DATA ANALYSIS USING R

Instructor: Dr. Ping He
Email: heping@uic.edu.hk
Teacher assistant: Mr. Fu Han
Email: hanfu@uic.edu.hk

Course website

ISpace: Course: Data Analysis Using R;

Learning Outcomes

1. Ability to organize, visualize and analyze data by statistical software R.
2. Skills to select and use an appropriate statistical method to solve the real-world problem.
3. Report writing and oral presentation skills
4. **Develop ability on writing R program to solve various real-life problem.**

Textbook

No

Reference

http://www.r-project.org/

Ispace: An Introduction to R; simpleR; Using R for Data Analysis and Graphics; (Pdf files)

TOPICS:

- Introduction to R
- From data to graphics
- Probability Distributions
- Hypothesis Testing and Confidence Interval Estimation
- Analysis of variance
- Multiple Regression
- Tabular data

Assessment:

- Quizzes 10%
- Assignments and Lab exercises 20%
- Mid-term test 20%
- Group project 10%
- Final examination 40%

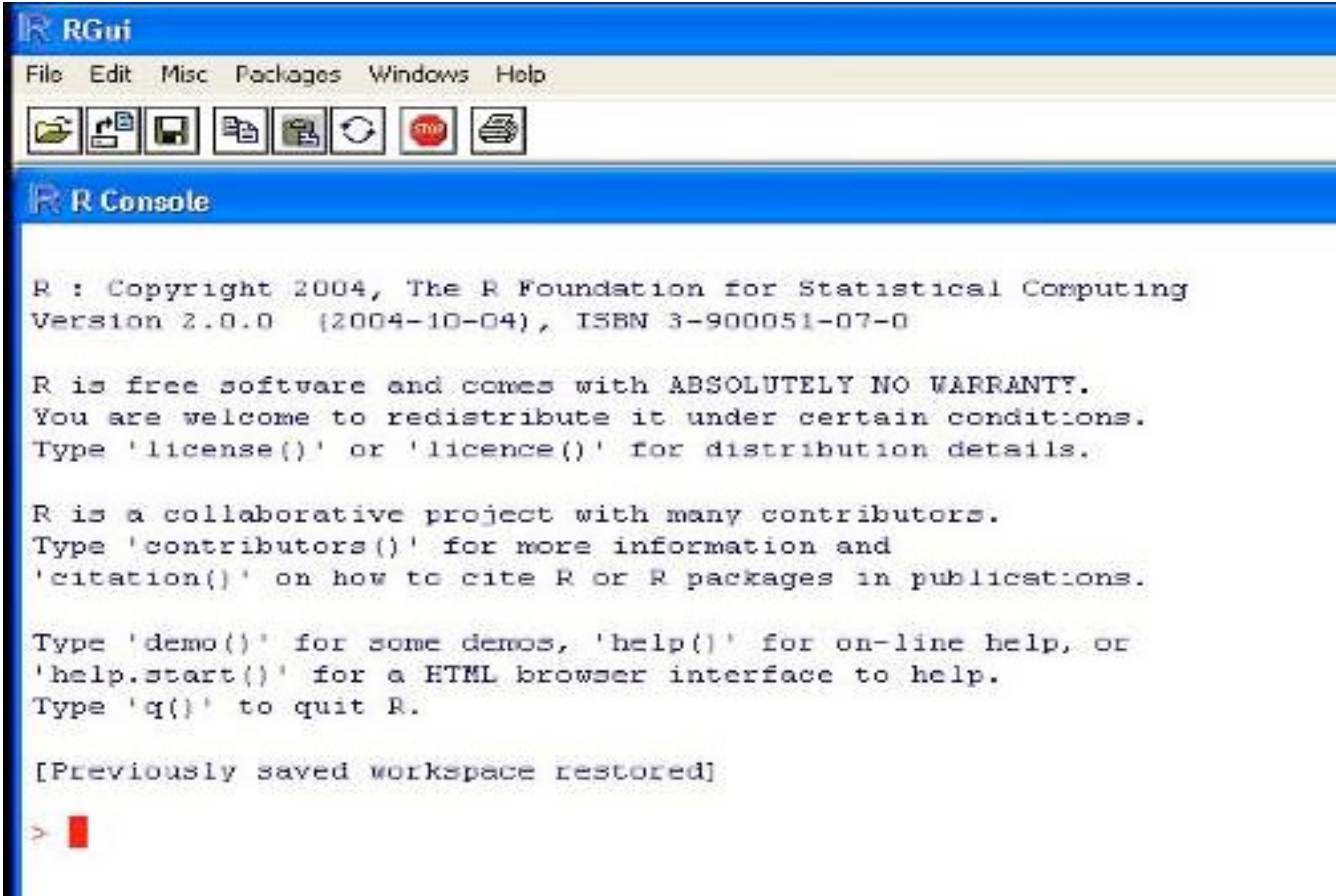
An Introduction to R

Website

- R www.r-project.org
  – software;
  – documentation;
  – RNews.

Installing R

http://cran.r-project.org


RGui
File Edit Misc Packages Windows Help
R Console
R : Copyright 2004, The R Foundation for Statistical Computing
Version 2.0.0 (2004-10-04), ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
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 a HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
>

USEFUL COMMANDS:

- Quit

  > **q()** to quit

  File _ Exit also works to quit in Windows

- File path is relative to working directory

  > **getwd()**

  > **setwd()**

  File _ Change dir also works in Windows

- List objects in workspace

  > **ls()**

Getting Help

- **Details about a specific command whose name you know (input arguments, options):**

```
> ? t.test
> help(t.test)
```

- **Find commands containing a regular expression:**

```
> apropos("var")
[1] "var.na" ".__M__varLabels:Biobase"
[3] "varLabels" "var.test"
[5] "varimax" "all.vars"
[7] "var" "variable.names"
[9] "variable.names.default""variable.names.lm"
```

Getting Help

- **HTML search engine lets you search for topics related to regular expressions:**
  > help.search("covariance")

- **See an example of usage:**
  > demo(graphics)
  > example(mean)

