Functions and Distributions in R

By

Dina Dawoud

and

Nagham Mohamma

Elementary Built-in Functions

(1)The mean of an observation variable is a numerical measure of the central location of the data values. It is the sum of its data values divided by data count.

Hence, for a data sample of size n, its sample mean is defined as follows:

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

Similarly, for a data population of size N, the population mean is:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Example

Find the mean eruption duration in the data set faithful.

Solution

We apply the mean function to compute the mean value of eruptions.

```
duration <-  faithful$eruptions     # the eruption durations
mean(duration)                      # apply the mean function

[1] 3.4878
```

(2)The sample median

```
 x <- c(14, 17, 15, 22, 16, 18, 12, 12, 10, 17)

 median(x)   # computes the median or 50th percentile of the data
```

(3) `var(x)`   # computes the variance of the data in x

(4)   `summary(x)`   # computes several summary statistics on the data in x

Relational operators

To test relations to decide whether they are TRUE or FALSE.
R allows for equality and inequality relations to be tested in using the relational operators: <, >, ==, >=, <=, !=

Examples:

(1) Type

```
a <- c(3, 6, 9)
```

(2) To test which elements are greater than 4, type

```
a > 4
```

(3) To test which elements are exactly equal to 4, type

```
a==4

[1] FALSE FALSE FALSE
```

(4) To test which elements are greater than or equal to 4, type

```
a >=4

[1] FALSE  TRUE  TRUE
```

(5) To test which elements are not equal to 4, type

```
a != 4

TRUE TRUE TRUE
```

(6) To print the elements of which are greater than 4, type

```
a[a > 4]
```

```
[1] 6 9
```

Type

```
b <- c(4, 6, 8)
```

To test which elements of a greater and equal than the corresponding elements of b.

Type

```
a[a >= b]
```

```
[1]  6 9
```

(7) To test which elements of a are less than the corresponding elements of b, type

```
a < b
```

```
[1]  TRUE FALSE FALSE
```

(8) To print the elements of a that are less than the corresponding elements of b, type

```
a[a < b]
```

```
[1] 3
```

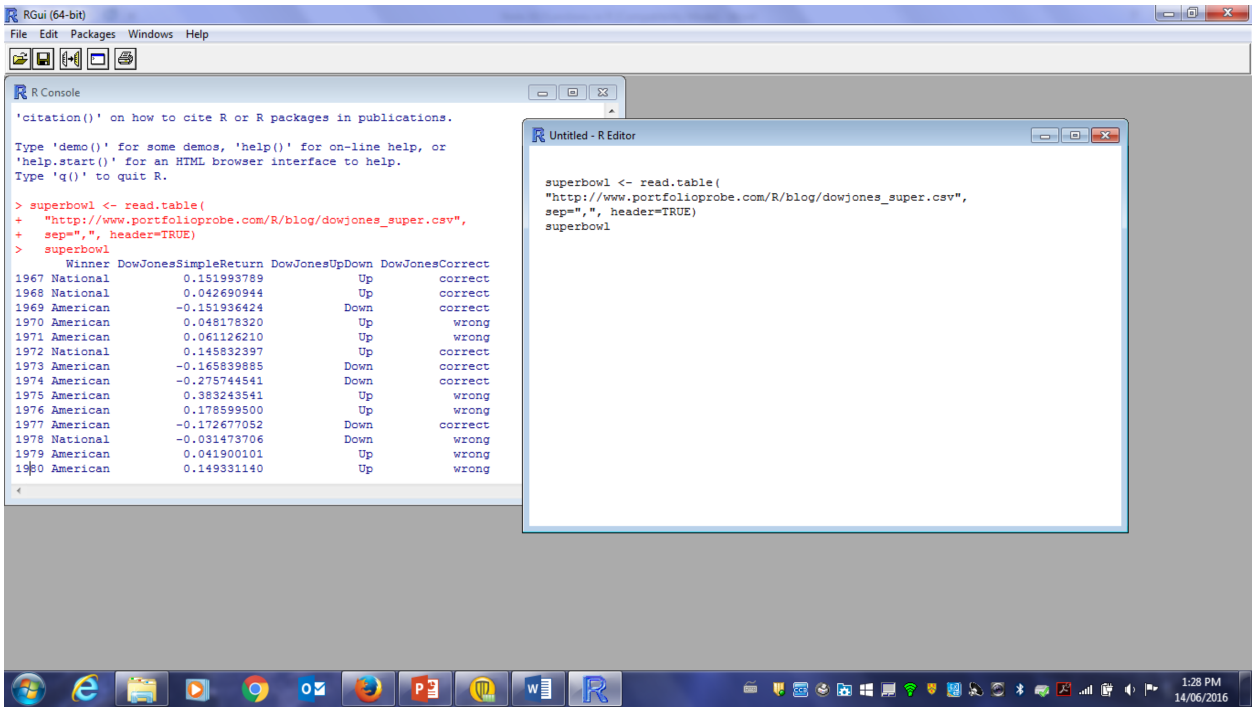
read.table function

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Example

```
superbowl <- read.table(
  "http://www.portfolioprobe.com/R/blog/dowjones_super.csv",
  sep=",", header=TRUE)
superbowl
```

The command above reads the data from the file and puts it into an object called superbowl.



```
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> superbowl <- read.table(
+   "http://www.portfolioprobe.com/R/blog/dowjones_super.csv",
+   sep=",", header=TRUE)
>   superbowl
       Winner DowJonesSimpleReturn DowJonesUpDown DowJonesCorrect
1967 National          0.151993789             Up         correct
1968 National          0.042690944             Up         correct
1969 American         -0.151936424           Down         correct
1970 American          0.048178320             Up           wrong
1971 American          0.061126210             Up           wrong
1972 National          0.145832397             Up         correct
1973 American         -0.165839885           Down         correct
1974 American         -0.275744541           Down         correct
1975 American          0.383243541             Up           wrong
1976 American          0.178599500             Up           wrong
1977 American         -0.172677052           Down         correct
1978 National         -0.031473706           Down           wrong
1979 American          0.041900101             Up           wrong
1980 American          0.149331140             Up           wrong
```

```
superbowl <- read.table(
"http://www.portfolioprobe.com/R/blog/dowjones_super.csv",
sep=",", header=TRUE)
superbowl
```

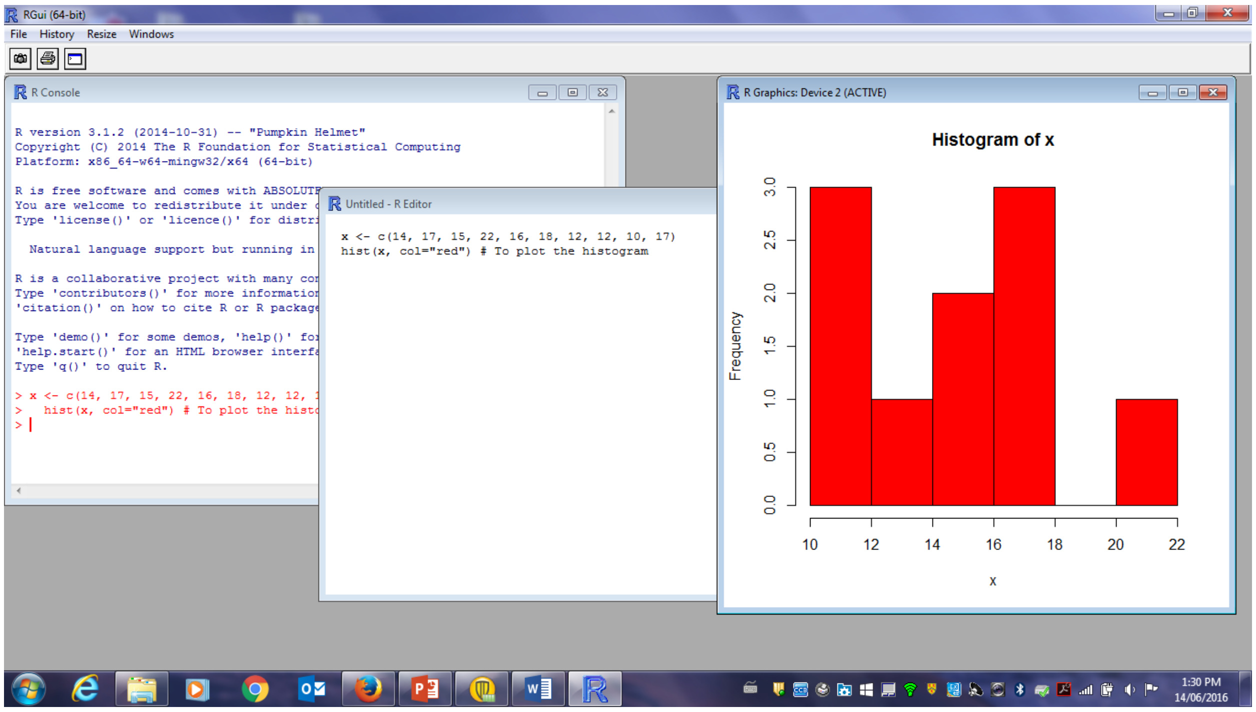
Our call to the read.table function has used three arguments:

- the location of the file — typically this would be a file on your file system.
- the sep argument says what character is used to separate items.
- the header argument says whether or not the first line in the file contains labels.
    - The result is a data frame.

