Data Manipulation List – R

Part 1 - Data Cleansing & Manipulation

* Import data

df <- read.table("<FileName>.txt",

header = FALSE, #header = TRUE / FALSE

sep="/", #separator

strip.white = TRUE,

na.strings = "EMPTY") #indicates which strings interpreted as NA

df <- read.csv("<FileName>.txt", col.names= c("X", "Y", "Z", "A","B"),header = TRUE)

df <- read.csv2("<FileName>.txt", header = TRUE) #read.csv2 is used for files with ;

If double quotes have been used for string data, use **quote="\"",**

If header is needs, use **col.names= c("X", "Y", "Z", "A","B"),**

df <- read.delim("<name and extension of your file>",

header = FALSE,

sep = "/",

quote "\"",

row.names = c("O", "P", "Q"),

fill = TRUE, #blank field will be added to rows with unequal length

strip.white = TRUE,

stringsAsFactors = TRUE,

nrows = 5) #only read the first 5 rows

df <- read.xlsx("<name and extension of your file>",

sheetIndex = 1)

* Export Data Frame to Excel

write.xlsx(df,

"<name and extension of your existing file>",

sheetName="Data Frame"

append=TRUE) #if write to existing file

* Check Data format

*Str(*df)

* Rename columns

library(plyr)

rename(df, c("Header1"="NewHeader1", "Header2"="NewHeader2")) #from plyr library

*colnames(*df*) <- c(“*Header1”*,* “Header2”*)*

* Check duplicates

length(unique(as.character(df$var))) #check distinct volume

*dup.ticket <- as.character(df[which(duplicated(as.character(df$var))), "var"]) #pull out duplicates for checking*

* Remove duplicates / or distinct data

# To get duplicate records

*df [duplicated(df),] #on entire data frame*

Or

*dup.name <- as.character(df [which(duplicated(as.character(df$var))), "Name"])*

*df[which(df$var %in% dup.name), ] #on one variable*

Or

df[!duplicated(df[c("var1","var2")]),] *#on two or more variables*

# To get the distinct records:

df[!duplicated(df$Var1), ] #based on one variable

Or

*unique(x) #only on data list*

# To count number of distinct:

*length(*df[!duplicated(df$Var), “Var” ]*) #use nrow if counting on Data Frame*

Or

*length(*unique(x)) *#only on data list*

* Create new data columns

var1 <- transform(var0, margin = round((vara/varb) \* 100, 1))

Or

*test.AddFlag <- data.frame(Survived= rep("None", nrow(test)), File=rep("test", nrow(test)), test[,])*

* Match / Merge two datasets

merge(df1, df2, by.x="var1", by.y="var2")

Or

data.frame(var1=df1$var1, var3=df2[match(df1$var2, df3$var4), ])

Or

colnames(df2) <- c("var2", "var4") #change the column name to the same as df1 for matching

library(dplyr)

left\_join(df1 , df2)

* Append datasets (either by rows or columns)

df\_byrows <- rbind(df1 , df2) #append by rows

df\_bycols <- cbind(df1 , df2) #append by columns

* Remove rows

df <- df[-c(2, 4, 6), ] # “-“ sign indicates remove (remove 2nd, 4th & 6th rows)

Or

df <- df[!df$Var > 4,] # remove rows (by !) where Var is > 4

* Check / Change Data format, i.e. time / date / currency

df$var <- round(df$var, 1) #round to 1 decimal

# convert numeric to currency format – 3 steps, including creating a function

class(df$Var) <- c("money", class(df$Var))

print.money <- function(x, ...) {

print.default(paste0("$", formatC(as.numeric(x), format="f", digits=2, big.mark=",")))

}

format.money <- function(x, ...) {

paste0("$", formatC(as.numeric(x), format = "f", digits=2, big.mark=","))

}

df$Var1 <- format.money(df$Var)

* Sort / Order / Rank data (ascending or descending)

# Order is ranking on position – putting the 3rd data first etc. ; rank is to produce the rank. Therefore

a[order(a)] = sort(a)

#To create Rank column and order by group

transform(df, rank1 = ave(rank\_Var, Group\_by\_var,

FUN = function(x) rank(-x, ties.method = "first"))) #change “-x” to “x” for ascending

