**Demo Algorithms and Statistical Models for Routine Data Analytics Feb 12, 2018**

**Overview**: We will apply the data analytic techniques studied to solve complete problems and illustrate the data science process. We will also learn to work with three important algorithms discussed in Chapter 3: (i) linear regression, (ii) K-means clustering algorithm and (iii) K-NN (K-nearest neighbor) classification algorithm.

**Discussion:**

**Problem 1: Consider the data**

S = {(x,y)= (1,25), (10,250), (100,2500), (200,5000)}

What next? What is value for X = 40?

We observe that

There is a linear pattern.

Coefficient relating x and y is 25

Seems deterministic

and y = 25\*x is the relationship, an obvious linear pattern.

Linear model is defined by the functional form:

y = β0 + β1\*x

Lets work on 2 demos:

A simple analysis of "cars" data already available in R environment Another demo to understand the evaluation metric for an "lm" . Goal is to visualize and understand lm and its statistics. The second demo is on synthetic data to understand the importance of the evaluation metrics and also to understand "overfitting"

For each demo we are interested in:

Cleaning and getting a data frame

1. Plot the data points
2. Visually assess the data plot
3. Model using "lm"
4. "lm" by default seeks to find a trend line that minimizes the sum of squares of the vertical distances between the approximated or predicted and observed y's.
5. We want to learn both the "trend" and the "variance"
6. Evaluate the measure of goodness of our model in R-squared and p: R-square measures the the proportion of variance. p-value assesses the significance of the result.
7. We discuss both these measures: we want R-squared to high (0.0-1.0 range) and p to be low or <0.05.
8. R-squared is 1-(total predicted error-squared/total mean error squared)

**Problem 2**: **Linear Regression**

**Example 1:**

x<-c(1,2,4,6,7,9,10,12,30,50)

y<-c(3,5,7,9,10,11,15,35,40,39)

model<-lm(y~x)

model #display the parameters of the model

plot(model)

pred.y = 0.806\*x + 6.841

x= 20

pred.y

**Example 2:**

x <- rnorm(15)

y <- x + rnorm(15)

predict(lm(y ~ x))

new <- data.frame(x = seq(-3, 3, 0.5))

predict(lm(y ~ x), new, se.fit = TRUE)

pred.w.plim <- predict(lm(y ~ x), new, interval="prediction")

pred.w.clim <- predict(lm(y ~ x), new, interval="confidence")

matplot(new$x,cbind(pred.w.clim, pred.w.plim[,-1]),

 lty=c(1,2,2,3,3), type="l", ylab="predicted y")

**Example 3: Install “nutshell” package for this exercise. Multiple regression.**

library(nutshell)

data(team.batting.00to08)

runs.mdl <- lm(formula=runs~singles+doubles+triples+homeruns+walks+hitbypitch+sacrificeflies+stolenbases+caughtstealing, data=team.batting.00to08)

runs.mdl

newdata<-data.frame(singles=1000, doubles=300,triples=35,homeruns=200,walks=600,hitbypitch=50,sacrificeflies=50,stolenbases=100,caughtstealing=40)

predict(runs.mdl,newdata)

**Example 4: Bad EDA: Working with real and current data. We will examine twitter mention of ebola and correlate with the actual cases ebola as per CDC.**

We will attempt **Linear Regression** on two datasets: the current rate of Ebola infections as per the WHO, and the amount of times Ebola has been mentioned on Twitter. We will use R to help us determine m and b. We present Ebola data from late August 2014 to end of October 2014[[1]](#footnote-1), and Twitter data during this time frame (approximately 60GB).

**Step 1, 2 and 3 of the Data Science Process**: The Twitter data was analyzed ahead of time via Apache Hadoop MapReduce, and the JSON of each Tweet was decoded. The date of the tweet and the content of the tweet were analyzed; and counts for the world “ebola” were tallied on a per day basis.

