

Appendix A

R: Getting started

There are many possible statistical programs that can be used in psychological research. They differ in multiple ways, at least some of which are ease of use, generality, and cost. Some of the more common packages used are SAS, SPSS, and Systat. These programs have GUIs (Graphical User Interfaces) that are relatively easy to use but that are unique to each package. These programs are also very expensive and limited in what they can do. Although convenient to use, GUI based operations are difficult to discuss in written form. When teaching statistics or communicating results, it is helpful to use examples that others may use, perhaps in other computing environments. This book, as well as other texts in the Using R series, describes an alternative approach that is widely used by practicing statisticians, the statistical environment R. This appendix is not meant as a complete user's guide to R, but merely the first step in using R for psychometrics in particular and psychological research in general.

Throughout the text, examples of analyses are given in R. But what is R and how to get it to work on your computer is perhaps the first question the reader faces.

A.1 R: A statistical programming environment

The [R Development Core Team \(2011\)](#) has developed an extremely powerful “language and environment for statistical computing and graphics” and a set of **packages** that operate within this programming environment (R). The R program is an open source version of the statistical program S and is very similar to the statistical program based upon S, S-PLUS (also known as S+). Although described as merely “an effective data handling and storage facility [with] a suite of operators for calculations on arrays, in particular, matrices” R is, in fact, a very useful interactive package for data analysis. When compared to most other stats packages used by psychologists, R has at least three compelling advantages: it is free, it runs on multiple platforms (e.g., Windows, Unix, Linux, and Mac OS X and Classic), and combines many of the most useful statistical programs into one quasi integrated environment. R is free¹, open source software as part of the GNU² Project. That is, users are free to use, modify, and distribute the program, within the limits of the GNU non-license). The program itself and

¹ Free as in speech rather than as in beer. See <http://www.gnu.org>

² GNU's Not Unix

detailed installation instructions for Linux, Unix, Windows, and Macs are available through CRAN (Comprehensive R Archive Network) at <http://www.r-project.org>³

Although many run R as a language and text oriented programming environment, there are GUIs available for PCs, Linux and Macs. See for example, *R Commander* by John Fox or *R-app* for the Macintosh developed by Stefano Iacus and Simon Urbanek. Compared to the basic PC environment, the Mac GUI is to be preferred.

R is an integrated, interactive environment for data manipulation and analysis that includes functions for standard descriptive statistics (means, variances, ranges) and also includes useful graphical tools for Exploratory Data Analysis. In terms of inferential statistics R has many varieties of the General Linear Model including the conventional special cases of Analysis of Variance, MANOVA, and linear regression. Statisticians and statistically minded people around the world have contributed **packages** to the R Group and maintain a very active news group offering suggestions and help. The growing collection of **packages** and the ease with which they interact with each other and the core R is perhaps the greatest advantage of R. Advanced features include correlational **packages** for multivariate analyses including Factor and Principal Components Analysis, and cluster analysis. Advanced multivariate analyses **packages** that have been contributed to the R-project include one for Structural Equation Modeling (**sem**), Multi-level modeling (also known as Hierarchical Linear Modeling and referred to as non linear mixed effects in the **nlme4** package) and taxometric analysis. All of these are available in the free **packages** distributed by the R group at CRAN. Many of the functions described in this book are incorporated into the **psych** package. Other **packages** useful for psychometrics are described in a **task-view** at CRAN. In addition to be a environment of prepackaged routines, R is a interpreted programming language that allows one to create specific functions when needed.

R is also an amazing program for producing statistical graphics. A collection of some of the best graphics is available at the webpage <http://addictedtor.free.fr/graphiques/> with a complete gallery of thumbnail of figures.

A.2 General comments

R is not overly user friendly (at first). Its error messages are at best cryptic. It is, however, very powerful and once partially mastered, easy to use. As additional **packages** are added, it becomes even more useful. Packages available at CRAN may be found and downloaded by using the **package manager** command (or menu option for the Mac R-Gui). To download a number of relevant packages all at once, the **taskviews** function will install packages recommended for particular applications (e.g., **Psychometrics**, **Socialsciences**, etc).

Commands may be entered directly into the “Console” window and executed immediately. In this regard, R can be thought of as a very advanced graphing calculator. Alternatively, the Mac and PC versions have a text editor window that allows you to write, edit and save your commands. For all systems, if you use a normal text editor (As a Mac user, I use BBEDIT, PC users can use Notepad or TINN-R, many users prefer the ESS – emacs

³ The [R Development Core Team \(2011\)](#) releases an updated version of R about every six months. That is, as of March, 2012, the current version of 2.14.2 will be replaced with 2.15.0 sometime in March of 2012. Bug fixes are then added with a sub version number (e.g. 2.12.2 fixed minor problems with 2.12.1). It is recommended to use the most up to date version, as it will incorporate various improvements and operating efficiencies.

speaks statistics– editor), you can write out the commands you want to run, comment them so that you can remember what they do the next time you run a similar analysis, and then copy and paste them into the R console. You can add a comment to any line by using a `#`. Anything following the `#` is printed but ignored for execution. This allows you to document your commands as you write them. The `history` window keeps track of recent commands. The R code throughout this text is meant to be copied and pasted into R.