Or

library(data.table)

DT <- as.data.table(df)

DT[, rank1:=rank(-rank\_Var,ties.method="first"),by= Group\_by\_var]

Or

transform(dt, rank1 = ave(rank\_Var, Group\_by\_var, FUN=function(x) order(x,decreasing=T)))

#To get frequency table and rank by Count

*df.count <- data.frame(sort(table(df$Var), decreasing = T))*

* Create groups / categorical data (and summing the values)

df$group <- with(df, ifelse(name %in% group1, "group1",

ifelse(name %in% group2, "group2", "group3" )))

aggregate(value ~ group, sum, data=df)

* Replace Null data with Mean / Median

#To replace missing data with mean within a column

Column1[is.na(Column1)] <- mean(Column1, na.rm = TRUE)

#To replace missing data with mean within all columns

for(i in 1:ncol(data)){

data[is.na(data[,i]), i] <- mean(data[,i], na.rm = TRUE)

}

#Although the best way would be:

for(i in 1:ncol(data)){

data[is.na(data[,i]), i] <- mean(data[,i],

na.rm = T)+rnorm(sum(is.na(data[,i])))\*sd(data[,i], na.rm = T)

}

* Search in text (strings) / Replace string (two steps)

## a) Create extract function

extractfunction <- function(name) {

name <- as.character(name)

if(length(grep("Miss.", name)) > 0) {return ("Miss") }

else if(length(grep("Mr.", name)) > 0) {return("Mr")}

else if(length(grep("Mrs.", name)) >0) {return("Mrs") }

else if(length(grep("Master.", name)) >0) {return("Master")}

else {return("Other")}

}

## b) Apply extract function to obtain function

title <- NULL

for (i in 1:nrow(train.TickVol)) {

title <- c(title, extractfunction(train.TickVol[i, "Name"]))

}

* Substring (e.g. take the first character of the string)

substr(df$Var, start, stop)

* Combine keys / Composite keys / concatenate

> x <- c("Hello", "World")

> x

[1] "Hello" "World"

> paste(x, collapse="--")

[1] "Hello--World"

> paste(x, "and some more", sep="|-|", collapse="--")

[1] "Hello|-|and some more--World|-|and some more"

* Remove space

library(stringr)

str\_replace\_all(x, fixed(" "), "")

str\_replace\_all(x, space(), "")

* Upper case / lower case

toupper(v)

#To apply withina number of columns

data.frame(lapply(df, function(v) {

if (is.character(v)) return(toupper(v))

else return(v)

}))

* Take a subset from a data frame

head(subset(mtcars, select = 'gear'))

Part 2 - Data Calculations

(http://www.gardenersown.co.uk/education/lectures/r/correl.htm)

* Checking data

#To look at the first few rows

*Head(df)*

#Check number of rows/columns

*dim(data)*

* Date format

*as.Date("2001-01-01", format = "%Y-%m-%d")*

* To calculate percentage

*count <- function(x, n){ length((which(x == n))) }  
perc <- function(x, n){ 100\*length((which(x == n))) / length(x) }*

* Calculate column sum where there is missing data

sum(people$Weight,na.rm=TRUE)

#Column sum

colSums (x, na.rm = FALSE, dims = 1)

rowSums (x, na.rm = FALSE, dims = 1)

colMeans(x, na.rm = FALSE, dims = 1)

rowMeans(x, na.rm = FALSE, dims = 1)

#To sum all variables in the data frame

# get means for variables in data frame mydata  
# excluding missing values

sapply(mydata, mean, na.rm=TRUE)

#Or, use aggregate to sum by groups

Var\_sum <- aggregate(df[,3:4], list(df$groupby\_var), FUN = sum)

#Or apply

apply(df, margin, mean) #Margin: 1 indicates rows, 2 indicates columns; c(1,2) columns&rows

* Calculate cumulative sum

*df$csum <- ave(df$value, df$id, FUN=cumsum)*

* Count number of distinct / duplicate data on one or more columns

#Number of distinct data

length(unique(df$Var*)* #On one column

sapply(Data.Combined, function(x) length(unique(x))) #on all columns

OR

*lapply(Data.Combined, function(x) length(unique(x)))* #on all columns

* Count number of matched data base on one or more fields

#Between two list

v1 <- c("x", "x", "y", "y", "z")

v2 <- c("x", "x", "x", "z", "z")

intersect(v1, v2)