This data was combined with WHO Ebola data into a single CSV file.The since the Ebola dataset had a limited number of datapoints, the Twitter dataset was reduced in the above file to fit appropriately. Now, let’s take a look at the columns in the CSV file:
Date: The Date (in 2014) when the datapoint was recorded
DaysSince825: The number of days since 8/25. We need this variable because variables need to be
DMentions: (Delta Mentions) The number of tweets mentioning Ebola on that date
Dcases: (Delta-Cases) – the number of new Ebola cases recorded on that date
Ddeaths: (Delta-Deaths) – the number of Ebola deaths recorded on that date
numeric for R to generate an equation
Cases: The total cases of Ebola to date
Deaths: The Ebola-caused deaths to date
Mentions: The number of tweets mentioning Ebola to date

**Step 4 Exploratory data analysis:** Now, let’s load the data into R:

ebola <- read.csv(..)
summary(ebola)
days <- ebola$DaysSince825
mentions <- ebola$Mentions
cases <- ebola$Cases
deaths <- ebola$Deaths
dmentions <- ebola$Dmentions
ddeaths <- ebola$Ddeaths
dcases <- ebola$Dcases

There are plenty of columns here to work with and attempt to draw conclusions from. First, let’s look at a few basic plots to better understand the data and perform some EDA (exploratory data analysis).
First, let’s just look at the data over time:
plot(days,cases)

plot(days,dmentions)

plot(days,deaths)



Now, let’ look at the mentions and days. Can you suggest why this sudden increase in activity here?


The reported deaths over time is also similar:


Now, let’s plot the relationship between Ebola cases and the number of mentions of Ebola on twitter:

**tweets <- read.csv()**

**days<-tweets$DaysSince825**

**mentions<-tweets$Mentions**

**plot(days,mentions)**

**Step 5: Apply statistical methods: linear regression in this case:**

Let us do some analytics. A best fit line is a line of the form y = mx+b , and is as close to as many of the data points as possible. Let’s have R use our data and the least square approach to predict our data:
Note that coefs[1] is the slope of the line, and coefs[2] is the y-intercept of the line. You can try this approach for different values of x and y and see if you can show a linear relationship. NOTE: **This is on the first set of data we called ebola.**x <- cases
y <- mentions
model <- lm(y ~ x)
coefs <- coef(model)
plot(x,y)
abline(coefs[1],coefs[2])

model



This line seems to fit the data pretty well. If we enter coefs into the R console, we can see the slope (denoted, somewhat confusingly, as x) and the intercept (denoted as intercept):



Let’s take a look at the inverse relationship – and see if we can predict the number of cases based on the number of tweets thus far on twitter:
x <- mentions
y <- cases
model <- lm(y ~ x)
coefs <- coef(model)
plot(x,y)
abline(coefs[1],coefs[2])

This yields a relatively nice looking line as well: **However is this valid interpretation?**



**================End of Linear-regression========================================**

1. **K-Means Clustering machine learning algorithm.**

**Example 1:**

library(cluster)

library(fpc)

data(iris)

dat <- iris[, -5] # without known classification

# Kmeans cluster analysis

clus <- kmeans(dat, centers=3)

with(iris, pairs(dat, col=c(1:3)[clus$cluster]))

plotcluster(dat, clus$cluster)



**Example 2**: A detour to learn to generate synthetic data and save the generated data.

#synthetic data

runif(10,0.0,100.0) # generate 10 numbers between 0.0 and 100.0 : uniform distribution

rpois(100,56)# generate 100 numbers with mean of 56 according to Poisson distribution

rnorm(100,mean=56,sd=9)

sample(0:100,20)

sts<-sample(state.name,10)

sts

**Example 3:**

age<-c(23, 25, 24, 23, 21, 31, 32, 30,31, 30, 37, 35, 38, 37, 39, 42, 43, 45, 43, 45)

clust<-kmeans(age,centers=4)

plotcluster(age,clust$cluster)

Try the K-means with the other synthetic data: Examine the outputs of the model, goodness of the clustering in between\_SS/total\_SS percent. Higher the percent, better the fit.