Built-in graphics functions

Example (1)

```
 x <- c(14, 17, 15, 22, 16, 18, 12, 12, 10, 17)
hist(x) # To plot the histogram
```



```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTE
You are welcome to redistribute it under c
Type 'license()' or 'licence()' for distri

  Natural language support but running in

R is a collaborative project with many cor
Type 'contributors()' for more informatior
'citation()' on how to cite R or R package

Type 'demo()' for some demos, 'help()' for
'help.start()' for an HTML browser interfa
Type 'q()' to quit R.

> x <- c(14, 17, 15, 22, 16, 18, 12, 12, 1
>   hist(x, col="red") # To plot the histo
> |
```

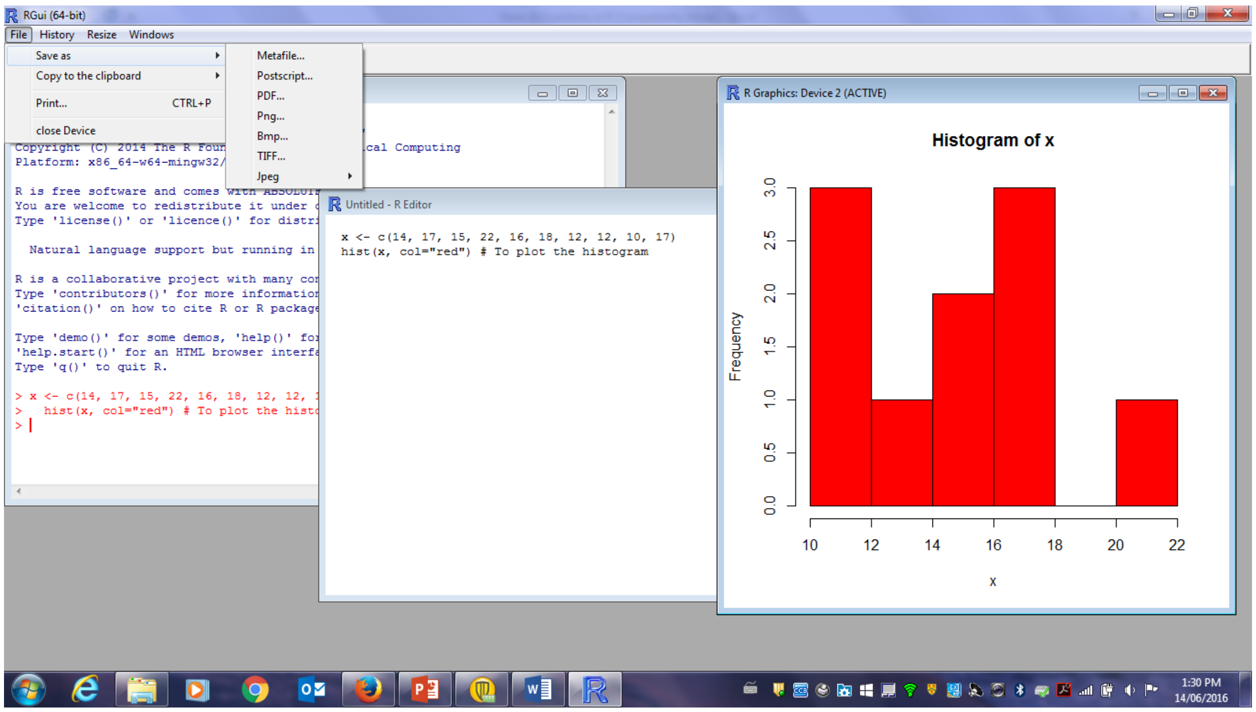
```
x <- c(14, 17, 15, 22, 16, 18, 12, 12, 10, 17)
hist(x, col="red") # To plot the histogram
```

Saving Plots in R

After you have finished your analysis in R, you may wish to save your data and/or graphs in a different format for use in other programs.

If you're actually sitting in front of a Windows or Mac computer, the graphical user interface makes it easy to save files. Under Windows, right click inside the graph window, and choose either "Save as metafile ..." or "Save as postscript ..." If using Word, make sure to save as a metafile, or pdf.

On a Mac, click on the graphics window to make sure it's the active one, then go to File -> Save in the menu bar, and choose a location to save the file. It will be saved as a pdf file, which you can double click to open in Preview, and then use the File -> Save As menu choice to convert to another format.



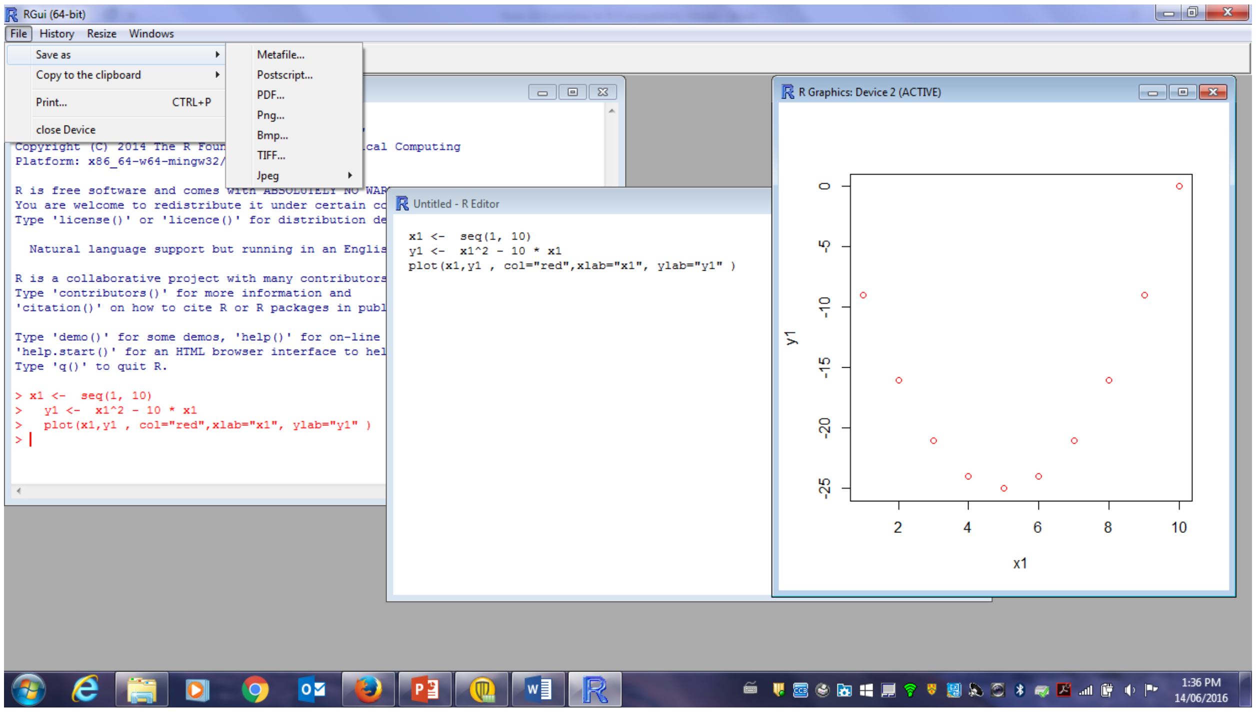
```
Copyright (C) 2014 The R Foun
Platform: x86_64-w64-mingw32/

R is free software and comes
You are welcome to redistribute it under
Type 'license()' or 'licence()' for distr

  Natural language support but running in

R is a collaborative project with many co
Type 'contributors()' for more informatio
'citation()' on how to cite R or R packag

Type 'demo()' for some demos, 'help()' fo
'help.start()' for an HTML browser interf
Type 'q()' to quit R.

> x <- c(14, 17, 15, 22, 16, 18, 12, 12, 1
>   hist(x, col="red") # To plot the hist
> 
```

```
.cal Computing
```

```
x <- c(14, 17, 15, 22, 16, 18, 12, 12, 10, 17)
hist(x, col="red") # To plot the histogram
```

