# MLA4: It's tough to make predictions, especially about the future.

*Mariano Dominguez*

*August 25, 2015*

---

### Overview of Supervised Learning

- There is a set of variables that might be denoted as **inputs**, which are measured or preset. These have some influence on one or more **outputs**.
- The **goal** is to use the inputs **to predict** the values of the outputs.
- In the statistical literature the inputs are often called **predictors**, a term that we will use interchangeably with inputs, and more classically the **independent variables**.
- In the pattern recognition literature the term **features** is preferred. The outputs are called the **responses**, or classically the **dependent variables**

---

### Linear Models and Least Squares

- The linear model has been a mainstay of statistics for over the past 30 years and remains one of the most imortant tools. Gievn a vector of inputs $X^T = (X_1, X_2, ..., X_p)$, we predict the output $Y$ via the model: $\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$

The term $\hat{\beta}_0$ is the intercept, also known as the **bias** in machine learning. Often it is convenient to include the constant variable 1 in $X$, include $\hat{\beta}_0$ in the vector of coefficients $\hat{\beta}$, and then write the linear model in vector form as an inner product: $\hat{Y} = X^T\hat{\beta}$.

Specifying statistical models like regression models is quite easy in R. An R formula is written as "y ~ model", where "y" will be the response variable and "model" will be all of the predictors that are to be included in the model.

We will use the linear model framework, hence we will be using the lm function.

---

### Simple Linear Regression

We will use 92 stars from the Hipparcos dataset that are associated with the Hyades. Based on the values of right ascension, declination, principal motion of right ascension, and principal motion of declination. We exclude one additional star with a large error of parallax measurement:

```
loc <- "http://astrostatistics.psu.edu/datasets/"
hip <- read.table(paste(loc,"HIP_star.dat",sep=""),
header=T,fill=T)
attach(hip)
```

```
## The following objects are masked from hip (pos = 3):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 4):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 5):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 6):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 7):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 8):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 9):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 10):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 11):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 13):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 14):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 15):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 16):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 17):
##
```

```
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 18):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 19):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
##
## The following objects are masked from hip (pos = 20):
##
##     B.V, DE, e_Plx, HIP, Plx, pmDE, pmRA, RA, Vmag
```

```r
filter1 <- (RA>50 & RA<100 & DE>0 & DE<25)
filter2 <- (pmRA>90 & pmRA<130 & pmDE>-60 & pmDE< -10)
filter <- filter1 & filter2 & (e_Plx<5)
sum(filter)
```

