**Assignment 1**

1. Suppose you keep track of your mileage each time you fill up your car’s gas tank. At your last fill-ups, the mileage values were : 65311 , 65624 , 65908 , 66219 , 66499 , 66821 , 67145 , 67447 respectively. Enter these numbers into an R vector. Use the function diff() on the data. What is the result?

x = c(65311 , 65624 , 65908 , 66219 , 66499 , 66821 , 67145 , 67447)

diff(x)

1. Use the function length() to find the lengths of the miles and x vectors. Explain the result.

length(x)

1. Use R functions to find the maximum and minimum number of miles between fill-ups, the average number of miles between fill-ups, the standard deviation.

y = diff(x)

max(y)

min(y)

mean(y)

sd(y)

summary(x)

1. You have tracked your commute times for two weeks (10 days) and you recorded the following times in minutes:17 16 20 24 22 15 21 15 17 22. Enter this into a vector called commute. Use R functions to find the average and a statistical summary.

Commute = c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)

mean(Commute)

summary(Commute)

1. Suppose the 24 was a mistake; it should have been 18. How can you fix this? Do so, and then find the new average.

Commute[4] = 18

mean(Commute)

1. How many times was your commute 20 minutes or more?

length(Commute[Commute>= 20])

1. What percent of your commutes are less than 17 minutes? How can you answer this with R?

length(Commute[Commute < 17])/length(Commute)

1. Briefly explain the result of the following commands:

commute + 1

commute[-6]

sum(commute>21 | commute<16)

commute[commute>=17]

sum(commute[commute>20])

commute[4:6]

The function above adds one to every value in the vector.

The function above shows every value except 6th item in the vector.

The function above shows the total cumulative of values which are either greater than 21 or less than 16 in the vector.

The function above shows all values which are greater than or equal to 17 in the vector.

The function above shows the total cumulative time of commute of 20 minutes or more.

The function above shows 4th, 5th and 6th items of the vector.

1. Let the data x be given by

x = c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5)

Use R to compute the following functions. Note, we use x1 to denote the first element of x (which is 1) etc.

(X1 + X2 + X3. . .+X10)/10. Use sum

y = sum(x) /10

print(y)

1. Find log10(Xi) for each i?

Log(x)

1. Find (X4.4 )/2.875 for each .

(x^4.4) / 2.875

1. Find the difference between the largest and smallest values of x. (This is the range. You can use max and min or guess a built in command.)

max(x)-min(x)

1. Your cell phone bill varies from month to month. Suppose that your bills had the following monthly amounts last year:

bill = c(46, 33, 39, 37, 46, 30, 48, 32, 49, 35, 30, 48)

Run the following calculations, and explain its result:

range(bill)

range(bill)[2] - range(bill)[1]

This computes the difference between the smallest and largest cell phone bills last year.

1. How many months was the amount greater than $40? What percentage was this?

length(bill[bill>40])/length(bill)

Part 1

1. More types

typeof() function returns the type

is.foo() functions return Booleans for whether the argument is

of type foo

as.foo() (tries to) “cast” its argument to type foo — to translate it

sensibly into a foo-type value

typeof(7)

[1] "double"

is.numeric(7)

[1] TRUE

is.na(7)

[1] FALSE

is.character(7)

[1] FALSE

is.character("7")

[1] TRUE

is.na("seven")

[1] FALSE

as.character(5/6)

[1] "0.833333333333333"

as.numeric(as.character(5/6))

[1] 0.8333333

1. First data structure: vectors

Group related data values into one object, a data structure.

A vector is a sequence of values, all of the same type.

x = c(7, 8, 10, 45)

x

[1] 7 8 10 45

is.vector(x)

[1] TRUE

c() function returns a vector containing all its arguments in

order

x[1] is the first element, x[4] is the 4th element

x[-4] is a vector containing all but the fourth element.