Example (2)

```
x1 <-  seq(1, 10)
y1 <-  x1^2 - 10 * x1
plot(x1,y1 , col="red",xlab="x1", ylab="y1" )
```

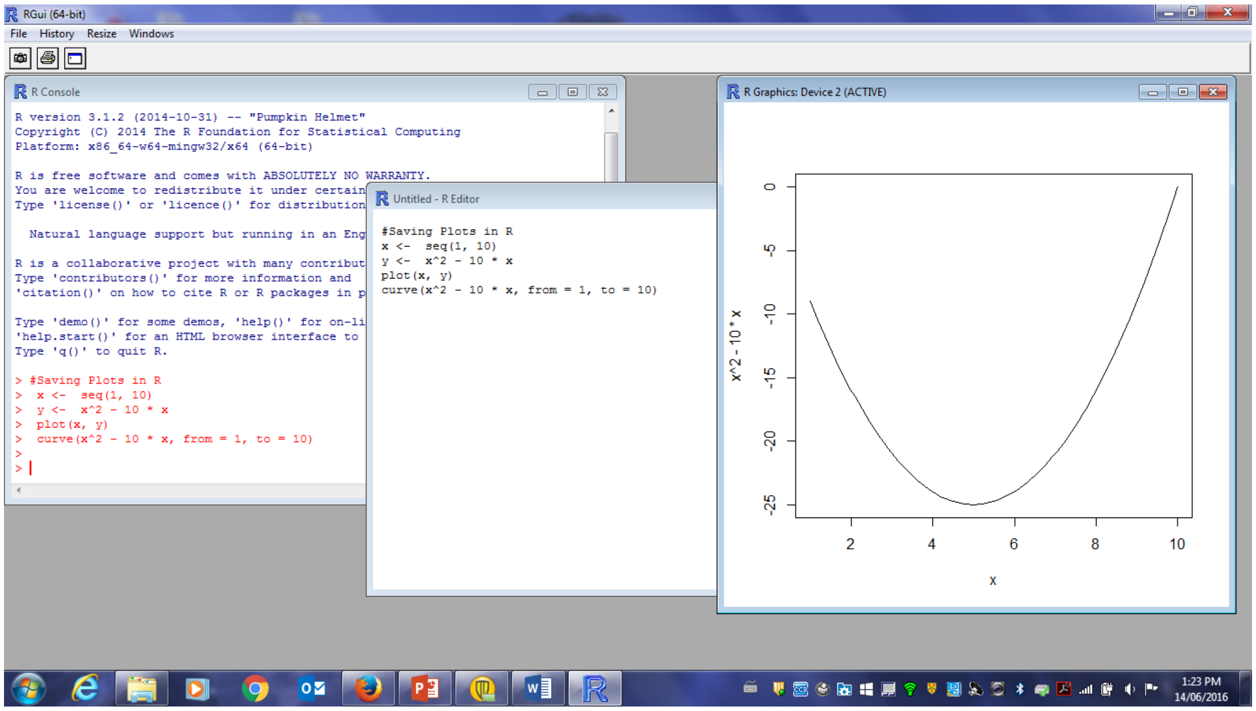


```
Copyright (C) 2014 The R Foun                    cal Computing
Platform: x86_64-w64-mingw32/

R is free software and comes with ABSOLUTELY NO WAR
You are welcome to redistribute it under certain co
Type 'license()' or 'licence()' for distribution de

  Natural language support but running in an Englis

R is a collaborative project with many contributors
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publ

Type 'demo()' for some demos, 'help()' for on-line
'help.start()' for an HTML browser interface to hel
Type 'q()' to quit R.

> x1 <-  seq(1, 10)
>   y1 <-  x1^2 - 10 * x1
>   plot(x1,y1 , col="red",xlab="x1", ylab="y1" )
> 
```

```
x1 <-  seq(1, 10)
y1 <-  x1^2 - 10 * x1
plot(x1,y1 , col="red",xlab="x1", ylab="y1" )
```

Example (3)

```
x <-  seq(1, 10)
y <-  x^2 - 10 * x
plot(x, y)
curve(x^2 - 10 * x, from = 1, to = 10)
```



```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain
Type 'license()' or 'licence()' for distribution

  Natural language support but running in an Eng

R is a collaborative project with many contribut
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in p

Type 'demo()' for some demos, 'help()' for on-li
'help.start()' for an HTML browser interface to
Type 'q()' to quit R.

> #Saving Plots in R
>  x <-  seq(1, 10)
>  y <-  x^2 - 10 * x
>  plot(x, y)
>  curve(x^2 - 10 * x, from = 1, to = 10)
>
> |
```

```
#Saving Plots in R
x <-  seq(1, 10)
y <-  x^2 - 10 * x
plot(x, y)
curve(x^2 - 10 * x, from = 1, to = 10)
```

Adding to plots

Several functions exist to add components to the plot region of existing graphs:

- points(x, y, ...)
- lines(x, y, ...) adds line segments
- text(x, y, labels, ...) adds text into the graph
- abline(a, b, ...) adds the line $y = a + bx$
- abline(h=y, ...) adds a horizontal line
- abline(v=x, ...) adds a vertical line
- par(mfrow = c(2,2)) Construct a 2x 2layout of graphs
- mar=c(side1, side2, side3, side4) sets the margins of the plot to the given numbers of lines of text on each side.

You can create a plot with the dataset superbowl :

```
plot(DowJonesSimpleReturn ~ Winner, data=superbowl)
```

That command makes a rather austere plot. You can make it prettier with some minor additions

```
plot(100 * DowJonesSimpleReturn ~ Winner,
   data=superbowl, col="powderblue", ylab="Return (%)")
```

```
boxplot(x) # Box plot
```

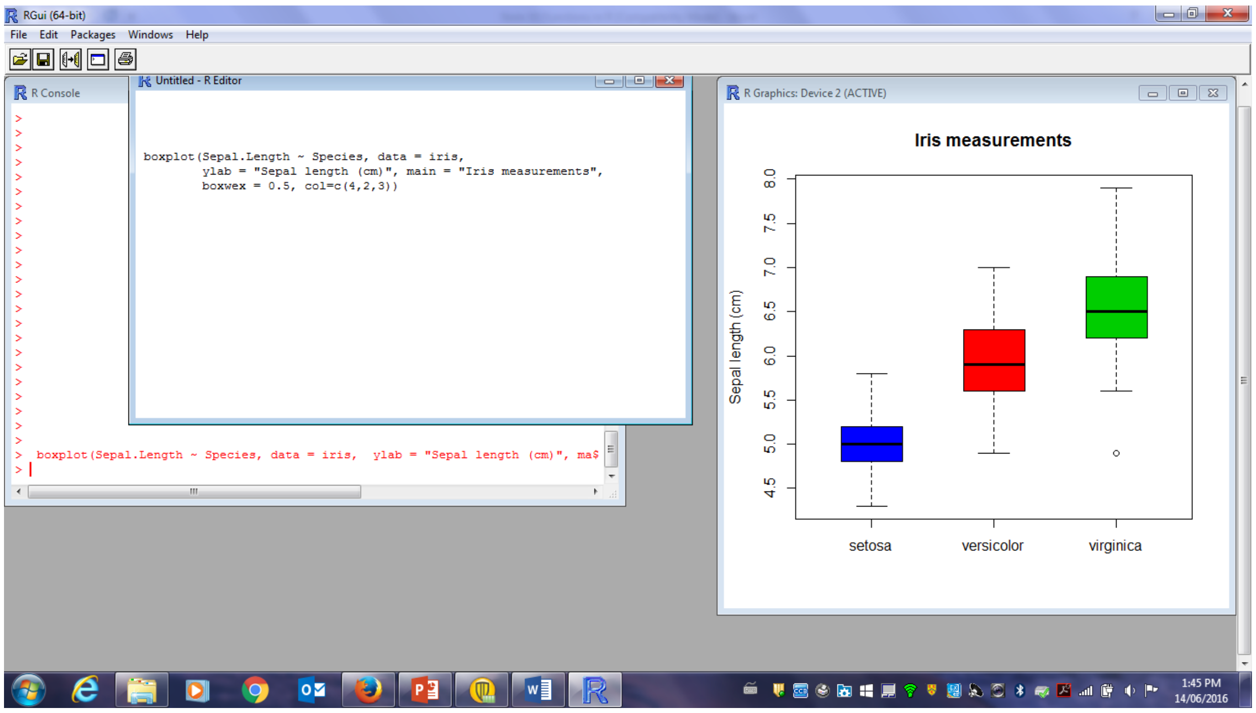
Note:
Boxplots are convenient for comparing distributions of data in two or more categories, with a number (say 10 or more) of numerical observations per category.