# [1] "x" "z"

length(intersect(v1, v2))

# [1] 2

OR

df1.2 <- df2[df2$KeyID %in% df1$KeyID,]

length(unique(df1.2$KeyID))

#For multiple vectors

v1 <- c("x", "x", "y", "y", "z")

v2 <- c("x", "x", "x", "z", "z")

v3 <- c("x", "y", "y", "z")

vector.list <- list(v1, v2, v3)

Reduce("intersect", vector.list)

# [1] "x" "z"

* Count number of Nulls

*sum(is.na(df$Var)) #use !is.na for not Nulls*

* Calculate distance between two points

euc.dist <- function(x1, x2) sqrt(sum((x1 - x2) ^ 2))

dist <- NULL

for(i in 1:nrow(x1)) dist[i] <- euc.dist(x1[i,],x2[i,])

dist

#For large dataset

library(foreach)

foreach(i = 1:nrow(x1), .combine = c ) %do% euc.dist(x1[i,],x2[i,])

* Statistics Summary, e.g. mean, median, min, max, frequency  
    
  summary(mydata)

OR

sapply(mydata, mean, na.rm=TRUE)

OR

library(Hmisc)  
describe(mydata)   
# n, nmiss, unique, mean, 5,10,25,50,75,90,95th percentiles   
# 5 lowest and 5 highest scores

OR

library(pastecs)  
stat.desc(mydata)   
# nbr.val, nbr.null, nbr.na, min max, range, sum,   
# median, mean, SE.mean, CI.mean, var, std.dev, coef.var

OR

library(psych)  
describe(mydata)  
# item name ,item number, nvalid, mean, sd,   
# median, mad, min, max, skew, kurtosis, se

* Calculate mean / median by groups

Var\_sum <- aggregate(df[,3:4], list(df$groupby\_var), FUN = sum)

* Produce Cross-tab plot

x<-sample(1:20,20)+rnorm(10,sd=2)

y<-x+rnorm(10,sd=3)

z<-(sample(1:20,20)/2)+rnorm(20,sd=5)

df<-data.frame(x,y,z)

plot(df[,1:3])

* Calculate Cross- Covariance between data

#For Co-Variance

covMat1Mat2(object, X1, X2, nugget.flag=FALSE)

* Calculate Correlation between data

#Correlation Matrix

d <- data.frame(x1=rnorm(10),

x2=rnorm(10),

x3=rnorm(10))

library(ellipse)

M <- cor(d[,1:3], method="pearson") # get correlations

M

cor.test(d$x1,d$x2,method="pearson") # to test significance (p-value) by pearson

cor.test(df$x,df$y,method="spearman") # Or to test by pearson, spearman or kendall

library(corrplot) # presenting the correlation matrix

corrplot(M, method="circle")

corrplot(M, method="ellipse")

corrplot(M, method="number")

* Calculate Pearson’s Chi-squared Test / Goodness of Fit test

attach(df)

your.chi = chisq.test(var1, p=var2, rescale.p=T) OR

your.chi = chisq.test(observed.data, p=expected.values, rescale.p=T)

# As before we can extract the expected values and the residuals:

your.chi$exp  
your.chi$res

your.chi$obs

* Calculate ANOVA for Linear Model Fits

attach(df)

your.aov = aov(Var1 ~ Var2)

your.aov

summary(your.aov)

* Calculate t-test

attach(df)

t.test(var1, var2, var.equal=T)

Part 3 - Data Visualisation

* Frequency table / Cross-tab

table(df$Var) #just to find out the volume

as.data.frame(df$Var) # to turn the frequency table as a data frame

OR

Var\_Freq <- data.frame(df$Var) #could be used to append back to the data

# using count in “plyr” package

install.packages('plyr')

library(plyr)

count(df, 'Var') #this will return as a data frame

#OR for N-way

> t = count(mtcars, c('cyl', 'gear', 'vs'))