- **To see code for a function, type the name**
  with no parentheses or arguments:
  > plot

R as a Calculator

- > 2+2
  [1] 4
- > 2*3*4*5
  [1] 120
- > log2(32)
  [1] 5
- > print(sqrt(2))
  [1] 1.414214
- > pi
  [1] 3.141593
- >1000*(1+0.075)^5 - 1000
  [1] 435.6293
- >sin(c(30,60,90)*pi/180)
  [1] 0.5000000 0.8660254 1.0000000

Arithmetic

+ -	add, subtract
* /	multiply, divide
^	exponentiation
%%	modulus
%/%	integer divide,
`abs()`	absolute value
`cos()`, `sin()`, `tan()`	cosine, sine, tangent of angle x
`exp()`	exponential function, e^x
`log()`	natural (base-e) logarithm
`log10()`	common (base-10) logarithm
`sqrt()`	square root

IMPORTANT CONCEPT

- Command: function

  plot(mydata1), q()

- Objects:

  vector, *matrix*, *factor*, *list*, and *data frame*

- Modes

  logical, numeric, and character.

MODES

- Variables:

  >a<-49 **Numeric**

  >sqrt(a)

  [1] 7

  >b<-”The dog ate my homework”

  >sub(“dog”,”cat”,b) **Character**

  [1] “The cat ate my homework” **string**

  >**d<-(1+1==3)**

  >d **Logical**

  [1] FALSE

Objects

- **>ls()**
- List all data objects currently available
- **>rm()**

  removes the data object
- **>typeof()**

  Determine the type of an object
- **>class()**

  Determine the type of an object

VECTORS:

- **> a <- c(1,2,3)**
- **> a*2**

  [1] 2 4 6
- **>t1 <- c(1,2,3,4,5)**
- **>t1**

- In R, a single number is the special case of a vector with 1 element.
- Other vector types: character strings, logical:
  c(T,F,F,F,T,T,F)

  c(”Canberra”,”Sydney”,”Newcastle”,”Darwin”)

Vectors (Expression)

- > **c(1,2,3,4,5)**

  [1] 1 2 3 4 5

  > **1:5**

  [1] 1 2 3 4 5

  > **seq(1, 5, by=1)**

  [1] 1 2 3 4 5

  > **seq(1, 5, length=5)**

  [1] 1 2 3 4 5

Joining Vectors

- `> x <- c(2,3,5,2,7,1)`
- `> y <- c(10,15,12)`
- `> z <- c(x, y)`
- `> z`

  [1] 2 3 5 2 7 1 10 15 12
- `>length(Z)`
- **[1] 9**

SUBSETS OF VECTORS

a. **> x <- seq(-1, 1, by=.1)**

> x

[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> x[5:10]

[1] -0.6 -0.5 -0.4 -0.3 -0.2 -0.1

> x[c(5,7:10)]

[1] -0.6 -0.4 -0.3 -0.2 -0.1

> x[-(5:10)]

We remove the elements whose index lies between 5 and 10

[1] -1.0 -0.9 -0.8 -0.7 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> x>0

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE [13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

>x[x>0]

[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

Example

1. Create a sequence of numbers from 3 to 30 in steps of 3.

Solution:

Create a sequence of numbers from 1 to 30 that does not include the numbers that where mentioned above (those that are divisible by 3)

VECTOR ARITHMETIC

The elementary arithmetic operators are the usual +, -, *, / and ^

a <- c(1,2,3)
Try to a*2, a-2, a+2, a^2

In addition all of the common arithmetic functions are available.
log, exp, sin, cos, tan, sqrt.
Try to sqrt(a)

Vector arithmetic

Name	**Operations**
round	round up for positive and round down for negative
sort	sort the vector in ascending or descending order
sum	return the sum of the vector
cumsum	cumulative sum
cumprod	cumulative product
min, max	return the smallest and largest values
range	return a vector of length 2 containing the min and max
mean	return the sample mean of a vector
var	return the sample variance of a vector
sd	return the sample standard deviation of a vector

FUNCTIONS OPERATE ON THE ELEMENTS OF VECTORS:

- >union(v1,v2)
- >intersect(v1,v2)
- >setdiff(v1,v2)
- >setequal(v1,v2),
- >is.element(element1,v1)
  (or, >element1 %in% v1).

Repeat Vector

- **>rep(1,10)**

  [1] 1 1 1 1 1 1 1 1 1 1
- **> rep(1:5,3)**

  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
- **>rep(1:5,each=3)**

  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
- **> rep(1:5,2,each=3)**

  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5

Example: Use vector commands to answer the following questions:

Exercise: Computes values of $y = \frac{(x-1)}{(x+1)}$ for x = 1, 2, · · · , 10.

Exercise The sum of the geometric series $1+r+r^2+r^3+...+r^n$ approaches the limit $1/(1-r)$ for $r < 1$ as $n \to \infty$. Take $r = 0.5$ and $n = 10$, and write a **one-statement** command that creates the vector $G = (r^0, r^1, r^2, ..., r^n)$. Compare the sum (using `sum()`) of this vector to the limiting value $1/(1-r)$. Repeat for $n = 50$.

Appendix: Frequent used operator

<table>
<tr><td>|</td><td>Or</td></tr>
<tr><td>&</td><td>And</td></tr>
<tr><td><</td><td>Less</td></tr>
<tr><td>></td><td>Greater</td></tr>
<tr><td><=</td><td>Less or =</td></tr>
<tr><td>>=</td><td>Greater or =</td></tr>
<tr><td>!</td><td>Not</td></tr>
<tr><td>!=</td><td>Not equal</td></tr>
<tr><td>==</td><td>Is equal</td></tr>
</table>

```
>(5>3)|(5>6)
[1] TRUE
>(5>3)&(5>6)
[1] FALSE
```

MATRICES

- Matrix: rectangular table of data of the same type (Two dimension)

- A matrix:

> matrix(1:10, ncol=5)

```
1   3   5   7   9
2   4   6   8  10
```

Caution: by default, the elements of a matrix are given vertically, column after column

Create Matrix

- > matrix(1:9, nrow=3, ncol=3)