Example (3)

The iris dataset in R is a well-studied dataset of measurements of 50 flowers from each of three species of iris.

```
iris
head(iris)
?iris
```

```
boxplot(Sepal.Length ~ Species, data = iris,  ylab = "Sepal length (cm)", main = 
"Iris measurements",  boxwex = 0.5, col=c(4,2,3))
```



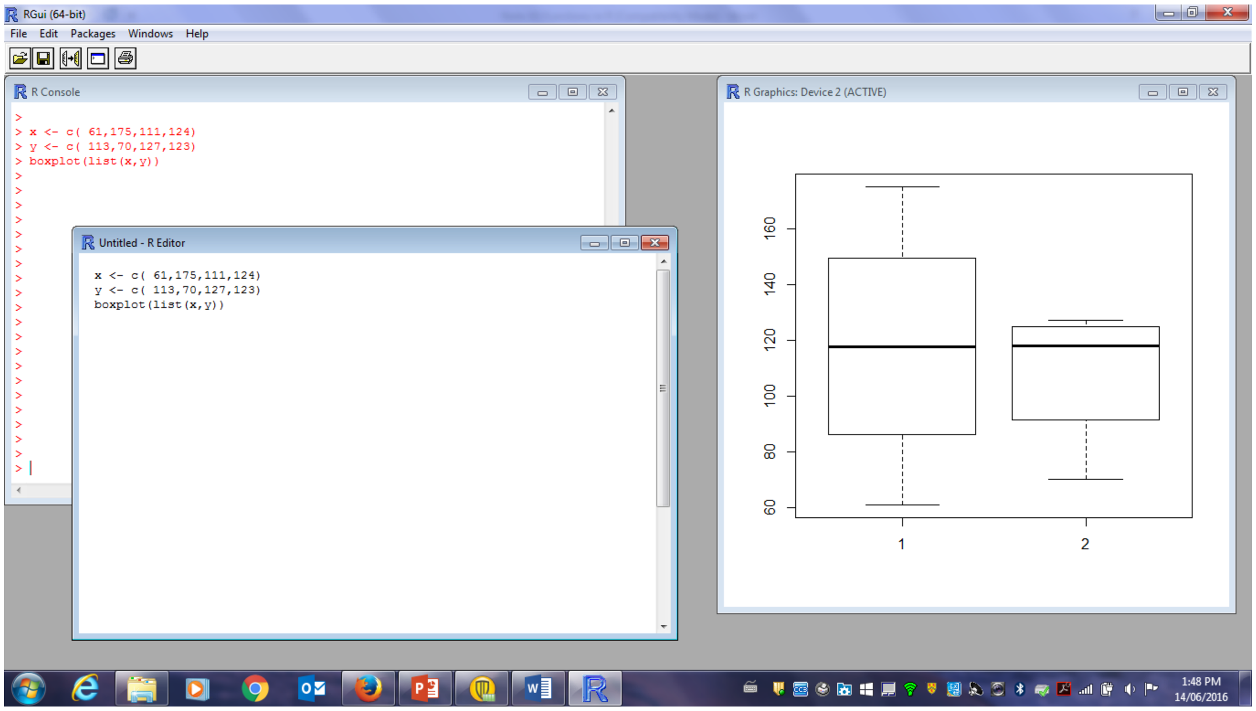
```
boxplot(Sepal.Length ~ Species, data = iris,
        ylab = "Sepal length (cm)", main = "Iris measurements",
        boxwex = 0.5, col=c(4,2,3))
```

```
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>  boxplot(Sepal.Length ~ Species, data = iris,  ylab = "Sepal length (cm)", ma$
> |
```

Example (4)

To compare between the boxplot for both x, y use

```
x <- c( 61,175,111,124)
y <- c( 113,70,127,123)
boxplot(list(x,y))
```



```
>
> x <- c( 61,175,111,124)
> y <- c( 113,70,127,123)
> boxplot(list(x,y))
```

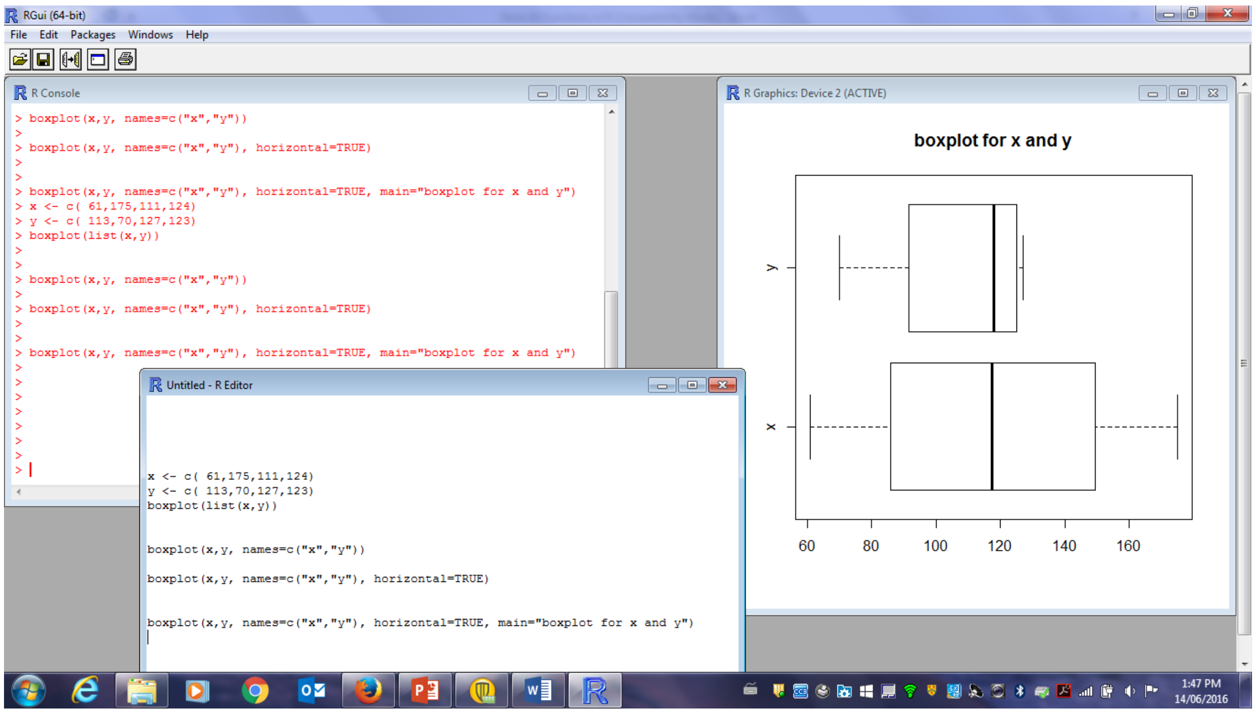
```
x <- c( 61,175,111,124)
y <- c( 113,70,127,123)
boxplot(list(x,y))
```

OR

boxplot(x,y, names=c("x","y"))

boxplot(x,y, names=c("x","y"), horizontal=TRUE)

boxplot(x,y, names=c("x","y"), horizontal=TRUE, main="boxplot for x and y")