(i)

age<-rpois(1000,56)

(ii)

age<-data.frame(x<-sample(1:500,4300),y<-rpois(300,450))

clust<-kmeans(age,centers=3)

plotcluster(age,clust$cluster)

clust$centers

clust$size

clust

**========================End of K-means==========================================**

1. **K-NN, K nearest neighbor classification machine learning algorithm.**

**Example 1: We will use a built-in data set in R called iris to work on this algorithm. “iris” is a dataset of 150 elements of 4 independent variables and three classes of irises. “iris3” is a 50 data sets of four variables {1:4] and another dimension for the class [1:3]. We will follow the K-NN algorithm discussed in the text book.**

data(iris3)

iris3

iris

train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])

test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])

cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))

model <-knn(train, test, cl, k = 3, prob=TRUE) # now **LEARN how to classify; train (and validate)**

attributes(.Last.value)

plot(model)

query1<-c(5.0, 3.2, 4.9, 2.0) #set unknown data

knn(train, query1, cl, k = 3, prob=TRUE) # now classify query1

query2<-c( 5.1, 3.8 , 1.9, 0.4)

knn(train, query2, cl, k = 3, prob=TRUE)

**Example 2: For this example of K-NN we will generate some synthetic data.**

income<-sample(1000:1000000, 1000) # 1000 income values in the range 1000-100000

age<-sample(21:100,1000,replace=T)

gender<-sample(0:1,1000,replace=T)

data3<-data.frame(age,gender,income) # form a data frame that is required for application of k-nn

**View the data3 from the workspace window**

income<-sample(1000:1000000, 1000) # 1000 income values in the range 1000-100000

age<-sample(21:100,1000,replace=T)

gender<-sample(0:1,1000,replace=T)

data3<-data.frame(age,gender,income) # form a data frame that is required for application of k-nn

train<-data3[1:500,] # half the data is training set

test<-data3[500:1000,] #other half is test set

cl<-factor(sample(0:1,500,replace=T)) # randomly select the classes to be either 1 or 0

model<-knn(train, test, cl, k = 3, prob=TRUE)

plot(model)

query<-c(56,1,90000)

classifyMe <-knn(train, query, cl, k = 3, prob=TRUE)

classifyMe

query<-c(56,1,9000)

classifyMe <-knn(train, query, cl, k = 3, prob=TRUE)

classifyMe

**# How does the value of K affect the clustering? Lower value underfits, higher value overfits, so choosing an optimal value is important. Try the code with different K values: 3,5**

**# illustrating over-fitting: Try with k =5, k = 7**

model<-knn(train, test, cl, k = 7, prob=TRUE)

plot(model)

query<-c(56,1,90000)

classifyMe <-knn(train, query, cl, k = 7, prob=TRUE)

classifyMe

query<-c(56,1,9000)

classifyMe <-knn(train, query, cl, k = 7, prob=TRUE)

classifyMe

write.csv(data3, "knn.csv", row.names=FALSE)

1. **Miscellaneous Plots and arrangements**

**Example 1: Multiple types of plots in one layout**

attach(mtcars)

par(mfrow=c(2,2))

plot(wt,mpg, main="Scatterplot of wt vs. mpg")

plot(wt,disp, main="Scatterplot of wt vs disp")

hist(wt, main="Histogram of wt")

boxplot(wt, main="Boxplot of wt")

detach(mtcars)



**More data analysis:**

**When you launch on a data analytics project here are some of the questions to ask:**

* What are the basic variables?
* What are underlying processes?
* What influences what?
* What are the predictors?
* What causes what?
* **What do you want to know?: Needs a domain expert**
1. <http://en.wikipedia.org/wiki/Ebola_virus_epidemic_in_West_Africa> [↑](#footnote-ref-1)