Using R for Windows

R is the most commonly used statistical package among researchers in Statistics. It is freely distributed open source software. For detailed information about downloading and installing R, please see the R project website at http://www.r-project.org.

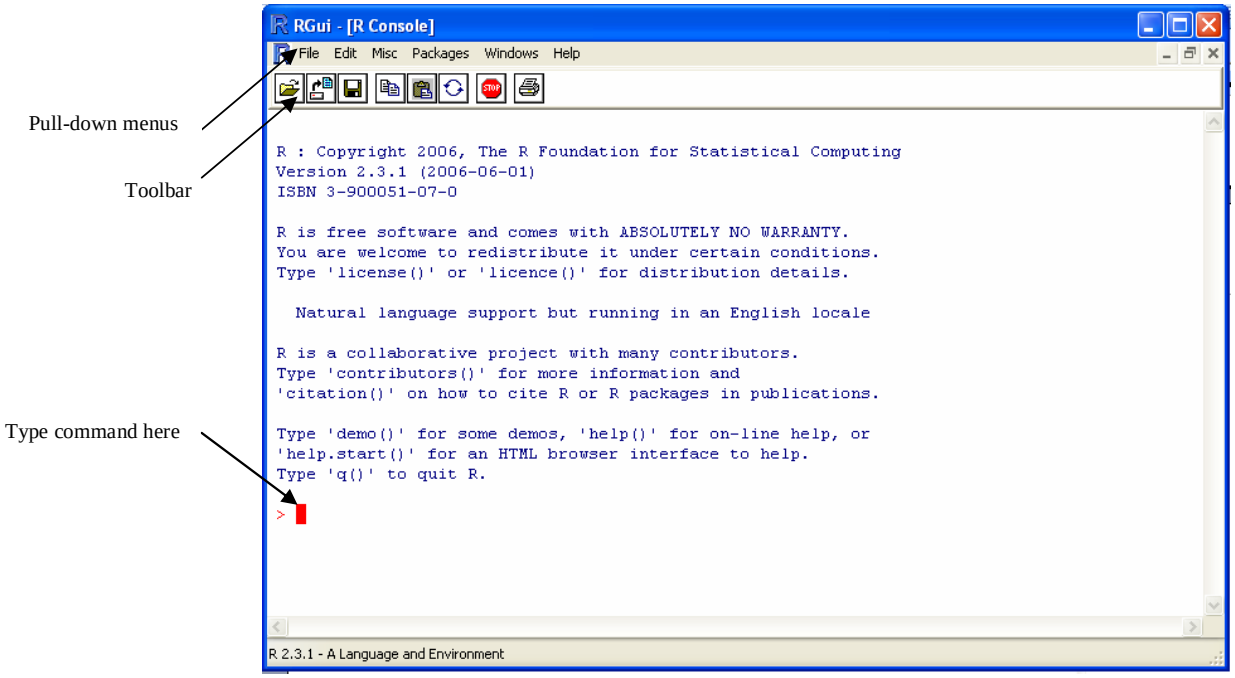
R is a programming language, therefore the challenges for beginners to learn R lie in writing the commands for analysis. Some investment of effort is required before mastering R. One of the great advantages of R is its flexibility. R allows you to download add-on packages for targeted analysis, write your own packages, and share your packages with others. This document covers the basic features and statistical commands of R.

Table of Contents

Getting Started with R for Windows

Similar to SPSS, Stata and SAS, R is a command- and window-driven application. That is, it has a graphical interface consisting of pull-down menus, allows users to issue commands for procedures, and displays results. You can perform numerical and graphical tasks in R by writing commands in the R console window. A basic familiarity with the Windows operating system is all you need to get started with R. Once installed, you can access R on a PC from the Start menu, or by double-clicking on the R icon (shown below) on the desktop, if applicable.

When you start R, a screen like the one below will appear:



Like most Windows programs, R has a toolbar and a menu bar with pull-down menus that you can use to access many of the features of the program. The toolbar contains buttons for more commonly used procedures. To see what each button does, hold the mouse over the button for a moment and a description of what the button does will appear. The following is a summary of the main pull-down menus and their functions:

Menu	***Functions***
File	Open source R code, create, open and save script, load and save workspace, load and save history, display files, change working directory, print files, and exit R.
Edit	Copy, paste, select all, clear console, data editor, R configuration window editor.
Misc	Stop current computation, buffer output, list objects in the memory, remove all objects, and list search path.
Packages	Load, install, update packages, set CRAN mirror, select repositories, install packages, install packages from local zip files.
Windows	Cascade and tile R console windows. Arrange icons and switch among windows
Help	Get help on R procedures, commands, and connect to the R website for more help information.

