k-Nearest Neighbors

Nik Bear Brown

In this lesson we'll learn the theory behind using k-nearest neighbors (kNN) as a supervised classification technique. We'll then use kNN to classify the UCI wine dataset in R.

# Additional packages needed

To run the code you may need additional packages.

* If necessary install the followings packages.

install.packages("ggplot2");  
install.packages("class");

library(ggplot2)  
library(class)

# Data

We will be using the [UCI Machine Learning Repository: Wine Data Set](https://archive.ics.uci.edu/ml/datasets/Wine). These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The attributes are:  
1) Alcohol  
2) Malic acid  
3) Ash  
4) Alcalinity of ash  
5) Magnesium  
6) Total phenols  
7) Flavanoids  
8) Nonflavanoid phenols  
9) Proanthocyanins  
10) Color intensity  
11) Hue  
12) OD280/OD315 of diluted wines  
13) Proline

Feel free to tweet questions to [@NikBearBrown](<https://twitter.com/NikBearBrown>)

# Load our data  
data\_url <- 'http://nikbearbrown.com/YouTube/MachineLearning/M07/wine.csv'  
wn <- read.csv(url(data\_url))  
head(wn)

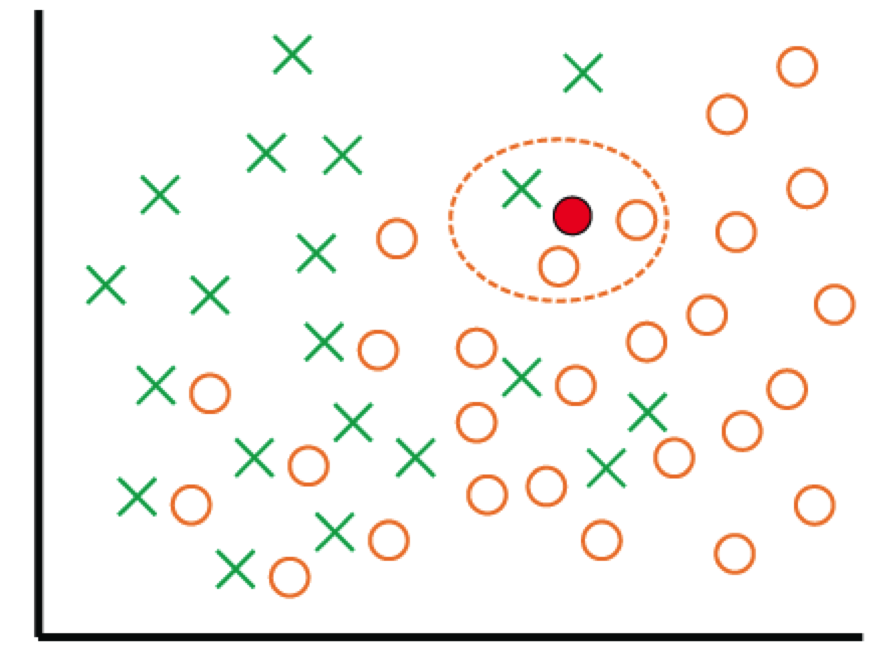
## Cultivar Alcohol Malic.acid Ash Alcalinity.ash Magnesium Total.phenols  
## 1 1 14.23 1.71 2.43 15.6 127 2.80  
## 2 1 13.20 1.78 2.14 11.2 100 2.65  
## 3 1 13.16 2.36 2.67 18.6 101 2.80  
## 4 1 14.37 1.95 2.50 16.8 113 3.85  
## 5 1 13.24 2.59 2.87 21.0 118 2.80  
## 6 1 14.20 1.76 2.45 15.2 112 3.27  
## Flavanoids Nonflavanoid.phenols Proanthocyanins Color.intensity Hue  
## 1 3.06 0.28 2.29 5.64 1.04  
## 2 2.76 0.26 1.28 4.38 1.05  
## 3 3.24 0.30 2.81 5.68 1.03  
## 4 3.49 0.24 2.18 7.80 0.86  
## 5 2.69 0.39 1.82 4.32 1.04  
## 6 3.39 0.34 1.97 6.75 1.05  
## OD280.OD315 Proline  
## 1 3.92 1065  
## 2 3.40 1050  
## 3 3.17 1185  
## 4 3.45 1480  
## 5 2.93 735  
## 6 2.85 1450

# k-Nearest Neighbors (kNN)

A simple supervised learning algorithm is [k-Nearest Neighbors](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) algorithm (k-NN). KNN is a non-parametric method used for classification and regression.

