Workshop 5
K-Nearest Neighbour Regression

In this workshop you will learn how to fit a non-parametric regression model using the K-Nearest Neighbour method and compare the method with linear regression models.

Start R in the usual way. Install the package `FNN` and load it into R. You will also be plotting 3-dim graphics using the R package `rgl`. You should already have this installed but check it now using

```
require(rgl)
```

One explanatory variable

We will start with a simple (easy) simple (one variable) regression example. The `x` variable is a vector of integers between 1 and 20 and `y` is a vector of normal random variables with mean equal to the corresponding `x` value plus 10.

```
x<-1:20
y<-rnorm(length(x),mean=x+10)
xgrid<-data.frame(x)
```

The third command defines the values of the explanatory variables used to evaluate the predictor function. To start with we obtain the fitted values, i.e. calculate the predictor function at the the observed `x` values. The function in Library `FNN` to fit a K-nearest neighbour regression model is called `knn.reg` and to obtain the correct fitted values we need to pass the x values again to the argument called `test=`. We will investigate what happens when the `test` argument is omitted in the *mean squared error* Section.

Fit the KNN regression with $K = 1$ and compare the fitted values with the outcome variable

```
knnr.out<-knn.reg(x,test=xgrid,y=y,k=1)
round(cbind(x,y,knnr.out$pred),2)
```

Repeat this with the value $k = 3$

To plot the predicted values as a function, we will specify a much finer grid for the predicted values. Again start with $k = 1$.

```
xgrid<-data.frame(x=seq(0,21,0.05))
```

```
knnr.out<-knn.reg(x,test=???,y=???,k=???)
plot(x,y)
lines(xgrid$x,knnr.out$pred)
```

Gradually increase the value of K and observe what happens to the predictor function. Which values of K correspond to under fitting and which values to over fitting?

For completeness and a bit of revision we will have a look at the linear regression for these data.

```
lm.obj<-lm(y~x)
summary(lm.obj)
plot(x,y)
abline(lm.obj,col=2)
knnr.out<-knn.reg(x,test=xgrid,y=y,k=19)
lines(xgrid[,1],knnr.out$pred,col=3)
```

Note that the linear regression model fits the data very well and requires only two parameters. Although for non-parametric models we do not use the concept of the number of "parameters" a comparison would be using just two constant functions i.e. $k = 19$ which massively under fits these data. This is somewhat unfair though, as the example data is ideal for a linear regression model.

Two explanatory variables

The following data were taken from a small study investigating the fitness rating for 10 participants. The measured explanatory variables are *weight* and *lung volume*. A small data set can be directly entered into R using the following code:

```
fit<-as.data.frame(matrix(c(1,87, 42,6, 73, 43,7, 66, 44,15,62,54,12,
     68,45,4,92,46,12,60,50,13,70,46,14,71,54,10,64,47),byrow=T,ncol=3))
names(fit)<-c("fitness","weight","lungvol")
summary(fit)
```

The linear model for fitness dependent on weight and lung volume is fitted using:

```
lm.fitness<-lm(fitness~weight+lungvol,data=???)
summary(???)
```

Both explanatory variables are significant at the 5% level. Use this output to write the predictor function as a function of the two predictor variables.

We will plot the two variables using (i) a *perspective plot* and (ii) a *3-d plot*. In both cases we need to compute the predicted values at points on a grid with *weight* taking values between 55 and 95 Kg and *lung volume* between 40 and 55 dl.

```
m1<-seq(55,95,length=20)
m2<-seq(???,???,length=20)
Xgrid<-expand.grid(weight=m1,lungvol=m2)
pred.grid<-predict(lm.fit,newdata=Xgrid)
tt<-cbind(Xgrid,pred.grid)
res<-persp(m1,m2,matrix(pred.grid,nrow=length(m1)),border=grey(0.6),
     xlab="Weight",ylab="Lungvol",zlab="fitness",theta=0,phi=15)
points(trans3d(fit$weight,fit$lungvol,fit$fitness,pmat=res),
     pch=16,col = c("DarkRed","orange")[1.5+.5*sign(lm.fit$residuals)])
```

The dark red points have a negative residual so lie under the fitted plane and the orange points have a positive residual so lie above the fitted plane. The function `persp` takes two arguments determining the position from which the data are viewed. `theta` controls the rotation of the x and y axes, and `phi` controls the height of the viewing point. Play about with these two parameters to get a better understanding of the data and the fitted plane.

