

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](#). 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/DATA/wine.csv'
wn <- read.csv(url(data_url))
head(wn)

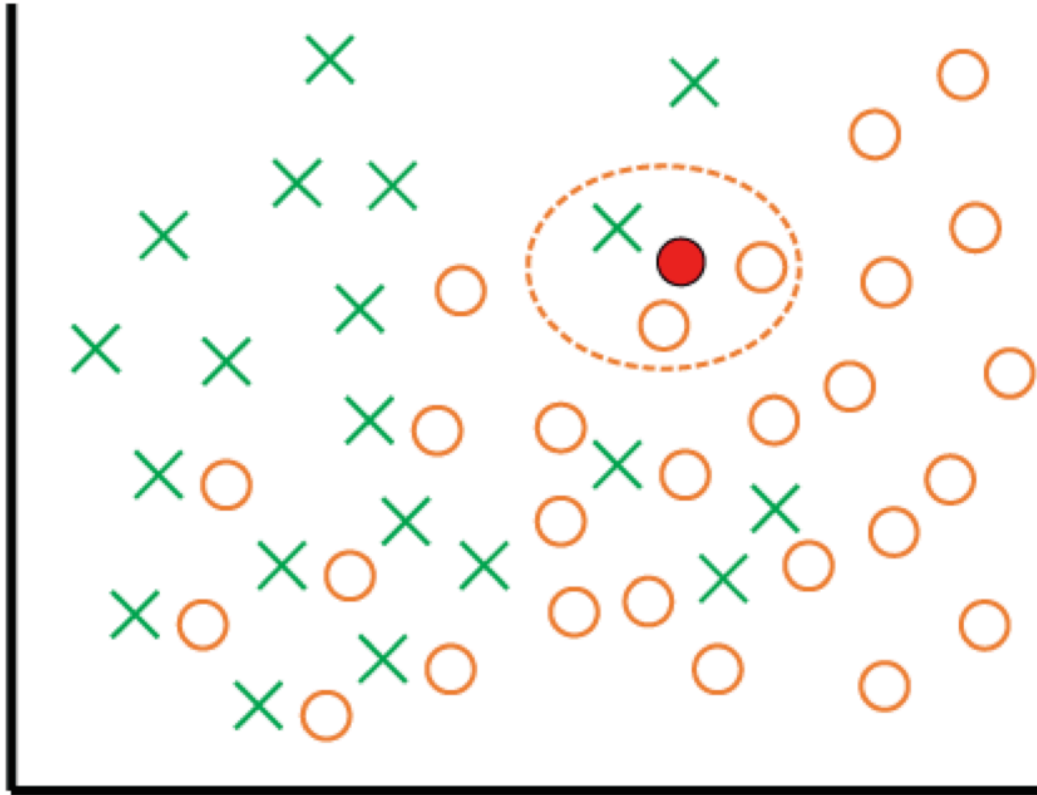
##   Cultivar Alcohol Malic.acid  Ash Alcalinity.ash Magnesium
Total.phenols
## 1         1   14.23         1.71 2.43         15.6         127
2.80
## 2         1   13.20         1.78 2.14         11.2         100
2.65
## 3         1   13.16         2.36 2.67         18.6         101
2.80
## 4         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](#) 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 labeled 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 labeled, as labeling often requires 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 preferred 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 R and S, find the k nearest Neighbor from S for every object in R. S refers to data that

has already been classified, the training set. R 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 S for every point in R needs to be calculated.

Nearest neighbor search

A simple solution to finding nearest neighbors is to compute the distance from the each point in S to every point in R 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 by limiting the search to a preset radius around a point culling out most of the points in S . If k neighbors aren't found in the radius then the bound can be iteratively expanded until k are found. Alternatively, 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 because it does 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 S and R . Any distance metric or dissimilarity measure can be used. The most common being the Euclidean distance between the points $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ is given by the pythagorean formula:

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}. \end{aligned}$$

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

- **Chebyshev distance** - measures distance assuming only the most significant dimension is relevant.