In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.



k-nearest neighbor voting

*k-nearest neighbor voting*

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN has the nice property that a labled subset of a data set could be used to label the whole data set. This is especially important in the analysis of “big-data.” Most big-data sets are only partially labled, as labeling often librarys human annotation. While many are looking to unsupervised learning the ‘future’ of big-data, k-Nearest Neighbors is an instance of a supervised learning algorithm that can be used with big-data.

The kNN classification problem is to find the k nearest data points in a data set to a given query data point. The point is then assigned to the group by a majority "vote." For this reason, pick an odd k is prefered as the odd vote can break ties. This operation is also known as a kNN join, and can be defined as: given two data sets and , find the k nearest Neighbor from for every object in . refers to data that has already been classified, the training set. refers to data that is needs to be classified.

The kNN algorithm can be fairly expensive, especially if one chooses a large k, as the k-nearest neighbors in for every point in needs to be calculated.

## Nearest neighbor search

A simple solution to finding nearest neighbors is to compute the distance from the each point in to every point in and keeping track of the "best so far". This algorithm, sometimes referred to as the naive approach, has a running time of O(|R||S|).

One can speed up the search to retrieve a "good guess" of the nearest neighbor. This is often done be limiting the search to a preset radius around a point culling out most of the points in . If k neighbors aren't not found in the radius then the bound can be iteratively expanded until k are found. Altnernatively, the vote could be made using fewer points when k points aren't found within a radius r.

## k-Nearest Neighbors is nonparametric "lazy learning "

K-Nearest Neighbors algorithm (kNN) is a nonparametric method for classifying objects based on the closest training examples in the feature space. kNN is nonparametric becuase it d oes not involve any estimation of parameters. This is sometimes called "lazy learning" or instance-based learning, as the mapping is approximated locally and all computation is deferred until classification.

# kNN Classification and Distance Metrics

Neighbors are defined by a distance or dissimilarity measure. In essence, the only thing that kNN librarys is some measure of "closeness" of the points in and . Any distance metric or dissimilarity measure can be used. The most common being the Euclidean distance between the points and is given by the pythagorean formula:

Any measure of "closeness", distance or dissimilarity measure can be used. For example,

* [Chebyshev distance](https://en.wikipedia.org/wiki/Chebyshev_distance) - measures distance assuming only the most significant dimension is relevant.
* [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance) - identifies the difference bit by bit of two strings
* [Mahalanobis distance](https://en.wikipedia.org/wiki/Mahalanobis_distance) - normalizes based on a covariance matrix to make the distance metric scale-invariant.
* [Manhattan distance](https://en.wikipedia.org/wiki/Taxicab_geometry) - measures distance following only axis-aligned directions.
* [Minkowski distance](https://en.wikipedia.org/wiki/Minkowski_distance) - is a generalization that unifies Euclidean distance, Manhattan distance, and Chebyshev distance

.. and many more.

# kNN Algorithm

## Distance function

The distance function depends on your needs, but in general choosing features and distance metrics in which being "close" makes some sense in your domain are the distance metrics and features to choose. The type of variable, categorical, ordinal or nominal should be considered when choosing a sensible measure of closeness.

## k nearest neighbors

Given an data point p, a training data set , and an integer k, the k nearest neighbors of p from , denoted as kNN(p, S), are a set of k objects from such that:

## kNN join

Given two data sets R and S (where S is a training data set) and an integer k, the kNN join of R and S is defined as:

kNNjoin(R, S) = {(r, s)|∀r ∊ R, ∀s ∊ kNN(r, S)}

Basically, this combines each object r ∊ R with its k nearest neighbors from S.

# Steps in kNN Classification

The kNN algorithm can be summarized in the following simple steps:

* Determine k (the selection of k depends on your data and project libraryments; there is no magic formula for k).
* Calculate the distances between the new input and all the training data (as with k, the selection of a distance function also depends on the type of data).
* Sort the distance and determine the k nearest neighbors based on the kth minimum distance.
* Gather the categories of those neighbors.
* Determine the category based on majority vote.