Although being syntax driven seems a throwback to an old, pre Graphical User Interface type command structure, it is very powerful for doing production statistics. Once you get a particular set of commands to work on one data file, you can change the name of the data file and run the entire sequence again on the new data set. This is also very helpful when doing professional graphics for papers. In addition, for teaching, it is possible to prepare a web page of instructional commands that students can then cut and paste into R to see for themselves how things work. That is what may be done with the instructions throughout this book. It is also possible to write text in \LaTeX with embedded R commands. Then executing the `Sweave` function on that text file will add the R output to the \LaTeX file. This almost magical feature allows rapid integration of content with statistical techniques. More importantly, it allows for *literate programming* and *reproducible research* (Leisch and Rossini, 2003) in that the actual data files and instructions may be specified for all to see.

A.3 Using R in 12 simple steps

(These steps are not meant to limit what can be done with R, but merely to describe how to do the analysis for the most basic of research projects and to give a first experience with R).

1. Install R on your computer or go to a machine that has it.
2. Download the `psych` package as well as other recommended packages from CRAN using the `install.packages` function, or using the package installer in the GUI. To get packages recommended for a particular research field, use the `ctv` package to install a particular *task view*. Note, these first two steps need to be done only once!
3. Activate the `psych` package or other desired packages using e.g., `library(psych)`. This needs to be done every time you start R. Or, it is possible to modify the *startup parameters* for R so that certain libraries are loaded automatically.
4. Enter your data using a text editor and save as a text file (perhaps comma delimited if using a spreadsheet program such as Excel or OpenOffice)
5. Read the data file or copy and paste from the clipboard (using, e.g., `read.clipboard`).
6. Find basic descriptive statistics (e.g., means, standard deviations, minimum and maxima) using `describe`.
7. Prepare a simple descriptive graph (e.g, a box plot) of your variables.
8. Find the correlation matrix to give an overview of relationships (if the number is not too great, a scatter plot matrix or SPLOM plot is very useful, this can be done with `pairs.panels`).
9. If you have an experimental variable, do the appropriate multiple regression using standardized or at least zero centered scores.
10. If you want to do a *factor analysis* or *principal components* analysis, use the `factanal` or `fa` and `principal` functions.

11. To score items and create a scale and find various reliability estimates, use `score.items` and perhaps `omega`.
12. Graph the results.

A.4 Getting started

A.4.1 Installing R on your computer

Although it is possible that your local computer lab already has R, it is most useful to do analyses on your own machine. In this case you will need to download the R program from the R project and install it yourself. Using your favorite web browser, go to the R home page at <http://www.r-project.org> and then choose the Download from CRAN (Comprehensive R Archive Network) option. This will take you to list of mirror sites around the world. You may download the Windows, Linux, or Mac versions at this site. For most users, downloading the binary image is easiest and does not require compiling the program. Once downloaded, go through the *install* options for the program. If you want to use R as a visitor it is possible to install R onto a “thumb drive” or “memory stick” and run it from there. (See the R for Windows FAQ at CRAN).

A.4.2 Packages and Task Views

One of the great strengths of R is that it can be supplemented with additional programs that are included as *packages* using the `package manager`. (e.g., *sem* or *OpenMX* do structural equation modeling) or that can be added using the `source` command. Most packages are directly available through the CRAN repository. Others are available at the BioConductor (<http://www.bioconductor.org>) repository. Yet others are available at “other” repositories. The *psych* package (Revelle, 2011) may be downloaded from CRAN or from the <http://personality-project.org/r> repository.

The concept of a “task view” has made downloading relevant packages very easy. For instance, the `install.views("Psychometrics")` command will download over 20 packages that do various types of psychometrics. To install the *Psychometrics task view*⁴

```
> install.packages("ctv")
> library(ctv)
> install.views("Psychometrics")
```

For any other than the default packages to work, you must activate it by either using the Package Manager or the `library` command:

- e.g., `library(psych)` or `library(sem)`
- entering `?psych` will give a list of the functions available in the `psych` package as well as an overview of their functionality.

⁴ Here, as throughout the book, the “>” represents the R system prompt to enter a new line. Do not enter it, but just rather enter the text following the prompt.