Create a 3d plot for these data:

```
with(tt, plot3d(lungvol,weight,pred.grid,col="grey80",size=3,zlab="fitness"))
with(fit, plot3d(lungvol,weight,fitness,add=T,size=2,col="red",type="s"))
```

You can rotate the plot by dragging it with the mouse.

We will now fit a K nearest neighbour regression model. We define a data frame `X` with the two variables `weight` and `lungvol` and start with $K = 1$.

```
X<-fit[,c("weight","lungvol")]
knnr.out<-knn.reg(X,test=Xgrid,y=???,k=1)
res<-persp(m1,m2,matrix(knnr.out$pred,nrow=length(m1)),border=grey(0.6),
  xlab="Weight",ylab="Lungvol",zlab="fitness",theta=15,phi=15)
points(trans3d(fit$weight,fit$lungvol,fit$fitness,pmat=res),
       pch=16,col = c("DarkRed")[1.5+.5*sign(lm.fit$residuals)])
with(tt, plot3d(lungvol,weight,knnr.out$pred,col="grey80",size=3,zlab="fitness"))
with(fit, plot3d(lungvol,weight,fitness,add=T,size=2,col="red",type="s"))
```

Now increase the value of K each time obtaining the new plot.

What is the best value for K?

Mean Squared Error

Which is the best value of K? Re-run the K nearest neighbour regression model with $K = 1$ and obtaining predictions at the data points. Calculate the mean squared error for this particular model.

```
knnr.out<-knn.reg(???,test=X,y=???,k=1)
mean((fit$fitness-knnr.out$pred)^2)
```

Ouch! $K = 1$ is obviously over-fitting the data but the mean squared error suggests this is the best model (the MSE can not be negative). Choosing a model based on fitted MSE alone is not a good idea.

One way to avoid this problem is called *cross validation*, which can be used in many supervised learning methods. We will return to the one dimensional data from the first section, as it is easier to investigate what is going on. We will compare the output of `knn.reg` with and without the argument `test` using only one nearest neighbour $k = 1$.

```
xgrid<-data.frame(x)
knnr.out1<-knn.reg(x,y=y,k=1)
knnr.out2<-knn.reg(x,test=xgrid,y=y,k=1)
round(cbind(x,y,knnr.out1$pred,knnr.out2$pred),2)
```

Look closely at the two sets of predicted values, what do you notice?

When the `test` argument is not specified, then algorithm excludes the current x value when determining the K neighbours nearest to x. This is called *cross validation*. For each observation in turn, the model is fitted without that observation point, and then the predicted value obtained at that point. We can now compute the MSE for the cross validated model for different values of K. Once the best value of K is chosen, the non cross validated model needs to be run to obtain the predictions.

Another possible approach is to set aside a proportion of the data calling this the *test data* set and the remaining part the *training data* set. Here we only have 10 observations so splitting the observations into training and test data sets is not a sensible option.

```
msevec<-rep(NA,9)
for(k in 1:9){
  knnr.out<-knn.reg(X,y=fit$fitness,k=k)
  msevec[k]<-mean(???)
}
plot(1:9,msevec)
```

What is the best number of neighbours using this method? Plot the optimal fitted KNN regression in a 3D-plot.

Finally we can calculate the MSE for the multiple regression model.

```
mean((fit$fitness-lm.fitness$fitted)^2)
```

The fit is noticeably better (this is not always the case, these data fit the regression model well).

Optional extra

With a linear model we have a method of assessing which variables have significant effect on the outcome variable by considering the p-values. The p-value itself relies on the model assumptions and specification being correct, but can nevertheless be used as a decision rule when choosing between models.

Using KNN regression we can compare the MSE for different models.

For example repeat the *cross validation* fit for the model with just weight fitted. Calculate the MSE for each value of K creating a vector called `mse1vec`. Compare `msevec` with `mse1vec` graphically.

The difference between the best MSE value for the *weight and lung volume* model and the best MSE value for the *weight* only model gives an indication of how much better the combined model is. The combined model will nearly always give a lower MSE, but we need to consider if the cost of a more complicated model is worth it. This is a subjective problem but there are methods to overcome this, and we will look into this type of problem in the coming weeks.

Homework Question

If you have understood the KNN regression method this should be a quick and easy exercise. If you are unsure, then the practice will be good.

In the table below are values of three variables x_1, x_2 and y. Using x_1 and x_2 as the explanatory variables and y as the outcome variable, find the KNN regression estimate for the point $x_{.1} = 9$ and $x_{.2} = 7$ for values of $k = 3$ and $k = 5$

x_1	x_2	y
3	7	1
4	3	4
7	9	9
8	6	10
9	4	8
11	12	3
13	5	0
13	8	1