CHAPTER 14

AN INTRODUCTION TO R PROGRAMMING

J. Jayasankar, T. V. Ambrose and R. Manjeesh

Fishery Resources Assessment Division
ICAR-Central Marine Fisheries Research Institute

Introduction

R language is the GNU arm of S language, which has taken the computational world by storm in the last decade. Starting as a compendium of statistical tools, this language has grown up into a canopy lording over a research analysis environment thereby subsuming many hitherto complicated manoeuvres onto the realms of syntactical simplicity. As this an exponentially expanding field of development with ever exploding information downpour, it would be a near impossible task to frame it onto a short simple foundational discourse. However in the subsequent sections we would try to view the potential and the extent of practicality we would unravel the hidden features of the software through a GUI envelop also apart from the regular console and syntax based one. To get its power more understandable we would visualize its forays into the field of analytics using medium scale examples from marine fisheries data.

- R is "GNU S" — A language and environment for data manipulation, calculation and graphical display.
  - R is similar to the award-winning S system, which was developed at Bell Laboratories by John Chambers *et al.*,
  - a suite of operators for calculations on arrays, in particular matrices,
  - a large, coherent, integrated collection of intermediate tools for interactive data analysis,
  - graphical facilities for data analysis and display either directly at the computer or on hardcopy
  - a well developed programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- The core of R is an interpreted computer language.
  - It allows branching and looping as well as modular programming using functions.
  - Most of the user-visible functions in R are written in R, calling upon a smaller set of internal primitives.

It is possible for the user to interface to procedures written in C, C++ or FORTRAN languages for efficiency, and also to write additional primitives.

R, S and S-plus- a brief time line

- S: an interactive environment for data analysis developed at Bell Laboratories since 1976
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- Exclusively licensed by *AT&T/Lucent* to *Insightful Corporation*, Seattle WA. Product name: "S-plus".
- Implementation languages C, Fortran.
- See: *http://cm.bell-labs.com/cm/ms/departments/sia/S/history.html*
- R: initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of University of Auckland, New Zealand during 1990s.
- Since 1997: international "R-core" team of ca. 15 people with access to common CVS archive.

What R does and does not

- data handling and storage: numeric, textualo matrix algebrao
- hash tables and regular expressions
- high-level data analytic and statistical functions
- classes (Object Oriented "OO")
- graphics
- programming language: loops, branching, subroutines

- is not a database, but connects to DBMSs
- has no graphical user interfaces, but connects to Java, TclTko
- language interpreter can be very slow, but allows to call own C/C++ code
- no spreadsheet view of data, but connects to Excel/MsOffice
- no professional / commercial support

R and statistics

- Packaging: a crucial infrastructure to efficiently produce, load and keep consistent software libraries from (many) different sources / authors, which are updated at a best possible refresh rate

- Statistics: most packages deal with statistics and data analysis and there are many conduit and value addition libraries which augment the statistical inference
- State of the art: many statistical researchers provide their methods as R packages

Statistical Analysis

Data Analysis and Presentation happen to be the core strength of R software environment and the ease with which this is performed makes the environment as the ultimate winner. Faster computational routines and amenability of access and modification to interim steps and results makes the programming environment a winner.

- The R distribution contains functionality for large number of statistical procedures.
  - linear and generalized linear models
  - nonlinear regression models
  - time series analysis
  - classical parametric and nonparametric tests
  - clustering
  - smoothing
- R also has a large set of functions which provide a flexible graphical environment for creating various kinds of data presentations.

References For R

- The basic reference is The New S Language: A Programming Environment for Data Analysis and Graphics by Richard A. Becker, John M. Chambers and Allan R. Wilks (the "Blue Book") .
- The new features of the 1991 release of S (S version 3) are covered in Statistical Models in S edited by John M. Chambers and Trevor J. Hastie (the "White Book").
- Classical and modern statistical techniques have been implemented.
  - Some of these are built into the base R environment.
  - Many are supplied as packages. There are about 8 packages supplied

with R (called "standard" packages) and many more are available through the cran family of Internet sites (via http://cran.r-project.org).

- All the R functions have been documented in the form of help pages in an "output independent" form which can be used to create versions for HTML, LATEX, text *etc*.

– The document "An Introduction to R" provides a more user-friendly starting point.

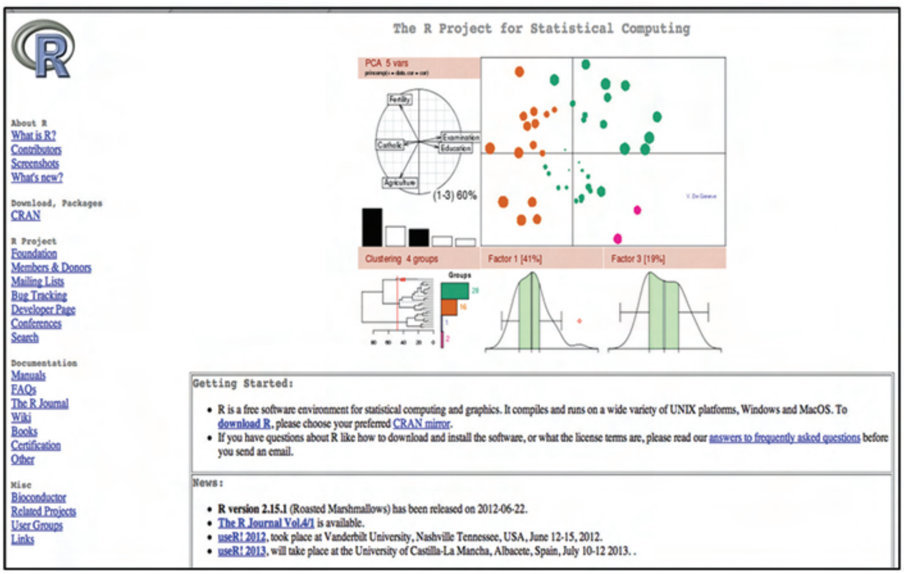
– An "R Language Definition" manual

– More specialized manuals on data import/export and extending R.

R installations

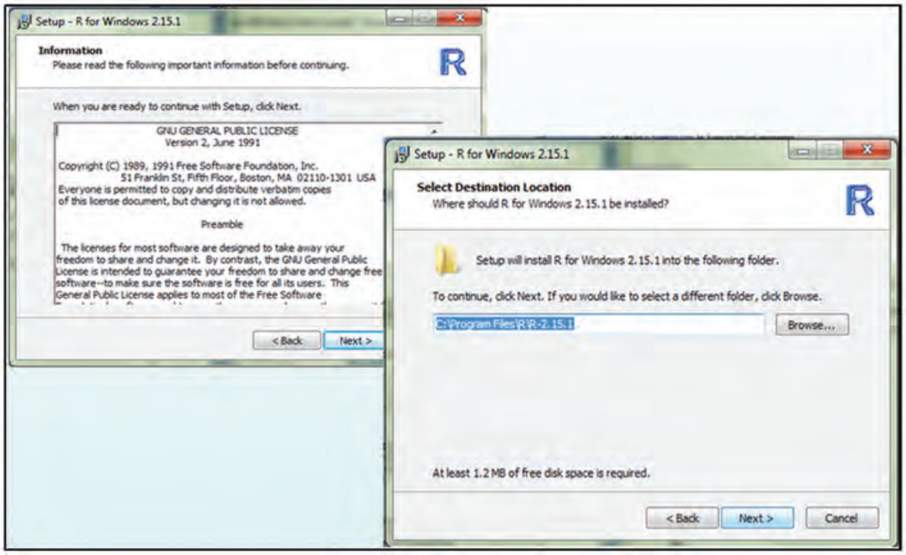
Getting Started

To install R on your MAC or PC the starting point has to be *http://www.r-project.org/*.

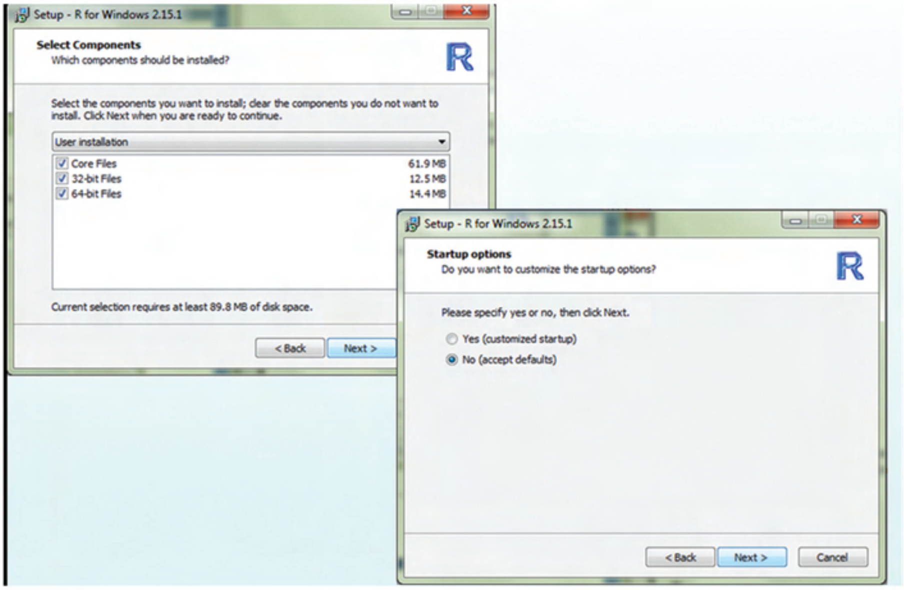


Depending on the choice of operating system the installer/ zip file with checksum may be downloaded and verified.

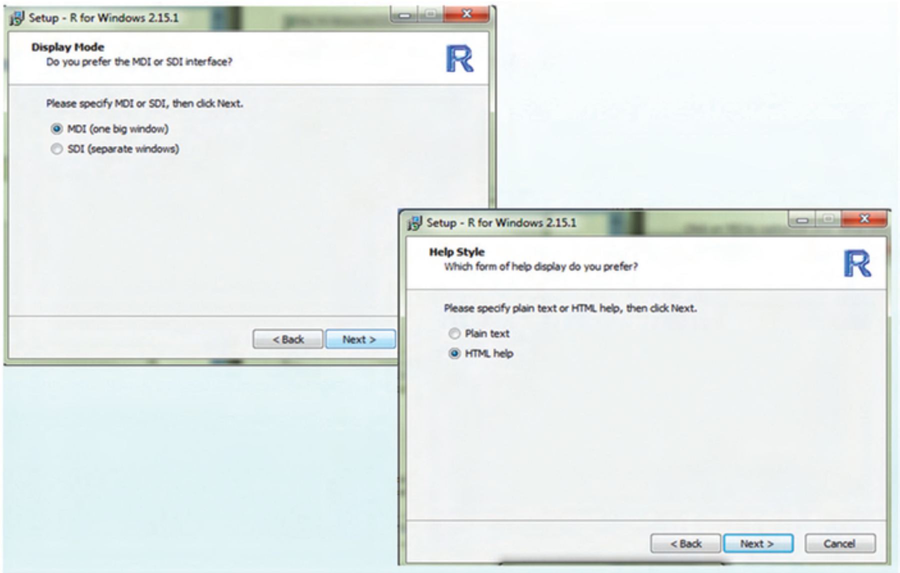
An effort to download R for Windows would have the following sequence of interactions with the portal, whose snapshots are given below:



Its always a good idea to download all the files.



