Introduction to
R software

Data frame and Graphical representation

Kundan Singh
Research Scholar
Department of Mathematics
Indian Institute of Technology, Patna

28 August 2022

DATA FRAME

Data frames contain complete data sets that are mostly created with other programs (spreadsheet-files, software SPSS-files, Excel-files etc.).

Variables in a data frame may be numeric (numbers) or categorical (characters or factors).

Data Frame

Example:

Package “MASS” describes functions and datasets to support Venables and Ripley, “Modern Applied Statistics with S” (4th edition 2002)

An example data frame painters is available in the library.
MASS (here only an excerpt of a data set):

```
> library(MASS)
> painters
```

Here, the names of the painters serve as row identifications, i.e., every row is assigned to the name of the corresponding painter.

However, these names are not variables of the data set. Here a subset of these names:

```
> rownames(painters)
> colnames(painters)
[1] "Composition" "Drawing" "Colour" "Expression" "School"
```

Data Frame

• The data set contains four numerical variables (Composition, Drawing, Colour and Expression), as well as one factor variable (School).

```
> is.numeric(painters$School)
[1] FALSE
```

Notice how we extract a variable (column) from data set.

```
> is.numeric(painters$Drawing)
[1] TRUE
> is.factor(painters$School)
[1] TRUE
```

Data Frame

Using the summary function, we can get a quick overview of descriptive measures for each variable:

```
> summary(painters)
```

The categories F and H, each present 4 times in the variable "School", are summed under the category Other as 8 with the corresponding frequency. i.e., only the 6 most frequent values are displayed.

Test if we are dealing with a data frame:

```
> is.data.frame(painters)
[1] TRUE
```

Data Frame

• Creating Data Frames Use the data.frame function to create a data frame by adding column vectors to the data frame.

Example

```
> x <- 1:16      # Vector
> y <- matrix(x, nrow=4, ncol=4)     # 4 X 4 matrix
> z <- letters[1:16]     # lowercase alphabets
> datafr <- data.frame(x, y, z)
> datafr
```

Data Frame

• Structure of the data:

Display information about the structure of the data frame (str). The result of str gives the dimension as well as the name and type of each variable.

```
> str(painters)
'data.frame' : 54 obs. of 5 variables:
$ Composition: int 10 15 8 12 0 15 8 15 4 17 ...
$ Drawing : int 8 16 13 16 15 16 17 16 12 18 ...
$ Colour : int 16 4 16 9 8 4 4 7 10 12 ...
$ Expression : int 3 14 7 8 0 14 8 6 4 18 ...
$ School : Factor w/ 8 levels "A","B","C","D",...: 1 1 1 1 1 1 1 1 1 1 ...
```

int means integer.

Data Frame

- Extract a variable from data frame using $

Example: Suppose we want to extract information on variable School from the data set painters.

```
> painters$School
[1] A A A A A A A A A A B B B B B B C C C C C C D D D D D
[28] D D D D D E E E E E E E F F F F G G G G G G G H H H H
Levels: A B C D E F G H
```

```
> summary(painters$School)    # Character variable
> summary(Composition)    # Numeric variable
```

Data Frame

• Subsets of a data frame can be obtained with subset() or with the second equivalent command:

> subset(painters, School=='F')

• Subsets of a data frame can be obtained with subset() or with the second equivalent command:

> subset(painters, Composition <= 6)

• Uninteresting columns can be eliminated.

> subset(painters, School=="F", select=c(-3,-5))

The third and the fifth column (Colour and School) are not shown.

Data Frame

• The command split partitions the data set by values of a specific variable. This should preferably be a factor variable.

Example: Following command splits painters with respect to School (A,B,C,... categories)

```
> splitted <- split(painters, painters$School)
```

• The objects splitted$A to splitted$H are themselves data frames:

```
> is.data.frame(splitted$A)
[1] TRUE
```

Data Handling

Setting up directories

- We can change the current working directory as follows:

```
> setwd("<location of the dataset>")
```

Example:

```
> setwd("F:/Simulation lab class/")
```

or

```
> setwd("F:\\Simulation lab class \\")
```

- The following command returns the current working directory:

```
> getwd()
[1] "F:/Simulation lab class/"
```

Importing Data Files

Suppose we have some data on our computer and we want to import it in R.