1. vector(length=6) returns an empty vector of length 6; helpful

for filling things up later.

weekly.hours = vector(length=5)

weekly.hours[5] <- 8

1. Operators apply to vectors “pairwise” or “elementwise”:

x = c(7, 8, 10, 45)

y <- c(-7, -8, -10, -45)

x+y

[1] 0 0 0 0

1. Recycling repeats elements in shorter vector when combined with longer

x = c(7, 8, 10, 45)

x + c(-7,-8)

[1] 0 0 3 37

1. To compare whole vectors, best to use identical() or all.equal():

x = c(7, 8, 10, 45)

y = c(-7, -8, -10, -45)

x == -y

[1] TRUE TRUE TRUE TRUE

identical(x,-y)

[1] TRUE

identical(c(0.5-0.3,0.3-0.1),c(0.3-0.1,0.5-0.3))

[1] FALSE

all.equal(c(0.5-0.3,0.3-0.1),c(0.3-0.1,0.5-0.3))

[1] TRUE

1. Functions on vectors

Lots of functions take vectors as arguments:

mean(), median(), sd(), var(), max(), min(), length(), sum():

return single numbers.

sort() returns a new vector.

hist() takes a vector of numbers and produces a histogram, a highly structured object, with the side-effect of making a plot.

Similarly ecdf() produces a cumulative-density-function object.

summary() gives a five-number summary of numerical vectors.

any() and all() are useful on Boolean vectors.

1. which() turns a Boolean vector in vector of TRUE indices:

x = c(7, 8, 10, 45)

y = c(-7, -8, -10, -45)

places = which(x > 9)

places

[1] 3 4

y[places]

[1] -10 -45

1. You can give names to elements or components of vectors:

x = c(7, 8, 10, 45)

names(x) = c("v1","v2","v3","fred")

names(x)

[1] "v1" "v2" "v3" "fred"

x[c("fred","v1")] displays 45 and 7.

names(x) is just another vector (of characters):

sort(names(x))

[1] "fred" "v1" "v2" "v3"

which(names(x)=="fred")

[1] 4

**Part 2**

1. Matrices:

A matrix is a vector with a dim attribute, i.e. an integer vector

giving the number or rows and columns.

To create matrices use matrix().

matrix(1:15, nrow = 4)

[,1] [,2] [,3] [,4]

[1,] 1 5 9 13

[2,] 2 6 10 14

[3,] 3 7 11 15

The functions dim(), nrow() and ncol() provide the attributes

of the matrix.

Rows and columns can have names, dimnames(),

rownames(), colnames().

factory = matrix(c(40,1,60,3),nrow=2 )

is.array(factory)

[1] TRUE

is.matrix(factory)

[1] TRUE

1. We can name either rows or columns or both, with rownames() and colnames()

These are just character vectors, and we use the same function to get and to set their values.

factory = matrix(c(40,1,60,3),nrow=2 )

rownames(factory) = c("labor","steel")

colnames(factory) = c("cars","trucks")

factory

1. Apply(), takes 3 arguments: the array or matrix, then 1 for rows

and 2 for columns, then name of the function to apply to each.

factory = matrix(c(40,1,60,3),nrow=2 )

factory

apply(factory,1,mean)

1. Lists:

Sequence of values, not necessarily all of the same type

my.distribution = list("exponential",7,FALSE)

my.distribution

Add to lists with c() (also works with vectors):

my.distribution = c(my.distribution,7)

my.distribution

[[1]]

[1] "exponential"

[[2]]

[1] 7

[[3]]

[1] FALSE

[[4]]

[1] 7

1. Chop off the end of a list by setting the length to something smaller (also works with vectors):

my.distribution = list("exponential",7,FALSE)

my.distribution = c(my.distribution,7)

my.distribution

length(my.distribution)

[1] 4

length(my.distribution) = 3

my.distribution

[[1]]

[1] "exponential"

[[2]]

[1] 7

[[3]]

[1] FALSE

1. Dataframes:

Dataframe = the classic data table, rows for cases, columns

for variables.