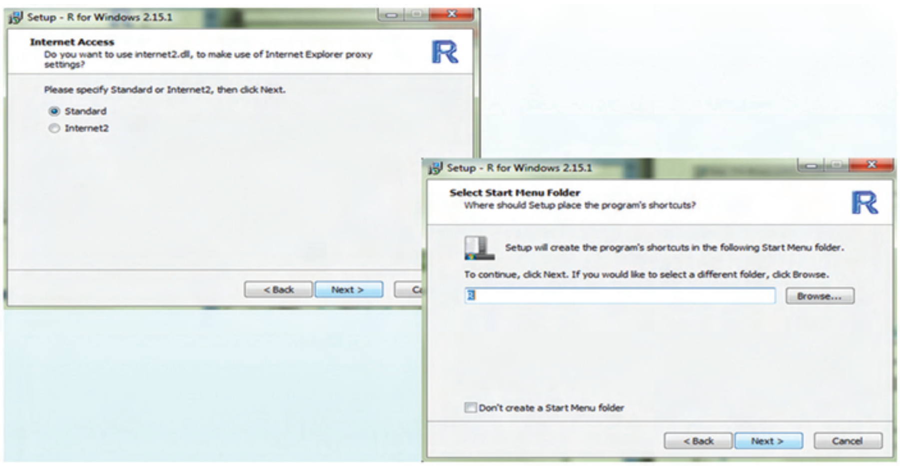
MDI is when the windows will be contained within one large window.

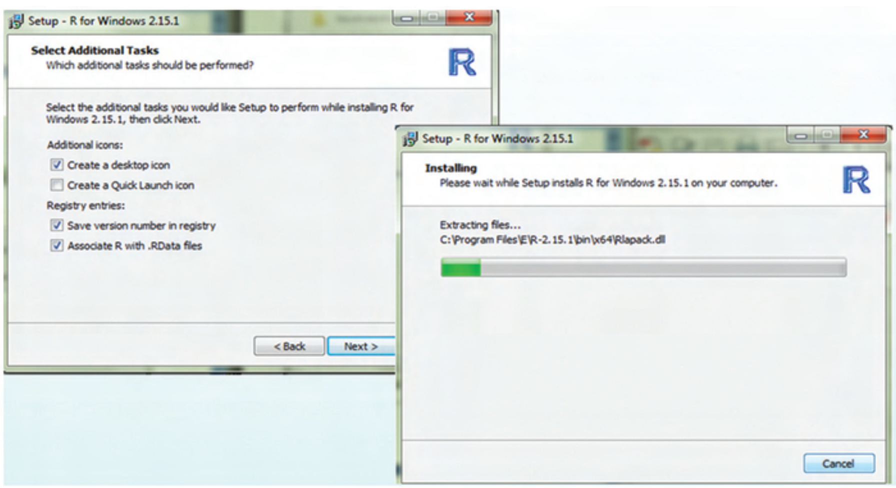


This is similar to how Excel is setup. SDI is a single document interface where every item will get its own window. This is similar to how SPSS is set up where it has separate data editor, viewer, and syntax windows. Once you choose which your prefer, click next.

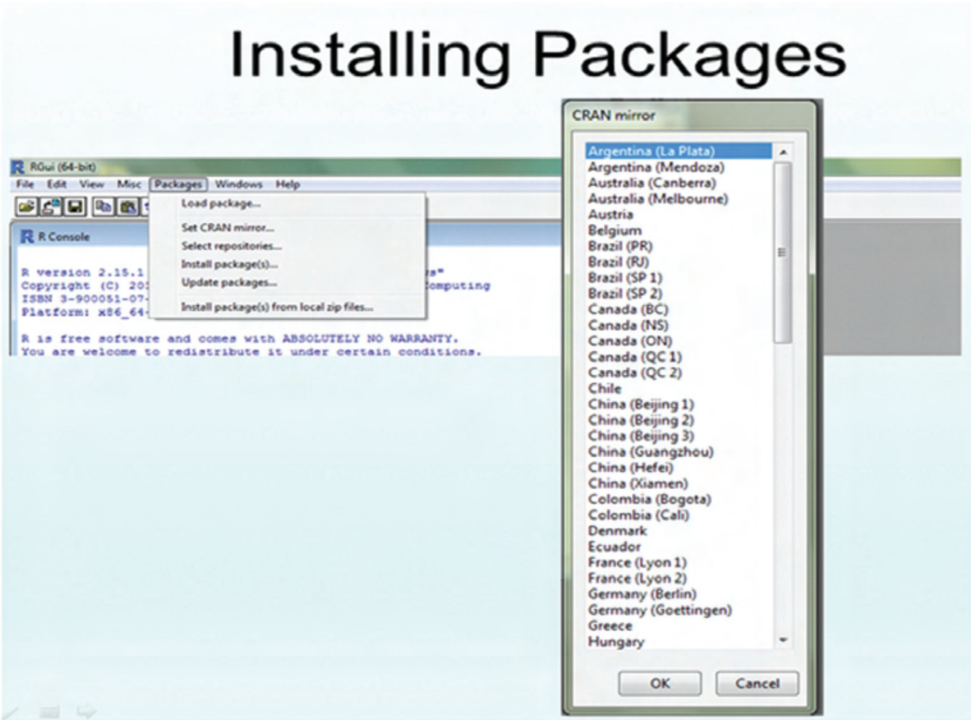
Choosing either html or plain text and clicking is the next step.

The installation may take awhile





To install packages on Windows, clicking on packages and install packages will be the next step.



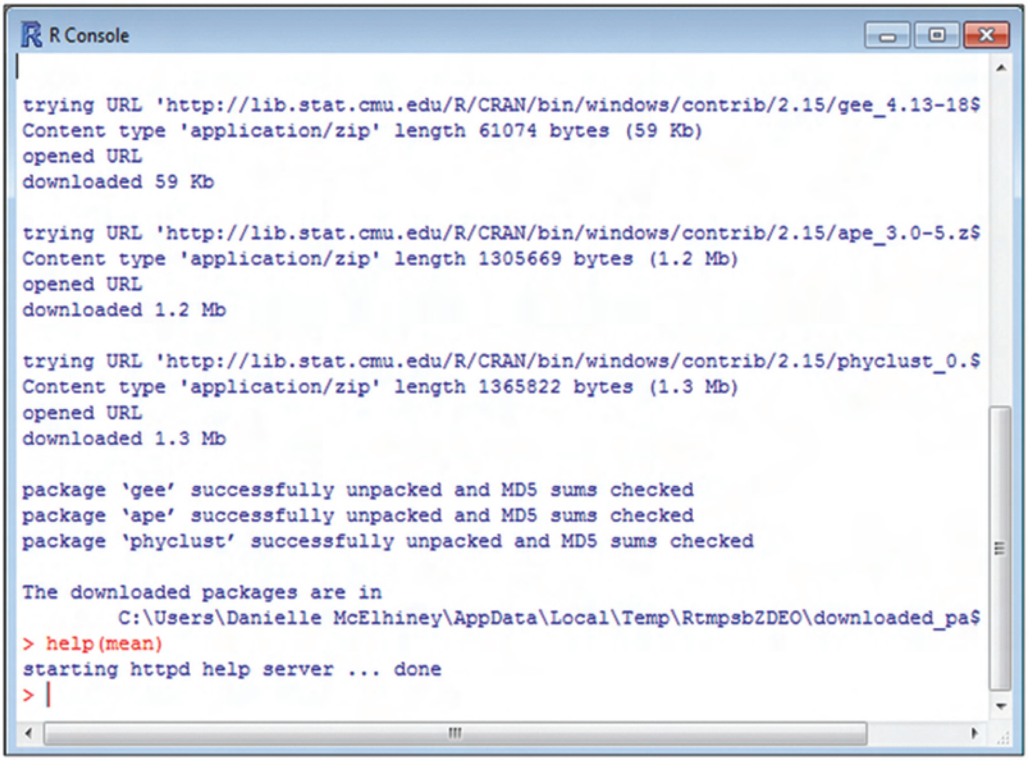
Scrolling down to country nearest and choosing a "mirror" that is close is the next step.

Scrolling down list until the requisite package is the next step, keeping in mind that R lists things in alphabetical order and by uppercase than lowercase. Once a package is clicked to load, R will install not only the package but all of the packages needed to run the package, including the dependencies.

To actually use the package, one has to go back to the package tab and click on load package.

Using Help Command

?solve translates on to giving details of help information about “solve” function whilst help.search or ?? allows searching for help in various ways.