- `objects(package:psych)` will list the functions available in a package (in this case, `psych`).

```
> library(psych)
> ?psych
> objects(package:psych)
```

If you routinely find yourself using the same packages everytime you use R, you can modify the Startup process by specifying what should happen `.First`. Thus, if you always want to have `psych` available,

```
.First <- function(library(psych))
```

and then when you quit, use the `save workspace` option.

A.4.3 Help and Guidance

R is case sensitive and does not give overly useful diagnostic messages. If you get an error message, don't be flustered but rather be patient and try the command again using the correct spelling for the command.

When in doubt, use the `help(somefunction)` function. This is identical to `? somefunction` where some function is what you want to know about. e.g.,

```
> ?read.table #ask for help in using the read.table function the answer is in the help window
> help(read.table) #another way of asking for help
> ??read #searches for all uses of the function in all your packages.
> apropos("read") #returns all available functions with that term in their name
> RSiteSearch("read") #opens a webbrowser and searches voluminous files
```

`RSiteSearch("keyword")` will open a browser window and return a search for "keyword" in all functions available in R and the associated packages as well (if desired) the R-Help News groups.

All packages and all functions will have an associated help window. Each help window will give a brief description of the function, how to call it, the definition of all of the available parameters, a list (and definition) of the possible output, and usually some useful examples. One can learn a great deal by using the help windows, but if they are available, it is better to study the *package vignette*.

A.4.4 Package vignettes

All packages have help pages for each function in the package. These are meant to help you use a function that you already know about, but not to introduce you to new functions. An increasing number of packages have a package *vignettes* that give more of an overview of the program than a detailed description of any one function. These vignettes are accessible from the help window and sometimes as part of the help index for the program. The two vignettes for the `psych` package are also available from the personality project web page. ([An overview of the psych package](#) and [Using the psych package as a front end to the sem package](#)).

A.5 Basic R commands and syntax

A.5.1 *R is just a fancy calculator*

One can think of R as a fancy graphics calculator. Enter a command and look at the output. Thus,

```
> 2 + 2 #returns the output
```

```
4
```

or, somewhat more fun, try graphing a function or two:

```
curve(sin, -2*pi, 2*pi)
curve(tan, main = "curve(tan) --> same x-scale as previous plot")
```

At the somewhat more abstract level, almost all operations in R consists of executing a *function* on an *object*. The result is a new object. This very simple idea allows the output of any operation to be operated on by another function.

Command syntax tends to be of the form:

```
variable = function (parameters) or
variable <- function (parameters)
```

The = and the <- symbol imply replacement, not equality. The preferred style is to use the <- symbol to avoid confusion with the test for equality (==).

The result of an operation will not necessarily appear unless you ask for it. The command

```
m <- mean(x)
```

will find the mean of x but will not print anything on the console without the additional request

```
m.
```

however, just asking `mean(x)`

will find the mean and print it.

A.5.1.1 R is also a statistics table

It has been suggested by some that you should never buy a statistics book that has probability tables in it, because that means that the author did not know about the various distributions in R. Many statistics books include tables of the *t* or *F* or χ^2 distribution. By using R this is unnecessary since these and many more distributions can be obtained directly. Consider the normal distribution as an example. `dnorm(x, mean=mu, sd=sigma)` will give the probability density of observing that x in a distribution with mean=mu and standard deviation= sigma. `pnorm(q, mean=0, sd=1)` will give the probability of observing the value q or less. `qnorm(p, mean=0, sd=1)` will give the quantile value of a value with probability p. `rnorm(n, mean, sd)` will generate n random observations sampled from the normal distribution with specified mean and standard deviation. Thus, to find out what z value has a .05 probability we ask for `qnorm(.05)`. Or, to evaluate the probability of observing a z value of 2.5, specify `pnorm(2.5)`. (These last two examples are one side p values).

A.5.2 Data structures

Data may be composed of

- Single elements which may be of type integer, real, complex, or character. Thus, `i <- 1`, `x <- 2.34`, `name <- "bill"` are possible elements.
- Vectors and lists are collections of elements can be formed by combining individual elements: `v <- c(i,x,name)` or by providing a rule: `y <- 10:20` (creates a vector with elements [10, 11, ..., 20]).

A suggestion about programming style that is not strictly R related: It is useful to label variables with names that will make sense to you later when you look at an analysis several months later. Thus, rather than calling variables `x`, `y`, and `z`, giving names that reflect that they are in fact impulsivity, anxiety, and performance tends to be more useful.

For a more complete list of R commands, see Appendix B. A limited number of examples are shown below. Examples of using R to do simple matrix operations are discussed in Appendix E.