	[,1]	[,2]	[,3]
[1,]	1	4	7
[2,]	2	5	8
[3,]	3	6	9

- > matrix(1:9, nrow=3, ncol=3, byrow=T)

	[,1]	[,2]	[,3]
[1,]	1	2	3
[2,]	4	5	6
[3,]	7	8	9

- > t(matrix(1:9, nrow=3, ncol=3)) transpose a matrix

	[,1]	[,2]	[,3]
[1,]	1	2	3
[2,]	4	5	6
[3,]	7	8	9

RBIND AND CBIND

- a<-c(1,2,3)
- b<-c(4,6,8)
- > cbind(a,b)   #combine by columns

```
     a b
[1,] 1 4
[2,] 2 6
[3,] 3 8
```

- > rbind(a,b)   #combine by rows

```
  [,1] [,2] [,3]
a    1    2    3
b    4    6    8
```

Subset of Matrix

```
>A<-matrix( 1:9, nrow=3, ncol=3 )
>A[2,3]                    #specify both the row and column
 [1] 8
>A[2:3, 1:2]               #specify range of rows and columns.
     [,1] [,2]
[1,]    2    5
[2,]    3    6
>A[2:3,]       # extract entire columns by leaving a blank after the comma
     [,1] [,2] [,3]
[1,]    2    5    8
[2,]    3    6    9
>A[,2:3]       # extract entire rows by leaving a blank before the comma
     [,1] [,2]
[1,]    4    7
[2,]    5    8
[3,]    6    9
```

Matrix product (* and %*%)

- suppose m1 is the matrix

  1 2 3
  4 5 6
- Then **m1 * 2** is 2 4 6
  8 10 12
- Matrix multiplication works too. Suppose m2 is the matrix

  1 2
  1 2
  1 2
- then **m1 %*% m2** is

  6 12
  15 30
- and **m2 %*% m1** is

  9 12 15
  9 12 15
  9 12 15

IMPORTANT COMMAND OF MATRIX

```
>m <- matrix( c(1,2,3,4), nrow=2 )
```

- Determinant:

```
>det(m)
[1] -2
```

- Transpose:

```
>t(m)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

- A diagonal matrix:

```
>diag(c(1,2))
     [,1] [,2]
[1,]    1    0
[2,]    0    2
```

IMPORTANT COMMAND OF MATRIX

- Identity matrix

  > diag(2)

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

- Trace of a matrix:

  > sum(diag(m))

  [1] 5

- Inverse of a matrix:

  > solve(m)

```
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

IMPORTANT COMMAND OF MATRIX

- Eigenvalues:

  > eigen(m)$values

```
[1]  5.3722813 -0.3722813
```

- Eigenvectors:

  > eigen(m)$vectors

```
           [,1]       [,2]
[1,] -0.5742757 -0.9093767
[2,] -0.8369650  0.4159736
```

ARRAYS

- **array: 3-,4-,..dimensional matrix**

> d <- array(1:18, dim=c(3,3,2))

> d

```
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
```

Example of Array

- > data(HairEyeColor)
- > HairEyeColor

```
, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black 32    11   10     3
  Brown 38    50   25    15
  Red   10    10    7     7
  Blond  3    30    5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black 36     9    5     2
  Brown 81    34   29    14
  Red   16     7    7     7
  Blond  4    64    5     8
```

- > is.array(HairEyeColor)
- [1] TRUE

Useful Command (Sampling with and without replacement using command "sample")

```
## Roll a die
> sample(1:6,10,replace=TRUE)
[1] 5 1 5 3 3 4 5 4 2 1
## toss a coin
> sample(c("H","T"),10,replace=TRUE)
[1] "H" "H" "T" "T" "T" "T" "H" "H" "T" "T"
## pick 6 of 54 (a lottery)
> sample(1:54,6) # no replacement
[1] 6 39 23 35 25 26
```

By default, "sample" samples without replacement each object having equal chance of being picked. You need to specify replace=TRUE if you want to sample with replacement. Furthermore, you can specify separate probabilities for each if desired.

FACTOR

- A factor is a vector coding for a qualitative variable.

> x <- factor(sample(c("Yes", "No", "Perhaps"), 5, replace=T))

> x

```
[1] Yes     No      Perhaps  Perhaps  No
Levels: No Perhaps Yes
```

- "levels" of this factor.

> levels(x)

```
[1] "No"      "Perhaps" "Yes"
```

Data Frames

- **Data frame: rectangular table with rows and columns; data within each column has the same type (e.g. number, text, logical), but different columns may have different types**

```
>install.packages("car")
>library(car)
>States
```

	region	pop	SATV	SATM	percent	dollars	pay
AL	ESC	4041	470	514	8	3.648	27
AK	PAC	550	438	476	42	7.887	43
AZ	MTN	3665	445	497	25	4.231	30
AR	WSC	2351	470	511	6	3.334	23
CA	PAC	29760	419	484	45	4.826	39
CO	MTN	3294	456	513	28	4.809	31
CN	NE	3287	430	471	74	7.914	43
DE	SA	666	433	470	58	6.016	35
DC	SA	607	409	441	68	8.210	39
FL	SA	12938	418	466	44	5.154	30
GA	SA	6478	401	443	57	4.860	29
HI	PAC	1108	404	481	52	5.008	32
ID	MTN	1007	466	502	17	3.200	25

The data States is a data frame

Data Frames

- Access the columns of a data.frame

> States$pop

```
[1]  4041   550  3665  2351 29760  3294  3287   666   607 12938  6478  1108  1007 11431  5544  2777  2478
[18]  3685  4220  1228  4781  6016  9295  4375  2573  5117   799  1578  1202  1109  7730  1515 17990  6629
[35]   639 10847  3146  2842 11882  1003  3487   696  4877 16987  1723   563  6187  4867  1793  4892   454
```

> States[,1]

```
[1] ESC PAC MTN WSC PAC MTN NE  SA  SA  SA  SA  PAC MTN ENC ENC WNC WNC ESC WSC NE  SA  NE  ENC WNC ESC WNC
[27] MTN WNC MTN NE  MA  MTN MA  SA  WNC ENC WSC PAC MA  NE  SA  WNC ESC WSC MTN NE  SA  PAC SA  ENC MTN
Levels: ENC ESC MA MTN NE PAC SA WNC WSC
```