Not just a matrix because columns can have different types.

Many matrix functions also work for dataframes (rowSums(),

summary(), apply()).

1. Creating an example dataframe:

library(datasets)

states = data.frame(state.x77, abb=state.abb, region=state.region, division=state.division)

data.frame() is combining here a pre-existing matrix (state.x77),

a vector of characters (state.abb), and two vectors of qualitative

categorical variables (factors; state.region, state.division).

1. Column names are preserved or guessed if not explicitly set.

colnames(states)

[1] "Population" "Income" "Illiteracy" "Life.Exp" "Murder"

[6] "HS.Grad" "Frost" "Area" "abb" "region"

[11] "division"

states[1,] = this shows the first row of the data set.

1. Dataframe access:

By row and column index

states[49,3]

[1] 0.7

By row and column names

states["Wisconsin","Illiteracy"]

[1] 0.7

All of a column:

head(states[,3])

[1] 2.1 1.5 1.8 1.9 1.1 0.7

head(states[,"Illiteracy"])

[1] 2.1 1.5 1.8 1.9 1.1 0.7

head(states$Illiteracy)

[1] 2.1 1.5 1.8 1.9 1.1 0.7

1. Parts or all of the dataframe can be assigned to:

summary(states$HS.Grad)

Min. 1st Qu. Median Mean 3rd Qu. Max.

37.80 48.05 53.25 53.11 59.15 67.30

1. Adding rows and columns:

We can add rows or columns to an array or data-frame with

rbind() and cbind(), but be careful about forced type conversions.

**Part 3**

1. Descriptive Statistics

Centrality and Spread measures for Quantitative Variables

Measures of Central Tendancy

Mean

Median

Mode

Percentiles

Quartiles

Measures of Spread (Dispersion)

Range

IQR (Interquartile Range)

Variance

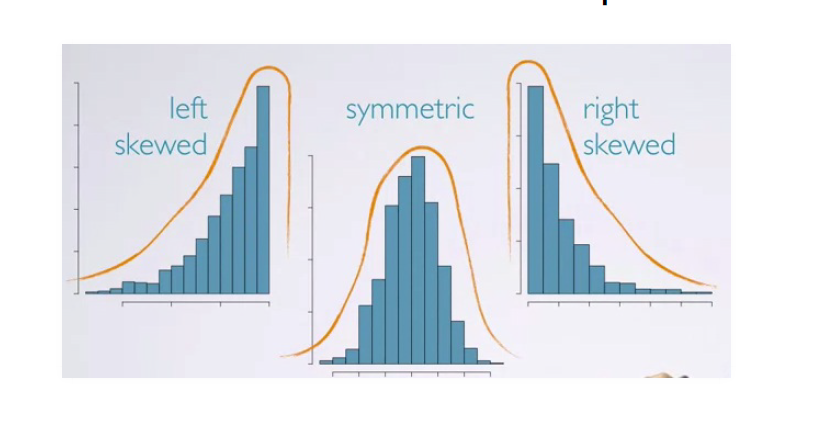
Standard Deviation

1. When do we use median instead of average ?

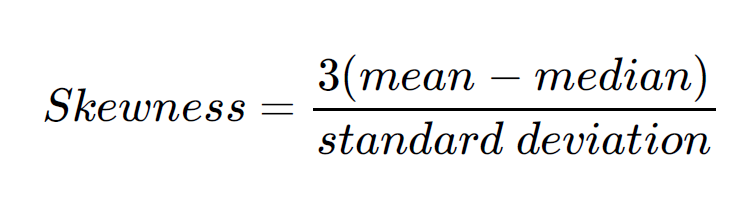
When mean is affected by each value in the dataset, including

extreme outliers.