- **Hamming distance** - identifies the difference bit by bit of two strings
- **Mahalanobis distance** - normalizes based on a covariance matrix to make the distance metric scale-invariant.
- **Manhattan distance** - measures distance following only axis-aligned directions.
- **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 S , and an integer k , the k nearest neighbors of p from S , denoted as $kNN(p, S)$, are a set of k objects from S such that:

$$\forall o \in kNN(p, S), \forall s \in S - kNN(p, S), |o, p| \leq |s, p|$$

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:

$$kNNjoin(R, S) = \{(r, s) | \forall r \in R, \forall s \in kNN(r, S)\}$$

Basically, this combines each object $r \in R$ 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)

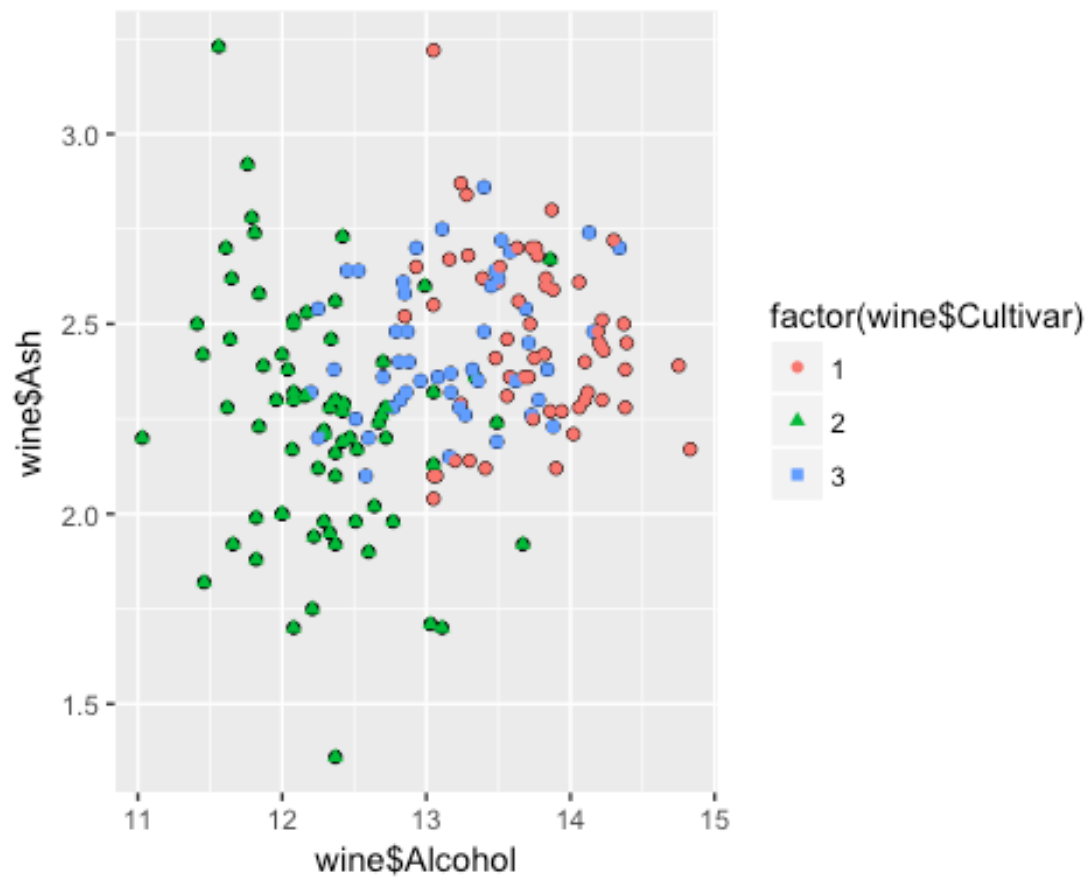

```
## [121] 0.421086790 0.613064781 0.444991069 0.572699155 0.472035346
## [126] 0.976167009 0.185936971 0.552378487 0.962700487 0.120071901
## [131] 0.215915280 0.418730139 0.083813882 0.388036222 0.518222434
## [136] 0.390524538 0.317318922 0.799042200 0.443909097 0.337957258
## [141] 0.682974269 0.576862602 0.735628112 0.194256585 0.943513720
## [146] 0.348561297 0.923960765 0.884016803 0.464547947 0.398462793
## [151] 0.600694906 0.217434656 0.745211611 0.130370027 0.377442376
## [156] 0.065865367 0.406120129 0.086493136 0.230928354 0.269886517
## [161] 0.070678699 0.139455020 0.999854616 0.412647564 0.863008609
## [166] 0.574303058 0.376764163 0.770976192 0.358862009 0.656123992
## [171] 0.567170520 0.034514746 0.004453351 0.177997533 0.578669026
## [176] 0.606805602 0.745585339 0.969962326

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

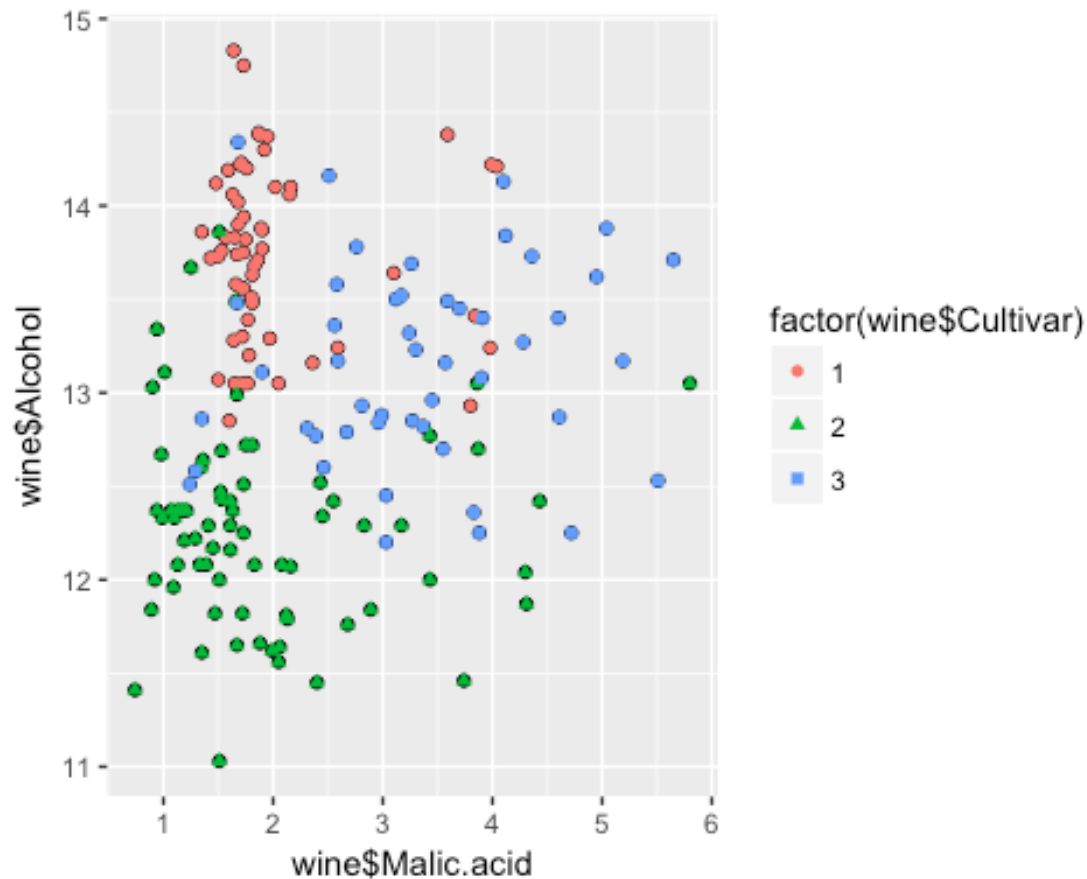
## [1] 3 1 1 3 1 1 2 1 3 3 2 3 3 2 1 1 2 1 2 1 1 3 3 2 1 2 2 2 2 1 2
1 2 3 2