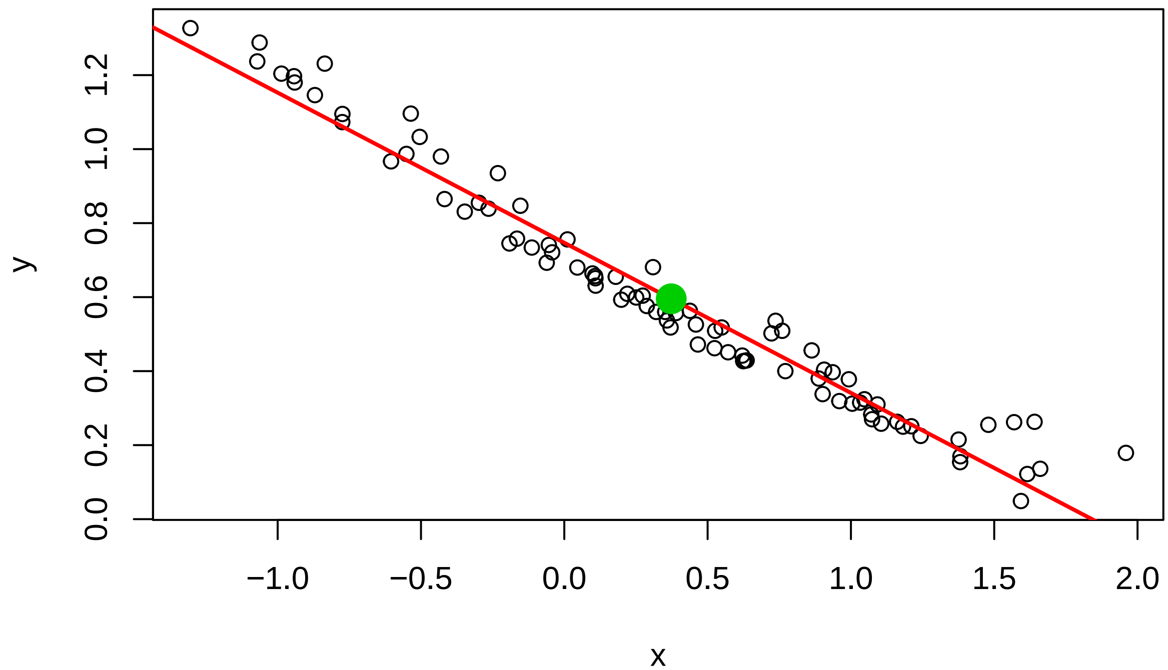
```
## [1] 92
```

Here is a quick example of linear regression relating $B - V$ to $\log(L)$.

```r
mainseqhyades <- filter & (Vmag>4 | B.V<0.2)
logL <- (15-Vmag-5 * log10(Plx)) / 2.5
x <- logL[mainseqhyades]
y <- B.V[mainseqhyades]
plot(x,y)
regline <- lm(y~x)
abline(regline,lwd=2,col=2)
summary(regline)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.08578 -0.04846 -0.01741  0.04004  0.22711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.746857   0.007712   96.85   <2e-16 ***
## x           -0.405698   0.009148  -44.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06487 on 86 degrees of freedom
## Multiple R-squared:  0.9581, Adjusted R-squared:  0.9576
## F-statistic:  1967 on 1 and 86 DF,  p-value: < 2.2e-16
```

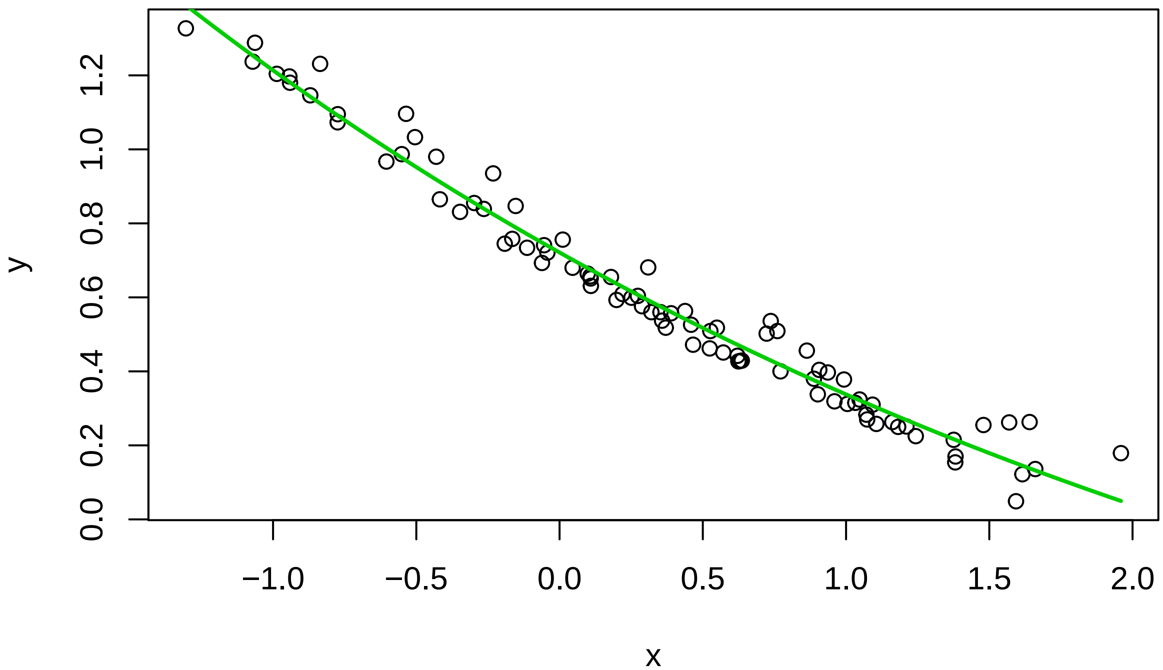
```r
points(mean(x),mean(y),col=3,pch=20,cex=3)
```