A.6 Entering or getting the data

For most data analysis, rather than manually enter the data into R, it is probably more convenient to use a spreadsheet (e.g., Excel or OpenOffice) as a data editor, save as a tab or comma delimited file, and then read the data from the file. Many of the examples in this tutorial assume that the data have been entered this way. Many of the examples in the help menus have small data sets entered using the `c()` command or created on the fly. It is also possible to read data in from a remote file server.

Using the `copy.clipboard()` function from the **psych** package, it is also possible to have a data file open in a text editor or spreadsheet program, copy the relevant lines to the clipboard, and then read the clipboard directly into R.

For the first example, we read data from a remote file server for several hundred subjects on 13 personality scales (5 from the Eysenck Personality Inventory (EPI), 5 from a Big Five Inventory (BFI), 1 Beck Depression, and two anxiety scales). The data are taken from a study in the Personality, Motivation, and Cognition Laboratory at Northwestern University. The file is structured normally, i.e. rows represent different subjects, columns different variables, and the first row gives subject labels. Had we saved this file as comma delimited, we would add the separation (`sep=","`) parameter.

To read a file from your local machine, change the `datafilename` to specify the path to the data. Using the `file.choose` command, you can set a local file name to the data file name anywhere on your computer.

```
#specify the name and address of the remote file
>datafilename <- "http://personality-project.org/r/datasets/maps.mixx.epi.bfi.data"
#Or, If I want to read a datafile from my desktop
>datafilename <- file.choose() #where you dynamically can go find the file

#now read the data file
>person.data <- read.table(datafilename,header=TRUE) #read the data file
```

```
>names(person.data) #list the names of the variables

> names(person.data) #list the names of the variables
[1] "epiE"      "epiS"      "epiImp"    "epilie"    "epiNeur"   "bfagree"   "bfcon"
[8] "bfext"     "bfneur"    "bfopen"    "bdi"       "traitanx"  "stateanx"
```

The data are now in the data.frame “person.data”. Data.frames allow one to have columns that are either numeric or alphanumeric. They are conceptually a generalization of a matrix in that they have rows and columns, but unlike a matrix, some columns can be of different “types” (integers, reals, characters, strings) than other columns.

A.7 Basic descriptive statistics

Basic descriptive statistics are most easily reported by using the `summary`, `mean` and Standard Deviations (`sd`) commands. Using the `describe` function available in the `psych` package is also convenient. Graphical displays that also capture this are available as a boxplot.

```
> summary(person.data) #print out the min, max, range, mean, median, etc. of the data
> round(mean(person.data),2) #means of all variables, rounded to 2 decimals
> round(sd(person.data),2) #standard deviations, rounded to 2 decimals
```

epiE	epiS	epiImp	epilie	epiNeur
Min. : 1.00	Min. : 0.000	Min. :0.000	Min. :0.000	Min. : 0.00
1st Qu.:11.00	1st Qu.: 6.000	1st Qu.:3.000	1st Qu.:1.000	1st Qu.: 7.00
Median :14.00	Median : 8.000	Median :4.000	Median :2.000	Median :10.00
Mean :13.33	Mean : 7.584	Mean :4.368	Mean :2.377	Mean :10.41
3rd Qu.:16.00	3rd Qu.: 9.500	3rd Qu.:6.000	3rd Qu.:3.000	3rd Qu.:14.00
Max. :22.00	Max. :13.000	Max. :9.000	Max. :7.000	Max. :23.00

bfagree	bfcon	bfext	bfneur	bfopen
Min. : 74.0	Min. : 53.0	Min. : 8.0	Min. : 34.00	Min. : 73.0
1st Qu.:112.0	1st Qu.: 99.0	1st Qu.: 87.5	1st Qu.: 70.00	1st Qu.:110.0
Median :126.0	Median :114.0	Median :104.0	Median : 90.00	Median :125.0
Mean :125.0	Mean :113.3	Mean :102.2	Mean : 87.97	Mean :123.4
3rd Qu.:136.5	3rd Qu.:128.5	3rd Qu.:118.0	3rd Qu.:104.00	3rd Qu.:136.5
Max. :167.0	Max. :178.0	Max. :168.0	Max. :152.00	Max. :173.0

bdi	traitanx	stateanx
Min. : 0.000	Min. :22.00	Min. :21.00
1st Qu.: 3.000	1st Qu.:32.00	1st Qu.:32.00
Median : 6.000	Median :38.00	Median :38.00
Mean : 6.779	Mean :39.01	Mean :39.85
3rd Qu.: 9.000	3rd Qu.:44.00	3rd Qu.:46.50
Max. :27.000	Max. :71.00	Max. :79.00

epiE	epiS	epiImp	epilie	epiNeur	bfagree	bfcon	bfext	bfneur
13.33	7.58	4.37	2.38	10.41	125.00	113.25	102.18	87.97

```

bfopen      bdi traitanx stateanx
123.43      6.78   39.01   39.85

  epiE      epiS      epiImp      epilie      epiNeur      bfagree      bfcon      bfext      bfneur
  4.14      2.69      1.88      1.50      4.90      18.14      21.88      26.45      23.34
bfopen      bdi traitanx stateanx
20.51      5.78   9.52   11.48

```

A.7.1 Using functions in the psych package

The psych package has been developed particularly for simple psychometrics and exploratory data of psychological data. It may be downloaded using the package installer from CRAN or from the <http://personality-project.org/r> personality project.