> States[[“pop"]]

```
[1]  4041   550  3665  2351 29760  3294  3287   666   607 12938  6478  1108  1007 11431  5544  2777  2478
[18]  3685  4220  1228  4781  6016  9295  4375  2573  5117   799  1578  1202  1109  7730  1515 17990  6629
[35]   639 10847  3146  2842 11882  1003  3487   696  4877 16987  1723   563  6187  4867  1793  4892   454
```

- Get the dimension of a data.frame

> dim(States)

[1] 51, 7

> names(States)

[1] "region" "pop"   "SATV"   "SATM"   "percent" "dollars" "pay"

SUBSET OF A DATA.FRAME

- >States[1:3,]

	region	pop	SATV	SATM	percent	dollars	pay
AL	ESC	4041	470	514	8	3.648	27
AK	PAC	550	438	476	42	7.887	43
AZ	MTN	3665	445	497	25	4.231	30

- >States[States$region=="SA"&States$SATM>450,]

	region	pop	SATV	SATM	percent	dollars	pay
DE	SA	666	433	470	58	6.016	35
FL	SA	12938	418	466	44	5.154	30
MD	SA	4781	430	478	59	6.184	38
VA	SA	6187	425	470	58	5.360	32
WV	SA	1793	443	490	15	5.046	26

Create a Data Frame (command:data.frame)

```
>x <-1:30
>y<-rep(c("female","male"),15)
> z <- sample(c(0:100),30,replace=T)
>  data.frame(num=x,sex=y,score=z)
```

	num	sex	score
1	1	female	61
2	2	male	34
3	3	female	23
4	4	male	10
5	5	female	50
6	6	male	36
7	7	female	7
8	8	male	3
9	9	female	83
10	10	male	41
11	…		

OPERATIONS ON VECTORS, ARRAYS OR MATRIX (USEFUL COMMAND)

"**Apply**" **function** applies a function (mean, quartile, etc.) to each column or row of a data.frame, matrix or array.

```
> df <- data.frame(x=rnorm(20),y=rnorm(20),z=rnorm(20))
> apply(df,2,mean)
      x        y        z
0.04937 -0.11279 -0.02171
> apply(df,2,range)

          x      y      z
[1,] -1.564 -1.985 -1.721
[2,]  1.496  1.846  1.107
```

Operations on vectors and arrays (useful command)

➢"**Tapply**" function groups the observations along the value of one (or several) factors and applies a function (mean, etc.) to the resulting groups. The "**by**" command is similar.

```
X=1:30
Y=rep(c("female","male"),15)
Z=sample(c(0:100),30,replace=T)
 data=data.frame(num=X,sex=Y,score=Z)
tapply(data$score, data$sex,mean)
```

Lists

- **Store complex data:**

```
> h <- list()
> h[["foo"]] <- 1
> h[["bar"]] <- c("a", "b", "c")
> h
$foo
[1] 1
$bar
[1] "a" "b" "c"
```

- **Choose element from a list**

```
> h[["bar"]]
[1] "a" "b" "c"
> h[[2]]
[1] "a" "b" "c"
```

MISSING VALUES

- The missing values are coded as "NA"

```
> x <- c(1,5,9,NA,2)
> x
[1] 1 5 9 NA 2
>is.na(x)
[1] FALSE FALSE FALSE  TRUE FALSE
> mean(x)
[1] NA
```

- Remove the missing values.

```
> mean(x, na.rm=T)
[1] 4.25
> na.omit(x)
[1] 1 5 9 2
```

R language: Control structures

- **Conditional statements**

  a. if(...) { ...

  } else { ...

  }

  b.switch

R LANGUAGE: CONTROL STRUCTURES

- Loop:

  a. for (i in 1:10) { ...}

  ```
  for (i in 1:20) {cat(i)}
  for (i in 1:20) {cat(i,"\t")}
  for (i in 1:20) {cat(i,"\n")}
  ```

  b. while(...) { ...}

  ```
  g<-0
  while (g<20)
  {g<-g+1)
  cat(g);cat("\n")}
  ```

  c. repeat { ... if(...) { break } ... }

  ```
  repeat {g <- g+1
  if (g > 19) break
  cat(g);cat("\n") }
  ```

Example

a. Add up all the numbers from 1 to 100 in two different ways: using **for** and using **sum**.

b. Multiply all the numbers from 1 to 50 in two different ways: using **for** and using **prod**.

FUNCTION

The basic template

```
function_name <- function (function_arguments) {
  function_body
  function_return_value
}
```

- **A function is defined as follows**.

f <- function(x) { x^2 + x + 1
}

The return value is the last value computed

- Use the "return" function.

f <- function(x) {
 return(x^2 + x + 1)
}

- Arguments can have default values.

f <- function(x, y=3) { ... }

Example:

```
ff <- function(x,Alpha=1,B=0)
 {
 out <- sin(x[1])-sin(x[2]-Alpha)+x[3]^2+B
 return(out)
 }
```

```
> ff(c(2,4,1),Alpha=3)
[1] 1.067826
> ff(c(2,4,1))
[1] 1.768177
```

EXAMPLE:

Write a R function that returns the value of the Haar wavelet, defined by

$$\psi^{(H)}(u) = \begin{cases} -1/\sqrt{2} & -1 < u \le 0 \\ 1/\sqrt{2} & 0 < u \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Use the vector (-0.96, -0.27, -0.53, -0.07, -2.22, -1.67, 0.08, -1.13, 0.13, -1.49) to check if your program is correct.

Solution

Strings

- **Print a string**: "print" function and "cat" function.