Note that the regression line passes exactly through the point (xbar, ybar).

Here is a regression of y on exp(-x/4):

```r
plot(x,y)
newx <- exp(-x/4)
regline2 <- lm(y~newx)
xseq <- seq(min(x),max(x),len=250)
lines(xseq,regline2$coef%*%rbind(1,exp(-xseq/4)),
lwd=2,col=3)
```

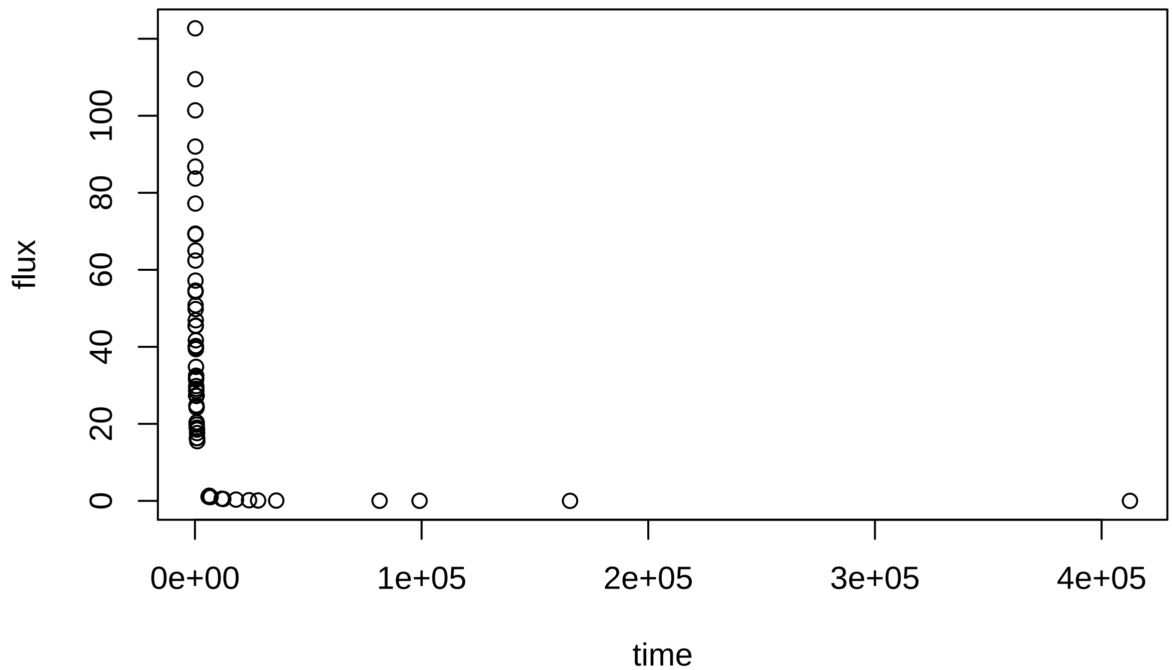



*** Task Let's now switch to a new dataset, one that comes from NASA's Swift satellite. This dataset is described at http://www.astrostatistics.psu. edu/datasets/GRB_afterglow.html. The statistical problem at hand is modeling the X-ray afterglow of gamma-ray bursts. First, read in the dataset:

```
grb <- read.table(paste(loc,"GRB_afterglow.dat",sep=""),
header=T,skip=1)
```