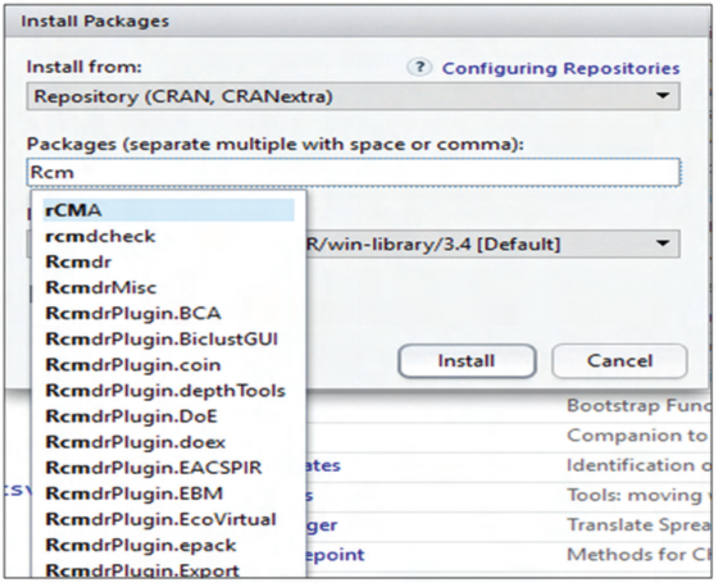


R Commander – A graphical interaction "skin" for R

R provides a powerful and comprehensive system for analysing data and when used in conjunction with the R-commander (a graphical user interface, commonly known as Rcmdr) it also provides one that is easy and intuitive to use. Basically, R provides the engine that carries out the analyses and Rcmdr provides a convenient way for users to input commands. The Rcmdr program enables analysts to access a selection of commonly-used R commands using a simple interface that should be familiar to most computer users. It also serves the important role of helping users to implement R commands and develop their knowledge and expertise in using the command line — an important skill for those wishing to exploit the full power of the program.(*http://www.rcommander.com/*)

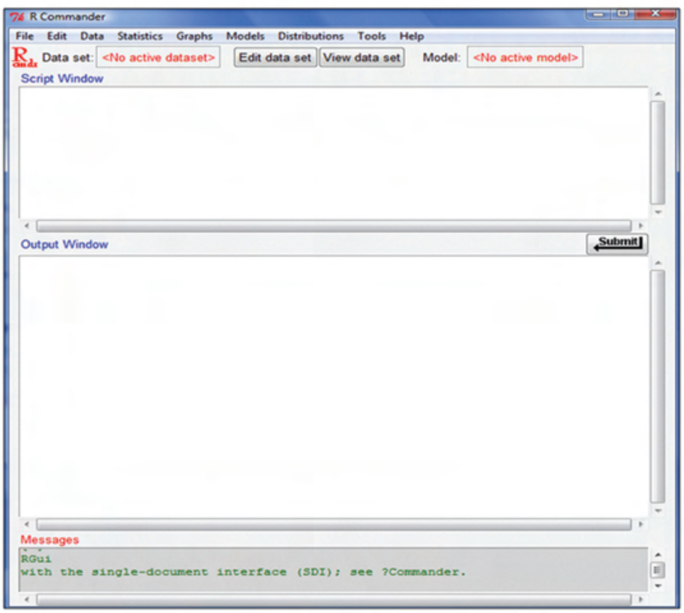
a) **Loading R Commander**

– Packages -> Install Packages -> Cran Mirror Selection -> Rcmdr



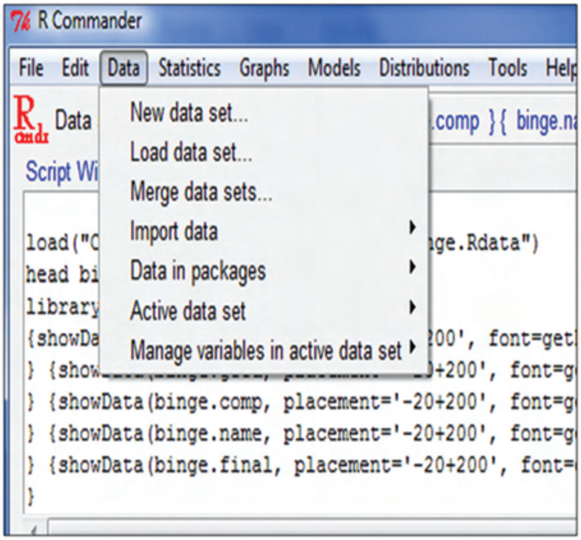
b) Opening R Commander

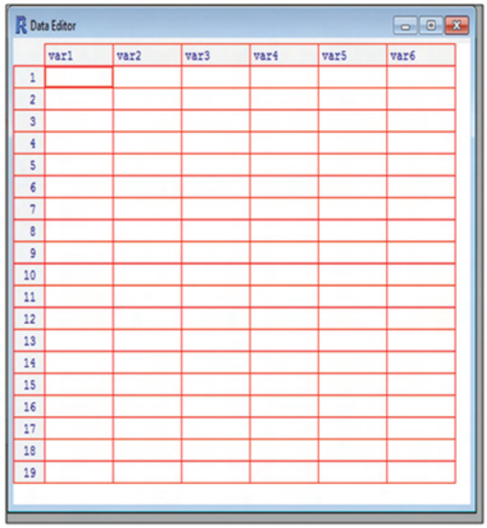
Open R -> Packages -> Load Packages -> Rcmdr



c) Loading Data

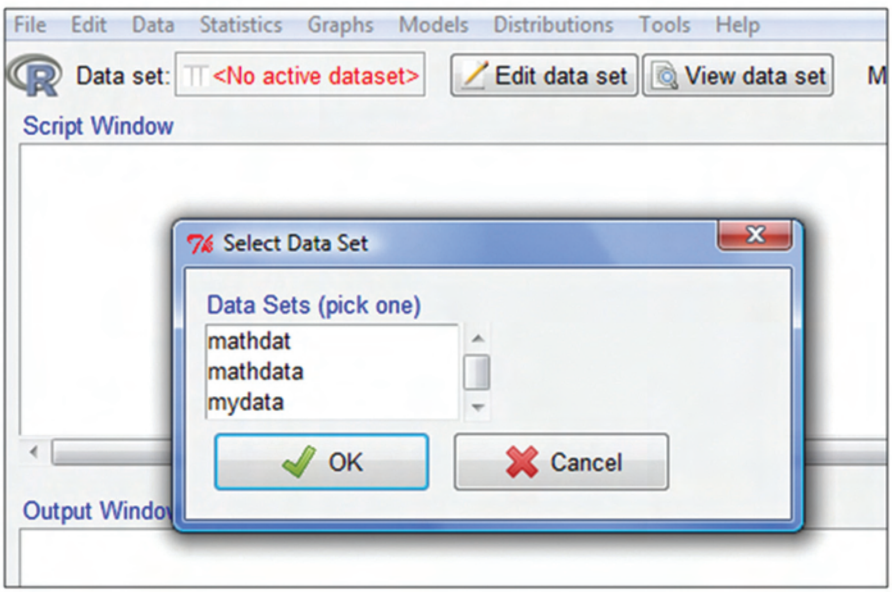
Data->Load data





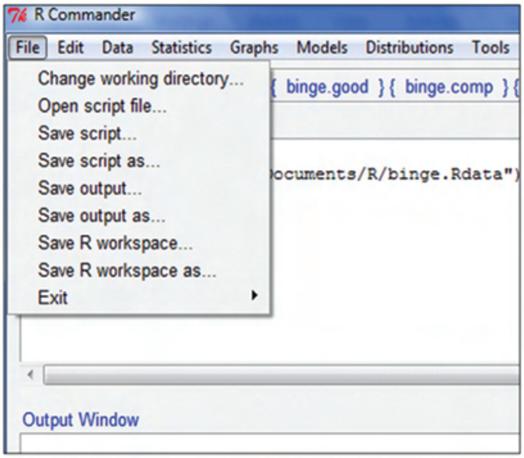
d) Active Data selection

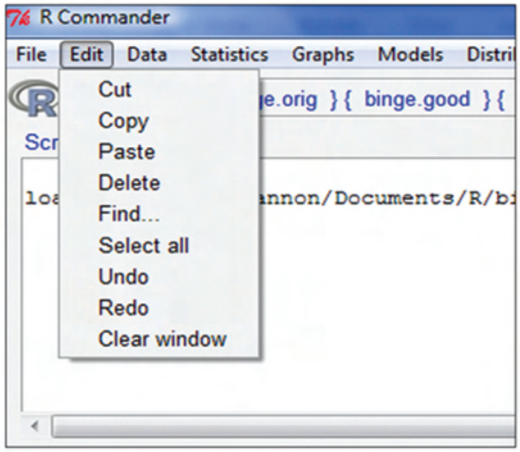
Data ->Active data set -> Select active data set



e) Menu driven File edit options

Script will save it as an R file .R and Output will save it as a text file. .txt

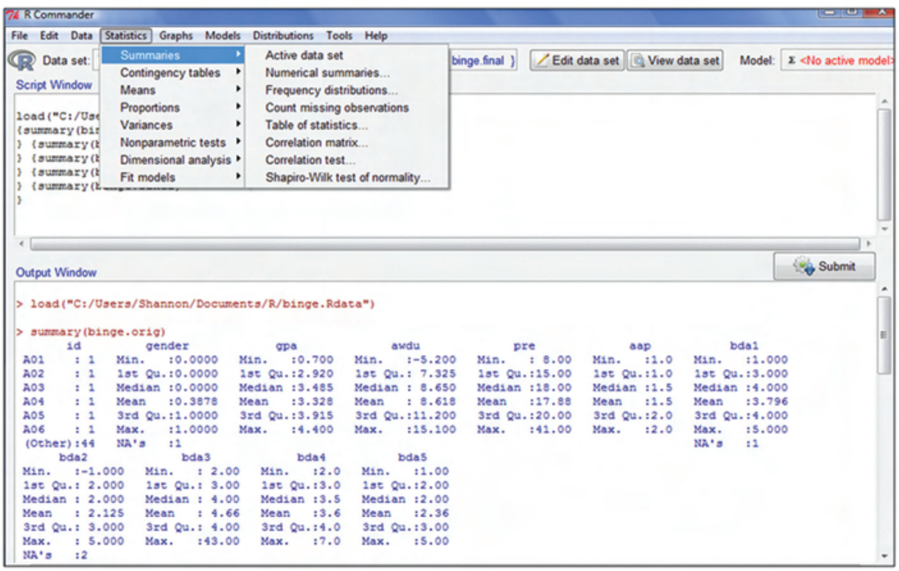




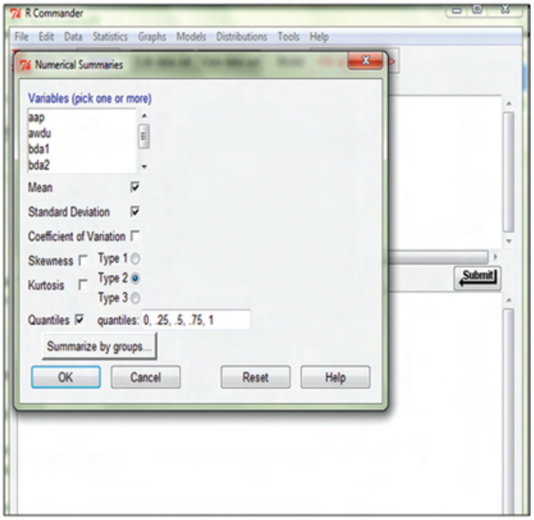
f) Summary of the data

Statistics -> Summaries

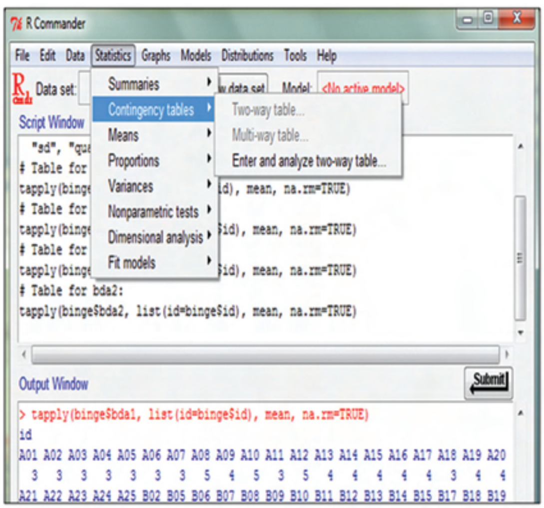
Numerical Summeries – can also provide mean, standard deviation, skewness, kurtosis *etc.*