Once downloaded and installed, it needs to be loaded before it can be used. The `library` command does this.

Among the functions within the psych package are `describe` and `pairs.panels`.

```

> library(psych)
> describe(person.data)

```

	var	n	mean	sd	median	min	max	range	se
epiE	1	231	13.33	4.14	14	1	22	21	0.27
epiS	2	231	7.58	2.69	8	0	13	13	0.18
epiImp	3	231	4.37	1.88	4	0	9	9	0.12
epilie	4	231	2.38	1.50	2	0	7	7	0.10
epiNeur	5	231	10.41	4.90	10	0	23	23	0.32
bfagree	6	231	125.00	18.14	126	74	167	93	1.19
bfcon	7	231	113.25	21.88	114	53	178	125	1.44
bfext	8	231	102.18	26.45	104	8	168	160	1.74
bfneur	9	231	87.97	23.34	90	34	152	118	1.54
bfopen	10	231	123.43	20.51	125	73	173	100	1.35
bdi	11	231	6.78	5.78	6	0	27	27	0.38
traitanx	12	231	39.01	9.52	38	22	71	49	0.63
stateanx	13	231	39.85	11.48	38	21	79	58	0.76

The `describe` function can be combined with the `by` function to provide even more detailed tables. This example reports descriptive statistics for subjects with lie scores < 3 and those ≥ 3 . The second element in the `by` command could be a categorical variable (e.g., sex).

```
by(person.data, epilie < 3, describe)
```

```
epilie < 3: FALSE
```

	var	n	mean	sd	median	min	max	range	se
epiE	1	90	12.64	4.00	13.0	1	21	20	0.42

epiS	2	90	7.61	2.81	8.0	0	13	13	0.30
epiImp	3	90	3.97	1.67	4.0	1	8	7	0.18
epilie	4	90	3.89	1.08	4.0	3	7	4	0.11
epiNeur	5	90	9.33	5.20	9.0	0	20	20	0.55
bfagree	6	90	128.12	16.55	129.0	87	167	80	1.74
bfcon	7	90	117.56	20.46	118.0	58	178	120	2.16
bfext	8	90	100.88	25.24	101.0	24	151	127	2.66
bfneur	9	90	82.22	22.80	81.5	35	144	109	2.40
bfopen	10	90	121.97	20.55	121.0	75	172	97	2.17
bdi	11	90	5.77	4.71	5.0	0	24	24	0.50
traitanx	12	90	37.01	9.06	36.0	22	71	49	0.95
stateanx	13	90	38.41	11.36	36.5	21	69	48	1.20

epilie < 3: TRUE

	var	n	mean	sd	median	min	max	range	se
epiE	1	141	13.77	4.17	14	4	22	18	0.35
epiS	2	141	7.57	2.62	8	1	13	12	0.22
epiImp	3	141	4.62	1.97	5	0	9	9	0.17
epilie	4	141	1.41	0.73	2	0	2	2	0.06
epiNeur	5	141	11.10	4.59	10	0	23	23	0.39
bfagree	6	141	123.00	18.88	124	74	165	91	1.59
bfcon	7	141	110.50	22.38	111	53	176	123	1.88
bfext	8	141	103.01	27.25	105	8	168	160	2.29
bfneur	9	141	91.64	23.01	94	34	152	118	1.94
bfopen	10	141	124.36	20.50	126	73	173	100	1.73
bdi	11	141	7.43	6.29	6	0	27	27	0.53
traitanx	12	141	40.28	9.62	39	23	71	48	0.81
stateanx	13	141	40.77	11.51	39	23	79	56	0.97

A.8 Simple Graphics

There are a variety of ways of graphically reporting the data. One is the box plot (`boxplot`) to show the Tukey 5 numbers (upper and lower hinges, upper and lower quartiles, median). Another way to grasp the distribution of the data is to overlay the actual data points with a `stripchart`.

```
boxplot(person.data[,1:5])
stripchart(person.data[,1:5],vertical=T,add=T,method="jitter",jitter=.2) #add in the points
```

Another way of describing the data is to graph them. `boxplot` show the top and bottom quartiles, medians, and the "hinges". histograms show the distribution in more detail. The `pairs.panels` command draws a matrix of scatter plots. (Note that just the first five variables are shown in the SPLOM to make it more readable).

```
pairs.panels(person.data[,1:5])
```