- > print("Hello\n")

  [1] "Hello\n"

- > cat("Hello\n")

  Hello

  - s <- "C:\\Program Files\\"
  - > print(s)

    [1] "C:\\Program Files\\"
  - > cat(s, "\n")

    C:\Program Files\

STRINGS

• The "nchar" function gives the length of a string there it is).

```
> nchar("Hello World!")
[1] 12
```

• The "substring" function extract part of a string (the second argument is the starting position, the third argument is the end position).

```
> s <- "Hello World"
> substring(s, 4, 6)
[1] "lo "
```

STRINGS

• The "gsub" function replaces each occurrence of a string (a regular expression, actually) by a string.

```
> s <- "foo bar baz"
> gsub("f", "t", s)
[1] "too bar baz"
> gsub("o", "tt", s)
[1] "ftttt bar baz"
```

• The "sub" only replaces the first occurrence.

```
> s <- "foo bar baz"
> sub(" ", "", s)
[1] "foobar baz"
```

DATA IMPORT

- Import data from readable formats
  - d <- read.table("filename.txt", header=T, sep=",")
  - d <-read.csv("filename.csv", header=T, sep=",")

- For the file only containing number or only strings, "scan" function is better choice.

EXAMPLE

Suppose there is a data file “austpop.txt” on a disk in drive a.

Year	NSW	Vic.	Qld	SA	WA	Tas.	NT	ACT	Aust.
1917	1904	1409	683	440	306	193	5	3	4941
1927	2402	1727	873	565	392	211	4	8	6182
1937	2693	1853	993	589	457	233	6	11	6836
1947	2985	2055	1106	646	502	257	11	17	7579
1957	3625	2656	1413	873	688	326	21	38	9640
1967	4295	3274	1700	1110	879	375	62	103	11799
1977	5002	3837	2130	1286	1204	415	104	214	14192

Read the data into R:

> austpop <- read.table(“a:/austpop.txt”, header=T)

> austpop

```
  Year  NSW Vic.  Qld   SA   WA Tas.  NT ACT Aust.
1 1917 1904 1409  683  440  306  193   5   3  4941
2 1927 2402 1727  873  565  392  211   4   8  6182
. . .
```

The object **austpop** is, in R parlance, a *data frame*.

A Short R Session

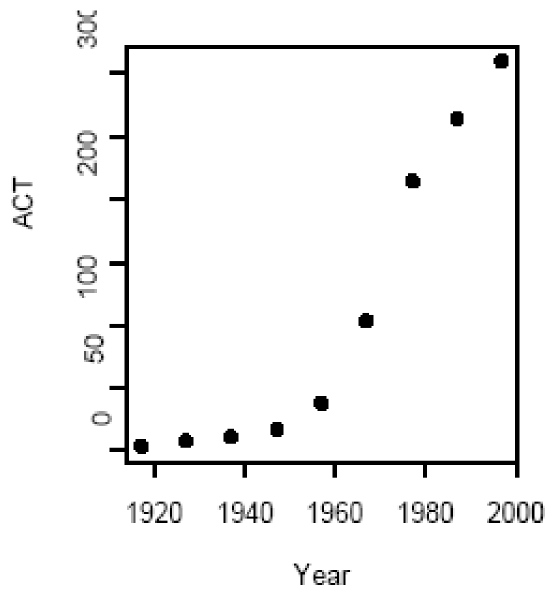
- **Get column names**

  > names(austpop)

```
[1] "Year"   "NSW"    "Vic."   "Qld"    "SA"     "WA"     "Tas."   "NT"
[9] "ACT"    "Aust."
```

- **Get the plot**

  >plot(ACT ~ Year, data=austpop, pch=16)



SIMPLE SUMMARY STATISTICS

Use str and summary to get an overview about R object

>install.packages("car")
>library(car)
>srt(States)

```
'data.frame':   51 obs. of  7 variables:
 $ region : Factor w/ 9 levels "ENC","ESC","MA",..: 2 6 4 9 6 4 5 7 7 7 ...
 $ pop    : int  4041 550 3665 2351 29760 3294 3287 666 607 12938 ...
 $ SATV   : int  470 438 445 470 419 456 430 433 409 418 ...
 $ SATM   : int  514 476 497 511 484 513 471 470 441 466 ...
 $ percent: int  8 42 25 6 45 28 74 58 68 44 ...
 $ dollars: num  3.65 7.89 4.23 3.33 4.83 ...
 $ pay    : int  27 43 30 23 39 31 43 35 39 30 ..
```

>summary(States)

>summary(States)

```
 region        pop              SATV            SATM            percent
SA    : 9   Min.   :  454   Min.   :397.0   Min.   :437.0   Min.   : 4.00
MTN   : 8   1st Qu.: 1215   1st Qu.:422.5   1st Qu.:470.0   1st Qu.:11.50
WNC   : 7   Median : 3294   Median :443.0   Median :490.0   Median :25.00
NE    : 6   Mean   : 4877   Mean   :448.2   Mean   :497.4   Mean   :33.75
ENC   : 5   3rd Qu.: 5780   3rd Qu.:474.5   3rd Qu.:522.5   3rd Qu.:57.50
PAC   : 5   Max.   :29760   Max.   :511.0   Max.   :577.0   Max.   :74.00
(Other):11
   dollars          pay
Min.   :2.993   Min.   :22.00
1st Qu.:4.354   1st Qu.:27.50
Median :5.045   Median :30.00
Mean   :5.175   Mean   :30.94
3rd Qu.:5.689   3rd Qu.:33.50
Max.   :9.159   Max.   :43.00
```

Exercise:
Use the data set **States** in library ‘car’.

a) Calculate the mean of SATM for each region.

b) Write your own short R program to estimate the coefficients in a regression model in which SAT math score is response and teachers’ salary and percentage of students taking the SAT exam are independent variables.