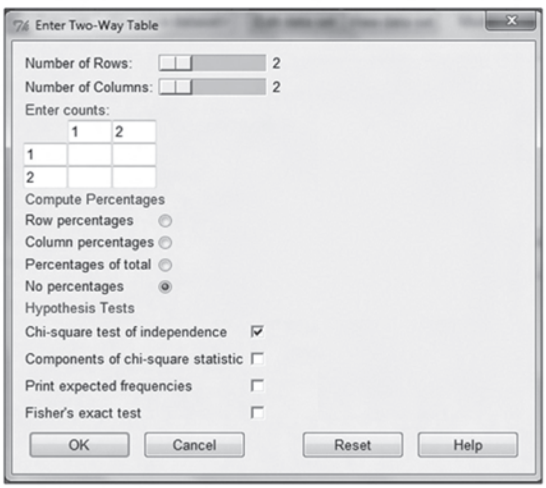


g) Mean, Standard Deviation, Skewness, Kurtosis



h) Contingency Tables

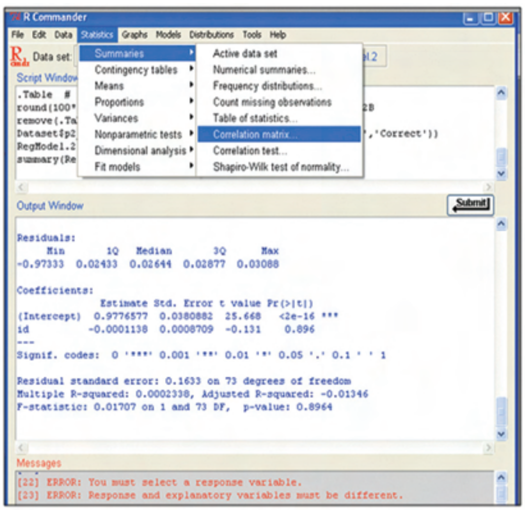


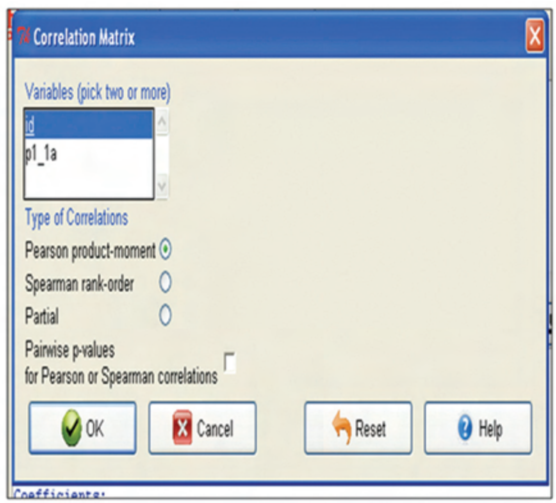


i) Correlations in R Commander

Correlation analysis can be done with R as follows.

Correlation is a bivariate analysis that measures the strengths of association between two variables and the direction of the relationship. In terms of the strength of relationship, the value of the correlation coefficient varies between +1 and -1. When the value of the correlation coefficient lies around ± 1, then it is said to be a perfect degree of association between the two variables. As the correlation coefficient value goes towards 0, the relationship between the two variables will be weaker. the direction of the relationship is simply the + (indicating a positive relationship between the variables) or - (indicating a negative relationship between the variables) sign of the correlation. Usually, in statistics, we measure four types of correlations: Pearson Correlation, Kendall rank correlation, Spearman correlation, and the Point-Biserial correlation. The software below allows you to very easily conduct a correlation.

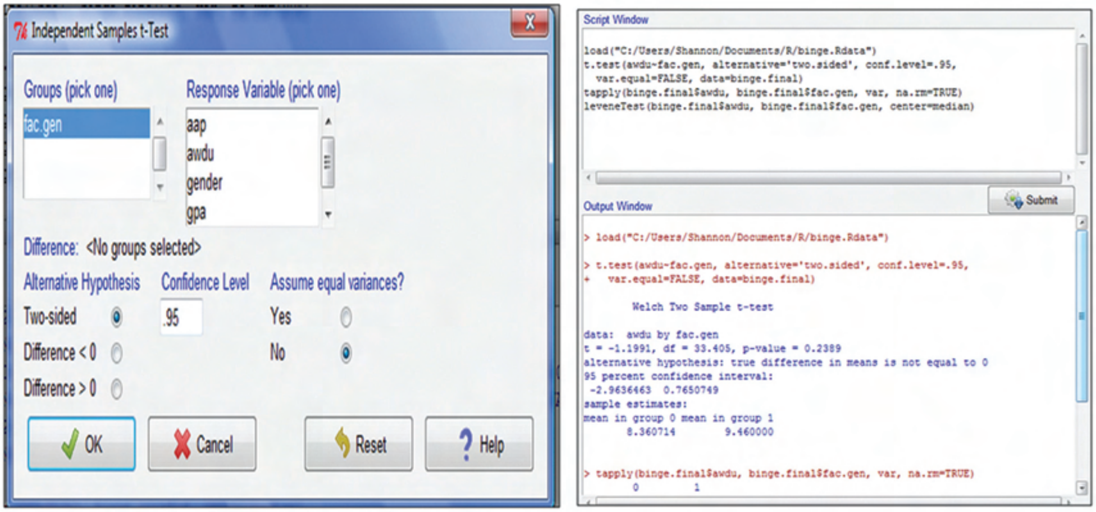




j) Independent T-Test

The independent t-test, also referred to as an independent-samples t-test, independent measures t-test or unpaired t-test, is used to determine whether the mean of a dependent variable (e.g., weight, anxiety level, salary, reaction time, *etc.*) is the same in two unrelated, independent groups (e.g., males vs females, employed vs unemployed, under 21 year olds vs those 21 years and older, *etc.*). Specifically, you use an independent t-test to determine whether the mean difference between two groups is statistically significantly different to zero.

Statistics->Independent T Test



k) One Way ANOVA

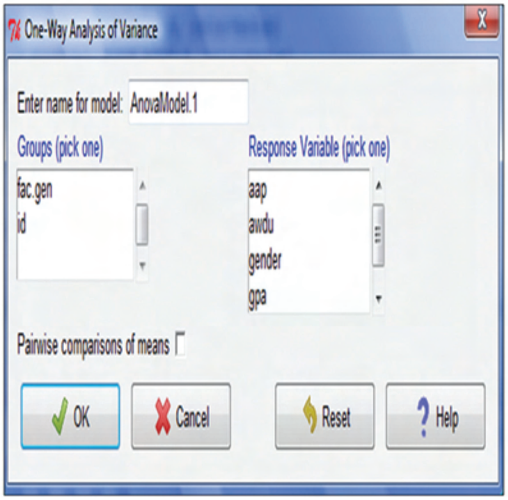
ANOVA(Analysis of Variance) is a statistical technique that assesses potential differences in a scale-level dependent variable by a nominal-level variable having 2 or more categories. For example, an ANOVA can examine potential differences in IQ scores by Country (US vs. Canada vs. Italy vs. Spain). The ANOVA, developed by Ronald Fisher in 1918, extends the *t* and the *z* test which have the problem of only allowing the nominal level variable to have just two categories. This test is also called the Fisher analysis of variance. ANOVAs are used in three ways: one –way Anova, two-way ANOVA, and N-way Multivariate ANOVA.

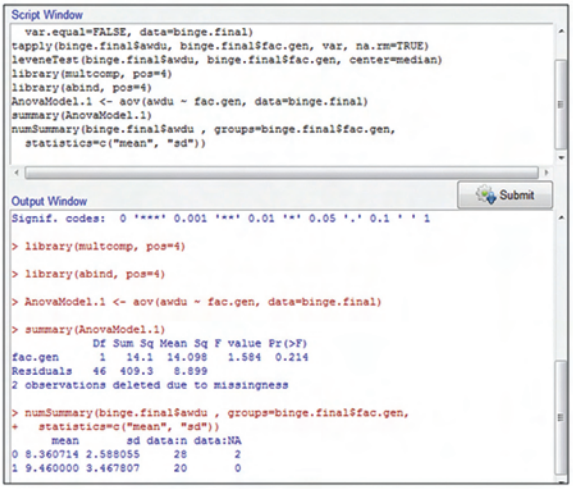
One-Way ANOVA

A one-way ANOVA refers to the number of independent variables—not the number of categories in each variables. A one-way ANOVA has just one independent variable. For example, difference in IQ can be assessed by Country, and County can have 2, 20, or more different Countries in that variable.

The software below allows you to easily conduct an ANOVA.

Statistics->One Way ANOVA





l) Factor Analysis

Factor analysis is a technique that is used to reduce a large number of variables into fewer numbers of factors. This technique extracts maximum common variance from all variables and puts them into a common score. As an index of all variables, we can use this score for further analysis. Factor analysis is part of general linear model(GLM) and this method also assumes several assumptions: there is linear relationship, there is no multicollinearity, it includes relevant variables into analysis, and there is true correlation between variables and factors. Several methods are available, but principal component analysis is used most commonly.

Types of factoring:

There are different types of methods used to extract the factor from the data set:

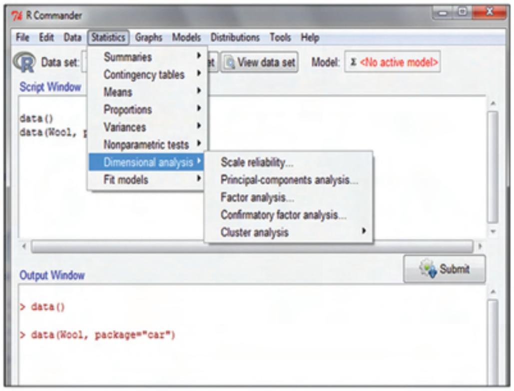
1. **Principal component analysis**: This is the most common method used by researchers. PCA starts extracting the maximum variance and puts them into the first factor. After that, it removes that variance explained by the first factors and then starts extracting maximum variance for the second factor. This process goes to the last factor.