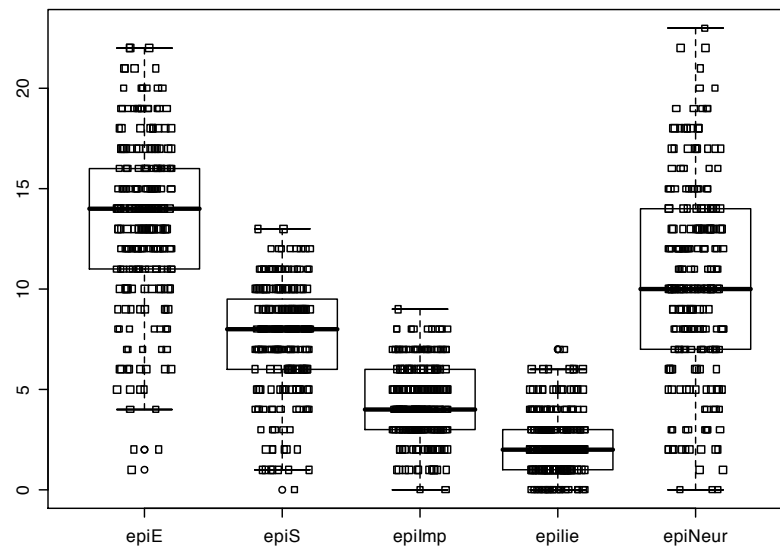


Fig. A.1 A boxplot with an added stripchart summarizes basic distributional properties of the data.

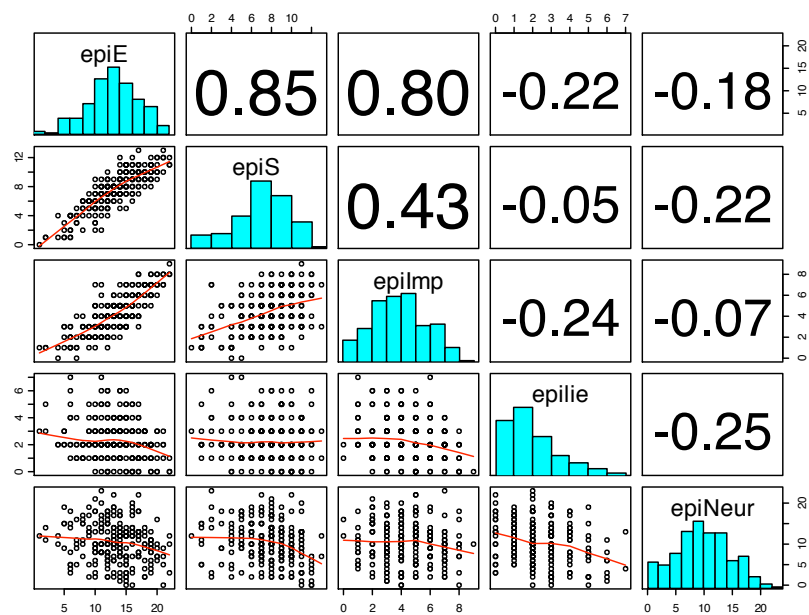


Fig. A.2 A scatter plot matrix of the data can be modified to give histograms as well as the correlations.

Appendix B

R commands

(Very rough summary of the most useful command)

B.1 Input and display

```
#read files with labels in first row
read.table(filename,header=TRUE) #read a tab or space delimited file
read.table(filename,header=TRUE,sep=',') #read csv files (comma separated)

x=c(1,2,4,8,16) #create a data vector with specified elements
y=c(1:8,1:4) #creat a data vector with 12 entries
matr=rbind(1:8,1:4) #create two rows in a 2 * 8 matrix
matc=cbind(1:8,1:4) #create two columns in a 8 * 2 matrix
n=10
x1=c(rnorm(n)) #create a n item vector of random normal deviates
y1=c(runif(n))+n #create another n item vector that has n added to each random uniform
distribution
z=rbinom(n,size,prob) #create n samples of size "size" with probability prob from the bino-
mialitem
sample(x, size, replace = FALSE, prob = NULL) #take a sample (with or without replace-
ment) of size from x

vect=c(x,y) #combine them into one vector of length 2n
mat=cbind(x,y) #combine them into a n x 2 matrix (column wise)
mat[4,2] #display the 4th row and the 2nd column
mat[3,] #display the 3rd row
mat[,2] #display the 2nd column
mat=cbind(rep(1:4,2),rep(4:1,2)) #create a 8 * 2 matrix with repeating elements
subset(data,logical) #those objects meeting a logical criterion
subset(data.df,select=variables,logical) #get those objects from a data frame that meet a
criterion
```