# k-Nearest Neighbors (kNN) in R

k-Nearest Neighbors (kNN) in R

head(wn)

## Cultivar Alcohol Malic.acid Ash Alcalinity.ash Magnesium Total.phenols  
## 1 1 14.23 1.71 2.43 15.6 127 2.80  
## 2 1 13.20 1.78 2.14 11.2 100 2.65  
## 3 1 13.16 2.36 2.67 18.6 101 2.80  
## 4 1 14.37 1.95 2.50 16.8 113 3.85  
## 5 1 13.24 2.59 2.87 21.0 118 2.80  
## 6 1 14.20 1.76 2.45 15.2 112 3.27  
## Flavanoids Nonflavanoid.phenols Proanthocyanins Color.intensity Hue  
## 1 3.06 0.28 2.29 5.64 1.04  
## 2 2.76 0.26 1.28 4.38 1.05  
## 3 3.24 0.30 2.81 5.68 1.03  
## 4 3.49 0.24 2.18 7.80 0.86  
## 5 2.69 0.39 1.82 4.32 1.04  
## 6 3.39 0.34 1.97 6.75 1.05  
## OD280.OD315 Proline  
## 1 3.92 1065  
## 2 3.40 1050  
## 3 3.17 1185  
## 4 3.45 1480  
## 5 2.93 735  
## 6 2.85 1450

summary(wn)

## Cultivar Alcohol Malic.acid Ash   
## Min. :1.000 Min. :11.03 Min. :0.740 Min. :1.360   
## 1st Qu.:1.000 1st Qu.:12.36 1st Qu.:1.603 1st Qu.:2.210   
## Median :2.000 Median :13.05 Median :1.865 Median :2.360   
## Mean :1.938 Mean :13.00 Mean :2.336 Mean :2.367   
## 3rd Qu.:3.000 3rd Qu.:13.68 3rd Qu.:3.083 3rd Qu.:2.558   
## Max. :3.000 Max. :14.83 Max. :5.800 Max. :3.230   
## Alcalinity.ash Magnesium Total.phenols Flavanoids   
## Min. :10.60 Min. : 70.00 Min. :0.980 Min. :0.340   
## 1st Qu.:17.20 1st Qu.: 88.00 1st Qu.:1.742 1st Qu.:1.205   
## Median :19.50 Median : 98.00 Median :2.355 Median :2.135   
## Mean :19.49 Mean : 99.74 Mean :2.295 Mean :2.029   
## 3rd Qu.:21.50 3rd Qu.:107.00 3rd Qu.:2.800 3rd Qu.:2.875   
## Max. :30.00 Max. :162.00 Max. :3.880 Max. :5.080   
## Nonflavanoid.phenols Proanthocyanins Color.intensity Hue   
## Min. :0.1300 Min. :0.410 Min. : 1.280 Min. :0.4800   
## 1st Qu.:0.2700 1st Qu.:1.250 1st Qu.: 3.220 1st Qu.:0.7825   
## Median :0.3400 Median :1.555 Median : 4.690 Median :0.9650   
## Mean :0.3619 Mean :1.591 Mean : 5.058 Mean :0.9574   
## 3rd Qu.:0.4375 3rd Qu.:1.950 3rd Qu.: 6.200 3rd Qu.:1.1200   
## Max. :0.6600 Max. :3.580 Max. :13.000 Max. :1.7100   
## OD280.OD315 Proline   
## Min. :1.270 Min. : 278.0   
## 1st Qu.:1.938 1st Qu.: 500.5   
## Median :2.780 Median : 673.5   
## Mean :2.612 Mean : 746.9   
## 3rd Qu.:3.170 3rd Qu.: 985.0   
## Max. :4.000 Max. :1680.0

length(wn)

## [1] 14

names(wn)

## [1] "Cultivar" "Alcohol" "Malic.acid"   
## [4] "Ash" "Alcalinity.ash" "Magnesium"   
## [7] "Total.phenols" "Flavanoids" "Nonflavanoid.phenols"  
## [10] "Proanthocyanins" "Color.intensity" "Hue"   
## [13] "OD280.OD315" "Proline"

table(wn$Cultivar)

##   
## 1 2 3   
## 59 71 48

wn$Cultivar

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2  
## [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3  
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [176] 3 3 3

length(wn$Cultivar)

