



R news and tutorials contributed by (750) R bloggers

- [Home](#)
- [About](#)
- [RSS](#)
- [add your blog!](#)
- [Learn R](#)
- [R jobs](#) ♦ ♦ ♦
- [Contact us](#)

Welcome!


Follow @rbloggers { 72.6K

Here you will find daily **news and tutorials about R**, contributed by over 750 bloggers.

There are many ways to **follow us** -

[By e-mail:](#)

[On Facebook:](#)



R blog...
76K likes

Be the first of your friends
to like this

If you are an R blogger yourself you are invited to [add your own R content feed to this site](#) (Non-English R bloggers should add themselves- [here](#))

[Jobs for R-users](#)

- [Customer Success Representative](#)
- [Movement Building Analyst](#)
- [Business Intelligence Analyst](#)
- [Innovation Fellow](#)
- [Postdoctoral position Stats, Comp. Biol. @ Madrid, Spain.](#)

Recent Posts



[Examples](#)

- [Predicting Car Battery Failure With R And H2O – Study](#)
- [Practical Data Science with R, half off sale!](#)
- [Rstudio & ThinkR roadshow – June 6 – Paris](#)
- [\[R\]eady for Production: a Joint Event with RStudio and EODA](#)
- [Royal Society of Biology: Introduction to Reproducible Analyses in R](#)
- [Spotlight on: Julia Silge, Stack Overflow](#)
- [Comparing Frequentist, Bayesian and Simulation methods and conclusions](#)
- [Analysing the HIV pandemic, Part 4: Classification of lab samples](#)
- [MRAN snapshots, and you](#)
- [Deep \(learning\) like Jacques Cousteau – Part 5 – Vector addition](#)
- [New Color Palette for R](#)
- [Easy quick PCA analysis in R](#)
- [Create a CLI for R with npm](#)
- [Bug when Creating Reference Maps with Choroplethr](#)

Other sites

- [Jobs for R-users](#)
- [SAS blogs](#)

How to Perform Hierarchical Clustering using R

December 18, 2017

By [Perceptive Analytics](#)

Like 375

Share

Share

(This article was first published on [R-posts.com](#), and kindly contributed to [R-bloggers](#)).

 Share

 Tweet



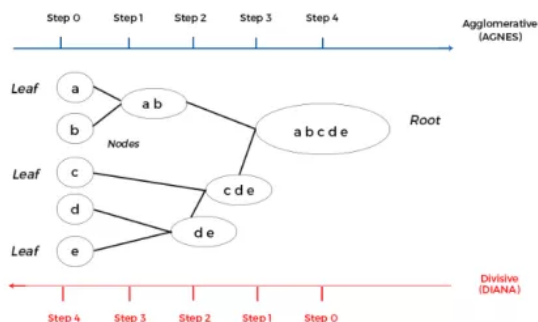
Clustering is a technique to club similar data points into one group and separate out dissimilar observations into different groups or clusters. In Hierarchical Clustering, clusters are created such that they have a predetermined ordering i.e. a hierarchy. For example, consider the concept hierarchy of a library. A library has many sections, each section would have many books, and the books would be grouped according to their subject, let's say. This forms a hierarchy. In Hierarchical Clustering, this hierarchy of clusters can either be created from top to bottom, or vice-versa. Hence, it's two types namely – Divisive and Agglomerative. Let's discuss it in detail.

Divisive Method

In Divisive method we assume that all of the observations belong to a single cluster and then divide the cluster into two least similar clusters. This is repeated recursively on each cluster until there is one cluster for each observation. This technique is also called DIANA, which is an acronym for Divisive Analysis.

Agglomerative Method

It's also known as Hierarchical Agglomerative Clustering (HAC) or AGNES (acronym for Agglomerative Nesting). In this method, each observation is assigned to its own cluster. Then, the similarity (or distance) between each of the clusters is computed and the two most similar clusters are merged into one. Finally, steps 2 and 3 are repeated until there is only one cluster left.



Please note that Divisive method is good for identifying large clusters while Agglomerative method is good for identifying small clusters. We will proceed with Agglomerative Clustering for the rest of the article. Since, HAC's account for the majority of hierarchical clustering algorithms while Divisive methods are rarely used. I think now we have a general overview of Hierarchical Clustering. Let's also get ourselves familiarized with the algorithm for it.