## [36] 2 3 2 1 3 3 1 3 2 2 2 1 2 3 2 2 3 1 1 1 2 3 3 2 2 1 3 2 2 3 3
2 2 3 3
## [71] 3 3 3 3 2 2 1 1 3 2 2 3 2 1 2 1 1 3 1 2 2 1 1 1 1 2 2 2 1 2 3
2 3 3 1
## [106] 3 2 2 2 3 2 2 2 3 2 1 2 1 1 3 1 1 2 1 3 2 1 2 1 1 2 2 2 2 3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

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
## Alkalinity.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 Alkalinity.ash  Magnesium
## 1  1.42811545  0.15544223  0.4136527    0.151234178 -0.61204853
## 2 -0.08698653  1.31017034  1.0333127   -0.267982252  0.15812565
## 3  0.89844565 -0.74864721  1.2155656    0.899834945  0.08810981
## 4 -0.28407297  0.04802567 -0.3153590    0.001514024 -0.96212770
## 5  0.29486844  1.47129519 -0.2789084   -0.597366590  0.22814148
## 6  0.61513390 -0.47115441  0.8875103    0.151234178 -0.26196936
## Total.phenols Flavanoids Nonflavanoid.phenols Proanthocyanins
## 1   -0.9828415 -1.3307885          0.6279146    -0.6130749
## 2    0.1835703  0.3811654          -0.8987620     0.6798202
## 3    1.1262866  1.2221252          -0.5773564     1.3786825
## 4   -1.4462105 -1.5210056          0.9493202    -1.6613683
## 5    0.5510698  0.6014167          -0.3363022     0.1207304
## 6    0.3753092  0.5813939          -0.6577078     0.1207304
## Color.intensity      Hue OD280.OD315      Proline
## 1    2.0023027 -1.4763404   -1.2699965  -0.27593266
## 2   -0.2407338  0.3174085    1.2793336  0.07337629
## 3    0.2768900  1.0174081    0.1384731  1.70877729
## 4    2.0885729 -1.6950903   -1.3826741 -0.87928449
## 5   -0.3011233 -0.6013409    0.5469294 -0.21242195
## 6   -0.6634600  0.7111583    1.7018745  0.31154148
```

```
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      : 0.00000      Mean      : 0.00000
