R: A short course

William Revelle
Northwestern University
March 30-April , 2009

Overview

I. Day 1: What is R: An introduction

II. Day 2: Graphical displays and EDA

III. Day 3: The general linear model and its special cases (ANOVA), multilevel models

IV. Day 4: Multivariate analysis

V. Day 5: R as a programming language

What is R: an introduction

I. What is it

II. How to get it

III. Entering or getting data

IV. Basic descriptive stats

R: statistics for all of us

I. What is it?

II. Why use it?

III. Common (mis)perceptions

IV. Examples for personality and individual differences research

R: What is it?

I. R: An international collaboration

II. R: the open source - public domain version of S/S+

III. R: written by statisticians (and all of us) for statisticians (and the rest of us)

IV. R: an extensible language

Common statistical programs

General	Specialized
R	AMOS
S+	EQS
SAS	LISREL
SPSS	MPlus
STATA	Mx
SYSTAT	your favorite program.

Common statistical programs most are costly

General	Specialized
R	AMO$
$+	EQ$
A	LI$REL
$P$$	MPlu$
$TATA	Mx
YTAT	your favorite program.

R: a way of thinking

(from the R point of view)

- “R is the lingua franca of statistical research. Work in all other languages should be discouraged.”
- “This is R. There is no if. Only how.”
- “Overall, SAS is about 11 years behind R and S-Plus in statistical capabilities (last year it was about 10 years behind) in my estimation.”

Taken from the R.-fortunes (selections from the R.-help list serve)

But it is open source - how can you trust it?

I. Q: When you use it [R], since it is written by so many authors, how do you know that the results are trustable?

II. A:The R engine [...] is pretty well uniformly excellent code but you have to take my word for that. Actually, you don't. The whole engine is open source so, if you wish, you can check every line of it. If people were out to push dodgy software, this is not the way they'd go about it.

Taken from the R.-fortunes (selections from the R.-help list serve)

What is R? : Technically

I. R is an open source implementation of S (S-Plus is a commercial implementation)

II. R is available under GNU Copy-left

III. The current version of R is 2.8.1 (2.9.0 is at the alpha release, 2.10 is in development)

IV. R is group project run by a core group of developers (with new releases semiannually)

(Adapted from Robert Gentleman)

R: History

I. 1991-93: Ross Dhaka and Robert Gentleman begin work on R project at U. Auckland

II. 1995: R available by ftp under the GPL

III. 96-97: mailing list and R core group is formed

IV. 2000: John Chambers, designer of S joins the R core (wins a prize for best software from ACM for S)

V. 2001-2005: Core team continues to improve base package

VI. 2009: Becoming the standard for statistics, although other languages are relevant (Python)

VII. Many (>1455) other contributed “packages”

R in Psychology Personality/Psychometrics

I. Some software packages tailored to psychology have been developed (e.g. those in the task view of psychometrics)

II. Some universities are now introducing R to graduate students

III. Some instructors are using it for undergraduates.

Why R?

I. Graphics for data exploration and interpretation

II. Data manipulation including statistics as data

III. Statistical analysis

   A. Standard univariate and multivariate generalizations of the linear model

   B. Multivariate-structural extensions

IV. Ease of programming for new applications

Ok, how do I get it

I. CRAN (Comprehensive R Archive Network)

   A. http://cran.r-project.org/

The Comprehensive R Archive Network

CRAN
[Mirrors]
[What's new?]
[Task Views]
[Search]

About R
[R Homepage]

Software
[R Sources]
[R Binaries]
[Packages]
[Other]

Documentation
[Manuals]
[FAQs]
[Contributed]
[Newsletter]

Frequently used pages

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Linux
- MacOS X
- Windows

Source Code for all Platforms

Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2008-12-22): R-2.8.1.tar.gz (read what's new in the latest version).
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.2 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Altough we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Universal R 2.8.1 for Mac OS X released on 2008/12/22

This binary distribution of R and the GUI supports both PowerPC and Intel based Macs. The corresponding binaries of R packages are available for both architectures as well. Starting with R 2.3.1, CRAN binaries support Mac OS X 10.4 (Tiger) and higher only. It is, however, possible to compile binaries for earlied OS X versions from sources.