HAC Algorithm

Given a set of N items to be clustered, and an $N \times N$ distance (or similarity) matrix, the basic process of Johnson's (1967) hierarchical clustering is –

1. Assign each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N .

In Steps 2 and 3 here, the algorithm talks about finding similarity among clusters. So, before any clustering is performed, it is required to



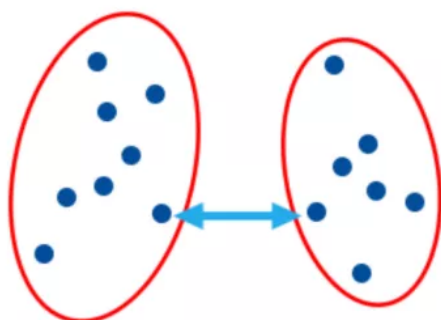
Minkowski, etc.). Then, the matrix is updated to specify the distance between different clusters that are formed as a result of merging. But, how do we measure the distance (or similarity) between two clusters of observations?

For this, we have three different methods as described below. These methods differ in how the distance between each cluster is measured.

Single Linkage

It is also known as the connectedness or minimum method. Here, the distance between one cluster and another cluster is taken to be equal to the shortest distance from any data point of one cluster to any data point in another. That is, distance will be based on similarity of the closest pair of data points as shown in the figure. It tends to produce long, "loose" clusters.

Disadvantage of this method is that it can cause premature merging of groups with close pairs, even if those groups are quite dissimilar overall.

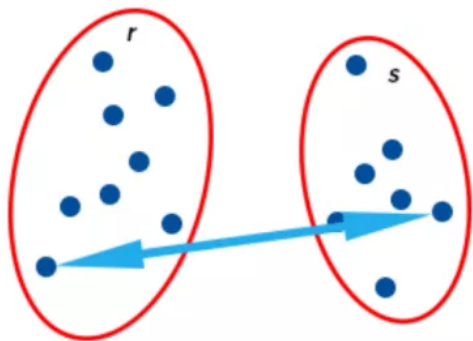


$$L(r,s) = \min(D(x_r, x_s))$$

Complete Linkage

This method is also called the diameter or maximum method. In this method, we consider similarity of the furthest pair. That is, the distance between one cluster and another cluster is taken to be equal to the longest distance from any member of one cluster to any member of the other cluster. It tends to produce more compact clusters.

One drawback of this method is that outliers can cause close groups to be merged later than what is optimal.



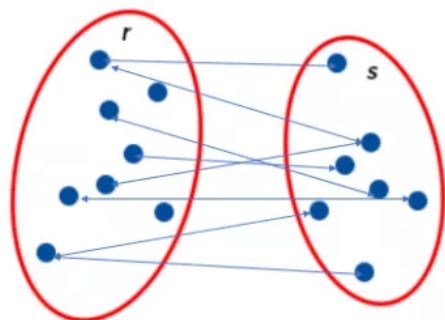
$$L(r,s) = \max(D(x_r, x_s))$$

Average Linkage



In Average linkage method, we take the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

A variation on average-link clustering is the UCLUS method of D'Andrade (1978) which uses the median distance instead of mean distance.



$$L(r, s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Ward's Method

Ward's method aims to minimize the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged. In other words, it forms clusters in a manner that minimizes the loss associated with each cluster. At each step, the union of every possible cluster pair is considered and the two clusters whose merger results in minimum increase in information loss are combined. Here, information loss is defined by Ward in terms of an error sum-of-squares criterion (ESS). If you want a mathematical treatment of this visit [this link](#).

The following table describes the mathematical equations for each of the methods —

Single Linkage	$D_{12} = \min_{i,j} d(X_i, Y_j)$	This is the distance between the closest members of the two clusters.
Complete Linkage	$D_{12} = \max_{i,j} d(X_i, Y_j)$	This is the distance between the members that are farthest apart (most dissimilar)
Average Linkage	$D_{12} = \frac{1}{k_1 k_2} \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} d(X_i, Y_j)$	This method involves looking at the distances between all pairs and averages all of these distances. This is also called UPGMA - Unweighted Pair Group Mean Averaging.
Centroid Method	$D_{12} = d(\bar{x}, \bar{y})$	This involves finding the mean vector location for each of the clusters and taking the distance between these two centroids.
Ward's Method	$D_{12} = \sqrt{\frac{2 \cdot k_1 \cdot k_2 }{ k_1 + k_2 }} \cdot \ \bar{x} - \bar{y}\ $	This method minimizes the total within-cluster variance. Those clusters are combined whose merger results in minimum information loss (ESS criterion).