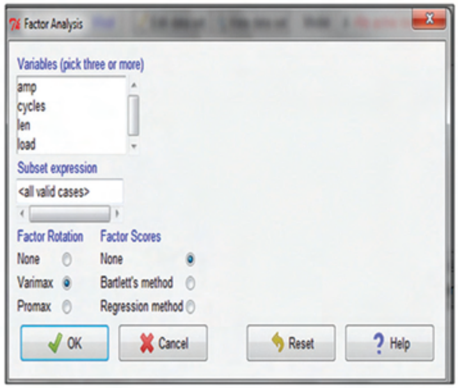
2. **Common factor analysis:** The second most preferred method by researchers, it extracts the common variance and puts them into factors. This method does not include the unique variance of all variables. This method is used in SEM.

3. **Image factoring:** This method is based on correlation matrix. OLS Regression method is used to predict the factor in image factoring.

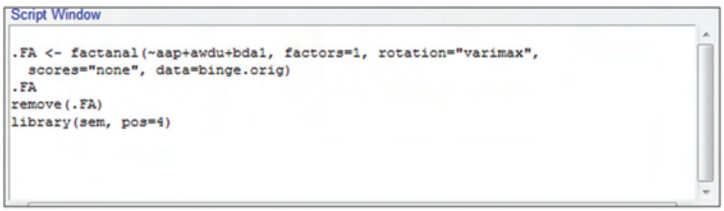
4. **Maximum likelihood method:** This method also works on correlation metrix but it uses maximum likelihood method to factor.

5. **Other methods of factor analysis:** Alfa factoring outweighs least squares. Weight square is another regression based method which is used for factoring.





Result are shown as follows



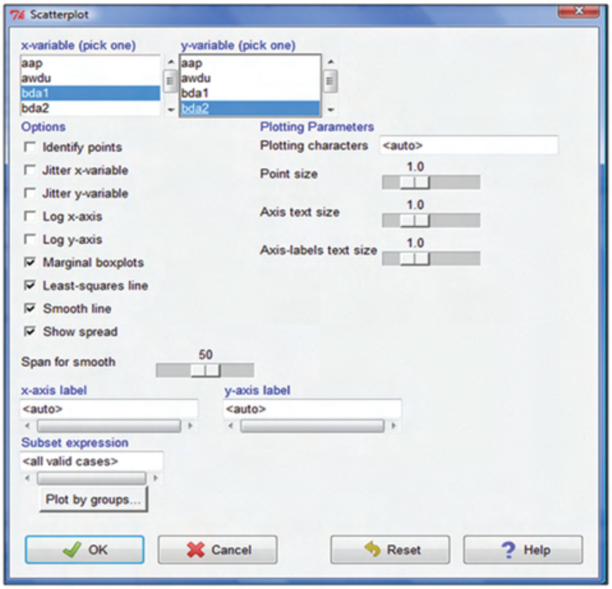
```
Script Window
.FA <- factanal(~aap+awdu+bdal, factors=1, rotation="varimax",
  scores="none", data=binge.orig)
.FA
remove(.FA)
library(sem, pos=4)
```

```
> .FA <- factanal(~aap+awdu+bdal, factors=1, rotation="varimax",
+   scores="none", data=binge.orig)

> .FA

Call:
factanal(x = ~aap + awdu + bdal, factors = 1, data = binge.orig,     scores = "none", rotation = "varimax")

Uniquenesses:
  aap  awdu  bdal
0.849 0.324 0.596

Loadings:
     Factor1
aap  0.388
awdu 0.822
bdal 0.636
```

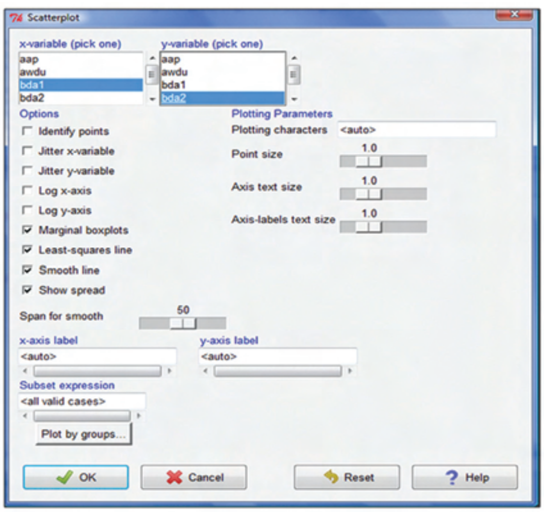
```
               Factor1
SS loadings      1.231
Proportion Var   0.410

The degrees of freedom for the model is 0 and the fit was 0

> remove(.FA)

> library(sem, pos=4)
```

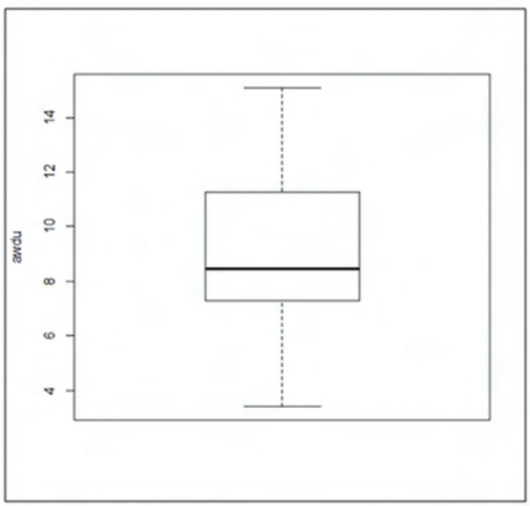
J) Graphs

Gparhs->Scatter plot



Gparhs->Box plot





R Basics

R is object base

Types of objects (scalar, vector, matrices and arrays Assignment of objects)

Building a data frame

Operation Symbols

Symbol	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%%	**Modulo (estimates remainder in a division)**
^	Exponential

R as a Calculator

1550+2000

[1] 3550

or various calculations in the same row

2+3; 5*9; 6-6

[1] 5

[1] 45

[1] 0

As Mathematics

1+1

[1] 2

2+2*7

[1] 16

(2+2)*7

[1] 28

As Variables

x<-2

x

[1] 2

```
y<-3
y
## [1] 3
5->z
(x*y)+z
## [1] 11
```

Numbers in R: NAN and NA

NAN (not a number) NA (missing value) -Basic handling of missing values

Missing values are noise to statistical estimations. We are going to learn a basic command for handling missing values.

```
x<-c(1,2,3,4,5,6,NA)
mean(x)
## [1] NA
mean(x,na.rm=TRUE)
## [1] 3.5
```

Objects in R

Objects in R obtain values by assignment.

This is achieved by the gets arrow, <-, and not the equal sign, =.

Objects can be of different kinds.

Built in Functions

R has many built in functions that compute different statistical procedures.

Functions in R are followed by ( ). Inside the parenthesis we write the object (vector, matrix, array, dataframe) to which we want to apply the function.

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))
##  [1] 32 33 34 35 36 37 38 39 40 41 42 43 44
# Find mean of numbers from 25 to 82.
print(mean(25:82))
## [1] 53.5
# Find sum of numbers frm 41 to 68.
```

```
print(sum(41:68))
## [1] 1526
```

Vectors

Vectors are variables with one or more values of the same type.

A variable with a single value is known as scalar. In R a scalar is a vector of length 1. There are at least three ways to create vectors in R: (a) sequence, (b) concatenation function, and (c) scan function.

Create two vectors of different lengths.

```
vector1 <- c(5,9,3)

vector2 <- c(10,11,12,13,14,15)

vector1

## [1] 5 9 3

vector2

## [1] 10 11 12 13 14 15
```

Arrays

Arrays are numeric objects with dimension attributes. The difference between a matrix and an array is that arrays have more than two dimensions.

```
# Take the above  vectors as input to the array.

result <- array(c(vector1,vector2),dim = c(3,3,2))

print(result)

## , , 1
##
##        [,1]   [,2]   [,3]
##  [1,]    5     10     13
##  [2,]    9     11     14
##  [3,]    3     12     15
##
```

```
## , , 2
##
##        [,1]  [,2]  [,3]
## [1,]     5    10    13
## [2,]     9    11    14
## [3,]     3    12    15
```

Matrices

A matrix is a two dimensional array. The command colnames

```
# Elements are arranged sequentially by row.
M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(M)
##        [,1]  [,2]  [,3]
## [1,]     3     4     5
## [2,]     6     7     8
## [3,]     9    10    11
## [4,]    12    13    14
```

String Characters

In R, string variables are defined by double quotation marks.

```
letters<-c("a","b","c")
letters
## [1] "a" "b" "c"
```

Subscripts and Indices

Select only one or some of the elements in a vector, a matrix or an array. We can do this by using subscripts in square brackets [ ].

In matrices or dataframes the first subscript refers to the row and the second to the column.

Dataframe

Researchers work mostly with dataframes. With previous knowledge you can built dataframes in R. Also, import dataframes into R.