Appendix: Some elementary functions (Univariable)

- min(x)
- max(x)
- median(x) # median
- mean(x) # mean
- var(x) # variance
- sd(x) # standard deviation
- rank(x) # rank
- sum(x)
- length(x)
- round(x)
- fivenum(x) # quantiles
- **quantile(x) # quantiles (different convention)**
- quantile(x, c(0,.33,.66,1))
- **cor(x,y)) # correlation**

APPENDIX: GENERATE RANDOM NUMBERS FROM NORMAL DISTRIBUTION

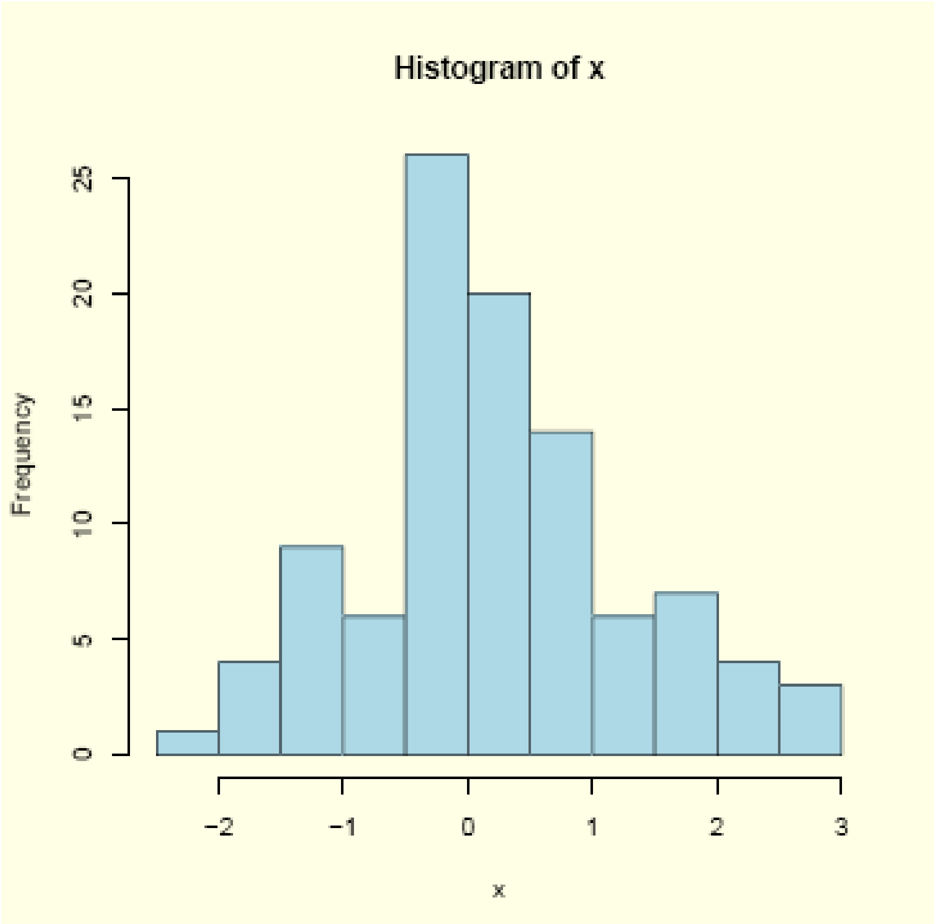
The function is called as rnorm(n,mean,sd) where one species the mean and the standard deviation.

Example

```
rnorm(1,100,16) # an IQ score
[1] 94.1719
rnorm(5,mean=280,sd=10)
[1] 277.2562 263.8982 264.3409 286.3676 279.1569
```

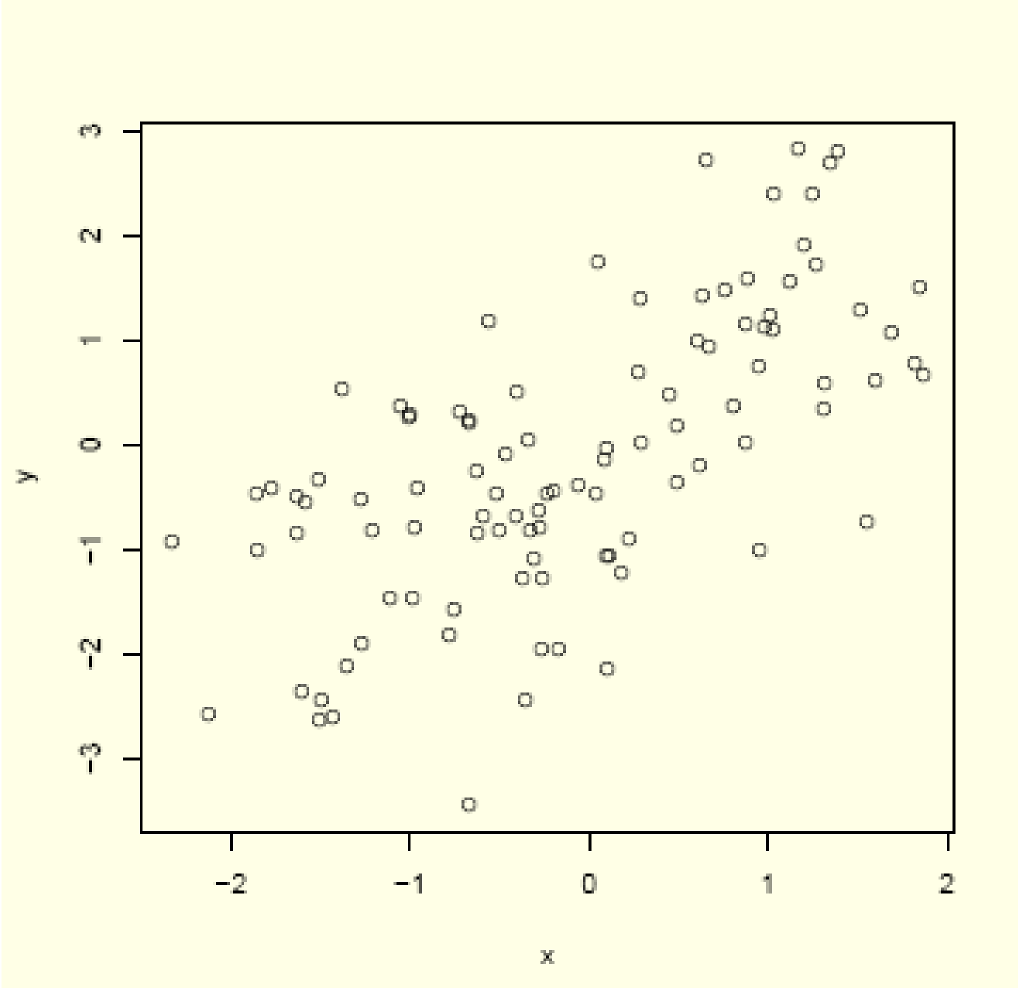
PLOT A HISTOGRAM

- x <- rnorm(100)
- hist(x, col = "light blue")