## 3rd Qu.: 0.83378      3rd Qu.: 0.6679      3rd Qu.: 0.69615
## Max.      : 2.25341      Max.      : 3.1004      Max.      : 3.14745
## Alkalinity.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 Alkalinity.ash Magnesium
Total.phenols
## 1 0.8236842  0.3498024 0.5989305      0.4845361 0.2282609
0.2413793
## 2 0.5000000  0.6047431 0.6898396      0.4123711 0.3478261
0.4931034
## 3 0.7105263  0.1501976 0.7165775      0.6134021 0.3369565
0.6965517
## 4 0.4578947  0.3260870 0.4919786      0.4587629 0.1739130
0.1413793
## 5 0.5815789  0.6403162 0.4973262      0.3556701 0.3586957
0.5724138
## 6 0.6500000  0.2114625 0.6684492      0.4845361 0.2826087
0.5344828
##   Flavanoids Nonflavanoid.phenols Proanthocyanins Color.intensity
```

```
## 1 0.07594937      0.5849057      0.26182965      0.7184300
## 2 0.43670886      0.2264151      0.49526814      0.2747440
## 3 0.61392405      0.3018868      0.62145110      0.3771331
## 4 0.03586498      0.6603774      0.07255521      0.7354948
## 5 0.48312236      0.3584906      0.39432177      0.2627986
## 6 0.47890295      0.2830189      0.39432177      0.1911263
```

```
##      Hue OD280.OD315      Proline
```

```
## 1 0.11382114      0.1611722 0.2724679
## 2 0.44715447      0.8241758 0.3509272
## 3 0.57723577      0.5274725 0.7182596
## 4 0.07317073      0.1318681 0.1369472
## 5 0.27642276      0.6336996 0.2867332
## 6 0.52032520      0.9340659 0.4044223
```

```
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 2 1 1 2 2 2 2 1 3 2 1 1 2 1 3 1 2 3 2 1 3 2 1 1 3 2 3

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

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

length(knn.m1)

## [1] 28

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

##
## wine.normalized.test.target knn.m1
## 1 11 0 0
## 2 0 10 1
## 3 0 0 6

```

Resources

- [Using R For k-Nearest Neighbors \(KNN\)](#)
- [Using the k-Nearest Neighbors Algorithm in R](#)
- [kNN PSU](#)