R Windows

The following is a summary of the functions and contents of three different types of windows in R. All of the windows are accessible from the pull-down **Windows** menu.

You may write R commands in the **R Console** window. This window also displays all the commands R has run through, the results, and error report. This window appears when you launch R. To bring this window to the foreground, click on the window or go to the **Windows** pull-down menu and choose the **R Console**. You can type and run commands in this window line by line.

Alternatively, you can write all the commands in **R Editor** window and allow R to run part or all of the commands at once. You open this window by clicking the **New Script** option in the **File** menu. If you want to redo your analysis at a later time, you can save the scripts. You can also print the scripts by clicking the **Print** option from the **File** menu.

The **R Graphics** window opens automatically when you create a graph. To bring a graph to the foreground, click on the graphics window or go to **Windows** pull-down menu and choose the **R Graphics window**. Graphs can be saved in various formats, such as jpg, png, bmp, ps, pdf, emf, pictex, x_g and so on.

Working with Data Files

Entering Data into R

There are several ways to read external data files into R. This section covers three common ways to read data files into R.

Reading Data from A Line

We use **scan()** to read data from a line. The following is an example that reads height and gender data of 5 children and puts them into a dataset named "data1". The following is an example with comments.

```
height<- scan()                    # Create variable: height
2.2 2.5 3.4 2.9                    # Type in data points with white space as separator
gender=scan()                      # Create another variable: gender
1 0 1 1                            # Type in data
data1= data.frame (height, gender)   # Create data file data1 which has two
                                          variables: height and gender
```

Importing Delimited Data Files

We usually use the **read.table()** function to read in delimited data files. The first line of a data file contains the variable names. From the second line onwards, each line represents one observation, which may consist of a row label and the values for each variable. The following is an example that reads in *text files* to R.

The name of the dataset — directory and name of external text file — omit row label — separator is "white space"

```
baseball <- read.table("C:\\Mydata\\baseball.data", header=T, sep="")
attach (baseball)
```

attach the dataset to the memory

The **read.table()** function can also be used to read a data file over the *internet*. The following is an example of reading data from a course website.

```
hayfever <- read.table('http://www-
stat/~jtaylo/courses/stats191/data/hayfever.table',header=T, sep="")
```

URL of the external file

Comma-delimited files often have an extension of .csv for "comma-separated values." To read these files, we use **read.csv**. For example:

```
baseball <- read.csv("C:\\Mydata\\baseball.csv", header=T, sep=",")
```

If the separator is semicolon, we use **read.csv2** command and **sep** now is defined as a ";".

```
baseball <- read.csv2("C:\\Mydata\\baseball.csv", header=T, sep = ";")
```

Similarly, we may use **read.delim()** or **read.delim2()** to read *tab delimited files* (.txt). We also need to change the field separator character (`sep = "\t"`). "Read.csv", "read.csv2", "read.delim", and "read.delim2" are identical to "read.table" except the defaults are different. We can use "read.table()" to read in comma, semicolon, or tab delimited data files when we choose the corresponding field separator.

Importing Fixed Width Format Files

Fixed width files are those data files that have no field delimiters. We may use the function **read.fwf()** to read this kind of data into R. By using the argument width, we let R know the width of each variable in the file. Fixed width format files do not include the variable names in the first row. R will automatically name the variable V1, V2, V3… after you read in the data. You could use **dimnames** to specify the variable names that you want.

```
datafile<-read.fwf(data, width=c(1,2,3)) #the 1st variable is in column 1, the 2nd
variable is in colum 2-3, the 3rd variable is in column 4-6.
```

Importing Data Files from Other Statistical Softwares

R can import data from other statistical packages. To do so, you need to first load the library "foreign." It allows you to import data from many popular statistical software packages such as Minitab, S, SAS, SPSS, Stata, Systat, dBase, and so on. To install a foreign package in R, you can type the commands:

```
> install.packages('foreign')
> library(foreign)
```

Or, you can choose the **Install packages** submenu of the **Packages** menu and select **Foreign to install**. After you install the foreign package, to view the functions in foreign library, type

```
> library(help=foreign)
```

For example, to read in an SPSS file "datafile.sav" into R, use the following commands:

```
datafile <- read.spss(file="C:\\Mydata\\datafile.sav")
```

Similarly, use the **read.dta** command to read *Stata binary files* and **read.ssd** to read S*AS files*.