## [1] 178

You can also embed plots, for example:

shuff<-runif(nrow(wn))  
shuff

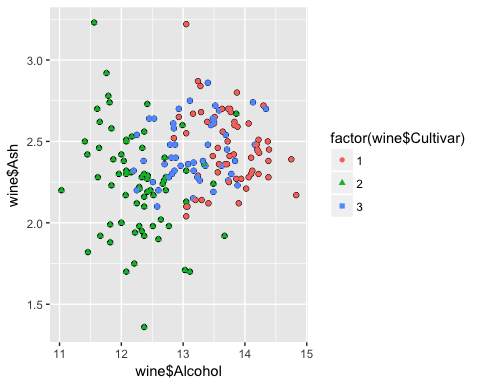
## [1] 0.879416736 0.465057831 0.203559469 0.857664812 0.446806759  
## [6] 0.993477742 0.357465548 0.431790816 0.601351195 0.299737555  
## [11] 0.065804084 0.950887116 0.072150465 0.090625413 0.893852573  
## [16] 0.762482274 0.574194463 0.144995949 0.767682299 0.141855116  
## [21] 0.339898303 0.417410148 0.072240409 0.639542649 0.452636903  
## [26] 0.644134149 0.766408274 0.332183770 0.199284510 0.843122139  
## [31] 0.848746219 0.930845051 0.159976850 0.469081585 0.364277273  
## [36] 0.305499332 0.725547439 0.585616025 0.428702026 0.401016329  
## [41] 0.228177729 0.563623512 0.698064011 0.299441274 0.415499139  
## [46] 0.520055889 0.153109725 0.739615300 0.214080374 0.162166949  
## [51] 0.082340735 0.375922605 0.539000284 0.056279513 0.429294613  
## [56] 0.129012763 0.134195874 0.257771960 0.311076509 0.578258805  
## [61] 0.719523014 0.801622628 0.483144780 0.911315740 0.160058855  
## [66] 0.858762778 0.490823320 0.893487326 0.208629933 0.896674890  
## [71] 0.425024987 0.862942249 0.753033714 0.882109831 0.149832761  
## [76] 0.360869391 0.552097062 0.216466527 0.924754099 0.523228708  
## [81] 0.753214470 0.167602310 0.941527774 0.883650711 0.263295079  
## [86] 0.502849103 0.072817653 0.355810725 0.803872135 0.904544791  
## [91] 0.592107939 0.701228071 0.590443582 0.227611112 0.502662175  
## [96] 0.036936287 0.529074259 0.250616603 0.262424937 0.105996758  
## [101] 0.699227039 0.362480839 0.108325217 0.355212895 0.496549245  
## [106] 0.118456276 0.557677326 0.085181800 0.963453403 0.152147554  
## [111] 0.635562747 0.676331046 0.721586145 0.080935735 0.387654620  
## [116] 0.098254921 0.466039377 0.530348545 0.454439861 0.057937427  
## [121] 0.280140246 0.502869837 0.988537036 0.808908303 0.244104668  
## [126] 0.518662793 0.820486286 0.164370501 0.698453487 0.260842148  
## [131] 0.960960730 0.416402775 0.449777470 0.226571414 0.532649183  
## [136] 0.895524792 0.026771151 0.845000878 0.549343386 0.006257911  
## [141] 0.789321971 0.071647472 0.927306648 0.445342876 0.876622166  
## [146] 0.097234364 0.329052628 0.832512605 0.267979169 0.042547621  
## [151] 0.635819976 0.773744501 0.858690942 0.494700995 0.019334461  
## [156] 0.828572836 0.709275471 0.661391530 0.914107190 0.710001386  
## [161] 0.453391860 0.868659350 0.823411378 0.920017180 0.135356334  
## [166] 0.305071812 0.655010787 0.261217887 0.459005100 0.818608793  
## [171] 0.955147237 0.794117657 0.337661183 0.099180498 0.934245064  
## [176] 0.510923482 0.433632229 0.539134596