Different formats of files can be read in R

- comma-separated values (CSV) data file,
- table file (TXT),
- Spreadsheet (e.g., MS Excel) file,
- files from other software like SPSS, Minitab etc.

Importing Data Files

Comma-separated values (CSV) files
First set the working directory where the CSV file is located.

```
setwd("<location of your dataset>")
```

```
> setwd("F:/Simulation lab class/")
```

To read a CSV file
Syntax: `read.csv("filename.csv")`

Example:

```
> setwd("F:/Simulation lab class")
> data <- read.csv("Table_1.csv")
> data
```

or

```
> data <- read.csv("F:/Simulation lab class/Table_1.csv")
```

Importing Data Files

• If the data file does not have headers in the first row, then use

```
data <- read.csv("datafile.csv", header=FALSE)
```

• We can also rename the header names manually:

```
> names(data) <- c("Column1","Column2","Column3",...)
```

Reading Tabular Data Files

We want to read a text file that contains a table of data.
read.table function is used and it returns a data frame.

```
read.table("FileName")
```

Example:

```
> data <- read.table("Table_2.txt")
> data
```

Graphical tools:

Graphical tools- various type of plots

- 2D & 3D plots,
- scatter diagram
- Pie diagram
- Histogram
- Bar plot
- Box plot

Graphics and Plots

Graphical tools:

In R, Such graphics can be easily created and saved in various formats.

- Bar plot
- Pie chart
- Box plot
- Grouped box plot
- Scatter plot
- Coplots
- Histogram
- Normal QQ plot ...

Bar Plots:

Visualize the relative or absolute frequencies of observed values of a variable.

It consists of one bar for each category.

The height of each bar is determined by either the absolute frequency or the relative frequency of the respective category and is shown on the y-axis.

barplot(x, width = 1, space = NULL,...)

```
> barplot(table(x))
> barplot(table(x)/length(x))
```

Bar plots:

> help("barplot")

barplot(height, width = 1, space = NULL, names.arg = NULL, legend.text = NULL, beside = FALSE, horiz = FALSE, density = NULL, angle = 45, col = NULL, border = par("fg"), main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, xpd = TRUE, log = "", axes = TRUE, axisnames = TRUE, cex.axis = par("cex.axis"), cex.names = par("cex.axis"), inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0, add = FALSE, args.legend = NULL, ...)

Bar plots: Example

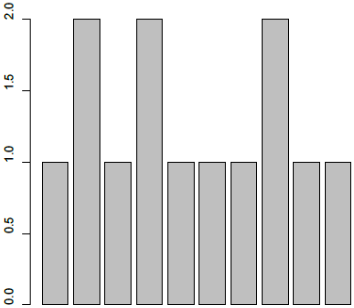
Code the 10 persons by using, say 1 for male (M) and 2 for female (F).

M,	F,	M,	F,	M,	M,	M,	F,	M,	M
1,	2,	1,	2,	1,	1,	1,	2,	1,	1

```
> gender <- c(1, 2, 1, 2, 1, 1, 1, 2, 1, 1)
> gender
[1] 1 2 1 2 1 1 1 2 1 1
```

Bar plots: Example

> barplot(gender)



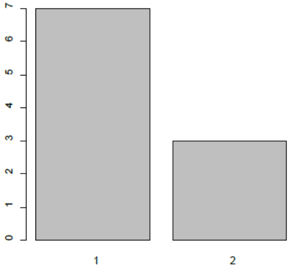
Bar plots: Example

> table(gender)

gender

1	2
7	3

> barplot(table(gender))