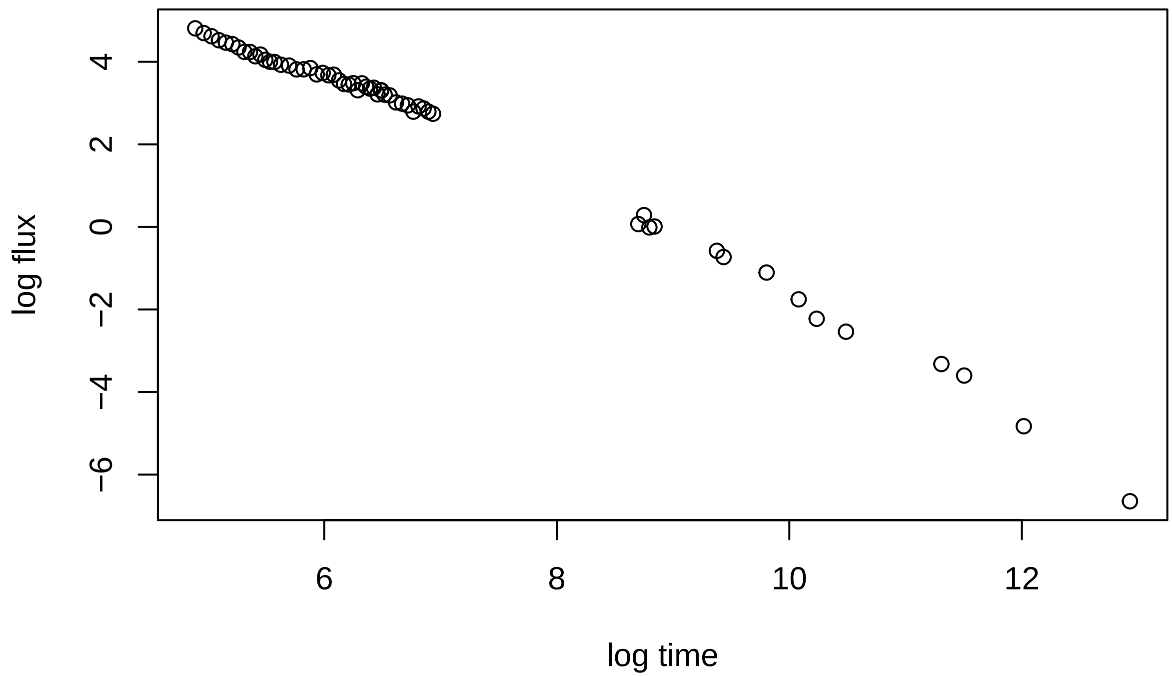
We use the skip=1 option since the raw file has some ancillary information entered on the first line. We will focus on the first two columns, which are times and X-ray fuxes:

```
plot(grb[,1:2],xlab="time",ylab="flux")
```




This plot is very hard to interpret because of the scales, so let's take the natural log of each variable:

```
x <- log(grb[,1])
y <- log(grb[,2])
plot(x,y,xlab="log time",ylab="log flux")
```




The relationship looks roughly linear, which is also substantiated by a test of the correlation coeficient:

```
cor.test(x,y)
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = -71.788, df = 61, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9964593 -0.9902880
## sample estimates:
##        cor
## -0.9941337
```

So let's try a linear model. **Exercise 1**: compute the linear regression parameters.

---

**Classification using nearest neighbors**

NNC are defined by characteristic of classifying unlabeled examples by assigning them to the class of the most similar labeled examples. They have been used succesfully for: + computer vission applications, including optical character recognition and facial recognition in both images and video, take a look at www.opencv.org.

+ predicting whether a person enjoys a movie which he/she has been recommended (as in the Netflix challenge)
+ Identifying patterns in genetic data, for use in detecting specific proteins or diseases.

In general NNC classifiers are well suited where relationships among features and the target classes are complicated , numerous or otherwise extremely difficult to understand.