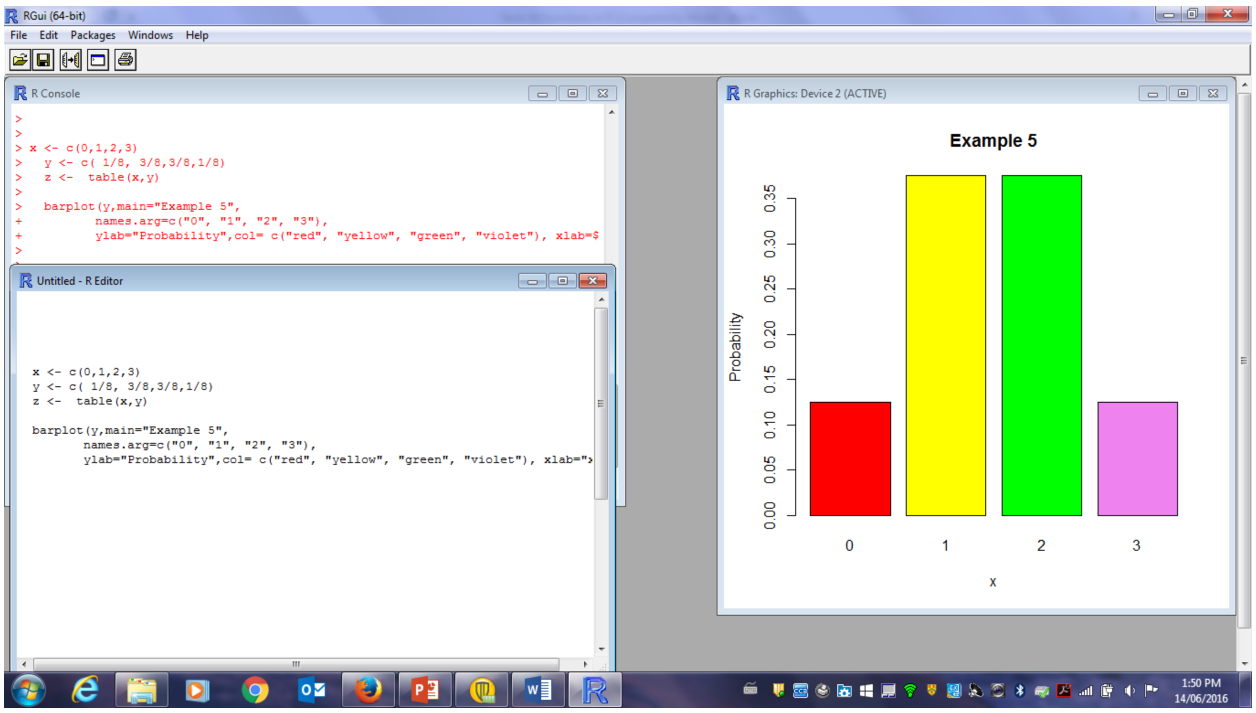


```
> boxplot(x,y, names=c("x","y"))
>
> boxplot(x,y, names=c("x","y"), horizontal=TRUE)
>
>
> boxplot(x,y, names=c("x","y"), horizontal=TRUE, main="boxplot for x and y")
> x <- c( 61,175,111,124)
> y <- c( 113,70,127,123)
> boxplot(list(x,y))
>
>
> boxplot(x,y, names=c("x","y"))
>
> boxplot(x,y, names=c("x","y"), horizontal=TRUE)
>
>
> boxplot(x,y, names=c("x","y"), horizontal=TRUE, main="boxplot for x and y")
>
>
>
>
>
>
>
>
>
```

```
x <- c( 61,175,111,124)
y <- c( 113,70,127,123)
boxplot(list(x,y))


boxplot(x,y, names=c("x","y"))

boxplot(x,y, names=c("x","y"), horizontal=TRUE)


boxplot(x,y, names=c("x","y"), horizontal=TRUE, main="boxplot for x and y")
```

Example (5) : Plot bargraph

```
x <- c(0,1,2,3)
y <- c( 1/8, 3/8,3/8,1/8)
z <-  table(x,y)

barplot(y,main="Example 2",
        names.arg=c("0", "1", "2", "3"),
        ylab="Probability",col= c("red", "yellow", "green", "violet"), xlab="x")
```



```
>
>
> x <- c(0,1,2,3)
>   y <- c( 1/8, 3/8,3/8,1/8)
>   z <-  table(x,y)
>
>   barplot(y,main="Example 5",
+          names.arg=c("0", "1", "2", "3"),
+          ylab="Probability",col= c("red", "yellow", "green", "violet"), xlab=$
>
```

```
x <- c(0,1,2,3)
y <- c( 1/8, 3/8,3/8,1/8)
z <-  table(x,y)

barplot(y,main="Example 5",
       names.arg=c("0", "1", "2", "3"),
       ylab="Probability",col= c("red", "yellow", "green", "violet"), xlab="x
```

Exercises

(a) Calculate the Mean, variance, and sd of

x: 14, 15, 18, 16, 13, 15, 16, 13, 10, 18, 16, 12, 15, 16, 16, 14, 12, 14, 18, 19

```
x <- c(14, 15, 18, 16, 13, 15, 16, 13, 10, 18, 16, 12, 15, 16, 16, 14, 12, 14, 18, 19)
mean(x) # To calculate the Mean
var(x)  # To calculate the variance
sd(x)
```

(b) To calculate the Mean with an outlier(Add an outlier to the varable X taking a numbers from 0:10 calculate the mean then add 100 to this data, and calculate the men and comment on the result

```
x <- c(0:10, 100)      # I add an outlier to the data
 x                     # To print x
 xm <- mean(x)         # Let the mean of x equal to xm
 xm                   # To print the value of xm
```

(c) Calculate the range of x

```
range(x) # will give you the min and the max of the data
Range <- max(x) - min(x) # The good way to calculate the range
Range
```

Discrete Random Variable

If the set of all possible values of a random variable X is countable, then X is discrete

Probability Functions

The probability that a discrete random variable *X* takes on a particular value *x*, that is, $P(X = x)$, is frequently denoted p(*x*). The function $p(x)$ is typically called the probability mass function, and its common abbreviation —the p.f.

Cumulative Distribution Function

The cumulative distribution function (cdf) for any random variable X is

F(x) is a nondecreasing function with

$$F(x) = P(X \leq x)$$

For a discrete random variable X, , is called the <u>cumulative distribution function</u> (cdf).

$$F(a) = \sum_{x_i \leq a} p(x_i)$$

Example (1)

# Discrete p.f and c.d.f.

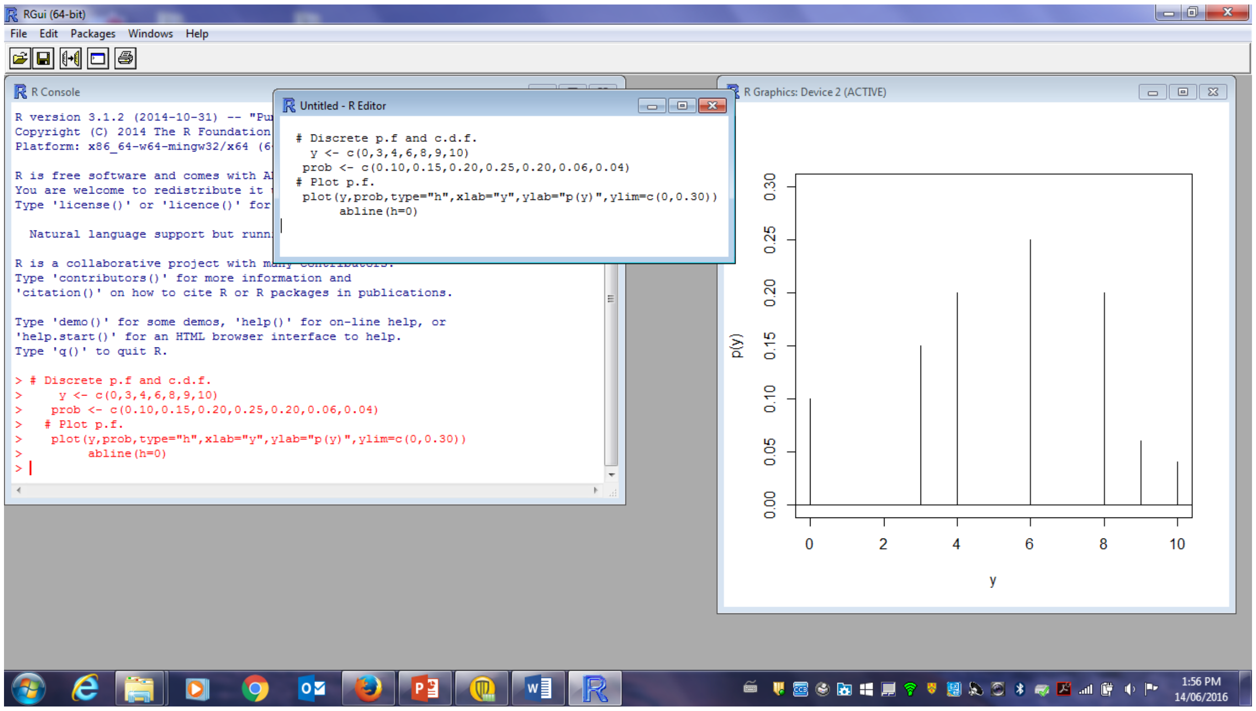
y <- c(0,3,4,6,8,9,10)

prob <- c(0.10,0.15,0.20,0.25,0.20,0.06,0.04)