1. Measurements of shape – Skeweness:



1. Skewness



1. GGplot for visualization

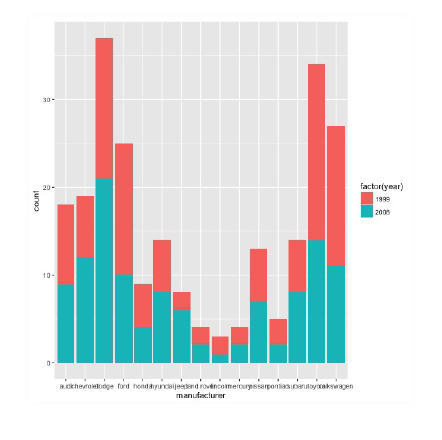
We usually use libraries that can generate nicer looking graphs.

The syntax is a little more complicated tough

# install.packages(ggplot2)

require(ggplot2)

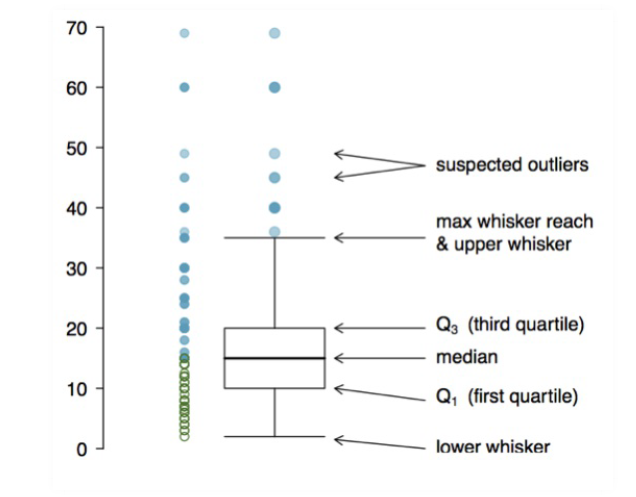
ggplot(data = mpg) + geom\_bar(aes(x = manufacturer, fill = factor(year)))



1. Box Plot:

The box in a box plot represents the middle 50% of the data,

and the thick line in the box is the median.



**Assignment 2**

1. For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

vector1 = c("5", "12", "7", "32")

max(vector1)

sort(vector1)

sum(vector1)

vector1 = c("5", "12", "7", "32") Loads inputs as characters.

max(vector1) Since values are characters, max function gives the last value when sorted alphabetically.

sort(vector1) sorts values alphabetically.

`sum(vector1) : Error occurs because sum() function expects a vector of numeric values.

1. For the next series of commands, either explain their results, or why they should produce errors.

vector2 = c("5",7,12)

vector2[2] + vector2[3]

Vectors are group of values of the same kind. Since the first value is declared as a character, the following ones are assummed so.

1. dataframe3 = data.frame(z1="5",z2=7,z3=12)

dataframe3[1,2] + dataframe3[1,3]

data frames can have columns of different kind so second and third columns are accepted as numeric. So the sum of 2nd and 3rd column of the 1st row is 19 ( 7 + 12).

Also, Dataframe is a classic data table and Z1, Z2 and z3 are column names which can have their own datas types. Therefore the result of 19 is correct as we are adding column z2 and z3 which have same data types.

1. list4 <- list(z1="6", z2=42, z3="49", z4=126)

list4[[2]]+list4[[4]]

list4[2]+list4[4]

Values are named, so to access the values one should write list4[[2]]+list4[[4]]

1. mtcars , another dataset under the datasets library in R, was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). Note: Although both are about cars, mtcars is a different dataset than the mpg dataset we worked with in the class.What is the type of mtcars ? Also is it a data frame?

data(mtcars)

typeof(mtcars)

## [1] "list"

is.data.frame(mtcars)

## [1] TRUE

1. How many rows and columns does mtcars have?

nrow(mtcars)

ncol(mtcars)

or

dimension=dim(mtcars)

dimension[1] #rows

## [1] 32

dimension[2] #columns

## [1] 11

1. What are the names of the columns of mtcars ?

colnames(mtcars)

or

names(mtcars)

1. What is the value of row 5, column 7 of mtcars ? What does the value signify ?

mtcars[5,7]

its the qsec of Hornet sportabout

1. Display the second row of mtcars in its entirety.

mtcars[2,]

1. Explain what this command does by running it on your data and examining the object.

Mydata = mtcars

names(mydata) = c("mileage","cylinder",seq(0,8))

It changed the column names to "mileage","cylinder",0,1,2,3,4,5,6,7,8.