```
# Create the data frame.
emp.data <- data.frame (
emp_id = c (1:5),
emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
salary = c(623.3,515.2,611.0,729.0,843.25),
 start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
"2015-03-27")),
stringsAsFactors = FALSE
)
# Print the data frame.
print(emp.data)
##    emp_id  emp_name   salary   start_date
## 1  1       Rick       623.30   2012-01-01
## 2  2       Dan        515.20   2013-09-23
## 3  3       Michelle   611.00   2014-11-15
## 4  4       Ryan       729.00   2014-05-11
## 5  5       Gary       843.25   2015-03-27
```

A journey wading through the amazing summarizing and analytical capabilities of R- a case study

Let the presumed data pertain to landings and standardized effort of a maritime state estimated by ICAR-CMFRI during the interregnum 1997 to 2013

calling file in R

```
klm<-read.csv("C:/Users/cmfri/Desktop/cpue_spcode_kldata.csv",header=TRUE)
```

To know header portion of the data set

```
head(klm)
##     year   month   species   raised     nomeff     stdcpue
## 1   1997   1       40        20595.35   122.0811   3.634042
## 2   1997   2       40        24201.10   114.3719   4.532246
## 3   1997   3       40        23497.64   255.0315   3.926130
```

```
## 4      1997   4    40   50176.75   154.7663   6.762821
## 5      1997   5    40   137626.24  314.6413   13.805531
## 6      1997   6    40   38149.38   649.1328   16.071358
```

To check the last few rows of the dataset

tail (klm)

```
##          year  month  species  raised  nomeff    stdcpue
## 245815   2013  7      4580     0       0.000000  0.000000
## 245816   2013  8      4580     1674    2.059835  1.667304
## 245817   2013  9      4580     0       0.000000  0.000000
## 245818   2013  10     4580     0       0.000000  0.000000
## 245819   2013  11     4580     0       0.000000  0.000000
## 245820   2013  12     4580     0       0.000000  0.000000
```

to know the observations in the data

length(klm)

```
## [1] 6
```

to know the structure of the dataframe

str(klm)

```
## 'data.frame':    245820 obs. of  6 variables:
##  $ year   : int  1997 1997 1997 1997 1997 1997 1997 1997 1997 1997 ...
##  $ month  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ species: int  40 40 40 40 40 40 40 40 40 40 ...
##  $ raised : num  20595 24201 23498 50177 137626 ...
##  $ nomeff : num  122 114 255 155 315 ...
##  $ stdcpue: num  3.63 4.53 3.93 6.76 13.81 ...
```

Descriptive statistics analysis

summary(klm)

```
##      year         month          species        raised
##  Min.   :1997   Min.   : 1.00   Min.   :   0   Min.   :      0
##  1st Qu.:2001   1st Qu.: 3.75   1st Qu.: 867   1st Qu.:  0
##  Median :2005   Median : 6.50   Median :1513   Median :  0
```

```
## Mean   :2005   Mean   : 6.50   Mean   :2201   Mean   :   42699
## 3rd Qu.:2009   3rd Qu.: 9.25   3rd Qu.:4016   3rd Qu.:        0
## Max.   :2013   Max.   :12.00   Max.   :9999   Max.   :71536031
##                                               NA's   :30
##      nomeff           stdcpue
## Min.   :      0.0   Min.   :    0.000
## 1st Qu.:      0.0   1st Qu.:    0.000
## Median :      0.0   Median :    0.000
## Mean   :    154.2   Mean   :    7.112
## 3rd Qu.:      0.0   3rd Qu.:    0.000
## Max.   :119100.1   Max.   :5600.000
##
```

If further enhanced list of summary statistics information about the data like third and fourth order moments, then the describe function of psych or summary function would come in handy.

```
library(psych)
describe(klm[,3:6])
##         vars      n     mean        sd median trimmed     mad min
## species    1 245820  2201.15   1951.83   1513 1941.16 1257.24   0
## raised     2 245790 42699.02 719150.48      0   62.52    0.00   0
## nomeff     3 245820   154.25   1543.66      0    0.16    0.00   0
## stdcpue    4 245820     7.11     52.38      0    0.11    0.00   0
##                max      range  skew kurtosis      se
## species     9999.0     9999.0  1.40     1.91    3.94
## raised  71536030.7 71536030.7 44.70  2681.18 1450.57
## nomeff    119100.1   119100.1 22.83   770.70    3.11
## stdcpue     5600.0     5600.0 21.65   971.06    0.11
```

If one wants to study monthly catch grouped information so that an idea about issues like which month (used as a group) would have etched up maximum landings/ catch, then simple literally rooted commands like describeBy (psych) or aggregate would come in handy.

```
library(psych)
describeBy(klm$raised,klm$month)
##
##  Descriptive statistics by group
## group: 1
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20485 41379.48 784622.6      0  146.65   0   0 51193526 51193526
##     skew kurtosis      se
## X1 46.55  2497.42 5482.05
## ------------------------------------------------------------------------
## group: 2
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20485 32904.06 535506.3      0  113.45   0   0 45468199 45468199
##     skew kurtosis      se
## X1 49.62  3259.68 3741.51
## ------------------------------------------------------------------------
## group: 3
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20485 39087.37 569052.1      0  162.51   0   0 31762665 31762665
##    skew kurtosis      se
## X1 38.4  1796.15 3975.89
## ------------------------------------------------------------------------
## group: 4
##    vars     n     mean     sd median trimmed mad min      max    range
## X1    1 20471 33795.18 477389      0   64.13   0   0 31931384 31931384
##     skew kurtosis      se
## X1 42.59  2353.01 3336.59
## ------------------------------------------------------------------------
```

```
## group: 5
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20485 37566.67 469275.5      0    96.2   0   0 30492626 30492626
##     skew kurtosis      se
## X1 33.18  1478.99 3278.76
## ------------------------------------------------------------
## group: 6
##    vars     n    mean       sd median trimmed mad min      max    range
## X1    1 20485 34552.2 655525.6      0   30.67   0   0 65432961 65432961
##     skew kurtosis      se
## X1 61.23  5239.89 4580.07
## ------------------------------------------------------------
## group: 7
##    vars     n    mean       sd median trimmed mad min      max    range
## X1    1 20485 32621.2 643003.1      0       0   0   0 49428947 49428947
##     skew kurtosis      se
## X1 42.19  2362.03 4492.57
## ------------------------------------------------------------
## group: 8
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20484 57397.86 713381.8      0   31.03   0   0 38795185 38795185
##     skew kurtosis      se
## X1 26.21   920.16 4984.42
## ------------------------------------------------------------
## group: 9
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20485 55833.65 901880.9      0    34.3   0   0 71536031 71536031
##     skew kurtosis      se
## X1 41.11  2415.63 6301.32
## ------------------------------------------------------------
```

```
## group: 10
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20484 57071.88 915432.9      0   89.05   0   0 55973676 55973676
##     skew kurtosis      se
## X1 34.05  1453.38 6396.16
## ------------------------------------------------------------
## group: 11
##    vars     n     mean     sd median trimmed mad min      max    range
## X1    1 20485 51210.52 915220      0  133.56   0   0 49127745 49127745
##     skew kurtosis      se
## X1 36.33  1488.92 6394.51
## ------------------------------------------------------------
## group: 12
##    vars     n     mean       sd median trimmed mad min      max    range
## X1    1 20471 38960.92 830555.4      0  134.37   0   0 66844967 66844967
##    skew kurtosis      se
## X1   56  3639.25 5804.96
```

Selecting subsets of data:

```
#to know the whole species entries
t<-klm$species
length(t)
## [1] 245820
# to know the june species entries
d<-klm$species[klm$month=="6"]
length(d)
## [1] 20485
```

to exclude some data

```
#exclude june catch and know the entries
e<-klm$species[klm$month!="6"]
```

length(e)

[1] 225335

correlation of the data

correlation between catch and effort for the whole period

attach(klm)

cor.test(raised,nomeff,method="pearson")

```
##
##  Pearson's product-moment correlation
##
## data:  raised and nomeff
## t = 434.94, df = 245790, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6572472 0.6617152
## sample estimates:
##      cor
## 0.659487
##multiple correlation
```

##Here we select the oilsardine catch.The oilsardine species code as 362

##we pick all the years monthly oil sardine

sp362<-klm[(klm$species=="362"),]

cordat<-sp362[,4:6]

cor(cordat)

```
 raised     nomeff    stdcpue
raised  1.0000000 0.45713639 0.61135090
nomeff  0.4571364 1.00000000 0.06860281
stdcpue 0.6113509 0.06860281 1.00000000
```

Linear regression & ANOVA

fit <- **lm**(raised~ year + month + nomeff, data=sp362)

show results

```
summary(fit)
##
## Call:
## lm(formula = raised ~ year + month + nomeff, data = sp362)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -24406856  -5945766   -838374   4725596  40857882
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.148e+09  2.787e+08  -7.706 5.93e-13 ***
## year         1.072e+06  1.389e+05   7.716 5.59e-13 ***
## month        7.997e+05  1.969e+05   4.062 6.97e-05 ***
## nomeff       3.997e+02  4.493e+01   8.897 3.44e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9689000 on 200 degrees of freedom
## Multiple R-squared:  0.4275, Adjusted R-squared:  0.4189
## F-statistic: 49.78 on 3 and 200 DF,  p-value: < 2.2e-16
# model coefficients
coefficients(fit)
##   (Intercept)          year         month        nomeff
## -2.147604e+09  1.072090e+06  7.997178e+05  3.997276e+02
# CIs for model parameters
confint(fit, level=0.95)
##                     2.5 %        97.5 %
## (Intercept) -2.697162e+09 -1.598046e+09
## year         7.980987e+05  1.346082e+06
## month        4.115344e+05  1.187901e+06
```

```
## nomeff       3.111348e+02  4.883205e+02
# predicted values
fitted(fit)
##          10609       10610       10611       10612       10613       10614
##    -3789651.96   -75345.54 15111313.36 13412874.31 17168949.26   120681.70
##          10615       10616       10617       10618       10619       10620
##    11475956.42  2176177.37  4491241.24 20281254.70 10248865.43  6278101.08
##          10621       10622       10623       10624       10625       10626
##     1848628.97  -945019.58 10648970.16 18599757.89  1915100.95  4945529.10
##          10627       10628       10629       10630       10631       10632
##     1844457.32  4524979.63  8480021.57 27270345.64 26410785.24  7449598.25
##          10633       10634       10635       10636       10637       10638
##     8195286.59 18056830.84 12504031.29  4797286.88   690139.61  7333241.94
##          10639       10640       10641       10642       10643       10644
##     9086615.20 12777192.22 16114211.77 21825496.12 23957847.88 30125417.82
##          10645       10646       10647       10648       10649       10650
##    16794955.21  8159428.15 18423291.70 38539644.49 22526843.37 15428828.71
##          10651       10652       10653       10654       10655       10656
##    19942372.43  8463199.11 16820433.97 16852255.88 19772511.73 16832240.83
##          10657       10658       10659       10660       10661       10662
##     6812947.52  2187489.33  3280344.12 24388104.43 18000977.41 15107404.98
##          10663       10664       10665       10666       10667       10668
##    11071325.90  8804492.99 11659447.99 15882452.30 13614255.15 14360781.30
##          10669       10670       10671       10672       10673       10674
##     4963345.25  3874425.71  8638896.83 15820079.63  9947652.94 10608928.30
##          10675       10676       10677       10678       10679       10680
##    11831223.68 10715678.08 18370843.69 18033007.59 24787443.71 20792659.27
##          10681       10682       10683       10684       10685       10686
##    10734553.89 14786524.50 23586068.72 15174415.81 14696669.45 21641645.35.88
##    26747332.20 27817053.16 27904369.27
# residuals
```