# Plot p.f.

plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))

abline(h=0)



```
# Discrete p.f and c.d.f.
  y <- c(0,3,4,6,8,9,10)
 prob <- c(0.10,0.15,0.20,0.25,0.20,0.06,0.04)
# Plot p.f.
 plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
      abline(h=0)
```

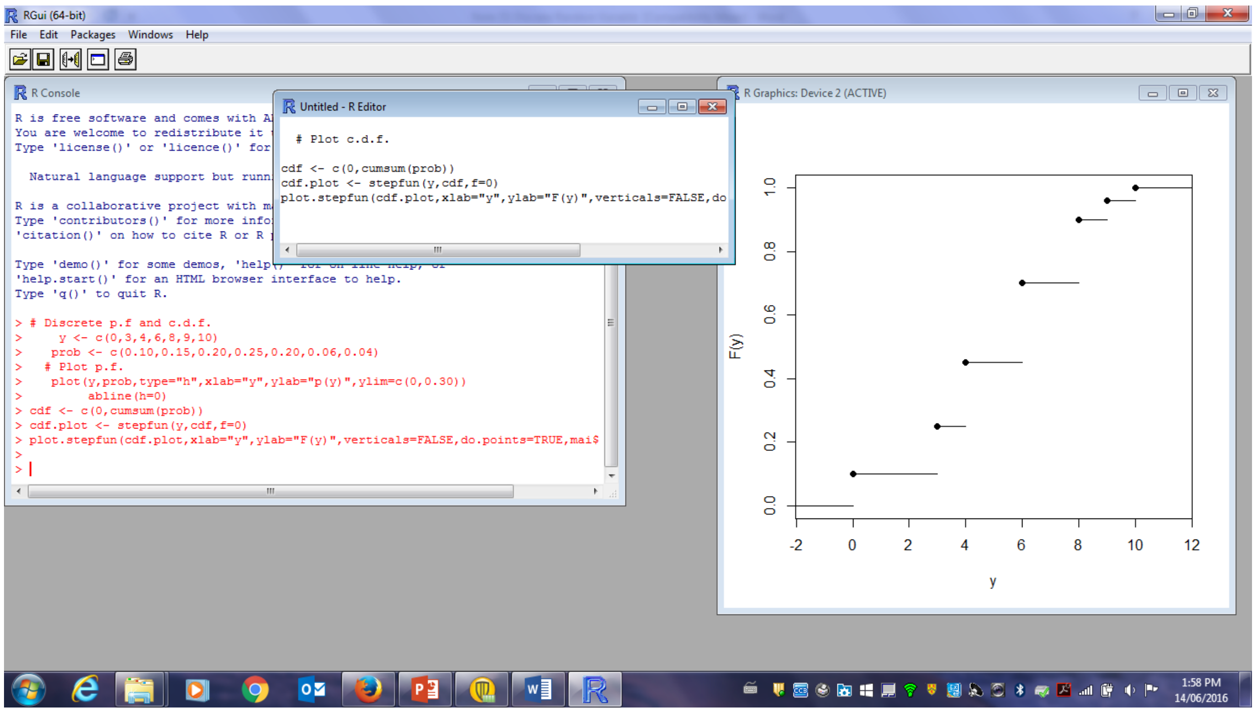
```
R version 3.1.2 (2014-10-31) -- "Pu
Copyright (C) 2014 The R Foundation
Platform: x86_64-w64-mingw32/x64 (6

R is free software and comes with A
You are welcome to redistribute it
Type 'license()' or 'licence()' for

  Natural language support but runn

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> # Discrete p.f and c.d.f.
>     y <- c(0,3,4,6,8,9,10)
>    prob <- c(0.10,0.15,0.20,0.25,0.20,0.06,0.04)
>   # Plot p.f.
>    plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
>         abline(h=0)
>
```

# Plot c.d.f.

cdf <- c(0,cumsum(prob))

cdf.plot <- stepfun(y,cdf,f=0)

plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main="",pch=16)



```
 # Plot c.d.f.

cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do
```

```
R is free software and comes with A
You are welcome to redistribute it
Type 'license()' or 'licence()' for

  Natural language support but runn

R is a collaborative project with m
Type 'contributors()' for more info
'citation()' on how to cite R or R

Type 'demo()' for some demos, 'help
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> # Discrete p.f and c.d.f.
>     y <- c(0,3,4,6,8,9,10)
>    prob <- c(0.10,0.15,0.20,0.25,0.20,0.06,0.04)
>   # Plot p.f.
>    plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
>         abline(h=0)
> cdf <- c(0,cumsum(prob))
> cdf.plot <- stepfun(y,cdf,f=0)
> plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,mai$
>
> 
```

# Example(2)

# Discrete p.f. and c.d.f.

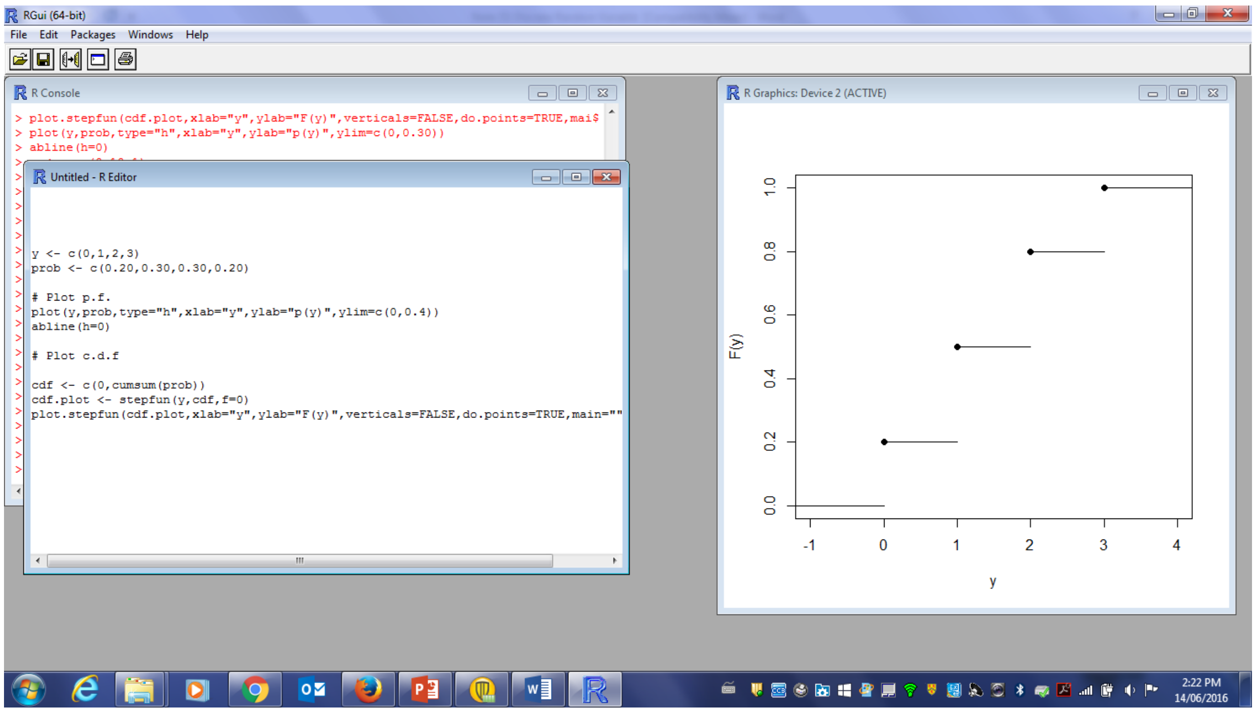
```
y <- c(0,1,2,3)
prob <- c(0.20,0.30,0.30,0.20)
```

# Plot p.f.

```
plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.4))
abline(h=0)
```

# Plot c.d.f

```
cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main="",pch=16)
```



```
> plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,mai$
> plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
> abline(h=0)
```

```
y <- c(0,1,2,3)
prob <- c(0.20,0.30,0.30,0.20)

# Plot p.f.
plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.4))
abline(h=0)

# Plot c.d.f

cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main=""
```