B.2 moving around

```
ls() #list the variables in the workspace
rm(x) #remove x from the workspace
rm(list=ls()) #remove all the variables from the workspace
attach(mat) #make the names of the variables in the matrix available
detach(mat) #releases the names
new=old[,-n] #drop the nth column
new=old[n,] #drop the nth row
new=subset(ol,logical) #select those cases that meet the logical condition
complete = subset(data,complete.cases(data)) #find those cases with no missing values
new=old[n1:n2,n3:n4] #select the n1 through n2 rows of variables n3 through n4)
```

B.3 data manipulation

```
x.df=data.frame(x1,x2,x3 ...) #combine different kinds of data into a data frame
as.data.frame()
is.data.frame()
x=as.matrix()
scale() #converts a data frame to standardized scores
factor() #converts a numeric variable into a factor (essential for ANOVA)
gl(n,k,length) #makes an n-level,k replicates, length long vectof factors
y <- edit(x) #opens a screen editor and saves changes made to x into y
fix(x) #opens a screen editor window and makes and saves changes to x
```

B.4 Statistics and transformations

```
max()
min()
mean()
median()
interp.median() #for interpolated values sum()
var() #produces the variance covariance matrix
sd() #standard deviation
mad() #(median absolute deviation)
fivenum() #Tukey fivenumbers min, lowerhinge, median, upper hinge, max
scale(data,scale=T) #centers around the mean and scales by the sd
colSums(), rowSums(), colMeans(), rowMeans() #see also apply(x,1,sum)
rowsum(x,group) #sum by group
cor(x,y,use="pair") #correlation matrix for pairwise complete data, use="complete" for complete cases
```

```
t.test(x,y) #x is a data vector, y is a grouping vector independent groups
t.test(x,y,pair=TRUE) #x is a data vector, y is a grouping vector – paired groups
pairwise.t.test(x,g) does multiple comparisons of all groups defined by g
aov(x y,data=datafile) #where x and y can be matrices
aov.ex1 = aov(Alertness Dosage,data=data.ex1) #do the analysis of variance or
aov.ex2 = aov(Alertness Gender*Dosage,data=data.ex2) #do a two way analysis of variance
summary(aov.ex1) #show the summary table
print(model.tables(aov.ex1,"means"),digits=3) #report the means and the number of sub-
jects/cell
boxplot(Alertness Dosage,data=data.ex1) #graphical summary appears in graphics window
```

```
lm(x y,data=dataset) #basic linear model where x and y can be matrices
lm(Y X) #Y and X can be matrices
lm(Y X1+X2)
lm(Y X|W) #separate analyses for each level of W
solve(A,B) #inverse of A * B - used for linear regression
solve(A) #inverse of A
```

B.5 Useful additional commands

```
colSums(x, na.rm = FALSE, dims = 1)
rowSums(x, na.rm = FALSE, dims = 1)
colMeans(x, na.rm = FALSE, dims = 1)
rowMeans(x, na.rm = FALSE, dims = 1)
rowsum(x, group, reorder = TRUE, ...) #finds row sums for each level of a grouping variable
apply(X, MARGIN, FUN, ...) #applies the function (FUN) to either rows (1) or columns (2)
on object X
apply(x,1,min) #finds the minimum for each row
apply(x,2,max) #finds the maximum for each column
col.max(x) #another way to find which column has the maximum value for each row
which.min(x)
which.max(x)
z=apply(big5r,1,which.min) #tells the row with the minimum value for every column
```

B.6 Graphics

```
stem() #stem and leaf diagram
```

```
par(mfrow=c(2,1)) #number of rows and columns to graph
```

```
boxplot(x,notch=T,names= grouping, main="title") #boxplot (box and whiskers)
```

```

hist() #histogram
plot()
plot(x,y,xlim=range(-1,1),ylim=range(-1,1),main=title)
par(mfrow=c(1,1)) #change the graph window back to one figure
symb=c(19,25,3,23)
colors=c("black","red","green","blue")
character=c("S","T","N","H")
plot(x,y,pch=symb[group],col=colors[group],bg=colors[condit],cex=1.5,main="main title")
points(mPA,mNA,pch=symb[condit],cex=4.5,col=colors[condit],bg=colors[condit])

curve()
abline(a,b)
abline(a, b, untf = FALSE, ...)
abline(h=, untf = FALSE, ...)
abline(v=, untf = FALSE, ...)
abline(coef=, untf = FALSE, ...)
abline(reg=, untf = FALSE, ...)

identify()
plot(eatar,eanta,xlim=range(-1,1),ylim=range(-1,1),main=title)
identify(eatar,eanta,labels=labels(energysR[,1]) ) #dynamically puts names on the plots
locate()
pairs() #SPLOM (scatter plot Matrix)

matplot () #ordinate is row of the matrix
biplot () #factor loadings and factor scores on same graph
coplot(x y|z) #x by y conditioned on z
symb=c(19,25,3,23) #choose some nice plotting symbols
colors=c("black","red","green","blue") #choose some nice colors

barplot() #simple bar plot
interaction.plot () #shows means for an ANOVA design

plot(degreedays,therms) #show the data points
by(heating,Location,function(x) abline(lm(therms degreedays,data=x))) #show the best fit-
ting regression for each group

x= recordPlot() #save the current plot device output in the object x
replayPlot(x) #replot object x
dev.control #various control functions for printing/saving graphic files
dput () #output the structure and contents of an object
getAnywhere(myfunction) #find where the function "myfunction" is located and then show
it methods(some function) #show generic functions objects(package:psych) #show the func-
tions in a pacakge