> t

cyl    gear    vs     freq

1      4      3         1       1

2      4      4         1       8

3      4      5         0       1

4      4      5         1       1

5      6      3         1       2

6      6      4         0       2

7      6      4         1       2

8      6      5         0       1

9      8      3         0       12

10    8      5         0       2

#Use xtabs for cross tabulation

> y = xtabs(~ cyl + gear, mtcars)

> y

gear

cyl      3     4     5

4      1     8     2

6      2     4     1

8     12    0     2

# #Cross tabulation of two variables with resulting average and SD

with(df, tapply(Var\_forCal, list(Var\_group1= Var\_group1, Var\_group2= Var\_group2), mean) )

with(df, tapply(Var\_forCal, list(Var\_group1= Var\_group1, Var\_group2= Var\_group2), sd) )

* For relative frequency

*prop.table(crossTab) # % of all in the crosstab*

prop.table(cTab, margin=1) # % by rows in the crosstab ( margin=2 for columns)

* Bar chart / Histogram

#Using bar plot

barplot(VADmeans, main="Road Deaths in Virginia",xlab="Categories", ylab="Mean Deaths")

OR

barplot(table(datalist))

OR

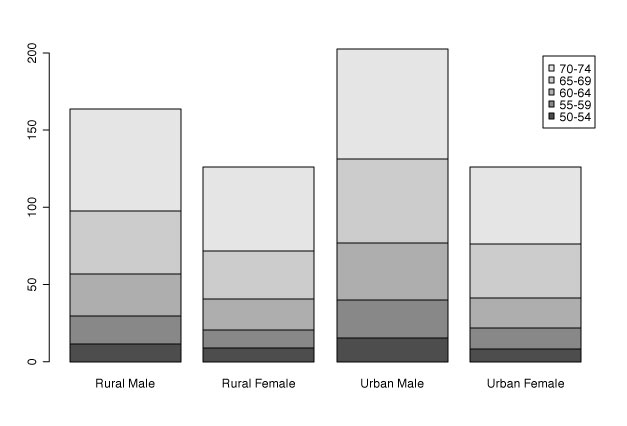
hist(test.data, probability = TRUE) #For %

OR

hist(test.data, 10, xlim=c(0,6), ylim=c(0,10)) #to limit the axis

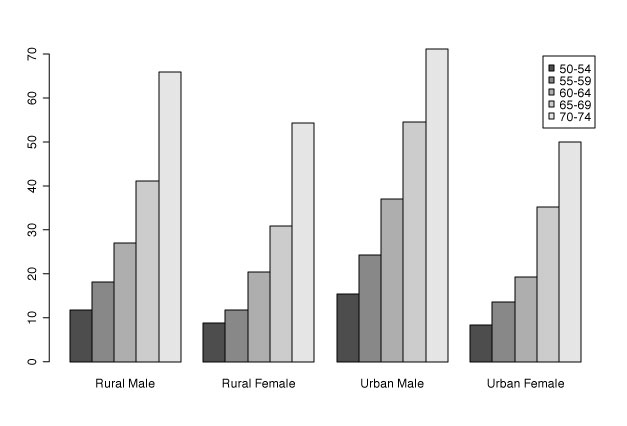
#### #Stacked charts

barplot(VADeaths, legend= rownames(VADeaths))



#bars by the side

barplot(VADeaths, legend= rownames(VADeaths), beside= TRUE)



#use gg-plot

install.package(ggplot2)

library(ggplot)

ggplot(train.TickVol, aes(x=Age, fill = Survived)) + #Histogram on numeric

facet\_wrap(~train.TickVol$Sex + train.TickVol$Pclass) +

geom\_histogram(binwidth = 10) +

xlab("Age") +

ylab("Total Count")

