Lab #1 – Regression Analysis

Useful commands

Command	Explanation
attach()	**Attaches database to R search path. No need for database$name syntax.**
summary()	**Returns summary statistics**
names()	**Returns list of names of variables in database**
lm(Y~X+Z)	**Syntax for linear regression where Y=dependent, X,Z=independent variables.**
plot(x,y)	**Scatter plot of variables x and y**
abline(intercept,slope)	**Add straight line to plot given intercept and slope. *Run right after running plot()**
curve(f(x))	**Plot function**

Example code for simple linear regression:

```
dat<-cars
attach(dat)
#run regression
fit<-lm(speed~dist)
#view regression output and graphs
summary(fit)
plot(speed~dist)
abline(fit, col="red", lwd=2)
plot(fit)
```

Exercise

1. Install package "MASS" which contains built-in datasets. install.packages("MASS")
2. Import the library: library(MASS)
3. Define object "Dat" as the database named "Boston"
4. Get to know the data set:
    a. Type ?Boston to read about the database in the workspace window "help" tab.
    b. How many rows/columns?
    c. Show summary statistics of variables
5. We want to predict median house prices (medv) by percent of lower status of population (lstat):
    a. Run a scatter plot of the independent variable on the Y-axis, and dependent variable on the x-axis. What can we learn about the relationship between the variables?

b. Run a simple linear regression. *Tip* - define it as an object, this will be useful later.
c. Print a summary of the regression results. (run "summary()" on regression object). Can you interpret the regression coefficient? What about the rest of the output?
d. Print diagnostic plots of regression results. (run "plot()" on regression object)
e. Extract predicted values. (Hint: use $ syntax similar to extracting a variable from a database).
f. Plot scatter plot of medv and lstat, then plot regression line. (Hint: to make line more visible change its color using argument col="color of choice"). Does this regression seem to be a good fit?

*Later we will run some different kinds of regressions and find a better fit.

Lab #2 – KNN Analysis

Useful commands

Command	Explanation
str()	**If given a data frame as argument, returns classes of all variables**
sample(x,size)	**Creates a random sample from vector x**
scale()	**Scales vector values. Default subtracts the mean, divides by sd (z-score)**
knn(train,test,cl,k)	**Runs k-nearest neighbors algorithm where train = training set (predictors only), test = test set (predictors only), cl = true classifications of training set, k = number of nearest neighbors.**
table()	**Table of frequencies for each variable**
prop.table(x,margin)	**Table of proportions for each variable. Margin = 1 for generating proportions across rows, 2 across columns.**

Example code for KNN:

```
dat <-iris[c("Sepal.Length","Sepal.Width","Species")]
#scaling
dat$Sepal.Length <- as.vector(scale(dat$Sepal.length))
dat$Sepal.Width <- as.vector(scale(dat$Sepal.Width))
#create train and test sets
index <- sample(x=1:nrow(dat), size=.3*nrow(dat))
test <- dat[index,]
train <- dat[-index,]
test_pred <- test[c("Sepal.Length","Sepal.Width")] #predictors only
train_pred <- train[c("Sepal.Length","Sepal.Width")] #predictors only
test_species <- test$Species #true test classification
train_species <- train$Species #true train classification
#run KNN
knn.1 <- knn(train = train_pred, test = test_pred, cl = train_species, k = 1)
#check accuracy of predictions (confusion table)
table(knn.1, test_species)
prop.table(table(knn.1, test_species),2) #percentages
```

Exercise:

1. Install package "ISLR" which contains "Default" data set.
   install.packages("ISLR")
2. Install package "class" which contains knn function.
   install.packages("class")
3. Import both libraries: library(PackageName)
4. Get to know the data like in question 4 of regression Exercise.
5. Subset the data to include only the "balance", "income" and "default" variables.
6. Normalize the numeric predictor variables using scale().
7. Split data into train and test set (remember, we need a train and test set with predictors only, and another train and test set with only the variable we want to predict).
8. Run knn with 1, 5, 20, and 70 k's. (*define each one as a different object)
9. Create proportion tables for each case.