# Lab #1 – Regression Analysis

Useful commands

| Command | Explanation |
|---|---|
| ?function_name | **Opens of the right the documentation of the function with explanation of input arguments and output values** |
| read.csv() | **Read csv file to R environment** |
| attach() | **Attaches database to R search path. No need for database$name syntax.** |
| dim() | **Returns number of rows and columns in data set** |
| summary() | **Returns summary statistics** |
| cor() | **Returns correlation matrix of variables** |
| 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()** |

Example code for simple linear regression:

```
dat<-read.csv("/.../cars.csv")
attach(cars)
#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. Read California Housing data set stored in a CSV file in .......
   *read.csv("path...../CAhousing.csv")*
2. Get to know your data:
   a. What is the size of the data set?
   b. Print summary statistics. Are there any anomalies?
   c. Learn about the variables from: https://developers.google.com/machine-learning/crash-course/california-housing-data-description
   d. Print correlation matrix. Which variables are correlated?
3. We want to predict median house prices in a block by the median of income in the block:

a. Run a scatter plot of the dependent variable on the Y-axis, and independent 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? (Which coefficients are significant? How good is the fit?)
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 median income and median house value, then plot regression line. (Hint: to make line more visible change its color using argument *col="color of choice"*, and width by using argument *lwd=number of choice*). Does this regression seem to be a good fit?

4. Revisit regression
   1. Return to the plot in 3a. Can you see any anomalies in the data? What can you learn about the median house value documentation? Draw a histogram with 100 breaks ( *hist(x, breaks=100)* )
   2. Fit a full regression model (i.e. with all the variables – you can use the syntax *formula = dependent.variable ~ .* in the formula argument (include the full stop in the formula!))
   3. Is the fit too good to be true? Look back at the correlation matrix. Should we omit some variables? Which ones?

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

## <u>Lab #2 – KNN Analysis</u>

Useful commands

| Command | Explanation |
|---|---|
| library() | **Import a package that has been already installed on the computer** |
| str() | **If given a data frame as argument, returns classes of all variables** |
| mean() | **Calculates the mean (average) of vector** |
| sd() | **Calculates the standard deviation of vector** |
| 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(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
```

<u>Exercise:</u>

1. Import both libraries: *ISLR* (for *Default* data set) and *class* (for *knn* function) *library(PackageName)*
2. Load the Default data by running: *dat <- Default*
3. Get to know your data:
   a. What is the size of the data set?
   b. Print summary statistics. Are there any anomalies? Are the variables centered around 0?
   c. Learn about the variables using the help command (*?Dataset_name* - note: this only works because the *Default* data set is part of the *ISLR* package!)
   d. Check the standard deviation of the variables. Are they standardized?
4. We want to predict default based on income and average monthly balance:
   a. Subset the data to include only the "balance", "income" and "default" variables.

b. Normalize the numeric predictor variables using *scale()*.
c. 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).
d. Run K-nearest neighbors with 1, 5, 20, and 70 k's (define each one as a different object).
e. Create frequency and proportion tables for each case.

5. Create frequency and proportion tables for the default variable in the training set. What is the percentage of "Yes" and "No" in the training set? Can you think of a "safe" prediction rule that will classify correctly at least 95% of the cases? Would you use such a classifier? How would you overcome this problem?

*Later we will show some different applications of knn and some interesting graphs.

## <u>Lab #3 – Kernel Regression</u>

In the previous question you implemented a K-Nearest Neighbors (KNN) regression model. In this question you would implement a somewhat related regression model, the Kernel regression.