Bar plots: Example

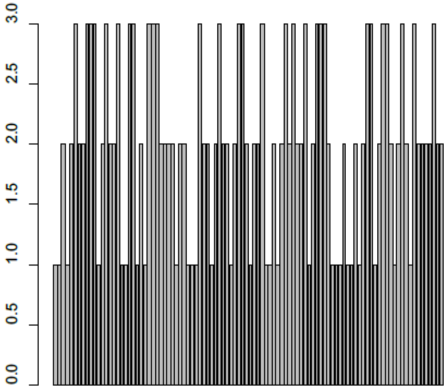
Consider data from Pizza. Take first 100 values from Direction and code Directions as

◇ East: 1
◇ West: 2
◇ Centre: 3

```
direction <- c(1, 1, 2, 1, 2, 3, 2, 2, 3, 3, 3, 1, 2, 3, 2, 2, 3, 1, 1, 3, 3, 1, 2,
1, 3, 3, 3, 2, 2, 2, 2, 1, 2, 2, 1, 1, 1, 3, 2, 2, 1, 2, 3, 2, 2, 1, 2, 3, 3, 2, 1,
2, 2, 3, 1, 1, 2, 1, 2, 3, 2, 3, 2, 2, 3, 1, 2, 3, 3, 3, 2, 1, 1, 1, 2, 1, 1, 2, 1,
2, 3, 3, 1, 2, 3, 3, 2, 1, 2, 3, 2, 1, 3, 2, 2, 2, 2, 3, 2, 2)
```

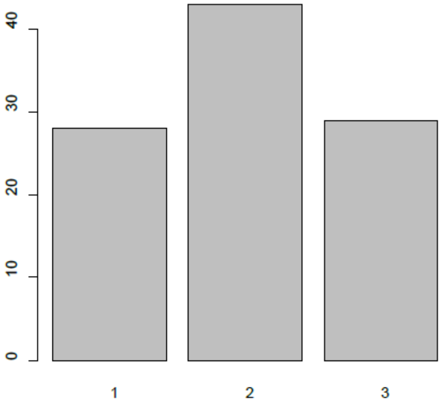
Bar plots: Example

> barplot(direction)



Bar plots: Example

```
> table(direction)
> barplot(table(direction))
```



Pie diagram:

Pie charts visualize the absolute and relative frequencies.

A pie chart is a circle partitioned into segments where each of the segments represents a category.

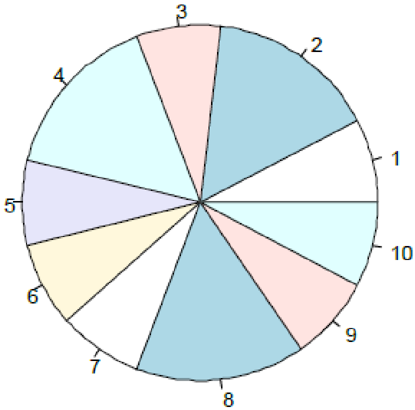
The size of each segment depends upon the relative frequency and is determined by the angle (frequency X 3600).

pie(x, labels = names(x), ...)

Pie diagram:

Example:

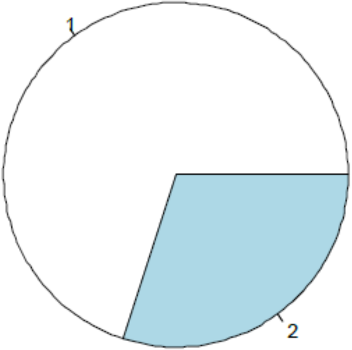
$>$ pie(gender)



Pie diagram:

Example:

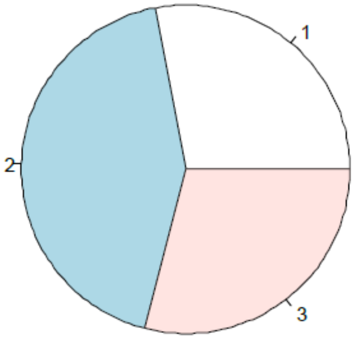
> pie(table(gender))



Pie diagram:

Example:

> pie(table(direction))



Histogram:

Histogram is based on the idea to categorize the data into different groups and plot the bars for each category with height.