wine<-wn[order(shuff),]  
wine$Cultivar

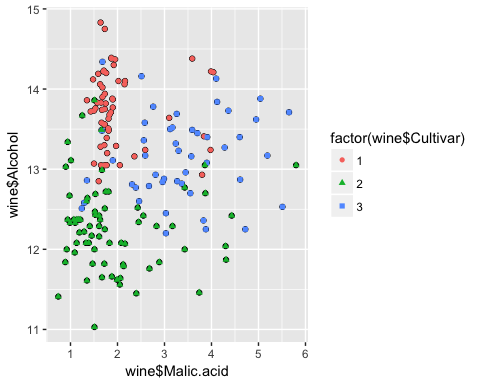
## [1] 3 3 3 2 3 1 2 1 3 1 1 2 2 1 2 1 3 2 3 2 2 2 1 1 3 1 1 2 2 1 1 2 1 2 2  
## [36] 1 1 2 1 2 3 2 1 2 2 1 2 3 2 2 3 2 1 1 3 1 1 3 1 3 1 2 2 1 2 2 1 1 2 1  
## [71] 1 3 1 2 1 1 1 3 3 1 3 1 3 2 3 1 2 1 2 2 3 2 2 2 2 3 2 1 2 2 2 3 1 3 3  
## [106] 2 2 1 1 2 1 2 2 1 2 3 1 1 3 3 2 1 2 2 2 3 3 2 2 1 1 2 2 1 1 1 3 3 3 2  
## [141] 2 2 3 2 3 3 3 1 3 1 1 3 2 2 3 3 1 2 2 2 1 3 2 2 2 3 3 2 3 1 3 2 1 3 3  
## [176] 2 2 1

You can also embed plots, for example:

qplot(wine$Alcohol,wine$Ash,data=wine)+geom\_point(aes(colour = factor(wine$Cultivar),shape = factor(wine$Cultivar)))



qplot(wine$Malic.acid,wine$Alcohol,data=wine)+geom\_point(aes(colour = factor(wine$Cultivar),shape = factor(wine$Cultivar)))



summary(wine)

## Cultivar Alcohol Malic.acid Ash   
## Min. :1.000 Min. :11.03 Min. :0.740 Min. :1.360   
## 1st Qu.:1.000 1st Qu.:12.36 1st Qu.:1.603 1st Qu.:2.210   
## Median :2.000 Median :13.05 Median :1.865 Median :2.360   
## Mean :1.938 Mean :13.00 Mean :2.336 Mean :2.367   
## 3rd Qu.:3.000 3rd Qu.:13.68 3rd Qu.:3.083 3rd Qu.:2.558   
## Max. :3.000 Max. :14.83 Max. :5.800 Max. :3.230   
## Alcalinity.ash Magnesium Total.phenols Flavanoids   
## Min. :10.60 Min. : 70.00 Min. :0.980 Min. :0.340   
## 1st Qu.:17.20 1st Qu.: 88.00 1st Qu.:1.742 1st Qu.:1.205   
## Median :19.50 Median : 98.00 Median :2.355 Median :2.135   
## Mean :19.49 Mean : 99.74 Mean :2.295 Mean :2.029   
## 3rd Qu.:21.50 3rd Qu.:107.00 3rd Qu.:2.800 3rd Qu.:2.875   
## Max. :30.00 Max. :162.00 Max. :3.880 Max. :5.080   
## Nonflavanoid.phenols Proanthocyanins Color.intensity Hue   
## Min. :0.1300 Min. :0.410 Min. : 1.280 Min. :0.4800   
## 1st Qu.:0.2700 1st Qu.:1.250 1st Qu.: 3.220 1st Qu.:0.7825   
## Median :0.3400 Median :1.555 Median : 4.690 Median :0.9650   
## Mean :0.3619 Mean :1.591 Mean : 5.058 Mean :0.9574   
## 3rd Qu.:0.4375 3rd Qu.:1.950 3rd Qu.: 6.200 3rd Qu.:1.1200   
## Max. :0.6600 Max. :3.580 Max. :13.000 Max. :1.7100   
## OD280.OD315 Proline   
## Min. :1.270 Min. : 278.0   
## 1st Qu.:1.938 1st Qu.: 500.5   
## Median :2.780 Median : 673.5   
## Mean :2.612 Mean : 746.9   
## 3rd Qu.:3.170 3rd Qu.: 985.0   
## Max. :4.000 Max. :1680.0

You can also embed plots, for example:

wine.scaled<-as.data.frame(lapply(wine[,c(2:14)], scale))  
head(wine.scaled)