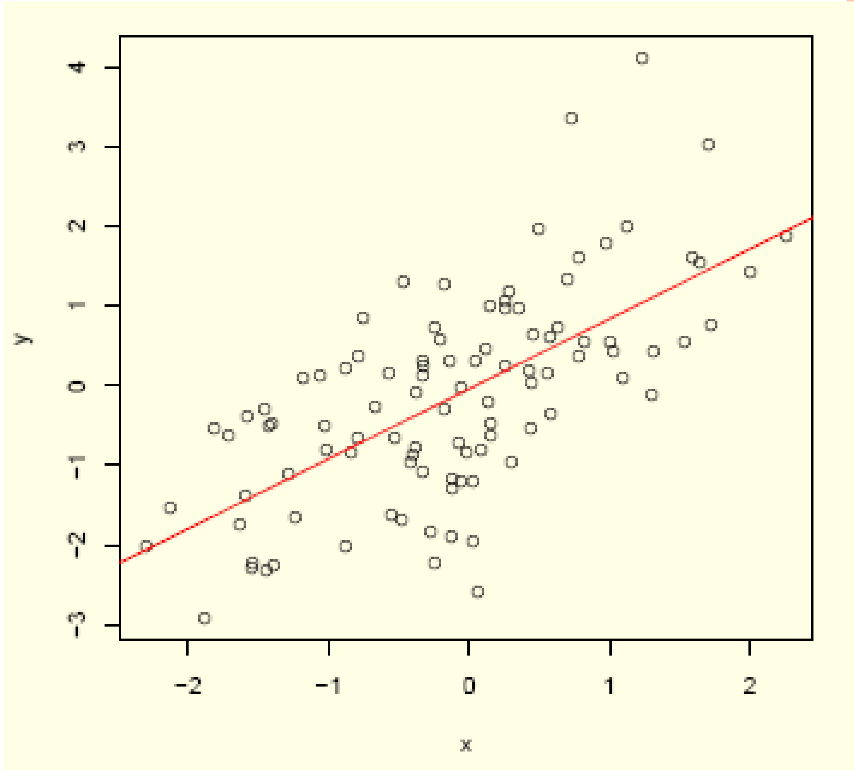
DISPLAY A SCATTER PLOT OF TWO VARIABLES

```
N <- 100
x <- rnorm(N)
y <- x + rnorm(N)
plot(y ~ x)
```



"REGRESSION LINE"

- **N <- 100**
- **x <- rnorm(N)**
- **y <- x + rnorm(N)**
- **plot(y ~ x)**
- **abline(lm(y ~ x), col = "red")**



Appendix: Single Linear Regression.

- A data set cntry.csv collects 15 countries' information. Two variables are contained in this data as follows:

  lifeexpf: female life expectancy

  Birthrat: births per 1000 population

- Download this data from ISpace. Read it into R.

a) Estimate the regression of female life expectancy on birthrate. Write out the regression equation

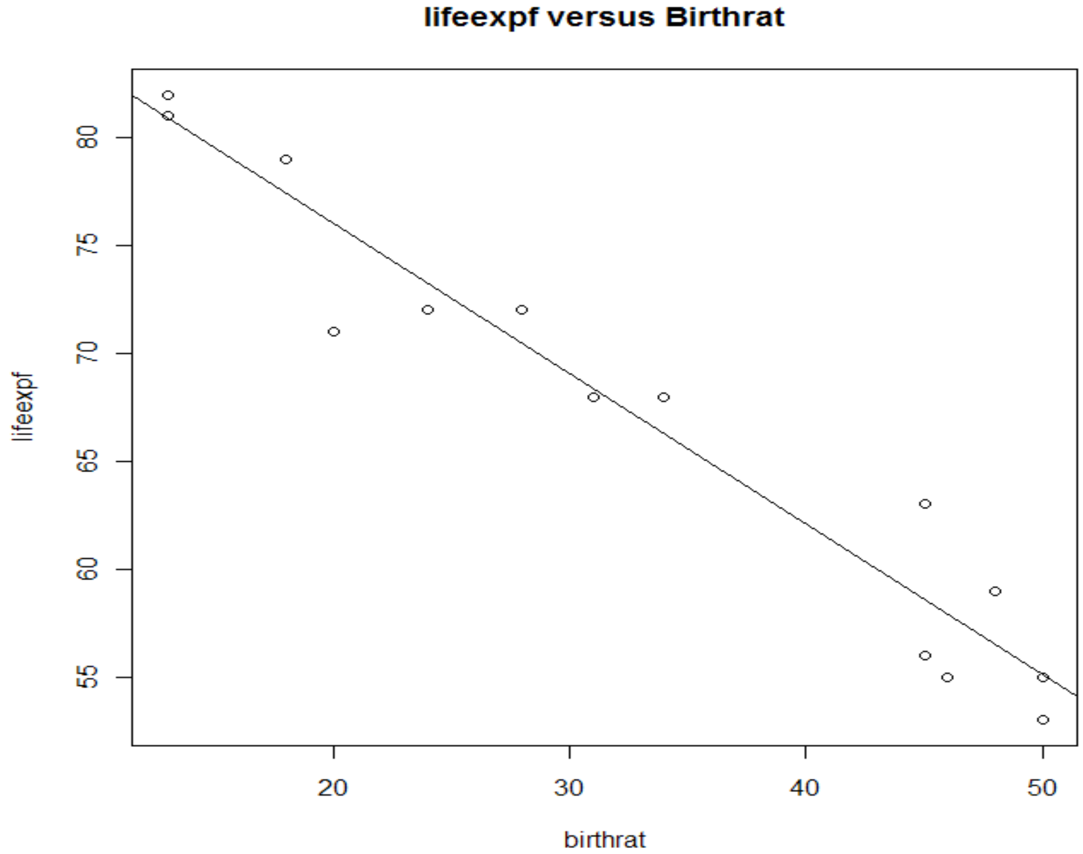
```
cntry<-read.csv(file.choose(),head=T)
lm.cntry<-lm(lifeexpf~birthrat,data=cntry)
```

```
> lm.cntry

Call:
lm(formula = lifeexpf ~ birthrat, data = cntry)

Coefficients:
(Intercept)     birthrat
    89.9852      -0.6973
```