1. In engines, engine size is evaluated by the total volume of its cylinders and it is referred to as displacement. The larger the displacement (hence the engine size), the more power the engine is likely to produce. To compare engines that are in different sizes we look at the engine power (hp) it generates (in horse power) for unit displacement (in liter). In the dataset, engine power is under hp column and engine displacement is under disp column. However the challenge is that in the dataset displacement (disp) is given in cubic inches. You first need to convert it to liter. To do that, first calculate the displacement in liter by considering that 1 cubic inch is roughly 0.0163871 liters.

Converttoliters = mtcars[,3]\*0.0163871

converttoliters

hp\_over\_displ= mtcars$hp/ converttoliters

mtcars2=data.frame(mtcars, hp\_over\_displ)

# Alternatively we can use cbind(exisiting datafame, new values)

1. Next, assign these values to a variable called disp\_l and add this vector as a column to your data(mydata)

mydata$disp\_I = hp\_over\_displ

1. Working with functions and operators. The colon operator will create a sequence of integers in order. It is a special case of the function seq() which you saw earlier in this assignment. Using the help command ?seq to learn about the function, design an expression that will give you the sequence of numbers from 1 to 10000 in increments of 372. Design another that will give you a sequence between 1 and 10000 that is exactly 50 numbers in length. The function rep() repeats a vector some number of times. Explain the difference between rep(1:3, times=3) and rep(1:3, each=3) .

seq(from=1, to=10000, by=372)

seq(from=1, to=10000, length.out=50)

rep(1:3, times=3)

rep(1:3, each=3)

# rep(1:3, times=3) repeats c(1, 2, 3) 3 times whereas rep(1:3, each=3) produces each 1 1 1 2 2 2 3 3 3

1. Yet another dataset for cars! MASS library has a dataset called Cars93 for 93 cars with model year 1993. Load the dataset following the instructions below.

library(MASS)

data(Cars93)

Begin by examining the data frame with the command View(Cars93) to understand the underlying object. You will need to use functions and other commands to extract elements for this assignment. Obtain a summary() of the full data structure. Can you tell from this how many rows are in the data? If so, say how; if not, use another method to obtain the number of rows.

summary(Cars93)

1. ars - checked the Mans.trans.avail binary variable. 32 No + 61 Yes
2. What is the mean and median price of a car with a rear-wheel drive train? Which one is a better measure for centrality? What is the standard deviation for a similar car?

rearwheel = subset(Cars93,DriveTrain ="Rear" )

rearwheel

mean(rearwheel$Price)

median(rearwheel$Price)

hist(rearwheel$Price)

Median is better because the distribution is skewed to the right - that is there are few relatively high priced cars that affects the mean.

1. What is the minimum horsepower of all cars with capacity for 7 passengers? With a capacity of at least 6 passengers?

sd(rearwheel$Price)

minhorsepr7 <- subset(Cars93,Passengers == 7 )

min(minhorsepr7$Horsepower)

minhorsepr6 <- subset(Cars93,Passengers == 6 )

min(minhorsepr6$Horsepower)

**Part 4**

1. Data preprocessing alone can account for 10 – 60% of all the time and effort for the entire data mining process.
2. Raw Data: Not necessarily all correct, not formatted for systems to accept.
3. Technically Correct Data: Can be read, but doesn't mean values are correct (e.x.

negative age values, missing data).