Viewing Data in R

After you load a data file into the memory, you can view it by typing the name of the data file. R will display the data file in the console window.

```
> airquality
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
```

```
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
......
```

To check the dimension of the data:

```
> dim(airquality)
[1] 153         6
```

Number of cases — number of variables in the dataset

To print variable names in the data:

```
> names(airquality)
[1] "Ozone"   "Solar.R"  "Wind"    "Temp"    "Month"   "Day"
```

Selecting Subsets of a Dataset

Viewing subsets of a Dataset

To view the 2nd and 3rd columns of the dataset:

```
> airquality[,2:3]
    Solar.R Wind
1       190  7.4
2       118  8.0
3       149 12.6
4       313 11.5
... data deleted ...
```

To view the 2nd and 3rd rows of the dataset:

```
> airquality[2:3,]
  Ozone Solar.R Wind Temp Month Day
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
```

To view the 2nd, 14th and 26th rows of the dataset:

```
> airquality[c(2,14,26),]
   Ozone Solar.R Wind Temp Month Day
2     36     118  8.0   72     5   2
14    14     274 10.9   68     5  14
26    NA     266 14.9   58     5  26
```

To view observations where Wind> 18 and Temp> 60:

```
> airquality[airquality$Wind>18&airquality$Temp>60,]
   Ozone Solar.R Wind Temp Month Day
9      8      19 20.1   61     5   9
48    37     284 20.7   72     6  17
```

Subsetting a Dataset

Subset() is a very useful function to select subsets of a dataset. For example, if you are only interested in dates when the temperature is higher than 95, you may subset the data by typing:

```
> air.sub <- subset(airquality, Temp > 95, select = Temp:Day)
```

new dataset — dataset name — condition — variables that we are interested in

```
> air.sub
    Temp Month Day
120   97     8  28
122   96     8  30
```

You may also extract a smaller dataset with observations of the first day of each month and delete the variable "Wind" from the dataset:

```
> air2.sub <- subset(airquality, Day == 1, select = -Wind)
> air2.sub
    Ozone Solar.R Temp Month Day
1      41     190   67     5   1
32     NA     286   78     6   1
62    135     269   84     7   1
93     39      83   81     8   1
124    96     167   91     9   1
```

Exporting Data

We usually use **write.table()** to export an R dataset into a different format. For example, to export air.sub from the previous example as .csv format to the "C:\Mydata" directory:

```
>  write.table(air.sub, "C:\\Mydata\\airsub.csv")
```

To export it as .txt format:

```
>  write.table(air.sub, "C:\\Mydata\\airsub.txt")
```

Analyzing Data

R has many functions for statistical analyses and graphics. The results and graphs produced by R are displayed on the screen and can also be saved for later use. You can perform many statistical procedures in R by typing commands in the Console window or run the Scripts in the R Editor window. Some advanced functions need to be installed from the package before they can be executed. Please also keep in mind that the R language is case sensitive. It discriminates between uppercase and lowercase letters in the names of the objects.

Descriptive Statistics

To get quick descriptive statistics for all variables in the dataset, we can use the following command.

```
> summary(airquality)
     Ozone           Solar.R           Wind             Temp      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
```

We can get the mean, variance, standard deviation, minimum and maximum of a single variable.

```
> mean(Day)
[1] 15.80392
> var(Day)
[1] 78.57972
> sd(Day)
[1] 8.86452
> min(Day)
```

```
[1] 1
> max(Day)
[1] 31
```

We can also get the correlation matrix of the third, fourth and fifth variable in the dataset. If you just type **cor**(airquality), R will give you the correlation matrix of all variables.

```
> cor(airquality [c(3,4,5)])
            Wind       Temp      Month
Wind   1.0000000 -0.4579879 -0.1782926
Temp  -0.4579879  1.0000000  0.4209473
Month -0.1782926  0.4209473  1.0000000
```

Regression

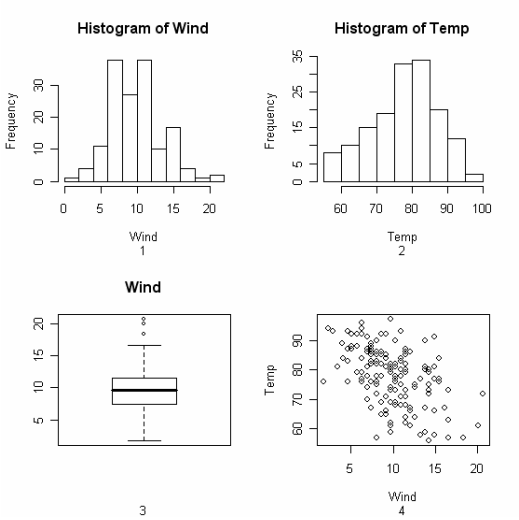
We use the command **lm(Y~X+Z)** to perform a linear regression. The summary statistics for the linear regression model displays the regression model with the residuals, estimates of coefficients, and R-squared.

```
> model = lm(Wind~Temp)
> summary(model)
Call:
lm(formula = Wind ~ Temp)

Residuals:
    Min      1Q  Median      3Q     Max
-8.5784 -2.4489 -0.2261  1.9853  9.7398

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.23369    2.11239  10.999  < 2e-16 ***
Temp        -0.17046    0.02693  -6.331 2.64e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.142 on 151 degrees of freedom
Multiple R-Squared: 0.2098,     Adjusted R-squared: 0.2045
F-statistic: 40.08 on 1 and 151 DF,  p-value: 2.642e-09
```