## Alcohol Malic.acid Ash Alcalinity.ash Magnesium  
## 1 -0.1978477 0.5582544 0.88751034 1.3489954 0.08810981  
## 2 -0.5181131 -0.9366262 -0.97146956 0.1512342 0.22814148  
## 3 -0.9246039 2.1336974 0.63235624 0.4506745 -0.75208020  
## 4 -0.6536100 -0.7307445 -0.60696370 -0.1482061 4.35907571  
## 5 0.0977820 1.3996841 -0.02375431 0.6003946 0.92829983  
## 6 0.9477173 -0.3905920 1.14266445 -0.7171427 1.06833150  
## Total.phenols Flavanoids Nonflavanoid.phenols Proanthocyanins  
## 1 0.03976608 -1.4309028 1.3510772 -1.3643519  
## 2 -1.30240632 -1.4509256 1.3510772 -0.3335300  
## 3 -1.46218874 -1.5610513 1.3510772 -1.3818234  
## 4 0.32737445 0.2410054 -0.3363022 2.9511225  
## 5 -1.41425402 -0.6400001 -0.1755994 -0.7877905  
## 6 1.12628658 0.7615996 0.2261576 0.1556735  
## Color.intensity Hue OD280.OD315 Proline  
## 1 -0.05956551 -0.2950911 -0.65026988 -0.49822018  
## 2 1.09646103 -1.6513403 -1.49535170 -0.33944338  
## 3 -0.52111342 -0.9075907 -1.88972321 -0.08540051  
## 4 -1.06030491 0.8861582 0.02579557 0.60369079  
## 5 1.87289677 -1.6950903 -1.80521503 -0.62524162  
## 6 0.53570189 0.7549083 0.44833648 1.99457553

summary(wine.scaled)

## Alcohol Malic.acid Ash   
## Min. :-2.42739 Min. :-1.4290 Min. :-3.66881   
## 1st Qu.:-0.78603 1st Qu.:-0.6569 1st Qu.:-0.57051   
## Median : 0.06083 Median :-0.4219 Median :-0.02375   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.83378 3rd Qu.: 0.6679 3rd Qu.: 0.69615   
## Max. : 2.25341 Max. : 3.1004 Max. : 3.14745   
## Alcalinity.ash Magnesium Total.phenols   
## Min. :-2.663505 Min. :-2.0824 Min. :-2.10132   
## 1st Qu.:-0.687199 1st Qu.:-0.8221 1st Qu.:-0.88298   
## Median : 0.001514 Median :-0.1219 Median : 0.09569   
## Mean : 0.000000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.600395 3rd Qu.: 0.5082 3rd Qu.: 0.80672   
## Max. : 3.145637 Max. : 4.3591 Max. : 2.53237   
## Flavanoids Nonflavanoid.phenols Proanthocyanins   
## Min. :-1.6912 Min. :-1.8630 Min. :-2.06321   
## 1st Qu.:-0.8252 1st Qu.:-0.7381 1st Qu.:-0.59560   
## Median : 0.1059 Median :-0.1756 Median :-0.06272   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.8467 3rd Qu.: 0.6078 3rd Qu.: 0.62741   
## Max. : 3.0542 Max. : 2.3956 Max. : 3.47527   
## Color.intensity Hue OD280.OD315 Proline   
## Min. :-1.6297 Min. :-2.08884 Min. :-1.8897 Min. :-1.4890   
## 1st Qu.:-0.7929 1st Qu.:-0.76540 1st Qu.:-0.9496 1st Qu.:-0.7824   
## Median :-0.1588 Median : 0.03303 Median : 0.2371 Median :-0.2331   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.4926 3rd Qu.: 0.71116 3rd Qu.: 0.7864 3rd Qu.: 0.7561   
## Max. : 3.4258 Max. : 3.29241 Max. : 1.9554 Max. : 2.9631

You can also embed plots, for example:

normalize<- function(x) {  
 return((x-min(x))/(max(x)-min(x)))  
}  
wine.normalized<-as.data.frame(lapply(wine[,c(2:14)],normalize))  
head(wine.normalized)