1. Consistent Data: Data ready for analysis and interpretation.
2. Loading Data:

The followings commands are good for loading fixed-width or

csv-like formats, (not for XML-like formats).

read.table

read.delim

read.delim2

read.csv

read.csv2

1. With read.table you can load data either from a web source or from a local file.

data.url = "http://yegingenc.com/lectures/data/SampleStudentGrades.txt"

sample.grades = read.table(data.url, header = T)

head(sample.grades)

1. Cleaning Data 1: Technically correct data:

Converting raw data to technically correct data. That is making sure

That variables are correctly populated and each value belongs to a variable.

Data types in the system match to the variable types. (Text variables are

stored as text, numeric variables are stored as numbers etc.)

1. summary() is a good function to get descriptive stats for each variable in the data set. Based on the variable type, it will give different statistics. For numeric variables (e.x. Grades), mean, median and quartiles. For categorical variables (e.x. Semester), it gives frequencies.
2. The problem is tough that sometimes numeric values are used for other purposes, for example student\_id. (i.e. Mean student\_id doesn't make sense).So we change it to factor or make it the row name.

sample.grades$Student\_ID <- as.factor(sample.grades$Student\_ID).

**Assignment 3**

1. Download the grades data from " http://yegingenc.com/lectures/data/SampleStudentGrades.txt" and save it to a ‘.csv’ file. Load the data to R. 4 – Which variables are numerical?

scores <- read.table('http://yegingenc.com/lectures/data/SampleStudentGrades.txt', header = TRUE)

scores

1. Which variables are numerical?

Grades are numerical.

1. What are average and standard deviations for each semester?

Fall14semester <- subset(scores, Semester=="14\_Fall")

Fall14semester

Fall15semester <- subset(scores, Semester=="15\_Fall")

Fall15semester

Spring15semester <- subset(scores, Semester=="15\_Spring")

Spring15semester

mean(Fall14semester$Grades)

mean(Fall15semester$Grades)

mean(Spring15semester$Grades)

sd(Fall14semester$Grades)

sd(Fall15semester$Grades)

sd(Spring15semester$Grades)

##Alternatively we can take advantage of libraries like dplyr and magrittr

library(magrittr)

library(dplyr)

data %>%

group\_by(Semester) %>%

summarise(Avg = mean(Grades), SD = sd(Grades))

1. Plot a histogram for the grades? Judging by the histogram you just created what can you say about the distribution?

hist(scores$Grades)

The data is negatively skewed in histogram.

1. Calculate the skewness of the data.

skewness = function(x) {

return(3 \* (mean(x) - median(x))/sd(x))

}

skewness(scores$Grades)

1. Using transformation techniques you just learnt try to normalize the grades distribution.

# Removing the outliers using transformation for variable grades which is not normally distributed

#Log Transformation

log.tr\_grades = log(scores$Grades)

#Square Transformation

sqrt.tr\_grades = sqrt(scores$Grades)

#Inverse Square root Transformation

inv.sqrt.tr\_grades = 1/sqrt(scores$Grades)

# Creating histograms for the transformed data for comparison

hist(scores$Grades, main = "Raw Data Fulltime Undergrad ")

hist(log.tr\_grades, main = "Log Transformtion")

hist(sqrt.tr\_grades, main = " Square-root Transformation")

hist(inv.sqrt.tr\_grades, main = "Inverse Square-root Transformation")

**Assignment 4**

Step 1 : Load the file

Rajcereal = read.csv('cereals.CSV',header = TRUE, sep = ',')

Step 2: find the location of missing values

Rajcereal[row.names(Rajcereal)[rowSums(is.na(Rajcereal))>0],colnames(Rajcereal)[colSums(is.na(Rajcereal))>0] ]

# I chose 58th row which has NA value for 2 columns. I chose 58th row which has NA value for Carbo.

Step 3: find the category of the missing value row

gocarbo = subset(Rajcereal, as.numeric(as.factor(Rajcereal$Type)) == 2)

Step 4: calculate the avg of the total value for that category

mean(gocarbo$Carbo, na.rm = TRUE)

Step 5: Populate the missing value based on its category avg.