#Or bar chart on factor

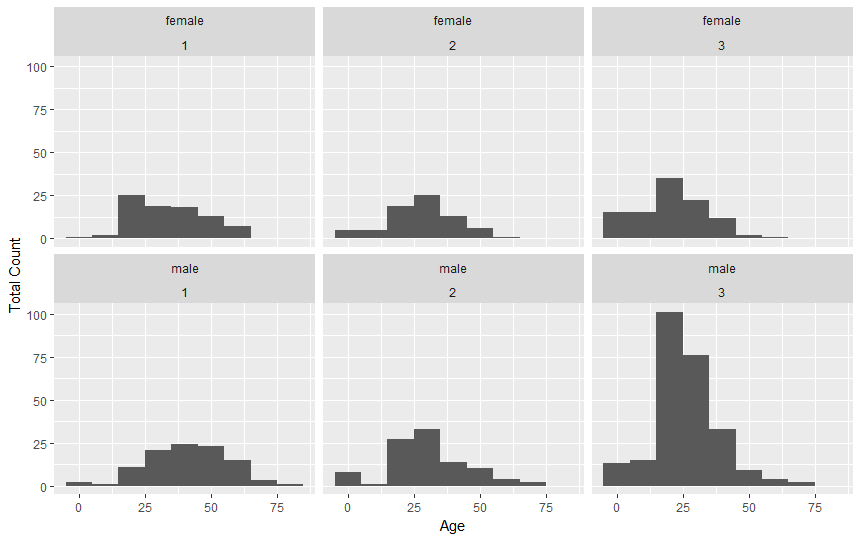
ggplot(train.TickVol, aes(x=Age, fill = Survived)) + #Bar chart on factor

facet\_wrap(~train.TickVol$Sex + train.TickVol$Pclass) +

geom\_bar(binwidth = 10) +

xlab("Age") +

ylab("Total Count")

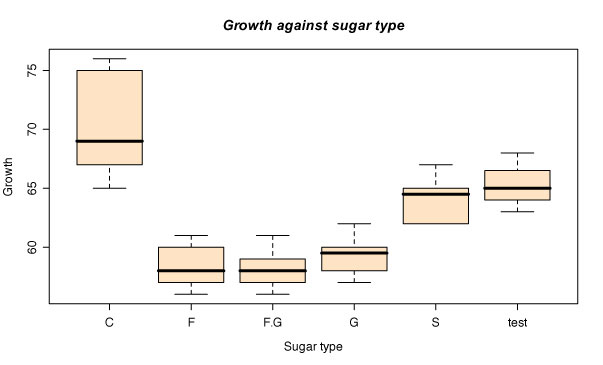


### Box and whisker plots

boxplot(test.data, xlab="Single sample", ylab="Value axis", main="Simple Box plot", col="lightblue")

#For multiple box=plot

boxplot(growth ~ sugar, data=fly, xlab="Sugar type", ylab="Growth", col="bisque", range=0)



#Or for horizontal box plot

boxplot(growth ~ sugar, data=fly, ylab="Sugar type", xlab="Growth", col="mistyrose", range=0, horizontal=TRUE)

* Line graphs / time series

*plot(cars, type="o", col="blue", ylim=c(0,12)) #just the plot*  
*lines(trucks, type="o", pch=22, lty=2, col="red") #line graph needs to be used with plot*

# Create Line Chart - more complex  
  
*# convert factor to numeric for convenience   
Orange$Tree <- as.numeric(Orange$Tree)   
ntrees <- max(Orange$Tree)  
  
# get the range for the x and y axis   
xrange <- range(Orange$age)   
yrange <- range(Orange$circumference)   
  
# set up the plot   
plot(xrange, yrange, type="n", xlab="Age (days)",  
   ylab="Circumference (mm)" )   
colors <- rainbow(ntrees)   
linetype <- c(1:ntrees)   
plotchar <- seq(18,18+ntrees,1)  
  
# add lines   
for (i in 1:ntrees) {   
  tree <- subset(Orange, Tree==i)   
  lines(tree$age, tree$circumference, type="b", lwd=1.5,  
    lty=linetype[i], col=colors[i], pch=plotchar[i])   
}   
  
# add a title and subtitle   
title("Tree Growth", "example of line plot")  
  
# add a legend   
legend(xrange[1], yrange[2], 1:ntrees, cex=0.8, col=colors,  
   pch=plotchar, lty=linetype, title="Tree")*

#Easier to use ggplot

ggplot(data=dat, aes(x=time, y=total\_bill, group=1)) +

geom\_line() +

geom\_point()

OR

ggplot(data=dat1, aes(x=time, y=total\_bill, group=sex, colour=sex)) + #Use sex for colours

