informative. Regarding the line "clusters <- kmeans(data14[,2:3], 5)" could you explain the purpose of "data14[,2:3]"?

▲ 1 REPLY

Sakthivel Palani
14/07/2018 03:09 AM

Hi john ,

its good question, here data14[,2:3] ? we have to use cluster method particular 2nd and 3rd column only ,like latitude and longitude.

▲ 1 REPLY

jungsongjob
18/10/2018 02:40 PM

If you had kmeans(data14, 5) , then will it do clustering based on all columns in the dataset, instead of clustering based on lat and long?
If so, wouldn't using entire dataset to cluster better than just lat and long?

▲ 1

Anmol Khurana

Want to leave a comment?

▲ 3 REPLY

Anmol Khurana
14/07/2018 06:02 PM
im getting this error - Warning message:
Removed 10062 rows containing missing values (geom_point)

▲ 1 REPLY

Muhammad Haad Bodla
22/07/2018 08:12 AM
Hi

Great tutorial, where can i get data set you have used?

▲ 1 REPLY

Muhammad Haad Bodla
23/07/2018 09:03 AM
anyone??

▲ 1 REPLY

Antoine Soetewey
25/07/2018 07:37 AM
You will find the code to import the data set in this tutorial, after the sentence "Let's now load up the data:".

▲ 2 REPLY

Chris Leisner
23/07/2018 04:56 PM
Suppose that, in the K-means algorithm, you want to use a distance measurement other than Euclidean distance. For example, you might want to use a distance measure that takes into account the curvature of the earth, rather than a distance measure that is only accurate for points in a plane. Is there any way to run K-means with a different distance measure?

▲ 0 REPLY

Want to leave a comment?

Thanks for the great tutorial. Really clear and well explained.

There is one aspect I don't understand though: the article suggests that the clustering is 'impressive' because it maps on to the real world boundaries, to some extent. This doesn't seem to me to be the measure of success (and I see in response to another question the author does say that any similarity is actually incidental).

Wouldn't it be *more* interesting if the algorithm showed clusters that bore no resemblance at all to geographical boundaries, and isn't this more the purpose of the clustering (ie to uncover groupings that we weren't already aware of in the data)?

▲ 1 REPLY

Vedaste BUCYENSENGE
26/09/2018 05:36 AM
Hello guys! If you want to build algorithm with disarrange data how should I build it?

▲ 1 REPLY

Vedaste BUCYENSENGE
26/09/2018 05:38 AM
when i tried to use K-means i got an error! any who could guide me , let me know via my mail!

bvedaste@aims.ac.rw

Best

▲ 1 REPLY

Abel Alade
10/10/2018 11:43 PM
Thank you for this systematic presentation of K-means clustering.

The challenge I have is that of date stratification with the statements:

data14$Date.Time < -mdy_hms(data14$Date.Time)

Want to leave a comment?

data14$Hour < -factor(hour(data14$Date.Time))

data14$Minute < -factor(minute(data14$Date.Time))

data14$Second < -factor(second(data14$Date.Time))

The error message I have is -

```
Warning message:
  1748065 failed to parse.
```

So I cannot continue the tutorial from that point. Can somebody please help me out?

▲ 1 REPLY

---

shyla k
27/10/2018 12:53 PM

Error in obtaining the map of NYC - geocode failed with status OVER_QUERY_LIMIT, location = "New York".

please let me know how to resolve this?

▲ 2 REPLY

Shamlan Alroomi
28/10/2018 09:51 PM

Help please. Error in obtaining the map of NYC too.

▲ 1 REPLY

MUNIKUMAR N M
30/10/2018 09:30 PM

Hi Sejal,

Want to leave a comment?

▲ 1 REPLY

Miguel Flores
15/11/2018 08:32 PM
The get_map function no longer works without an api key. It is not clear how to use get_map or get_googlemaps

▲ 3 REPLY

Omer Mish
20/01/2019 03:56 AM
Hi Sejal,

when I ran the date/time settings in the beginning, I got a mixed table that identified the year as the hours, the weekday as the month and so forth. Do you have any idea what should I do?

▲ 0 REPLY

Subscribe to RSS

About Terms Privacy

Want to leave a comment?

Want to leave a comment?