Where,

- X_1, X_2, \dots, X_k = Observations from cluster 1
- Y_1, Y_2, \dots, Y_l = Observations from cluster 2
- $d(x, y)$ = Distance between a subject with observation vector x and a subject with observation vector y
- $\|\cdot\|$ = Euclidean norm



Data Preparation

To perform clustering in R, the data should be prepared as per the following guidelines –

1. Rows should contain observations (or data points) and columns should be variables.
2. Check if your data has any missing values, if yes, remove or impute them.
3. Data across columns must be standardized or scaled, to make the variables comparable.

We'll use a data set called 'Freedman' from the 'car' package. The 'Freedman' data frame has 110 rows and 4 columns. The observations are U. S. metropolitan areas with 1968 populations of 250,000 or more. There are some missing data. (Make sure you install the car package before proceeding)

```
1 data <- car::Freedman
2
3 To remove any missing value that might be pres
4 data <- na.omit(data)
```

This simply removes any row that contains missing values. You can use more sophisticated methods for imputing missing values. However, we'll skip this as it is beyond the scope of the article. Also, we will be using numeric variables here for the sake of simply demonstrating how clustering is done in R. Categorical variables, on the other hand, would require special treatment, which is also not within the scope of this article. Therefore, we have selected a data set with numeric variables alone for conciseness.

Next, we have to scale all the numeric variables. Scaling means each variable will now have mean zero and standard deviation one. Ideally, you want a unit in each coordinate to represent the same degree of difference. Scaling makes the standard deviation the unit of measurement in each coordinate. This is done to avoid the clustering algorithm to depend to an arbitrary variable unit. You can scale/standardize the data using the R function scale:

```
1 data <- scale(data)
```

Implementing Hierarchical Clustering in R

There are several functions available in R for hierarchical clustering. Here are some commonly used ones:

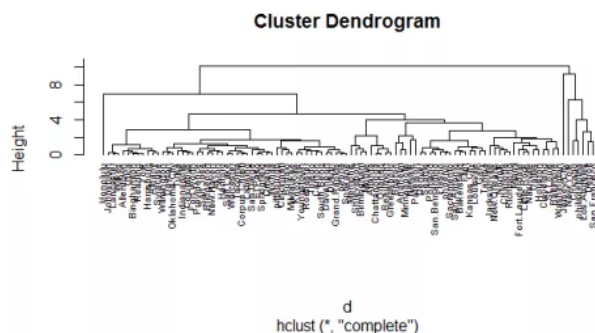
- 'hclust' (stats package) and 'agnes' (cluster package) for agglomerative hierarchical clustering
- 'diana' (cluster package) for divisive hierarchical clustering

Agglomerative Hierarchical Clustering

For 'hclust' function, we require the distance values which can be computed in R by using the 'dist' function. Default measure for dist function is 'Euclidean', however you can change it with the method argument. With this, we also need to specify the linkage method we want to use (i.e. "complete", "average", "single", "ward.D").

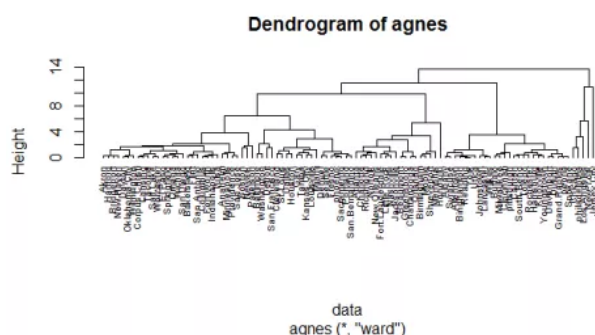
```
1 # Dissimilarity matrix
2 d <- dist(data, method = "euclidean")
3 # Hierarchical clustering using Complete Linka
4 hc1 <- hclust(d, method = "complete" )
5 # Plot the obtained dendrogram
6 plot(hc1, cex = 0.6, hang = -1)
```