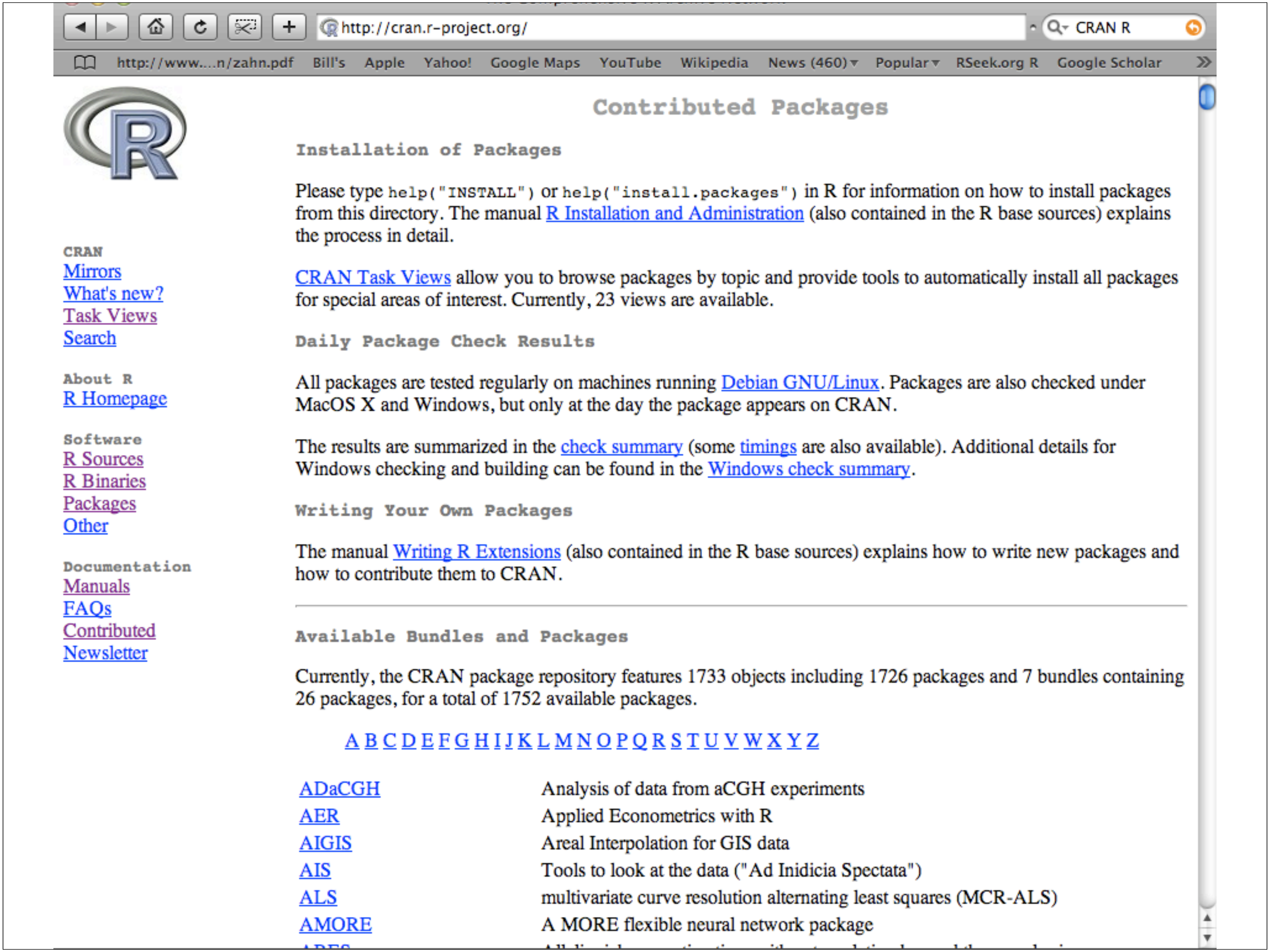
Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type

```
md5 R-2.8.1.dmg
```

in the *Terminal* application to print the MD5 checksum for the R-2.8.1.dmg image.

Files:

R-2.8.1.dmg (latest version) MD5- hash: c43581d6ebede51fee2da8fb5292b2c7 (ca. 63MB)	Universal binary of **R 2.8.1** for Mac OS X 10.4.4 and higher. This is a disk image containing the installer of R for Mac OS X 10.4.4 or higher. This image also contains Tcl/Tk libraries (for X11) and GNU Fortran 4.2.3 for both PowerPC and Intel Macs. This binary was tested on both Mac OS X 10.4 (Tiger) and Mac OS X 10.5 (Leopard). Depending on your browser, you may need to press the control key and click on this link to download the file. To install R simply double-click on icon of the multi-package "R.mpkg" contained in the `R-2.8.0.dmg` disk image.
R-2.8.1-mini.dmg MD5- hash: 4cd1ff20d029a06523d9345a7e1bf582 (ca. 27MB)	Universal binary of **R 2.8.1** for Mac OS X, upgrade package without supplemental tools. This is a subset of the above image. Unless a full R 2.8.0 installer was used before, it is not possible to compile packages from Fortran sources when this smaller subset is used. Only binary installs will work correctly. Also Tcl/Tk will not work unless installed separately. The



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

http://www....n/zahn.pdf Bill's Apple Yahoo! Google Maps YouTube Wikipedia News (460) Popular RSeek.org R Google Scholar

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

Contributed Packages

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this directory. The manual R Installation and Administration (also contained in the R base sources) explains the process in detail.

CRAN Task Views allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 23 views are available.

Daily Package Check Results

All packages are tested regularly on machines running Debian GNU/Linux. Packages are also checked under MacOS X and Windows, but only at the day the package appears on CRAN.

The results are summarized in the check summary (some timings are also available). Additional details for Windows checking and building can be found in the Windows check summary.