---

**The kNN algorithm strenths:**

- Simple and effective
- Make no ssumption about the underlying data
- Fast training phase ### kNN algorithm weakness:
- Does not produce a model
- Slow classification phase
- Requires a large amount of memory
- missing data requires additional processing

---

**kNN**

- This algorithm begins with a training dataset made up of examples that are classified into several categories, as labeled by a nominal variable.
- Assume that we have a test dataset containing unlabeled examples that otherwisr have the same features as the training data.
- For each record in the test dataset, kNN identifies k records in the training data that are the "nearest" in similarity, where k is an integer specified in advance.
- Locating a point nerest neighbors requires a **distance function** like the **Euclidian distance** or the **Manhattan distance**, read about this using ?dist.
- The unlabeled test instance is assigned the class of the majority of the k-neigbors.

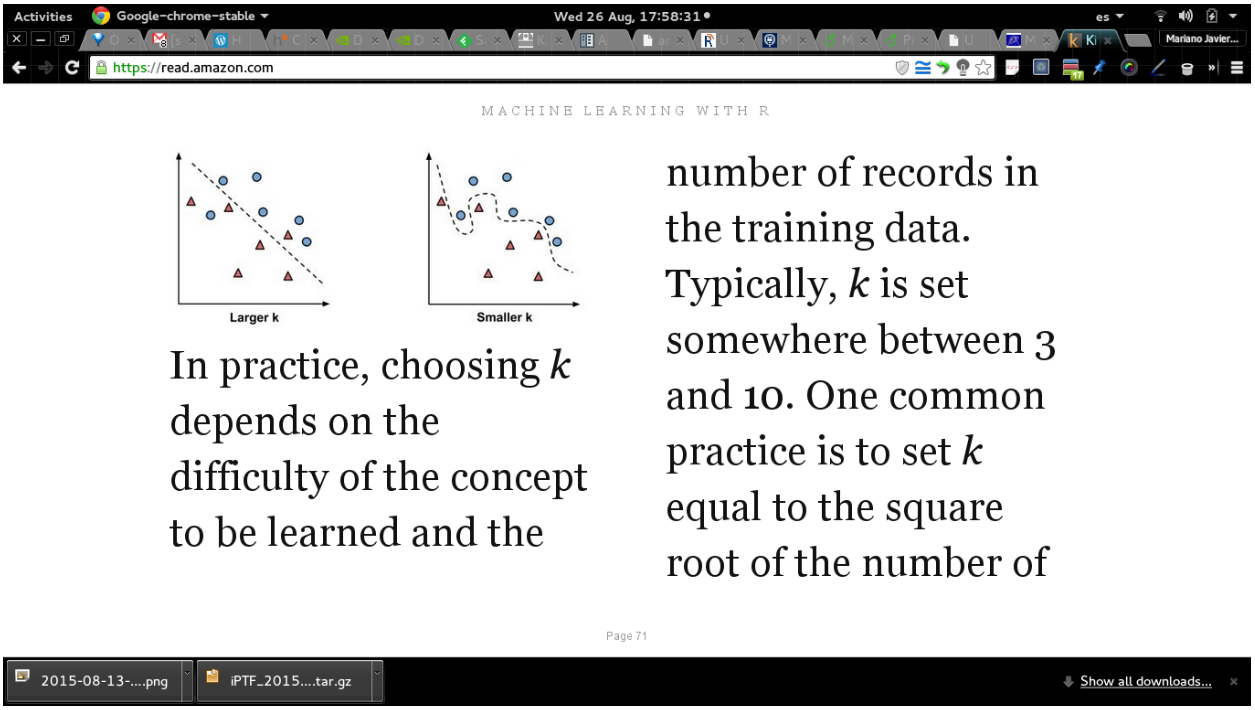
---

**Choosing an appropiate k.**

Decide how many neighbors to use for kNN determines how well the model will generalize to future data The balance between overfitting and underfitting the training data is known as the **bias-variance tradeoff**.

Choosing a large $k$ reduces the impact of variance caused by noisy data, but can bias the learner such that it runs the risk of ignoring small important patterns.