# Example (3)

# Binomial p.f and c.d.f.

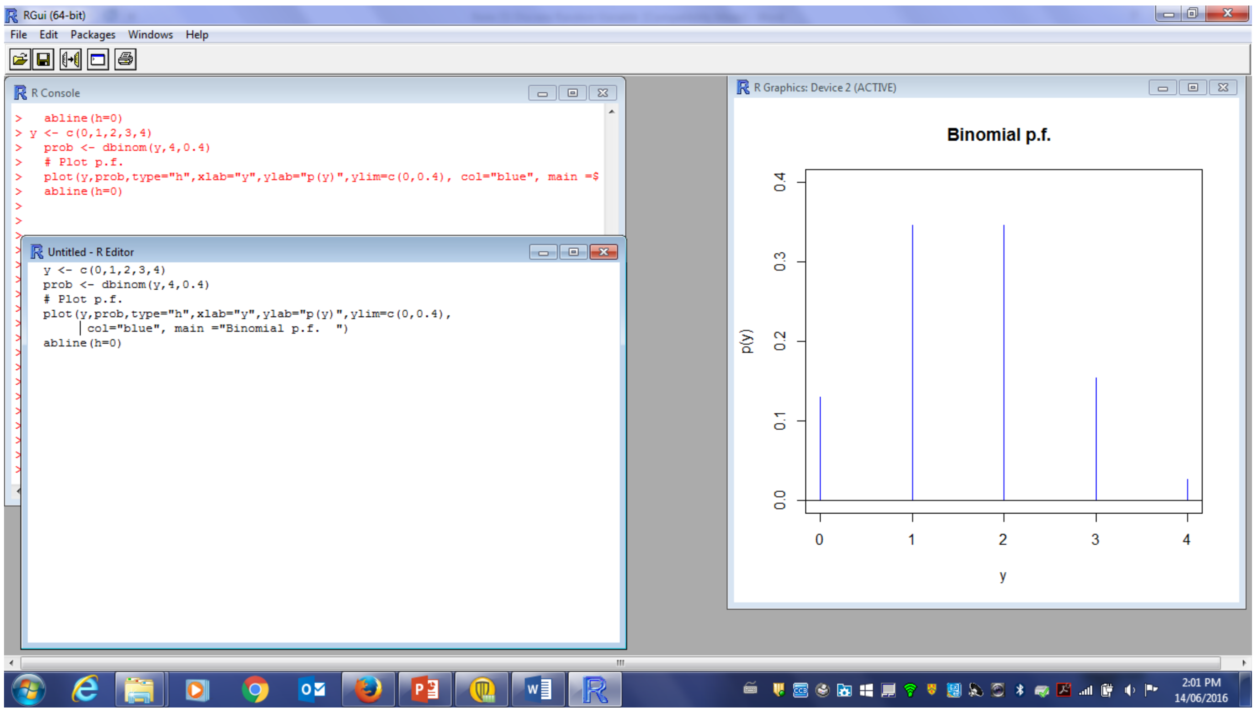
y <- c(0,1,2,3,4)

prob <- dbinom(y,4,0.4)

# Plot p.f.

plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.4) , col="blue", main ="Binomial p.f. ")

abline(h=0)



```
>   abline(h=0)
> y <- c(0,1,2,3,4)
>   prob <- dbinom(y,4,0.4)
>   # Plot p.f.
>   plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.4), col="blue", main =$
>   abline(h=0)
>
>
```

```
y <- c(0,1,2,3,4)
prob <- dbinom(y,4,0.4)
# Plot p.f.
plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.4),
      col="blue", main ="Binomial p.f.  ")
abline(h=0)
```

# Plot c.d.f

cdf <- c(0,cumsum(prob))

cdf.plot <- stepfun(y,cdf,f=0)

plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main="",pch=16)

# Example(4)
# Binomial p.f and c.d.f

y <- seq(0,30,1)

prob <- dbinom(y,30,0.1)

# Plot p.f.

plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.25))

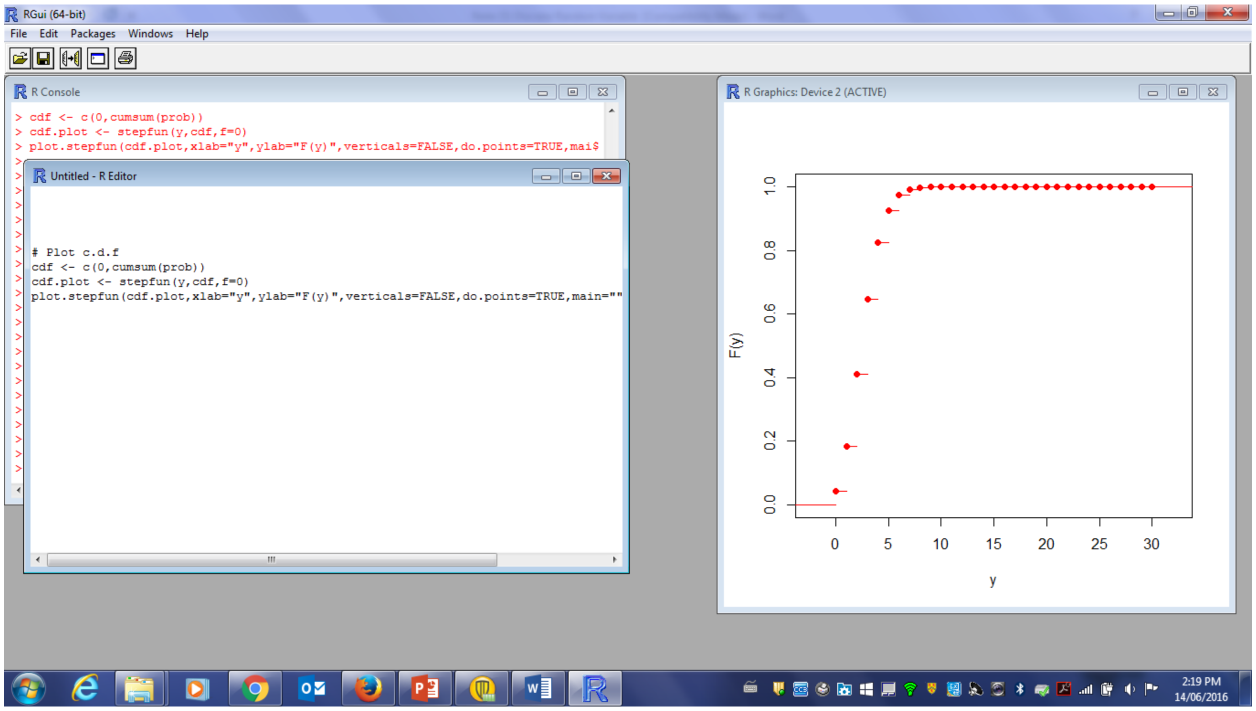
abline(h=0)

# Plot c.d.f

cdf <- c(0,cumsum(prob))

cdf.plot <- stepfun(y,cdf,f=0)

plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main="",pch=16)



```
> cdf <- c(0,cumsum(prob))
> cdf.plot <- stepfun(y,cdf,f=0)
> plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,mai$
```

```
# Plot c.d.f
cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main=""
```