The area of the bars (= height X width) is proportional to the relative frequency.

So the widths of the bars need not necessarily to be the same

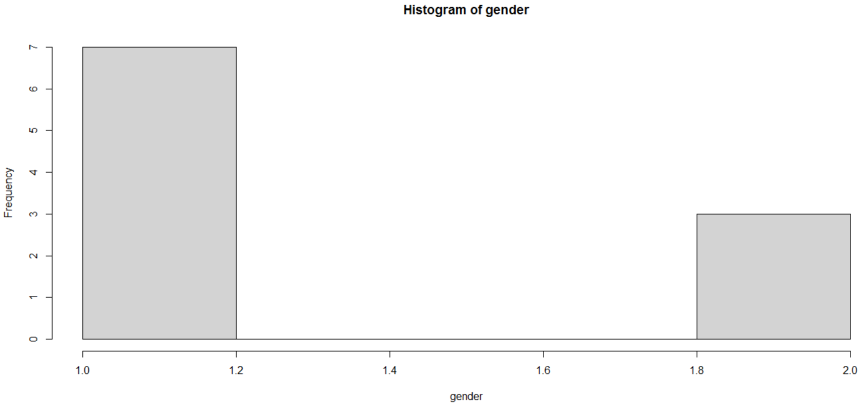
```
hist(x)   # show absolute frequencies
hist(x, freq=F)   # show relative frequencies
```

See help("hist") for more details

Histogram:

Example:

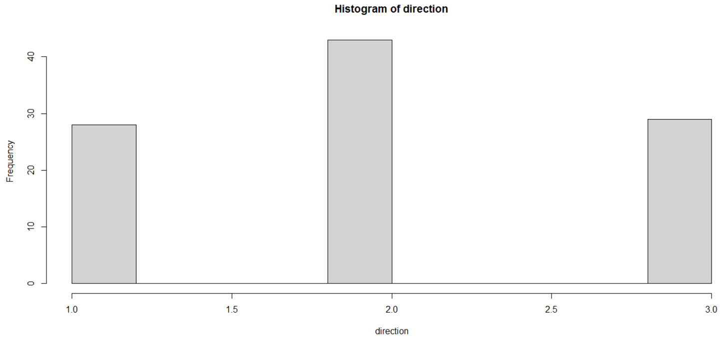
> hist(gender)



Histogram:

Example:

> hist(direction)



Boxplots, Skewness and Kurtosis

Summary of observations

In R, quartiles, minimum and maximum values can be easily obtained by the summary command

summary(x)   x: data vector

It gives information on

◇ minimum,
◇ maximum
◇ first quartile
◇ second quartile (median) and
◇ third quartile.

Summary of observations

Example

```
> marks <- c(68, 82, 63, 86, 34, 96, 41, 89, 29, 51, 75, 77, 56, 59, 42)
> summary(marks)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
29.0	46.5	63.0	63.2	79.5	96.0

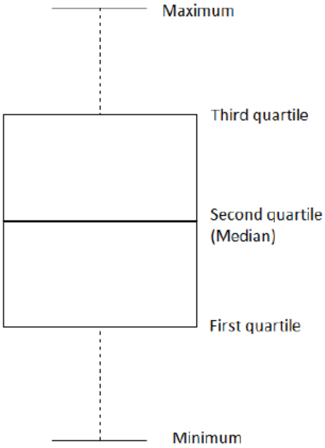
```
> marks1 <- c(628, 812, 613, 186, 34, 986, 41, 89, 29, 51, 795, 77, 56,
509, 420)
> summary(marks1)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
29.0	53.5	186.0	355.1	620.5	986.0

Boxplot

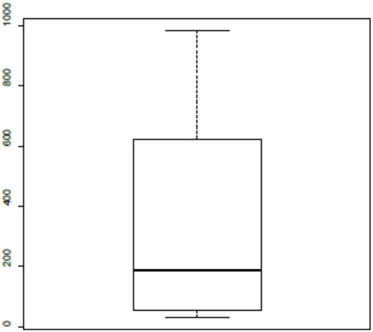
Box plot is a graph which summarizes the distribution of a variable by using its median, quartiles, minimum and maximum values.

boxplot() draws a box plot.