The following figure illustrates more generally how the **decision boundary** (depicted by a dashed line) is affected by larger and smaller $k$ values.


MACHINE LEARNING WITH R



In practice, choosing $k$ depends on the difficulty of the concept to be learned and the number of records in the training data. Typically, $k$ is set somewhere between 3 and 10. One common practice is to set $k$ equal to the square root of the number of

Page 71

---

**Preparing the data for use.**

- Tip: A less common, but interesting solution to this problem is to choose a larger $k$, but apply a **weighted voting** process in which the vote of closed neighbours is considered more authorative than the vote of far away neighbors.

Features are typically transformed to a standart range prior to apply the kNN algorithm. THe rationale for this step is that the distance formula is dependent in how the features are measured.

In particular, if certain features have much larger values than others, the distances measurements will be strongly dominated by the larger values.

---

**Rescaling the features**

- What we need is a way of shrinking the varius features such that each one contributes relativelly equally to the distance formula.

The traditional method for kNN is **minmax normalization**. This process transform a feature such that all of its values fall in a range between 0 and 1.

Another common tranformation is called **z-score standardization**. Substrac the mean value of each feature and divide by its standard deviation. This scores fall in an unbounded range of negative and positive numbers.

The Euclidean distance formula is not defined for nominal data, therefore we need to convert nominal feature into a numeric format. For instance **dummy coding**.

Classification algorithms based on the kNN are considered lazy learning algorithms because no abstraction occurs.

---

**Diagnosing Breast Cancer**

- We will investigate the utility of ML for detecting cancer by applying kNN algorithm to measurements of biopsed cells from women with abnormal breast masses.

We will utilize the "Breast Cancer Winscosin Diagnostic" dataset from the UCI ML Repository http://archive.ics.uci.edu/ml which includes 569 examples of cancer biopsies each with 32 features (differents characteristics of the cell nuclei) and the diagnosis coded as M(alignant) or B(enign)

```
data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc
data <- data[-1]
str(data)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ V2 : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ V3 : num  18 20.6 19.7 11.4 20.3 ...
##  $ V4 : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ V5 : num  122.8 132.9 130 77.6 135.1 ...
##  $ V6 : num  1001 1326 1203 386 1297 ...
##  $ V7 : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ V8 : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ V9 : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ V10: num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ V11: num  0.242 0.181 0.207 0.26 0.181 ...
##  $ V12: num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ V13: num  1.095 0.543 0.746 0.496 0.757 ...
##  $ V14: num  0.905 0.734 0.787 1.156 0.781 ...
##  $ V15: num  8.59 3.4 4.58 3.44 5.44 ...
##  $ V16: num  153.4 74.1 94 27.2 94.4 ...
##  $ V17: num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ V18: num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ V19: num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ V20: num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ V21: num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ V22: num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ V23: num  25.4 25 23.6 14.9 22.5 ...
##  $ V24: num  17.3 23.4 25.5 26.5 16.7 ...
##  $ V25: num  184.6 158.8 152.5 98.9 152.2 ...
##  $ V26: num  2019 1956 1709 568 1575 ...
##  $ V27: num  0.162 0.124 0.144 0.21 0.137 ...
##  $ V28: num  0.666 0.187 0.424 0.866 0.205 ...
##  $ V29: num  0.712 0.242 0.45 0.687 0.4 ...
##  $ V30: num  0.265 0.186 0.243 0.258 0.163 ...
##  $ V31: num  0.46 0.275 0.361 0.664 0.236 ...
##  $ V32: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

Regardless the machine learning method, ID variables should always be excluded. Neglecting to do could lead to erroneous findings because the ID can be used to "predict" each and likely suffer from overfitting.

The next variable, diagnosis is of particular interest, as is the outcome we hope to predict

```r
table(data$V2)
```

```
##
##   B   M
## 357 212
```

Also take a look to the rest of variables, ranges etc.