# Example(5)

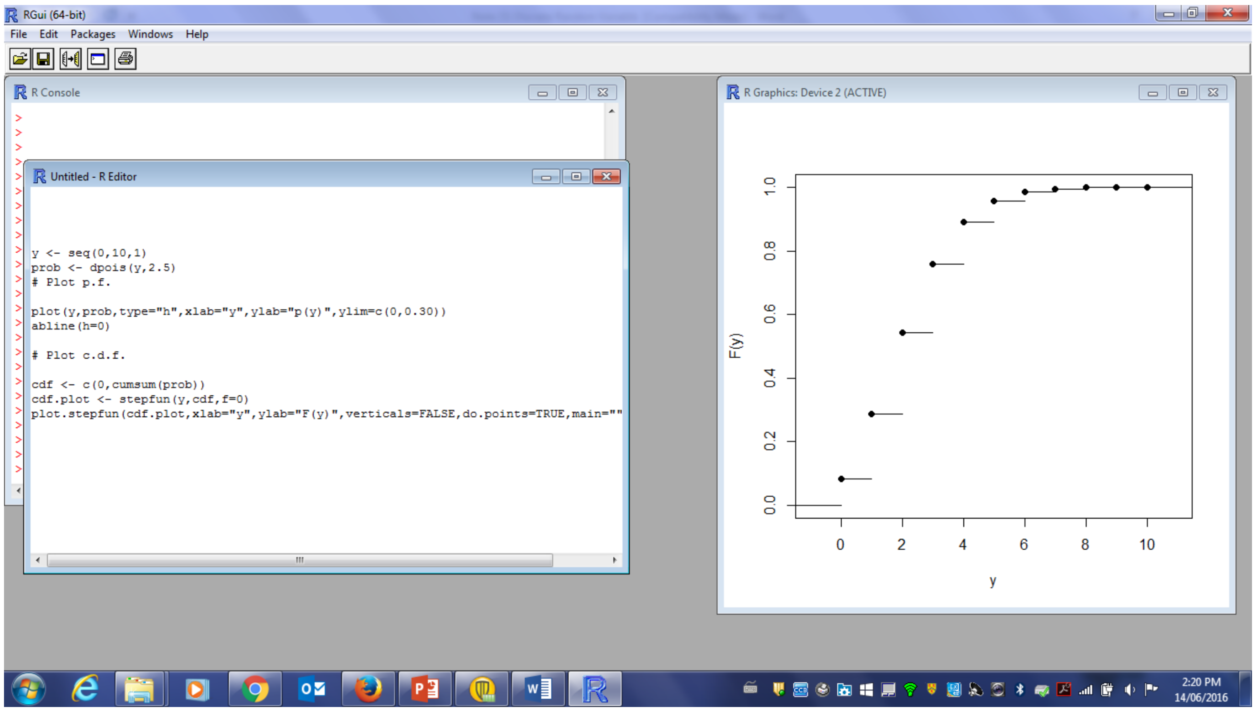
# Poisson p.f and c.d.f

y <- seq(0,10,1)
prob <- dpois(y,2.5)
# Plot p.f.

plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
abline(h=0)

# Plot c.d.f.

cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main="",pch=16)



```
y <- seq(0,10,1)
prob <- dpois(y,2.5)
# Plot p.f.

plot(y,prob,type="h",xlab="y",ylab="p(y)",ylim=c(0,0.30))
abline(h=0)

# Plot c.d.f.

cdf <- c(0,cumsum(prob))
cdf.plot <- stepfun(y,cdf,f=0)
plot.stepfun(cdf.plot,xlab="y",ylab="F(y)",verticals=FALSE,do.points=TRUE,main=""
```

Distributions in R

To get a full list of the distributions available in R you can use the following command:

```
help(Distributions)
```

For every distribution there are four commands. The commands for each distribution are prepended with a letter to indicate the functionality:

“d”	returns the height of the probability density function
“p”	returns the cumulative density function
“q”	returns the inverse cumulative density function (quantiles)
“r”	returns randomly generated numbers

The Binomial distribution

```
help(Binomial)
```

Example (1)

The distribution function, dbinom, with plot

```
x <- seq(0, 50, by=1)

y <- dbinom(x, 50, 0.2)

plot(x, y, col="blue", main=" The distribution function of Binomail with p=0.2", type= "h")
```

Example (2)

#(a) Toss the same coin 20 times, what's the probability of getting exactly 10 heads?

$$\binom{20}{10}(.5)^{10}(.5)^{10} = .176$$

dbinom(x, size, prob, log = FALSE)

```
dbinom(10, 20, 0.5)
```

# (b)Toss a coin 5times, what's the probability of getting at most 4 or fewer heads?

- The cumulative probability distribution function in R: **B(4; 5, 0.3)**

```
pbinom(4,5,0.3)  #  0.99757   as in the table
```

# (c)Toss a coin 5times, what's the probability of getting at most 2 or fewer heads?

```
pbinom(2, 5, 0.5)
```

B(2; 5, 0.5) # 0.5 as in the table

Example(3)

```
# Generate observations from  binomial distributions
# with p=0.4, and n=10

set.seed(123)
 n <-   10
 p <- 0.4
 y <- rbinom(6, n, p)
```

The Poisson distribution

Example (4)

Consider a computer system with Poisson job-arrival stream at an average of 2 per minute. Determine the probability that in any one-minute interval there will be
(i) 0 jobs
(ii) exactly 2 jobs
(iii) at most 3 arrivals.
(iv) more than 3 arrivals

(i)
No job arrivals:
$P(X=0) = e^{-2} = 0.135$

```
dpois(0, 2)
```

(ii)
Exactly 2 jobs

```
dpois(2, 2)
```

(iii)
At most 3 arrivals
$P(X\leq 3) = P(0) + P(1) + P(2) + P(3) = e^{-2} + e^{-2}21 + e^{-2}2^2/2! + e^{-2}2^3/3!$
$= 0.1353 + 0.2707 + 0.2707 + 0.1805 = 0.8571$

```
ppois(3,2)
[1] 0.8571235
```

(iv) more than 3 arrivals:
$P(X>3) = 1-P(X\leq 3) = 1-0.8571 = 0.1429$

```
1- ppois(3,2)
```

Example (5)

# Poisson Probability Density Functions plots

```
 par(mfrow = c(2,2))  # multiframe
 x <- 0:12  #look at the first 12 probabilities
 plot (x, dpois(x, 2),col="red", xlab = "Number of Hits", ylab = "P(X = x)",  type
             = "h", main= "Web Site Hits: Poisson(2)")
 plot (x, dpois(x, 3),col="blue", xlab = "Number of Calls", ylab = "P(X = x)",
            type = "h", main= "Calls to Mobile: Poisson(3)")
 plot (x, dpois(x, 4),col="green", xlab = "Number of Submissions", ylab = "P(X =
x)", type = "h", main= "Job Submissions: Poisson(4)")
plot (x, dpois(x, 6),col="darkred",  xlab = "Number of Messages", ylab = "P(X =
x)",  type = "h", main= "Messages to Server: Poisson(6)")
```

The Normal distribution

```
 help(Normal)
```

Example (6)

# Normal probability distribution with plot

```
 par(mfrow = c(2,1)) # multiframe
  x <- seq(-20,20,by=.1)
 y <- dnorm(x)
  plot(x,y, col="blue",main= " Normal probability distribution with mean =0,
var=1" )
  y <- dnorm(x,mean=2.5,sd=0.1)
  plot(x,y, col="red", main=" Normal probability distribution with mean =2.5,
var=0.1" )
```

Example (7)

pnorm, given a number or a list it computes the probability that a normally distributed random number will be less than that number "Cumulative Distribution Function"

```
pnorm(1)
[1] 0.8413447

 pnorm(0,mean=2)
[1] 0.02275013

 pnorm(0,mean=2,sd=3)
[1] 0.2524925
```

Example (8)

# We generate a random sample of size 50 from a normal population with
# mean 4 and standard deviation 7.
# Note the command? rnorm will give you more information about how the
 # function sample works.

```
set.seed(123)   # to get the same data each time we run the codes
x <- rnorm(50, 4, 7)
```

# To list only the 4 fist values of x

```
 x[1:4]
```

# To list the first and the third values of x

```
 x[c(1, 3)]
```

 # To calculate the minimum and the maximum of x

```
 min(x)

 max(x)
```

Practice Problems

Question (1)

Suppose there are 15 multiple choice questions in a probability class midterm. Each question has 5 possible answers, and only one of them is correct. If a student attempts to answer every question at random, find the probability of having

(a) Exactly 4 correct answers.

(b) At most 4 correct answers.

Question (2)

Cars cross a bridge at a uniform rate and the number of cars in non-overlapping time intervals are independent. On average 12 cars pass the bridge each minute. Find the probability that 17 or more cars pass the bridge in a one minute time interval.

Question (1)

Solution

Since only one out of five possible answers is correct, the probability of answering a question correctly by random is 1/5=0.2. We can find the probability of having exactly 4 correct answers by random attempts as follows.

```
dbinom(4, size=15, prob=0.2)
0.1876042
```

To find the probability of having four or less correct answers by random attempts, we apply the function dbinom with x = 0…, 4.

```
 dbinom(0, size=15, prob=0.2) +  dbinom(1, size=15, prob=0.2) +  dbinom(2, size=
15, prob=0.2) +  dbinom(3, size=15, prob=0.2) +  dbinom(4, size=15, prob=0.2)

[1] 0.8357663
```

Alternatively, we can use the cumulative probability function for binomial distribution pbinom.

```
 pbinom(4, size=15, prob=0.2)

[1] 0.8357663
```

Question (2)

Solution

The probability of having sixteen or less cars crossing the bridge in a particular minute is given by the function ppois.

```
ppois(16, lambda=12)   # lower tail
[1] 0.89871
```

Hence the probability of having seventeen or more cars crossing the bridge in a minute is in the upper tail of the probability density function.

```
ppois(16, lambda=12, lower=FALSE)   # upper tail
[1] 0.10129
```

If there are twelve cars crossing a bridge per minute on average, the probability of having seventeen or more cars crossing the bridge in a particular minute is 10.1%.