Another alternative is the `agnes` function. Both of these functions are quite similar; however, with the `agnes` function you can also get the agglomerative coefficient, which measures the amount of clustering structure found (values closer to 1 suggest strong clustering structure).

```
1 # Compute with agnes (make sure you have the
2 hc2 <- agnes(data, method = "complete")
3
4 # Agglomerative coefficient
5 hc2$ac
6
7 ## [1] 0.9317012
8 Let's compare the methods discussed
9
10 # vector of methods to compare
11 m <- c("average", "single", "complete", "ward")
12 names(m) <- c("average", "single", "complete", "ward")
13
14 # function to compute coefficient
15 ac <- function(x) {
16   agnes(data, method = x)$ac
17 }
18 map_dbl(m, ac)
19
20 ## from 'purrr' package
21 ## average single complete ward
22
23 ## 0.9241325 0.9215283 0.9317012 0.9493598
24
25 Ward's method gets us the highest agglomerative coefficient
26 hc3 <- agnes(data, method = "ward")
27 plotree(hc3, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```



Divisive Hierarchical Clustering

The function `'diana'` in the `cluster` package helps us perform divisive hierarchical clustering. `'diana'` works similar to `'agnes'`. However, there is no `method` argument here, and, instead of agglomerative coefficient, we have divisive coefficient.

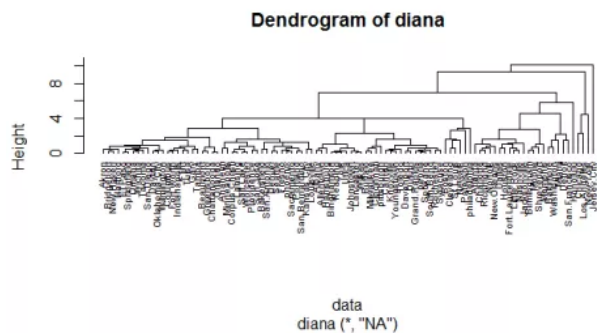
```
1 # compute divisive hierarchical clustering
2 hc4 <- diana(data)
3
4 # Divisive coefficient
```



```

7 | ## [1] 0.9305939
8 |
9 | # plot dendrogram
10 | ptree(hc4, cex = 0.6, hang = -1, main = "Den

```



A dendrogram is a cluster tree where each group is linked to two or more successor groups. These groups are nested and organized as a tree. You could manage dendrograms and do much more with them using the package “dendextend”. Check out the vignette of the package here:

https://cran.r-project.org/web/packages/dendextend/vignettes/Quick_Introduction.html

Great! So now we understand how to perform clustering and come up with dendrograms. Let us move to the final step of assigning clusters to the data points.

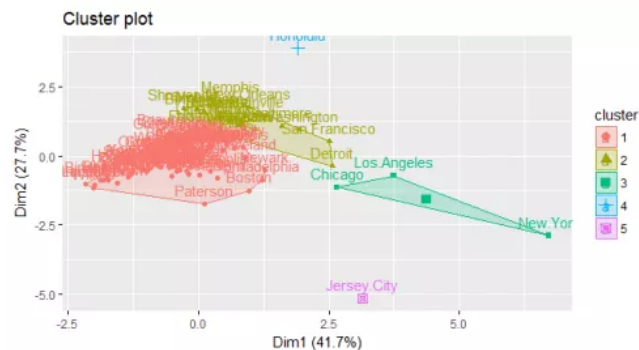
This can be done with the R function `cutree`. It cuts a tree (or dendrogram), as resulting from `hclust` (or `diana/agnes`), into several groups either by specifying the desired number of groups (`k`) or the cut height (`h`). At least one of `k` or `h` must be specified, `k` overrides `h` if both are given.

Following our demo, assign clusters for the tree obtained by `diana` function (under section Divisive Hierarchical Clustering).

```
1 | clust <- cutree(hc4, k = 5)
```

We can also use the `fviz_cluster` function from the `factoextra` package to visualize the result in a scatter plot.

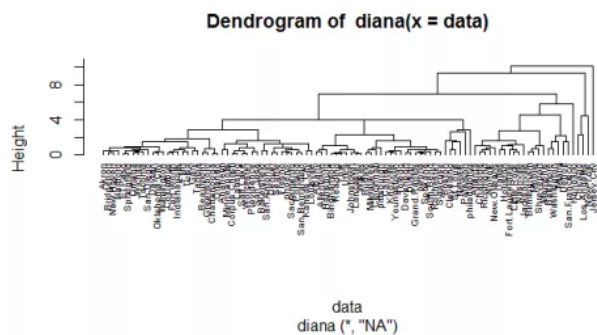
```
1 | fviz_cluster(list(data = data, cluster = clust
```



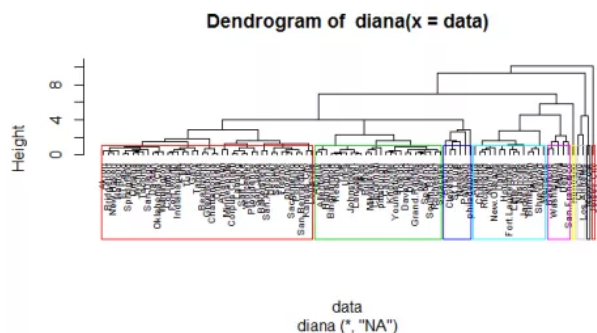
You can also visualize the clusters inside the dendrogram itself by putting borders as shown next