The two regression models are non-parametric, and predicts $y$ locally, using a linear smoother. A linear smoother is a regression function of the form:

$$\hat{\mu}(x) = \Sigma_i y_i \widehat{w}(x_i, x)$$

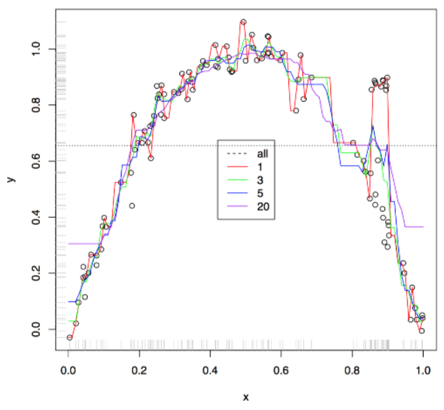
Where $y_i$ is the outcome value of $x_i$ and $\widehat{w}(x_i, x)$ is a function of $x$ and $x_i$.

Under the KNN, the prediction for $x$ would be $\hat{\mu}(x) = \Sigma_i y_i * \frac{1}{k} 1_{\{x_i \in N_k(x)\}}$ where $1_{\{.\}}$ is the indicator function that gets the value of 1 if the condition in the brackets is fulfilled and 0 otherwise. $N_k$ is the $k$ nearest neighbors to observation $x$ and $y_i$ is their corresponding outcome value. That is,

$$\widehat{w}_{knn}(x, \hat{x}) = \begin{Bmatrix} \frac{1}{k} & if\ x_i \in N_k(x) \\ 0 & otherwise \end{Bmatrix}$$

The KNN prediction is discontinuous - fit is not smooth as one neighbor will suddenly be removed.

**KNN regression with different number of K neighbors**



Source: Advanced Data Analysis from an Elementary Point of View, Shalizi

In Kernel regression the linear smoother is of the form

$$\hat{\mu}(x) = \Sigma_i y_i \frac{K(x_i, x)}{\Sigma_j K(x_j, x)}$$

Where $K(x_i, x)$ is the kernel function, a function that defines the relations between $x$ and $x_i$ and satisfies several properties.

In Kernel regression, instead of k neighbors we consider all observations and predict $y$ as in the linear smoother function from above. If many points at a similar distance, they get equal weight and if one point much closer, it gets full weight.

In this question we would consider two types of kernels, the Gaussian kernel and a Box kernel. The Gaussian kernel is

$$K_{gauss}(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

and the Box kernel is

$$K_{box}(u) = \left\{\begin{matrix} \frac{1}{2} & if\ |u| \leq 1 \\ 0 & otherwise \end{matrix}\right\}.$$

Given a choice of kernel, and a bandwidth $h$, kernel regression is defined by taking

$u := \frac{x_i - x}{h}$.

Pay attention that now, compared with the KNN, we do not need to choose (and tune) the number of K neighbors. Nevertheless, here we have to tune $h$, the bandwidth parameter.

- **Split CAhousing.csv into random train (80%) and test (20%) sets. You can implement the following code from stackoverflow, for example:**

  ```
  a # original data frame
  library(dplyr)
  train<-sample_frac(a, 0.7)
  sid<-as.numeric(rownames(train)) # because rownames() returns
  character
  test<-a[-sid,]
  ```

  Note: Usually, there is more than one way to solve a problem in programming. You can choose to split the sample in any other way, as long as it is correct. I recommend you run one line at a time and see what R is doing in each step.

- In the following subsections, use only the train dataset unless stated otherwise.

- **Use the "ksmooth" function from the *stats* library in order to predict the median house value from CAhousing.csv, using only the best predictor variable.**
- **Explore different values of $h$ and different kernels.**
- **Present three of the predictions in a plot (or combine them into one plot as the figure above suggest).**
- **How the prediction quality on the test set change under different selections of $h$? For each combination of Kernel-h compute the MSE for the test set you left aside earlier. Present your answers in a table**

  | Kernel | h | Train MSE | Test MSE |
  |---|---|---|---|
  | Gaussian | 0.2 | … | … |

  Note: Usually, we split our data into three parts: train, cross validation and test. We use the training set in order to fit the model, the cross validation set in order to tune our hyper-parameters and lastly, the test set allows us to assess our best algorithm's performance on a new dataset. Here, we constructed a toy example where we actually used the "Test Set" as a cross validation set, and used it to tune our parameters.
- **Can you spot the difference between the two kernels?**
- **Which model would you select and why?**