```

Table B.1 Functions used in this chapter. *are part of the psych package.

Function	Use	Example
?	Help about a function. Same as help.	?round or help(round)
%*%	Binary operator to do vector or matrix multiplication	A %*% B
%+%	*Binary operator to do vector or matrix like sums	A %+% B
==	Test of equality	A == B
<-	Assignment A is replaced by B. This notation is preferred to =	A <- B
=	Assignment A is replaced by B. (see <-)	A = B
[,]	Evaluate an element of a matrix , array or data.frame. A[i,j] is the ith row, jth column.	$x_{ij} = X[i, j]$
\$	Evaluate an element of a list or data.frame. A\$B is the element with name B in A.	a.b = A\$B
as.vector()	Make a set of numbers into a vector	A <- as.vector(A)
c()	Combine two or more items.	A <- c(B,C)
colnames() rownames()	Find or make the column names (also see rownames)	A <- colnames(B) finds colnames(A) <- B makes
colMeans() rowMeans()	Find column or row means of each column or row in a data.frame or matrix	Ac <- colMeans(A) Ar <- rowMeans(A)
curve()	Plot the curve for a specified function.	curve(1/(1+exp(-x)),-3,3)
data.frame()	Create a data.frame. (Similar to a matrix, but can have different types of elements)	A <- data.frame(x,y,z)
describe()	*Report basic descriptive statistics for a vector, matrix, or dataframe	describe(X)
diag()	Create or find the diagonal of a square matrix	A <- diag(B) finds diag(B) <- A creates
dim	Report the dimensions (rows,cols) of a data.frame or matrix.	n.rows <- dim(x.df)[1]
exp()	Raise e to the A power	exp(A)
for()	Execute a loop from start to finish	for (i in start:finish) {some operation using i}
function()	Create a new function to do something.	new.f <- function(x,y) { new <- x + y}
if() ... else	Do something if a logical condition holds. Do something else if it does not hold.	if(A < B) {print("B")} else {print("A")}
length()	Report the number of elements in a vector	n <- length(v)
list()	A general way of storing results.	A <- list(a=1,b=2,c=3.4)
log()	Find the natural logarithm of X	A <- log(X)
lower.tri()	Logical function, true if an element is in lower triangular submatrix of a matrix (see upper.tri)	A <- lower.tri(B)
matrix()	Create a matrix of m*n elements with m rows and n columns	A <- matrix(B,ncol=n)
mean() median()	Find the mean, or median of a data.frame, vector, or matrix	A.m <- mean(A)
model.fit()	*Calculate 3 alternative goodness of fit indices (see text)	
paste()	Combine several numeric or text variable into a string	A <- paste('B','is','4')
pairs.panels()	*Plot the scatter plot matrices (SPLOM) and report the correlations for a data.frame or matrix	pairs.panels(x.df)
pnorm()	What is the probability of an observation, z, given the normal distribution $-\infty < z < \infty$	p <- pnorm(z)
qnorm()	What is the z score associated with a particular quantile (probability) in a normal distribution $0 < x < 1$	z <- qnorm(x)
read.clipboard()	*Read a data matrix or data table from the clipboard	A <- read.clipboard()
rep()	Repeat A N times	rep(A,N)
round()	Round off numbers to n digits	round(x,n)
runif()	Create n random numbers, uniformly distributed between a and b. (Defaults to a=0, b=1)	runif(n,a,b)
sample()	Draw n samples (with or without replacement) from x	sample(2,100))
set.seed()	Supply a particular start value to the random number generator	set.seed(42)
seq()	Form the sequence from lower to upper by step size	x <- seq(lower,upper,step)
t()	Transpose a vector or matrix	ta <- t(A)
upper.tri()	Logical function, true if an element is in upper triangular submatrix of a matrix (see lower.tri)	A <- lower.tri(B)