Creating Graphs

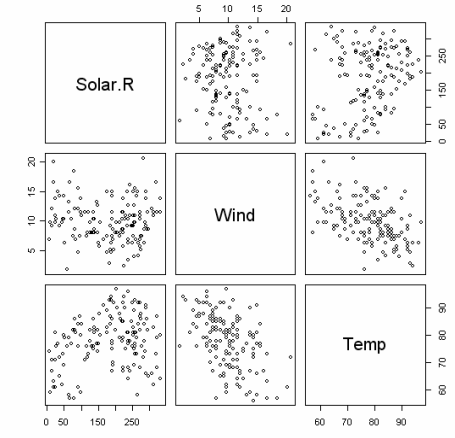
We use the commands **hist()**, **boxplot()** and **plot (X, Y)** to create a histogram, box plot and scatter plot, respectively. A useful function to display several graphs on one graphic sheet is **par()**. For example, you can use **par(mfrow=c(2,2))** to put four graphs on one sheet.

```
> par(mfrow=c(2,2))
> hist (Wind, sub=1)
> hist (Temp, sub=2)
> boxplot (Wind, main='Wind', sub=3)
> plot(Wind, Temp, sub=4)
```



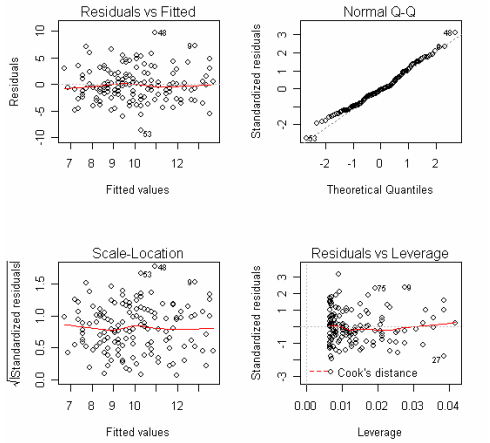
You may want to graph the scatter plot matrix for variables Solar.R, Wind and Temp.

```
> pairs(airquality [c(2,3,4)])
```



You may use the command **plot()** with the name of the object in the brackets to plot 4 basic diagnostic graphs for your regression model: residual versus fitted plot, normal qqplots of the residuals, standardized residual versus fitted plot, and scatter plots with the regression line overlaid.

```
> par(mfrow=c(2,2))
> plot(model)
```



Saving Output Files and Graphs

To save your commands and output in the R console window, you may go to the "save to file" submenu of the "File" menu. Your file will be saved in .txt format and you can open it later. Or you could highlight the area you want in the console window, then copy and paste it into Word or Notepad for subsequent analysis. R displays the graphs in the R Graphics window and you need to save the graphs separately from the text output. The graphs can be saved in various formats, such as jpg, png, bmp, ps, pdf, emf, pictex, x_g and so on.

Documentation and Books

You can read and photocopy R books and manuals in the Velma Denning Room (Green Library Bing Wing, room 120F) or purchase them at the Stanford Bookstore. Some R manuals are also available to be checked out from Green Library Reserves.

You can download the following R manuals from the R Project's manuals page: http://cran.r-project.org/manuals.html.

We also find the following user-contributed documentation helpful in learning R, all of which available as PDF files on http://cran.r-project.org/other-docs.html:

"R for Beginners" by Emmanuel Paradis, good for beginners

"R reference card" by Jonathan Baron, for more advanced users

"R reference card" by Tom Short, for more advanced users

The R Frequently Asked Question site also provides useful help.

http://cran.r-project.org/doc/FAQ/R-FAQ.html

SSDS Software Services at Stanford

The software consultants at Social Science Data and Software (SSDS) provide technical support for SPSS users at Stanford. Users can view documents, access information about our drop-in hours, and submit questions from our web page at:

http://ssds.stanford.edu/

Note: This document is based on R 2.4.1 for Windows XP.

Social Science Data and Software
Document revised: 2/08/2008