```
ptree(hc4, hang=-1, cex = 0.6)
```





```
rect.hclust(hc4, k = 9, border = 2:10)
```



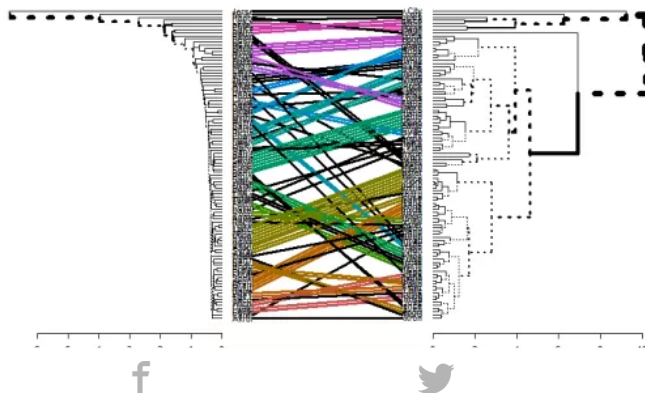
dendextend

You can do a lot of other manipulation on dendrograms with the [dendextend](#) package such as – changing the color of labels, changing the size of text of labels, changing type of line of branches, its colors, etc. You can also change the label text as well as sort it. You can see the many options in the [online vignette](#) of the package.

Another important function that the dendextend package offers is `tanglegram`. It is used to compare two dendrogram (with the same set of labels), one facing the other, and having their labels connected by lines.

As an example, let's compare the single and complete linkage methods for `agnes` function.

```
1 library(dendextend)
2
3 hc_single <- agnes(data, method = "single")
4 hc_complete <- agnes(data, method = "complete")
5
6 # converting to dendrogram objects as dendexter
7 hc_single <- as.dendrogram(hc_single)
8 hc_complete <- as.dendrogram(hc_complete)
9
10 tanglegram(hc_single, hc_complete)
```



This is useful in comparing two methods. As seen in the figure, one can relate to the methodology used for building the clusters by looking at this comparison.

You can find more functionalities of the dendextend package [here](#).

End Notes

Hope now you have a better understanding of clustering algorithms than what you started with. We discussed about Divisive and Agglomerative clustering techniques and four linkage methods namely, Single, Complete, Average and Ward's method. Next, we implemented the discussed techniques in R using a numeric dataset. Note that we didn't have any categorical variable in the dataset we used. You need to treat the categorical variables in order to incorporate them into a clustering algorithm. Lastly, we discussed a couple of plots to visualise the clusters/groups formed. Note here that we have assumed value of 'k' (number of clusters) is known. However, this is not always the case. There are a number of heuristics and rules-of-thumb for picking number of clusters. A given heuristic will work better on some datasets than others. It's best to take advantage of domain knowledge to help set the number of clusters, if that's possible. Otherwise, try a variety of heuristics, and perhaps a few different values of k.

Consolidated code

```

1  install.packages('cluster')
2  install.packages('purrr')
3  install.packages('factoextra')
4
5  library(cluster)
6  library(purrr)
7  library(factoextra)
8  data <- car::Freedman
9  data <- na.omit(data)
10 data <- scale(data)
11
12 d <- dist(data, method = "euclidean")
13 hc1 <- hclust(d, method = "complete")
14
15 plot(hc1, cex = 0.6, hang = -1)
16
17 hc2 <- agnes(data, method = "complete")
18 hc2$ac
19
20 m <- c("average", "single", "complete", "ward")
21 names(m) <- c("average", "single", "complete", "ward")
22
23 ac <- function(x) {
24   agnes(data, method = x)$ac
25 }
26
27 map_dbl(m, ac)
28
29 hc3 <- agnes(data, method = "ward")
30 pltree(hc3, cex = 0.6, hang = -1, main = "Den")
31
32 hc4 <- diana(data)
33 hc4$dc
34
35 pltree(hc4, cex = 0.6, hang = -1, main = "Den")
36
37 clust <- cutree(hc4, k = 5)
38 fviz_cluster(list(data = data, cluster = clust))
39
40 pltree(hc4, hang=-1, cex = 0.6)
41 rect.hclust(hc4, k = 9, border = 2:10)
42
43 library(dendextend)
44
45 hc_single <- agnes(data, method = "single")
46 hc_complete <- agnes(data, method = "complete")
47
48 # converting to dendrogram objects as dendexte
49 hc_single <- as.dendrogram(hc_single)

```