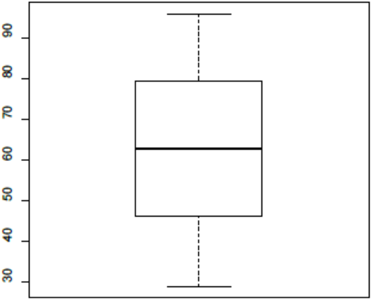


Example :

> boxplot(marks)
> boxplot(marks1)



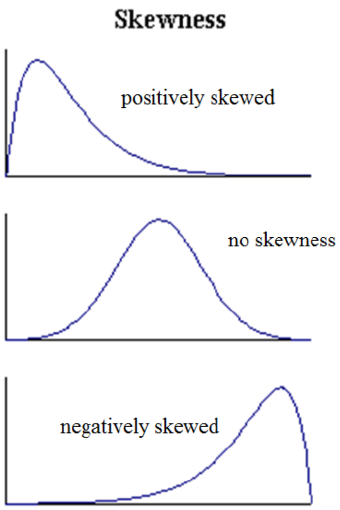
Boxplot(marks1)



Boxplot(marks)

Descriptive statistics:

- Structure and shape of data tendency (symmetricity, skewness, kurtosis etc.)
- Relationship study (correlation coefficient, rank correlation, corralation ratio, regression etc.)


Skewness
positively skewed
no skewness
negatively skewed
Kurtosis
platykurtic
mesokurtic
leptokurtic

Skewness :

Measures the shift of the hump of frequency curve.

Coefficient of skewness based on values $x_1,\ x_2, ...,\ x_n$.

$$\gamma_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}}$$

Mean

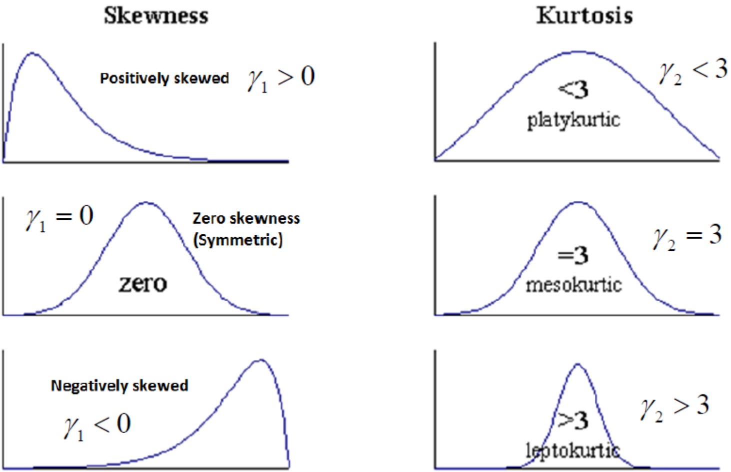
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Kurtosis :

Measures the peakedness of the frequency curve.

Coefficient of kurtosis based on values $x_1,\ x_2, ...,\ x_n$.

$$\gamma_2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2},\ -3 < \gamma_2 < 3$$



Skewness
Positively skewed $\gamma_1 > 0$
$\gamma_1 = 0$
Zero skewness
(Symmetric)
zero
Negatively skewed
$\gamma_1 < 0$
Kurtosis
$\gamma_2 < 3$
<3
platykurtic
$\gamma_2 = 3$
=3
mesokurtic
$\gamma_2 > 3$
>3
leptokurtic

Skewness and kurtosis :

First we need to install a package 'moments'

```
> install.packages("moments")
> library(moments)
```

skewness () : computes coefficient of skewness

kurtosis () : computes coefficient of kurtosis

Example:

```
> marks <- c(68, 82, 63, 86, 34, 96, 41, 89, 29, 51, 75, 77, 56, 59, 42)

> skewness(marks)
[1] -0.09869395

> kurtosis(marks)
[1] 1.830791
```

Bivariate and Three Dimensional Plots

Bivariate plots:

Provide first hand visual information about the nature and degree of relationship between two variables.