Rajcereal$Carbo[is.na(Rajcereal$Carbo)] <- mean(gocarbo$Carbo, na.rm = TRUE)

**Part 5**

1. Base package provides the simplest graphs: easy to remember, provides

low level of analysis. plot(), hist()…

1. Lattice and Ggplot is more options to create higher level of analysis.

# install.packages('lattice') #if not installed already

require(lattice)

histogram(~mpg$hwy | mpg$year).

1. Boxplot

boxplot(Rajcereal$Protein)

or

require(ggplot2)

boxplot(college\_data$F.Undergrad,college\_data$P.Undergrad,college\_data$Outstate,college\_data$S.F.Ratio,names=c("F.Undergrad","P.Undergrad","Outstate","Student Faculty Ratio"))

1. Scatter plots

require(ggplot2)

xyplot(hwy ~ cty, mpg)

or

require(lattice)

xyplot(SingleFemaleHousehold~Metropolitan.Residence, data = smdetails)

**Assignment 5**

1. Load the data to cereal data in R. Report on the measures of centrality and spread for the calories variable of the data with at least two measures each.

mean(Rajcereal$Calories)

median(Rajcereal$Calories)

table(Rajcereal$Calories)

range(Rajcereal$Calories)

diff(range(Rajcereal$Calories))

sd(Rajcereal$Calories)

1. Plot histograms for the calories and protein distributions of the items. What difference(s) do you see, briefly explain how we should interpret the difference ?

require(lattice)

par(mfrow = c(2, 1))

hist(C$Calories, col = "skyblue")

hist(C$Protein, col = "wheat")

While calories is left skewed, protein is right skewed. That means while items in the dataset are (realtively) higher in calories, they tend to be low in protein. While calories is more or less normally distributed, protein is right skewed. That means while items in the dataset are (realtively) higher in calories, they tend to be low in protein. This also shows us the protein content in cereals does not necessarily increase proportionately with calorie increase

1. Create a diagram to spot outliers for the protein distribution of the items (extremely high or low in protein content). What type of items do you think they are ? Why ?

boxplot(Rajcereal$Protein)

mean(Rajcereal$Protein)

q1 = quantile(Rajcereal$Protein, 0.25)

q3 = quantile(Rajcereal$Protein, 0.75)

q1

q3

outlierdeterminant = (IQR(Rajcereal$Protein))\*1.5

outlierdeterminant

# An outlier is defined as a data point that is located outside the fences ("whiskers") of the boxplot (e.g: outside # 1.5 times the interquartile range above the upper quartile and below the lower quartile). q1 and q3 are 2 and 3. #Here the outlier deteminant is 1.5 (i.e, 3 minus 2 multiplied by 1.5). Therefore, outliers would be below o.5 (i.e, #2-1.5) and above 4.5 (i.e, 3+1.5). So the ouliers are 5 and 6 which are above 4.5.

Prof sol:

require(lattice)

bp <- boxplot(C$Protein, col = "beige", horizontal = T,main = "Protein Distribution",xlab = "Protein")

bp$stats[,1]

subset(C,C$Protein>4)

They are probably ‘power’ sandwiches with meat in them since they are high in protein(protein > 4). These items has high protein content compared to other in distribution where the overall average stands around 2.5 and median of 3. These might be for customers working on weight loss or body building.

1. We are really interested amount of protein in items per calories. To investigate this, you need to calculate protein per calories by dividing protein amount byt the calorie (protein/calories). What is the average protein per calories amount for an item.

proteinpercalorie = round(Rajcereal$Protein/Rajcereal$Calories, 3)

proteinpercalorie

mean(proteinpercalorie)