```
52 | tanglegram(hc_single, hc_complete)
```

Author Bio:

This article was contributed by [Perceptive Analytics](#). Amanpreet Singh, Chaitanya Sagar, Jyothirmayee Thondamallu and Saneesh Veetil contributed to this article.

Perceptive Analytics provides data analytics, business intelligence and reporting services to e-commerce, retail and pharmaceutical industries. Our client roster includes Fortune 500 and NYSE listed companies in the USA and India.



Share



Tweet

To leave a comment for the author, please follow the link and comment on their blog: [R-posts.com](#).

[R-bloggers.com](#) offers [daily e-mail updates](#) about [R](#) news and [tutorials](#) on topics such as: [Data science](#), [Big Data](#), [R jobs](#), visualization ([ggplot2](#), [Boxplots](#), [maps](#), [animation](#)), programming ([RStudio](#), [Sweave](#), [LaTeX](#), [SQL](#), [Eclipse](#), [git](#), [hadoop](#), [Web Scraping](#)) statistics ([regression](#), [PCA](#), [time series](#), [trading](#)) and more...

If you got this far, why not **subscribe for updates** from the site?
Choose your flavor: [e-mail](#), [twitter](#), [RSS](#), or [facebook](#)...

Like 375

Share

Tweet

Share

Comments are closed.

Search R-bloggers

Search..

Go

Most visited articles of the week

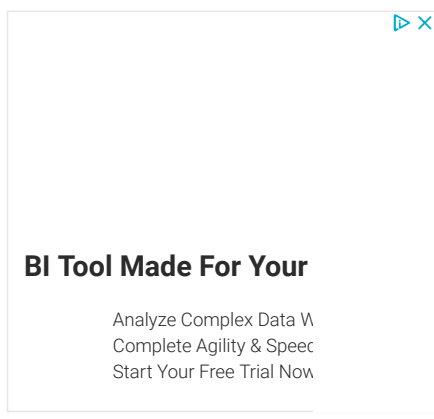
1. [How to write the first for loop in R](#)
2. [R Studio Shortcuts and Tips – part 2](#)
3. [Learning R: The Ultimate Introduction \(incl. Machine Learning!\)](#)
4. [Modern Data Science with R: A review](#)
5. [5 Ways to Subset a Data Frame in R](#)
6. [Part 2: Simple EDA in R with inspectdf](#)
7. [R – Sorting a data frame by the contents of a column](#)
8. [Using apply, sapply, lapply in R](#)
9. [Installing R packages](#)

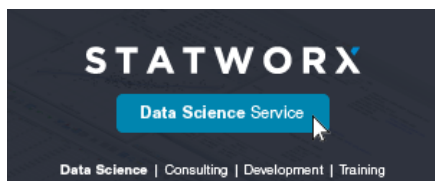
Sponsors





Quantide: statistical consulting and training





Our ads respect your privacy. Read our [Privacy Policy page](#) to learn more.

[Contact us](#) if you wish to help support R-bloggers, and place **your banner here**.

[Jobs for R users](#)

- [Customer Success Representative](#)
- [Movement Building Analyst](#)
- [Business Intelligence Analyst](#)
- [Innovation Fellow](#)
- [Postdoctoral position Stats, Comp. Biol. @ Madrid, Spain.](#)
- [Lead Data Scientist @ Washington, District of Columbia, U.S.](#)
- [PhD Fellowship @ Wallace University, US](#)

[Full list of contributing R-bloggers](#)

[R-bloggers](#) was founded by [Tal Galili](#), with gratitude to the [R](#) community.

Is powered by [WordPress](#) using a [bavotasan.com](#) design.

Copyright © 2019 **R-bloggers**. All Rights Reserved. [Terms and Conditions](#) for this website

