# K-Nearest Neighbors: Horseshoe crab

Milica Cudina

This is a well-known data set built into a library.

```
#install.packages("glmbb")
library(glmbb)
data(crabs)
```

Our task is to predict whether a female horseshoe crab will have a satellite (`y` equal to 1) based on all the other predictors: `color`, `spine`, `width`, `weight`.

Right now, `y` is an integer. So, let's make it a factor.

```
crabs$y<-as.factor(crabs$y)
```

Next, I will `attach` the data set for ease of access.

```
attach(crabs)
dim(crabs)
```

```
## [1] 173   6
```

```
n=length(crabs$y)
```

$K-$nearest neighbors is implemented via the `knn` command in the library `class`.

```
library(class)
```

We want to split the data into training and testing. This time, I will train on 2/3 of the data.

```
#setting the seed for replicability
set.seed(1)
#the training-set indices
train.ind=sample(n, floor(2*n/3))
```

The standardization of distance is hard here because some of the predictors are qualitative. We can use the command `scale` for `width` and `weight`.

```
standardized.width.train=scale(width[train.ind])
standardized.weight.train=scale(weight[train.ind])
standardized.width.test=scale(width[-train.ind])
standardized.weight.test=scale(weight[-train.ind])
```

We need to transform our categorical predictor using dummy variables. For that, the following library is useful:

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

First, I will put all of my categorical predictors into a separate data frame.

```r
temp.x=data.frame(spine, color)
```

Now, I will use the command `dummyVars` to create dummy variables for my categorical predictors in my array of predictors.

```r
dummy.x <- dummyVars(" ~ .", data = temp.x)
cat.x <- as.matrix(predict(dummy.x, newdata = temp.x))
#print(cat.x)
```

Now, I will collate the standardized numerical and the categorical variables.

```r
#predictors
train.X=data.frame(standardized.width.train, standardized.weight.train, cat.x[train.ind,])
test.X=data.frame(standardized.width.test, standardized.weight.test, cat.x[-train.ind,])

#responses
train.Y=crabs$y[train.ind]
test.Y=crabs$y[-train.ind]
```

It's time for the `knn` command to do its magic.

```r
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
#knn.pred
mean(test.Y == knn.pred)
```

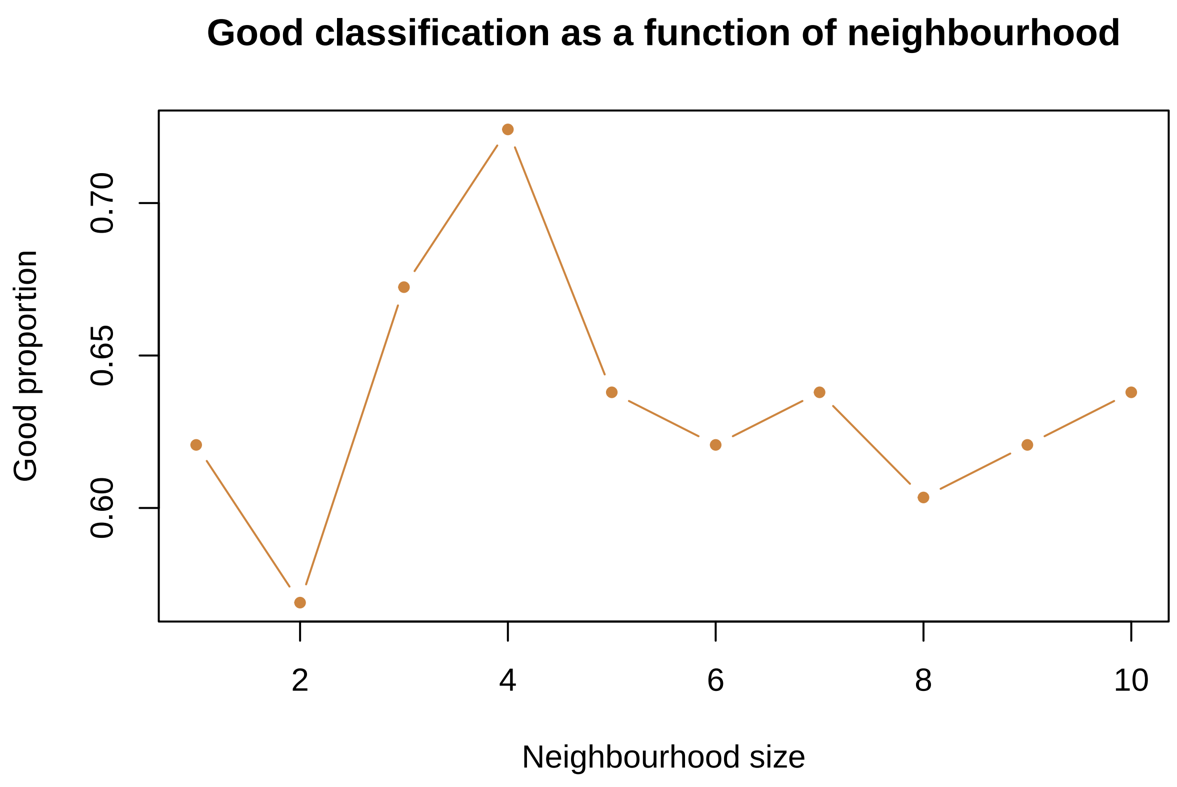
```
## [1] 0.6034483
```

The number of nearest neighbours we look at is a hyperparameter. Let's see how we perform as the $k$ varies.

```r
good.pred<-numeric(10)

for (k in 1:10){
  knn.pred <- knn(train.X, test.X, train.Y, k = k)
  good.pred[k]=mean(test.Y == knn.pred)
}

plot(good.pred,
     col="peru", type="b", pch=20,
     main="Good classification as a function of neighbourhood",
     xlab="Neighbourhood size", ylab="Good proportion")
```


Good classification as a function of neighbourhood
Good proportion
0.60
0.65
0.70
2
4
6
8
10
Neighbourhood size