$$\textit{life expectancy}=90-(0.70 \times \textit{birthrate})$$

b) Make a scatter plot of the variable lifeexpf versus variable Birthrat. Add the regression line to this plot.

```
attach(cntry)
plot(birthrat,lifeexpf,main="lifeexpf versus Birthrat")
abline(lm.cntry$coef)
```



c) Compute the coefficients of correlation.

cor(cntry[,2:3])

```
> cor(cntry[,2:3])
           lifeexpf   birthrat
lifeexpf  1.0000000 -0.9683468
birthrat -0.9683468  1.0000000
```

The coefficient of correlation is -0.968. That indicates a strong negative linear relationship between female life expectancy and birthrate.

d) Find and interpret the value of R^2

summary(lm.cntry)

```
> summary(lm.cntry)

Call:
lm(formula = lifeexpf ~ birthrat, data = cntry)

Residuals:
   Min      1Q Median      3Q     Max
-5.039 -1.684  0.080  1.553  4.394

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 89.98517    1.76457   50.99 2.32e-16 ***
birthrat    -0.69732    0.04985  -13.99 3.26e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.537 on 13 degrees of freedom
Multiple R-squared:  0.9377,    Adjusted R-squared:  0.9329
F-statistic: 195.7 on 1 and 13 DF,  p-value: 3.259e-09
```

R^2 is 0.9337. 93.37% of variability in female life expectancy can be explained by this regression

```
> summary(lm.cntry)

Call:
lm(formula = lifeexpf ~ birthrat, data = cntry)

Residuals:
   Min     1Q Median     3Q    Max 
-5.039 -1.684  0.080  1.553  4.394 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 89.98517    1.76457   50.99 2.32e-16 ***
birthrat    -0.69732    0.04985  -13.99 3.26e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 2.537 on 13 degrees of freedom
Multiple R-squared:  0.9377,     Adjusted R-squared:  0.9329 
F-statistic: 195.7 on 1 and 13 DF,  p-value: 3.259e-09
```

$$s = \sqrt{MSE} = \sqrt{\frac{SSE}{n-p}} = \sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{n-p}}$$

e) Compute 95% confidence interval for the slope coefficient in this simple regression. Test null hypothesis that, at the 5% level, the significance of the coefficient on birthrate in this regression.

confint(lm.cntry,level=0.95)

```
                  2.5 %     97.5 %
(Intercept) 86.1730414 93.7973030
birthrat    -0.8050175 -0.5896177
```

summary(lm.cntry)

```
>  summary(lm.cntry)

Call:
lm(formula = lifeexpf ~ birthrat, data = cntry)

Residuals:
   Min     1Q Median     3Q    Max
-5.039 -1.684  0.080  1.553  4.394

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 89.98517    1.76457   50.99 2.32e-16 ***
birthrat    -0.69732    0.04985  -13.99 3.26e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.537 on 13 degrees of freedom
Multiple R-squared:  0.9377,    Adjusted R-squared:  0.9329
F-statistic: 195.7 on 1 and 13 DF,  p-value: 3.259e-09
```

The value of t test statistics for the coefficient on birthrate is -13.99. P-value of this test is much less than 0.05. Therefore we reject this null hypothesis. The coefficient on birthrate should be negative.

f) Give the ANOVA table for this regression model. Give some explanations to this ANOVA table.

anova(lm.cntry)

```
> anova(lm.cntry)
Analysis of Variance Table

Response: lifeexpf
          Df  Sum Sq Mean Sq F value    Pr(>F)    
birthrat   1 1259.26 1259.26  195.65 3.259e-09 ***
Residuals 13   83.67    6.44                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The sum of squares for residuals is 83.67. The sum of squares for the regression is 1259.26.Therefore, most variability of female life expectancy has been explained by birthrate. F-test shows that this model is meaningful.

g) Create a data frame which contains actual value of life expectancy, fitted value of life expectancy and residuals obtained from this simple regression.

```
residuals.y<-residuals(lm.cntry)
predict.y<-fitted(lm.cntry)
result<-data.frame(lifeexpf,predict.y,residuals.y)
```

Alternative:

```
summary(lm.cntry)$residuals
```

```
> result
   lifeexpf predict.y residuals.y
1        55  57.90856 -2.90856201
2        55  55.11929 -0.11929156
3        59  56.51393  2.48607321
4        56  58.60588 -2.60587963
5        68  68.36833 -0.36832621
6        63  58.60588  4.39412037
7        53  55.11929 -2.11929156
8        79  77.43346  1.56654482
9        72  70.46028  1.53972095
10       72  73.24955 -1.24954950
11       68  66.27637  1.72362663
12       71  76.03882 -5.03881996
13       72  70.46028  1.53972095
14       82  80.92004  1.07995675
15       81  80.92004  0.07995675
```

h) Determine the 95% confidence interval for the mean value of life expectancy when value of birthrate is 40.

i) Determine the 95% prediction interval for the life expectancy when value of birthrate is 40.

```
newdata<-data.frame(birthrat=40)
conf.inv<-predict(lm.cntry,newdata,interval="confidence")
```

```
> conf.inv
       fit      lwr      upr
1 62.09247 60.48224 63.70269
```

```
pred.inv<-predict(lm.cntry,newdata,interval="predict")
```

```
> pred.inv
       fit      lwr     upr
1 62.09247 56.38004 67.8049
```

SAVE

```
save.image() # Save contents of workspace, into the file .RData

save.image(file="archive.RData")
# Save into the file archive.RData

save(celsius, fahrenheit, file="tempscales.RData")
```