```
residuals(fit)
##        10609       10610       10611       10612       10613
##   5952459.84 12255563.09 -3371411.14 -4445741.27 -8889076.47
##        10614       10615       10616       10617       10618
##    986134.71 -5748266.48  -336390.21  2807133.26  1645172.74
##        10619       10620       10621       10622       10623
##  -3629105.70 -4577842.81  3072907.21  3243308.73 -5672890.07
##        10624       10625       10626       10627       10628
## -15696727.40   289232.12  2042122.32  1117366.99  2926082.40
##        10629       10630       10631       10632       10633
##   5230228.43 -20382271.56 -5264124.44 -5075967.51  1491577.71
##        10634       10635       10636       10637       10638
##  -9837151.49 -6712232.19  -764792.30  -437886.38  2231690.27
##        10639       10640       10641       10642       10643
##  -1443831.23 -2440345.04 14926587.99 -6794617.92  2635516.43
##        10644       10645       10646       10647       10648
## -17311907.92 -5709093.26  4952910.28 -6048902.56 -6642668.40
##        10649       10650       10651       10652       10653
##  -9406029.73 11491464.13 29486574.30  2963737.40  3482526.36
##        10654       10655       10656       10657       10658
##    764926.90  5721591.58 -8014761.85  -334238.52  5160023.79
##        10659       10660       10661       10662       10663
##   3802703.26 -10108379.25 -2107670.27 -3238790.51  6520269.00
##        10664       10665       10666       10667       10668
##   6117951.47  3707721.08  4118584.97   744008.66 -2535146.08
##        10669       10670       10671       10672       10673
##   5587891.61   247621.47 -2882708.00   800991.54  -911955.00
# anova table
anova(fit)
## Analysis of Variance Table
##
## Response: raised
```

```
##  Df  Sum SqMean Sq F valuePr(>F)
## year        1 4.6080e+15 4.6080e+15  49.083 3.663e-11 ***
## month       1 1.9813e+15 1.9813e+15  21.104 7.689e-06 ***
## nomeff      1 7.4316e+15 7.4316e+15  79.159 3.445e-16 ***
## Residuals 200 1.8776e+16 9.3882e+13
## —-
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
# covariance matrix for model parameters
vcov(fit)
##              (Intercept)          year        month        nomeff
## (Intercept)  7.767104e+16 -3.872335e+13 28849322448.9 -1.085409e+09
##        year -3.872335e+13  1.930661e+10  -132736938.4  5.147853e+05
##       month  2.884932e+10 -1.327369e+08 38753042588.4 -5.204691e+05
##      nomeff -1.085409e+09  5.147853e+05     -520469.1  2.018502e+03
# regression diagnostics
influence(fit)
## $hat
##       10609       10610       10611       10612       10613       10614
## 0.042348953 0.032174152 0.030947216 0.024014063 0.027363125 0.031587019
##       10615       10616       10617       10618       10619       10620
## 0.018101845 0.031744185 0.029944584 0.028749417 0.028915850 0.042004060
##       10621       10622       10623       10624       10625       10626
## 0.036951680 0.032836278 0.020628210 0.029105061 0.025090117 0.020127986
##       10627       10628       10629       10630       10631       10632
## 0.028928511 0.025311220 0.021317185 0.041136744 0.038894083 0.038442958
##       10633       10634       10635       10636       10637       10638
## 0.024751425 0.032951924 0.018613317 0.018864207 0.027982400 0.015391058
##       10639       10640       10641       10642       10643       10644
## 0.014401572 0.013346093 0.015061997 0.022355644 0.027879390 0.046154691
##       10645       10646       10647       10648       10649       10650
## 0.031627027 0.018558780 0.023833019 0.112821017 0.025427226 0.010871644
##       10651       10652       10653       10654       10655       10656
```

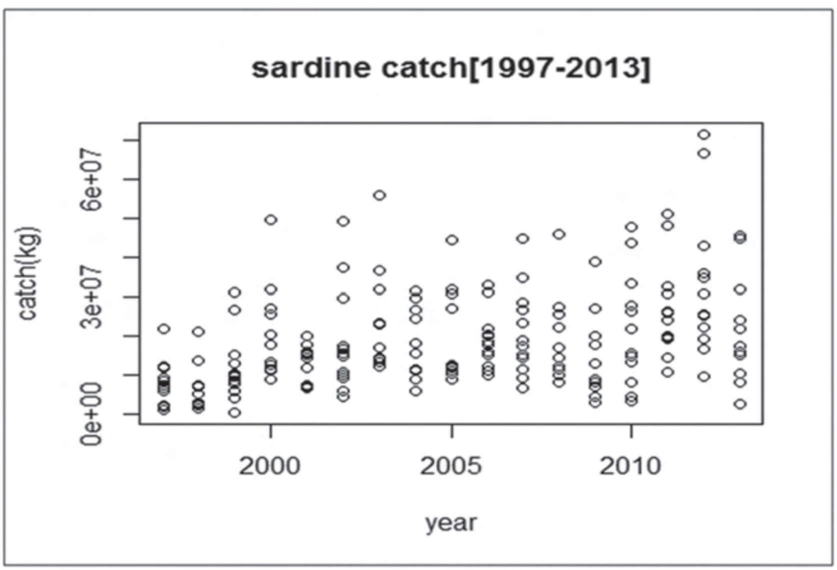
```
## 0.014936315 0.016434376 0.012730547 0.015052097 0.018993675 0.022811653
##       10657       10658       10659       10660       10661       10662
## 0.021590355 0.025598024 0.021891454 0.030677847 0.012303026 0.008431467
##       10663       10664       10665       10666       10667       10668
## 0.010270283 0.015731396 0.014200211 0.013621161 0.019758522 0.024082289
##       10669       10670       10671       10672       10673       10674
## $coefficients
##         (Intercept)          year         month        nomeff
## 10609  2.217824e+07 -1.095925e+04 -1.325088e+04  -3.148198546
## 10610  4.411931e+07 -2.183848e+04 -2.228032e+04  -4.498752468
## 10611 -1.067489e+07  5.318300e+03  5.379473e+03  -1.436946526
## 10612 -1.430707e+07  7.125744e+03  5.005198e+03  -1.244058740
## 10613 -2.792623e+07  1.393898e+046  .644383e+03  -3.898604484
## 10614  3.637567e+06 -1.803856e+03 -6.792737e+01  -0.548821439
## 10615 -1.912700e+07  9.531031e+03 -1.168978e+03  -0.136134257
## 10616 -1.236679e+06  6.142401e+02 -2.614444e+02   0.182574103
## 10617  1.017484e+07 -5.060185e+03  3.311361e+03  -1.300911103
## 10618  5.221933e+06 -2.616049e+03  2.285340e+03   0.594874799
## 10619 -1.269309e+07  6.332354e+03 -7.146199e+03   0.885644012
## 10620 -1.689093e+07  8.416379e+03 -1.142621e+04   2.385068449
## 10621  9.988869e+06 -4.931698e+03 -6.845283e+03  -1.449495213
## 10622  1.048887e+07 -5.182988e+03 -5.814728e+03  -1.523215775
## 10623 -1.631084e+07  8.103095e+03  8.519957e+03  -0.699865368
## 10624 -4.218674e+07  2.105372e+04  1.871018e+04  -8.082331986
## 10625  9.242638e+05 -4.579190e+02 -1.489350e+02  -0.132336511
## 10626  6.358893e+06 -3.155937e+03 -2.504379e+02  -0.691128004
## 10627  3.641035e+06 -1.805648e+03  3.989493e+02  -0.629386219
## 10628  9.337116e+06 -4.637748e+03  2.201757e+03  -1.355018464
## $sigma
##   10609   10610   10611   10612   10613   10614   10615   10616   10617
## 9704033 9673382 9710573 9708368 9692571 9713348 9704899 9713577 9711506
##   10618   10619   10620   10621   10622   10623   10624   10625   10626
```

```
##          9712887 9710099 9707947 9711071 9710794 9705104 9647742 9713585 9712507
##            10627   10628   10629   10630   10631   10632   10633   10634   10635
##          9713275 9711335 9706375 9600885 9706147 9706674 9713017 9687689 9701725
##            10636   10637   10638   10639   10640   10641   10642   10643   10644
##          9713453 9713556 9712299 9713060 9712046 9654918 9701385 9711759 9631991
##            10645   10646   10647   10648   10649   10650   10651   10652   10653
##          9704897 9707140 9703907 9700734 9690097 9679013 9482552 9711297 9710429
##            10654   10655   10656   10657   10658   10659   10660   10661   10662
##          9713454 9704972 9696589 9713578 9706537 9709783 9686303 9712444 9710871
##            10663   10664   10665   10666   10667   10668   10669   10670   10671
##          9702490 9703766 9710000 9709158 9713461 9711904 9705335 9713591 9711428
##            10672   10673   10674   10675   10676  106771    0678   10679   10680
##          9713440 9713390 9713495 9706020 9709067 9620081 9679152 9556146 9705788
##            10681   10682   10683   10684   10685   10686   10687   10688   10689
##          9703041 9712489 9696177 9713305 9713033 9713274 9711229 9713210 9707532
##            10690   10691   10692   10693   10694   10695   10696   10697   10698
##          9484558 9670016 9694154 9710393 9710677 9712970 9696964 9665645 9703363
##            10699   10700   10701   10702   10703   10704   10705   10706   10707
##          9699470 9711903 9695548 9685330 9698839 9696413 9712539 9713605 9645521
##            10708   10709   10710   10711   10712   10713   10714   10715   10716
##          9692194 9657695 9711752 9708527 9712793 9693026 9705844 9708928 9616936
##            10717   10718   10719   10720   10721   10722   10723   10724   10725
##          9700975 9709924 9687368 9702069 9706975 9713608 9712002 9705092 9711736
##
## $wt.res
##         10609        10610        10611        10612        10613
##    5952459.84  12255563.09  -3371411.14  -4445741.27  -8889076.47
##         10614        10615        10616        10617        10618
##     986134.71  -5748266.48   -336390.21   2807133.26   1645172.74
##         10619        10620        10621        10622        10623
##   -3629105.70  -4577842.81   3072907.21   3243308.73  -5672890.07
##         10624        10625        10626        10627        10628
```

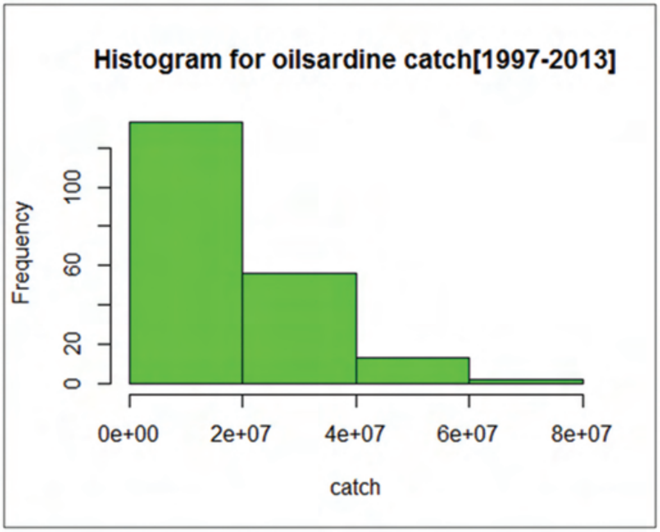
```
## -15696727.40    289232.12   2042122.32   1117366.99   2926082.40
##        10629        10630        10631        10632        10633
##   5230228.43 -20382271.56  -5264124.44  -5075967.51   1491577.71
##        10634        10635        10636        10637        10638
##  -9837151.49  -6712232.19   -764792.30   -437886.38   2231690.27
##        10639        10640        10641        10642        10643
##  -1443831.23  -2440345.04  14926587.99  -6794617.92   2635516.43
##        10644        10645        10646        10647        10648
## -17311907.92  -5709093.26   4952910.28  -6048902.56  -6642668.40
```

Plots in R

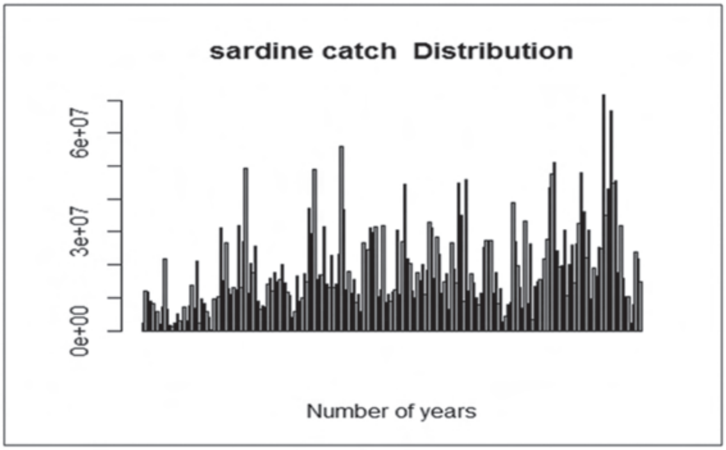
```
##scatter plot
sp3621<-sp362[c(1:2,4)]
attach(sp3621)
## The following objects are masked from klm:
##
##     month, raised, year
plot(year,raised,main="sardine catch[1997-2013]",xlab="year",ylab="catch(kg))
```