## Alcohol Malic.acid Ash Alcalinity.ash Magnesium Total.phenols  
## 1 0.4763158 0.4387352 0.6684492 0.6907216 0.3369565 0.4620690  
## 2 0.4078947 0.1086957 0.3957219 0.4845361 0.3586957 0.1724138  
## 3 0.3210526 0.7865613 0.6310160 0.5360825 0.2065217 0.1379310  
## 4 0.3789474 0.1541502 0.4491979 0.4329897 1.0000000 0.5241379  
## 5 0.5394737 0.6245059 0.5347594 0.5618557 0.4673913 0.1482759  
## 6 0.7210526 0.2292490 0.7058824 0.3350515 0.4891304 0.6965517  
## Flavanoids Nonflavanoid.phenols Proanthocyanins Color.intensity  
## 1 0.05485232 0.7547170 0.1261830 0.3105802  
## 2 0.05063291 0.7547170 0.3123028 0.5392491  
## 3 0.02742616 0.7547170 0.1230284 0.2192833  
## 4 0.40717300 0.3584906 0.9053628 0.1126280  
## 5 0.22151899 0.3962264 0.2302839 0.6928328  
## 6 0.51687764 0.4905660 0.4006309 0.4283276  
## Hue OD280.OD315 Proline  
## 1 0.33333333 0.32234432 0.2225392  
## 2 0.08130081 0.10256410 0.2582026  
## 3 0.21951220 0.00000000 0.3152639  
## 4 0.55284553 0.49816850 0.4700428  
## 5 0.07317073 0.02197802 0.1940086  
## 6 0.52845528 0.60805861 0.7824536

summary(wine.normalized)

## Alcohol Malic.acid Ash Alcalinity.ash   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3507 1st Qu.:0.1705 1st Qu.:0.4545 1st Qu.:0.3402   
## Median :0.5316 Median :0.2223 Median :0.5348 Median :0.4588   
## Mean :0.5186 Mean :0.3155 Mean :0.5382 Mean :0.4585   
## 3rd Qu.:0.6967 3rd Qu.:0.4629 3rd Qu.:0.6404 3rd Qu.:0.5619   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Magnesium Total.phenols Flavanoids Nonflavanoid.phenols  
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.1957 1st Qu.:0.2629 1st Qu.:0.1825 1st Qu.:0.2642   
## Median :0.3043 Median :0.4741 Median :0.3787 Median :0.3962   
## Mean :0.3233 Mean :0.4535 Mean :0.3564 Mean :0.4375   
## 3rd Qu.:0.4022 3rd Qu.:0.6276 3rd Qu.:0.5348 3rd Qu.:0.5802   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Proanthocyanins Color.intensity Hue OD280.OD315   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2650 1st Qu.:0.1655 1st Qu.:0.2459 1st Qu.:0.2445   
## Median :0.3612 Median :0.2910 Median :0.3943 Median :0.5531   
## Mean :0.3725 Mean :0.3224 Mean :0.3882 Mean :0.4915   
## 3rd Qu.:0.4858 3rd Qu.:0.4198 3rd Qu.:0.5203 3rd Qu.:0.6960   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Proline   
## Min. :0.0000   
## 1st Qu.:0.1587   
## Median :0.2821   
## Mean :0.3344   
## 3rd Qu.:0.5043   
## Max. :1.0000

nrow(wine)

## [1] 178

You can also embed plots, for example:

wine.normalized.train<-wine.normalized[1:150,]  
wine.normalized.test<-wine.normalized[151:178,]  
wine.normalized.train.target<-wine[1:150,c(1)]  
wine.normalized.test.target<-wine[151:178,c(1)]  
wine.normalized.test.target

## [1] 1 3 2 2 3 3 1 2 2 2 1 3 2 2 2 3 3 2 3 1 3 2 1 3 3 2 2 1

k<-5  
knn.m1<-knn(train = wine.normalized.train, test = wine.normalized.test,wine.normalized.train.target,k)  
knn.m1

## [1] 1 3 2 1 3 3 1 1 3 2 1 3 2 2 2 3 3 2 3 1 3 2 1 3 3 2 2 1  
## Levels: 1 2 3

length(knn.m1)

## [1] 28

cm<-table(wine.normalized.test.target,knn.m1)  
cm

## knn.m1  
## wine.normalized.test.target 1 2 3  
## 1 6 0 0  
## 2 2 9 1  
## 3 0 0 10

# Resources

* [Using R For k-Nearest Neighbors (KNN)](http://blog.datacamp.com/machine-learning-in-r/)
* [Using the k-Nearest Neighbors Algorithm in R](http://blog.webagesolutions.com/archives/1164)
* [kNN PSU](https://onlinecourses.science.psu.edu/stat857/node/129)