Relationship can be linear or nonlinear.

Scatter plot:

Plot command:
x, y: Two data vectors
plot(x, y)
plot(x, y, type)

type	
"p" for points	"l" for lines
"b" for both	"c" for the lines part alone of "b"
"o" for both 'overplotted'	"s" for stair steps.
"h" for 'histogram' like (or 'high-density') vertical lines	

Bivariate plots:

Scatter plot:

Plot command:
x, y: Two data vectors
plot(x, y)
plot(x, y, type) Get more details: help("type")

Other options:

main	an overall title for the plot.
suba	sub title for the plot.
xlaba	title for the x axis.
ylaba	title for the y axis.
aspthe	y/x aspect ratio.

Example:

Daily water demand in a city depends upon weather temperature.

We know from experience that water consumption increases as weather temperature increases.

Data on 27 days is collected as follows:

Daily water demand (in million litres)

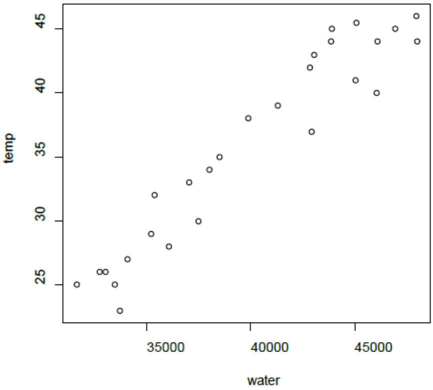
```
water <- c(33710,31666,33495,32758,34067,36069, 37497,33044,35216,
35383,37066,38037,38495, 39895,41311,42849,43038,43873,43923, 45078,
46935,47951,46085,48003,45050,42924,46061)
```

Temperature (in centigrade)

```
temp <- c(23,25,25,26,27,28,30,26,29,32,33,34, 35,38,39,42,43,44,
45,45.5,45,46,44,44,41,37,40)
```

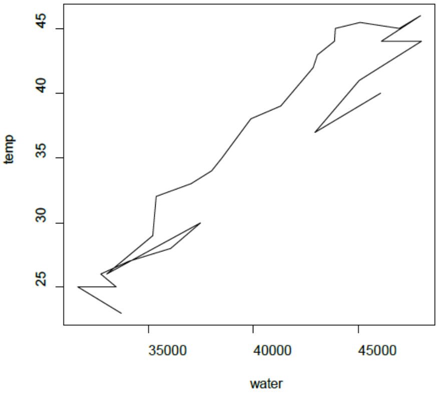
Scatter plot:

plot(water, temp)



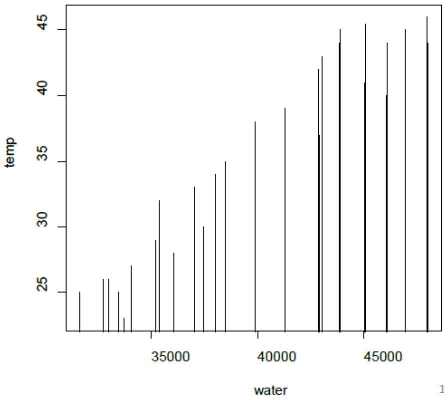
Scatter plot:

plot(water, temp, "l")
"l" for lines,



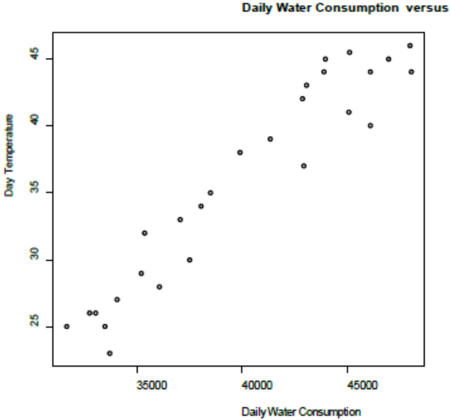
Scatter plot:

plot(water, temp, "h")
"h" for lines,



Scatter plot:

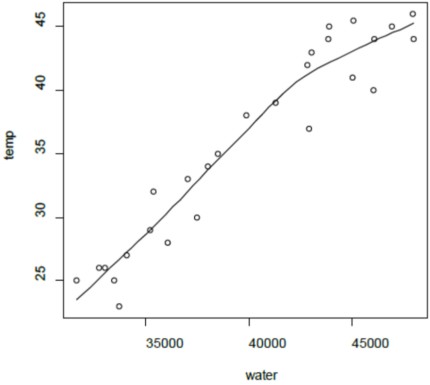
```
> plot(water, temp, xlab=" Daily Water Consumption ", ylab=" Day Temperature ", main=" Daily Water Consumption versus Day Temperature")
```



Smooth Scatter plot:

scatter.smooth(x,y) provides scatter plot with smooth curve

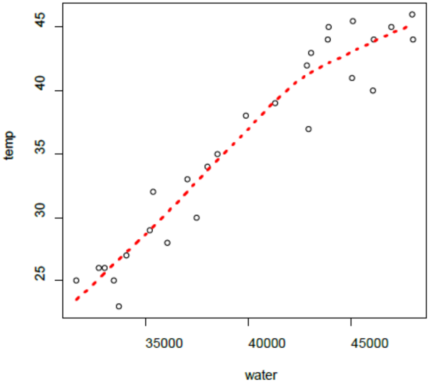
Example: scatter.smooth(water,temp)



Smooth Scatter plot:

Example:

```
> scatter.smooth(water, temp, lpars = list(col = "red", lwd = 3, lty = 3))
```



Remove Objects from Memory in R Programming – rm() Function

rm() function in R Language is used to delete objects from the memory. It can be used with ls() function to delete all objects. remove() function is also similar to rm() function.

```
rm(list = ls(all=TRUE))
```

R Plot Function

Usage

plot(x, y, . . .)

Example:
x <- seq(-pi,pi,0.1)
plot(x, sin(x))

Plot function

Adding Titles and Labeling Axes

```
main=" "   # title to our plot
xlab   # x-axis
ylab   # y-axis
```

Example :

```
plot(x, sin(x),main="Sine Function", ylab="sin(x)")
```

Plot function

Color and Plot Type

col define the color and black is the default color.

Similarly, We can change the plot type with the argument type.

"p" - points
"l" - lines
"b" - both points and lines
"c" - empty points joined by lines
"o" - overplotted points and lines
"s" and "S" - stair steps
"h" - histogram-like vertical lines
"n" - does not produce any points or lines

Plot function

legend() function

Calling plot() multiple times will have the effect of plotting the current graph on the same window replacing the previous one.

However, sometimes we wish to overlay the plots in order to compare the results.

This is made possible with the functions lines() and points() to add lines and points respectively, to the existing plot.

legend() function

Example

```
plot(x, sin(x),
main="Overlaying Graphs",
ylab="",
type="l",
col="blue")
lines(x,cos(x), col="red")
legend("topleft",c("sin(x)","cos(x)"),
fill=c("blue","red")
)
```

Plot function

Example

```
j < − 1:20
k < − j

par(mfrow = c(1, 3))

plot(j, k, type = "l", main = "type = 'l'")
plot(j, k, type = "s", main = "type = 's'")
plot(j, k, type = "h", main = "type = 'h'")
```

Plot function

You can also modify the text colors with the col.main, col.sub, col.lab and col.axis functions and even change the box color with the fg argument.

Example

```
j <- 1:20
k <- j
plot(j, k, main = "Title", sub = "Subtitle",
pch = 16,  # modify the symbol of the points
cex = 3,  # change the symbols size
lwd = 1,  # line width of the symbols
col = "red", # Symbol color
col.main = "green", # Title color
col.sub = "blue", # Subtitle color
col.lab = "sienna2", # X and Y-axis labels color
col.axis = "maroon4", # Tick labels color
fg = "orange") # Box color
```