```r
summary(data)
```

```
##  V2            V3               V4              V5
##  B:357   Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
##  M:212   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
##          Median :13.370   Median :18.84   Median : 86.24
##          Mean   :14.127   Mean   :19.29   Mean   : 91.97
##          3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
##          Max.   :28.110   Max.   :39.28   Max.   :188.50
##        V6               V7                V8                V9
##  Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
##  1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
##  Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
##  Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
##  3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
##  Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
##       V10               V11              V12               V13
##  Min.   :0.00000   Min.   :0.1060   Min.   :0.04996   Min.   :0.1115
##  1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324
##  Median :0.03350   Median :0.1792   Median :0.06154   Median :0.3242
##  Mean   :0.04892   Mean   :0.1812   Mean   :0.06280   Mean   :0.4052
##  3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789
##  Max.   :0.20120   Max.   :0.3040   Max.   :0.09744   Max.   :2.8730
##       V14              V15               V16               V17
##  Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713
##  1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169
##  Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380
##  Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
##  3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
##  Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
##       V18                V19               V20
##  Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##       V21                V22                 V23             V24
##  Min.   :0.007882   Min.   :0.0008948   Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870   Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949   Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400   Max.   :36.04   Max.   :49.54
```

```
##       V25             V26              V27               V28
##  Min.   : 50.41   Min.   : 185.2   Min.    :0.07117   Min.    :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean    :0.13237   Mean    :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.    :0.22260   Max.    :1.05800
##       V29              V30               V31              V32
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993   Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

---

### Data Tranformation

We need to create a normalize() function in R

```
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}
```

After executing the previus code, the function is available for use. Test the function in some vectors.

```
normalize(c(1,2,3,4,5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
normalize(c(10,20,30,40,50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

We can not apply the function ot the numeric features in the dataframe. The lapply() function of R takes a list and applies a function to each element of the list.

```
data_n <- as.data.frame(lapply(data[2:31], normalize))
summary(data_n$V3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2233  0.3024  0.3382  0.4164  1.0000
```

```
summary(data_n$V8)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1397  0.2247  0.2606  0.3405  1.0000
```

Bingo! In absence of new laboratory data, we will simulate this scenario by dividing our data into a **training dataset** that will we used to build the kNN model and a **test dataset** that we will use to estimate the predictive accuracy of the model.

```
data_train <- data_n[1:469, ]
data_test  <- data_n[470:569, ]
```

Notice that such datasets should be representative of the full set of data, i.e. **random sampling methods!**

---

#### Training a classifier

We exclude the target variable, but we will need to store the class these class labels in factor vectors

```
data_train_labels <- data[1:469, 1]
data_test_labels  <- data[470:569, 1]
```

For the **kNN algorithm** the training phase actually **involves no mode buiding** To classify our test instances we will use the class package with Euclidean distance, install it!

The test instance is classified by taking a vote among the k-nearest neighbors. A tie is broken at random. Now we can use the knn() function to classify the test data.

```
data_test_pred <- knn(train=data_train, test=data_test, cl=data_train_labels, k=21)
```

---

#### Evaluating model performance

- The next step of the process is to evaluate how well the predicted classes in data_test_pred match up the known values in data_test_labels vector.

```
library(gmodels)
CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                 | data_test_pred
```

```
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        77 |         0 |        77 |
##                  |     1.000 |     0.000 |     0.770 |
##                  |     0.975 |     0.000 |           |
##                  |     0.770 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         2 |        21 |        23 |
##                  |     0.087 |     0.913 |     0.230 |
##                  |     0.025 |     1.000 |           |
##                  |     0.020 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        79 |        21 |       100 |
##                  |     0.790 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

In the top left cell are the **true negative results**, the bottom down cell indicates the **true positive results** were the classifier and the clinically determined label agree that the mass is malignant. 98% accuracy for a few lines of R!

- Problems:
- Improve the performance (show the summary result) using z-score standardization provided by the R scale() function.
- Test for alternative values of k=1, 5, 11, 15, 21, 27 . Report the number of false negatives and positives and select the best value of k. Check if the result change using random patients to test, discuss.

---