```
##Histogram
hist(raised,main="Histogram for oilsardine catch[1997-2013]",
lab="catch",
col="green",
breaks=5)
```




```
##Bar plot
barplot(raised, main="sardine catch  Distribution",
xlab="Number of years")
```

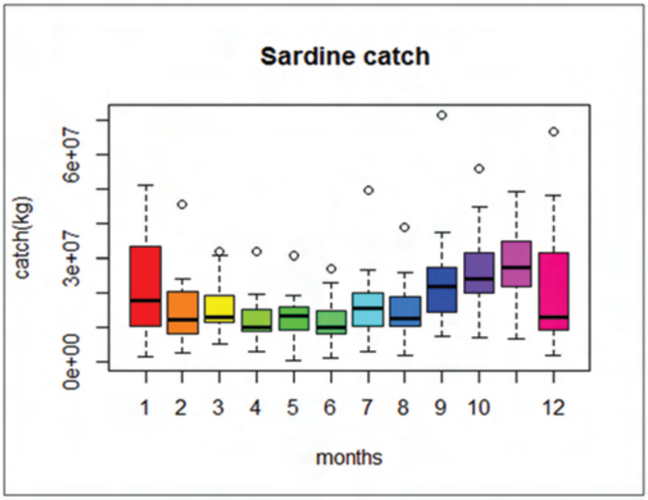


Boxplot in r

Boxplot of catch vs month

boxplot(raised~month,data=sp3621, main="Sardine catch ",

lab="months", ylab="catch(kg)",col=**rainbow**(**length**(**unique**(month))))



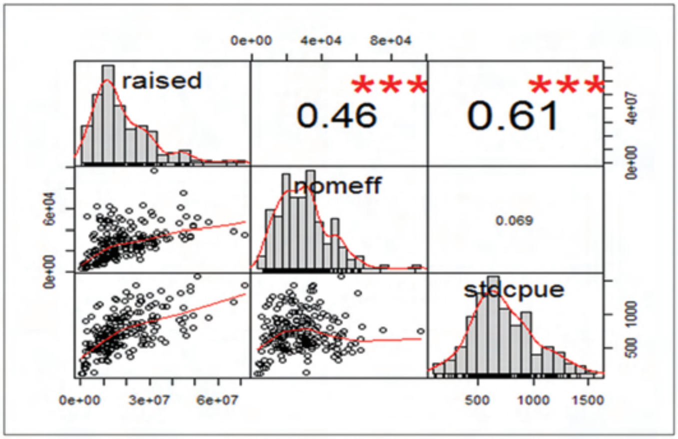
to plot a correlation in r

##we select sardine correlations

cordat<-sp362[,4:6]

library(PerformanceAnalytics)

chart.Correlation(cordat,method="pearson")



R for reading NetCDF data

NetCDF files contain one or more variables, which are usually structured as regular N-dimensional arrays. For example, you might have a variable named "Temperature" that is a function of longitude, latitude, and height. NetCDF files also contain dimensions, which describe the extent of the variables' arrays. In our Temperature example, the dimensions are "longitude", "latitude", and "height". Data can be read from or written to variables in arbitrary hyperslabs (for example, you can read or write all the Temperature values at a given height, or at a given latitude).

The R package 'ncdf4' allows reading from, writing to, and creation of netCDF files, either netCDF version 3 or (optionally) netCDF version 4. If you choose to create version 4 output files, be aware that older netcdf software might only be able to read version 3 files.

In fact this package can help extracting details from HDF5 format files too. This package can create NetCDF files from data.frames also. Nc_open() is the function to be used for opening a NetCDF fils and for creating a NetCDF file the function is nc_creat(). Once opened the attributes and variable names of the data can be got by using the generic print() command. To get specific variables the function is ncvar_get()

An example:

```
library(ncdf4)

ncold <- nc_open("states_population.nc")

data <- ncvar_get(ncold)

print("here is the data in the file:")

print(data)

nc_close( ncold )
```

The output is given below:

```
> ncold <- nc_open("states_population.nc")

> print(ncold)

File states_population.nc (NC_FORMAT_CLASSIC):

1 variables (excluding dimension variables):

int Pop[StateNo]

units: count

_FillValue: -1
```

long_name: Population

1 dimensions:

StateNo Size:50

units: count

long_name: StateNo

1 global attributes:

source: Census 2000 from census bureau web site

>

R in numerical methods

Taking cue from the fact that integration is infinitesimal addition, brutal algorithmic power of R has been put to use to find solutions of definite integrals. The most common function used for this purpose is integrate().

An example:

For the double integral given below

$$\int_0^1 \int_x^1 x\sin(y^2)dydx$$

A couple of lines as given below would do the job in R environment

```
integrate(function(x) {
sapply(x, function(x) {
integrate(function(y) x*sin(y^2),x,1)$value
})
},0,1)
```

The output is given below (with error measure)

```
> integrate(function(x) {
+  sapply(x, function(x) {
+  integrate(function(y) x*sin(y^2),x,1)$value
+     })
+ },0,1)
0.09105548 with absolute error < 1e-15
>
```

References

Bryant, F. B., and Yarnold, P. R. 1995. Principal components analysis and exploratory and confirmatory factor analysis. In L. G. Grimm & P. R. Yarnold (Eds.), *Reading and understanding multivariate analysis*. Washington, DC: American Psychological Association.

Crowley, M. J. 2007. *The R Book*. Chichester, New England: John Wiley & Sons, Ltd.

Dunteman, G. H. 1989. *Principal components analysis*. Newbury Park, CA: Sage Publications.

Fabrigar, L. R., Wegener, D. T., MacCallum, R. C., and Strahan, E. J. (1999). Evaluating the use of exploratory factor analysis in psychological research. *Psychological Methods,* 4(3), 272-299.

Fox, J. 2005. R commander: A basic-statistics user interface to R. *Journal of Statistical Software*. 14 (9): 1-42.

Gorsuch, R. L. 1983. *Factor Analysis*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Hair, J. F., Jr., Anderson, R. E., Tatham, R. L., and Black, W. C. 1995. *Multivariate data analysis with readings* (4th ed.). Upper Saddle River, NJ: Prentice-Hall.

Hatcher, L. 1994. *A step-by-step approach to using the SAS system for factor analysis and structural equation modeling*. Cary, NC: SAS Institute.

http://www.statisticssolutions.com.

Hutcheson, G., and Sofroniou, N. 1999. *The multivariate social scientist: Introductory statistics using generalized linear models*. Thousand Oaks, CA: Sage Publications.

Kim, J. O., and Mueller, C. W. 1978a. *Introduction to factor analysis: What it is and how to do it*. Newbury Park, CA: Sage Publications.

Kim, J. O., and Mueller, C. W. 1978b. *Factor Analysis: Statistical methods and practical issues*. Newbury Park, CA: Sage Publications.

Lawley, D. N., and Maxwell, A. E. 1962. Factor analysis as a statistical method. *The Statistician,* 12(3): 209-229.

Levine, M. S. 1977. *Canonical analysis and factor comparison*. Newbury Park, CA: Sage Publications.

Pett, M. A., Lackey, N. R., and Sullivan, J. J. 2003. *Making sense of factor analysis: The use of factor analysis for instrument development in health care research*. Thousand Oaks, CA: Sage Publications.

Rosales de Veliz L., David S.L., McElhiney D., Price E., and Brooks G. 2012. . *Introduction to R: The Basics* Contributions from Ragan. M., Terzi. F., & Smith. E."— Presentation transcript

Shapiro, S. E., Lasarev, M. R., and McCauley, L. 2002. Factor analysis of Gulf War illness: What does it add to our understanding of possible health effects of deployment, *American Journal of Epidemiology, 156*, 578-585.

Teetor, P. 2011. *25 Recipes for Getting Started with R*. Sebastopol, CA: O'Reilly Media Inc.

Teetor, P. 2011. *R cookbook*. Sebastopol, CA: O'Reilly Media Inc.

Velicer, W. F., Eaton, C. A., and Fava, J. L. 2000. Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components. In R. D. Goffin & E. Helmes (Eds.), *Problems and solutions in human assessment: Honoring Douglas Jackson at seventy.* Boston, MA: Kluwer.

Widaman, K. F. 1993. Common factor analysis versus principal component analysis: Differential bias in representing model parameters, *Multivariate Behavioral Research,* 28, 263-311.