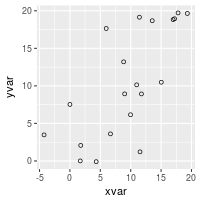
geom\_line() +

geom\_point()

* Scatter plots

ggplot(dat, aes(x=xvar, y=yvar)) +

geom\_point(shape=1) # Use hollow circles

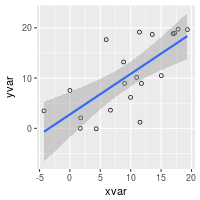


ggplot(dat, aes(x=xvar, y=yvar)) +

geom\_point(shape=1) + # Use hollow circles

geom\_smooth(method=lm) # Add linear regression line

# (by default includes 95% confidence region)

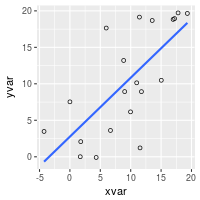


ggplot(dat, aes(x=xvar, y=yvar)) +

geom\_point(shape=1) + # Use hollow circles

geom\_smooth(method=lm, # Add linear regression line

se=FALSE) # Don't add shaded confidence region

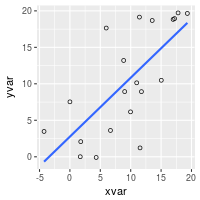


ggplot(dat, aes(x=xvar, y=yvar)) +

geom\_point(shape=1) + # Use hollow circles

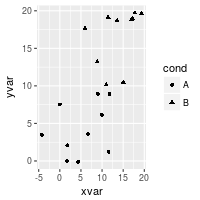
geom\_smooth() # Add a loess smoothed fit curve with confidence region

#> `geom\_smooth()` using method = 'loess'



*# Set shape by cond*

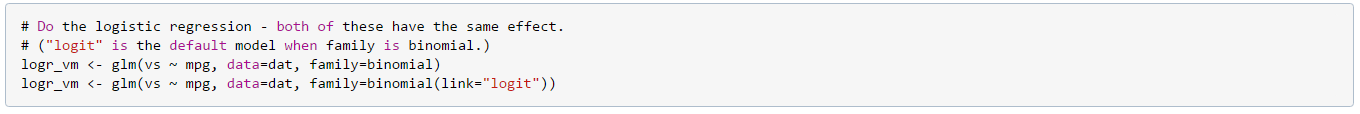
ggplot(dat, aes(x=xvar, y=yvar, shape=cond)) + geom\_point() #may use colour for groups

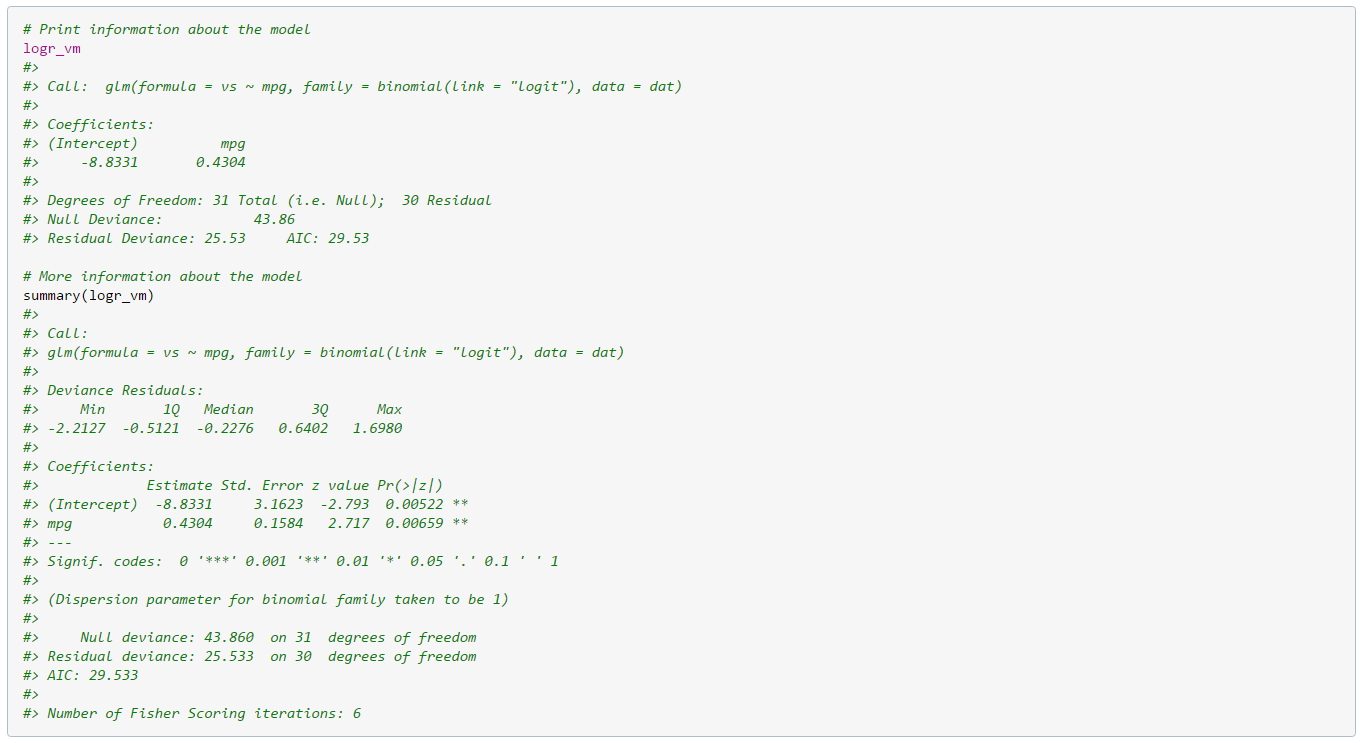


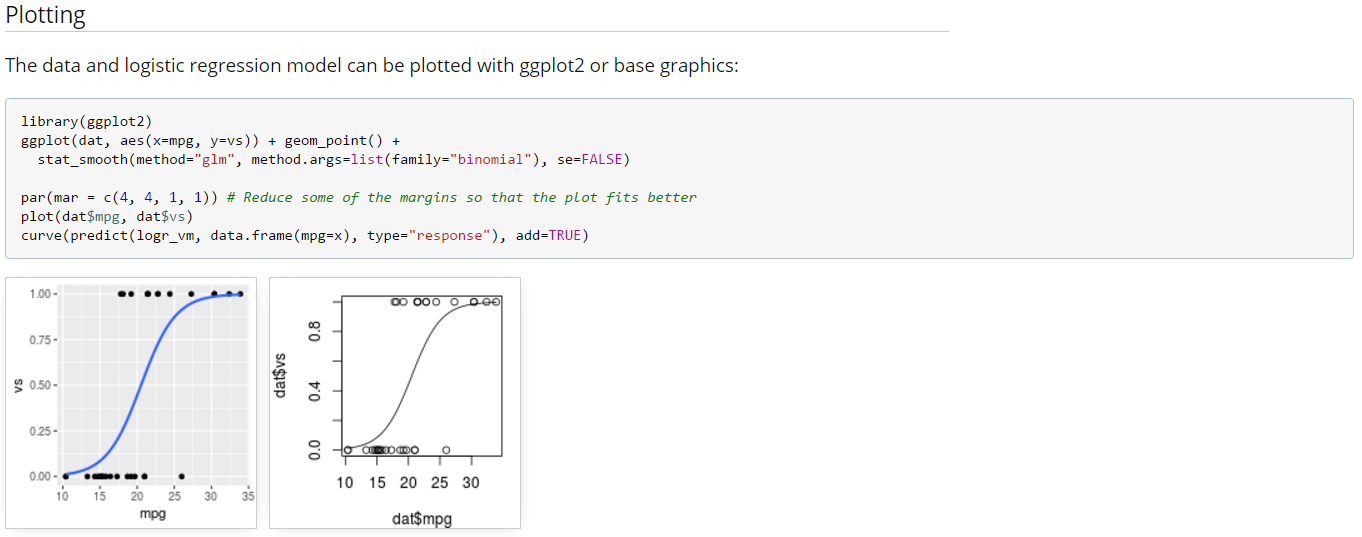
Part 4 – Building Model  
(http://www.cookbook-r.com/Statistical\_analysis/Logistic\_regression/)

* Decision tree
* Linear Regression









* Logit Regression
* Cluster
* Propensity Model