Writing Your Own Packages

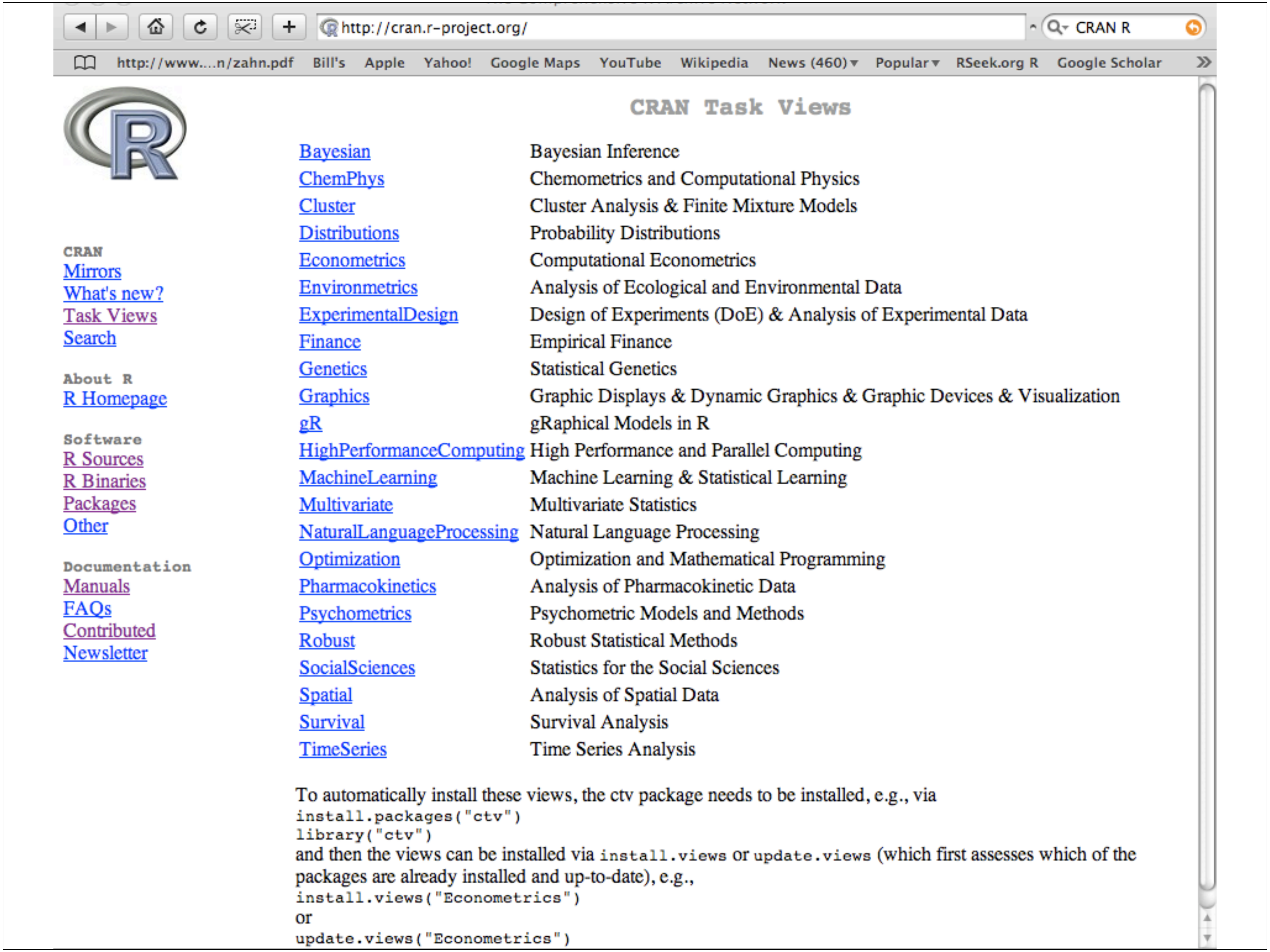
The manual Writing R Extensions (also contained in the R base sources) explains how to write new packages and how to contribute them to CRAN.

Available Bundles and Packages

Currently, the CRAN package repository features 1733 objects including 1726 packages and 7 bundles containing 26 packages, for a total of 1752 available packages.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

ADaCGH	Analysis of data from aCGH experiments
AER	Applied Econometrics with R
AIGIS	Areal Interpolation for GIS data
AIS	Tools to look at the data ("Ad Inidicia Spectata")
ALS	multivariate curve resolution alternating least squares (MCR-ALS)
AMORE	A MORE flexible neural network package



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

CRAN R

http://www....n/zahn.pdf Bill's Apple Yahoo! Google Maps YouTube Wikipedia News (460) Popular RSeek.org R Google Scholar

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
gR	gRaphical Models in R
HighPerformanceComputing	High Performance and Parallel Computing
MachineLearning	Machine Learning & Statistical Learning
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Psychometrics	Psychometric Models and Methods
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis

To automatically install these views, the ctv package needs to be installed, e.g., via

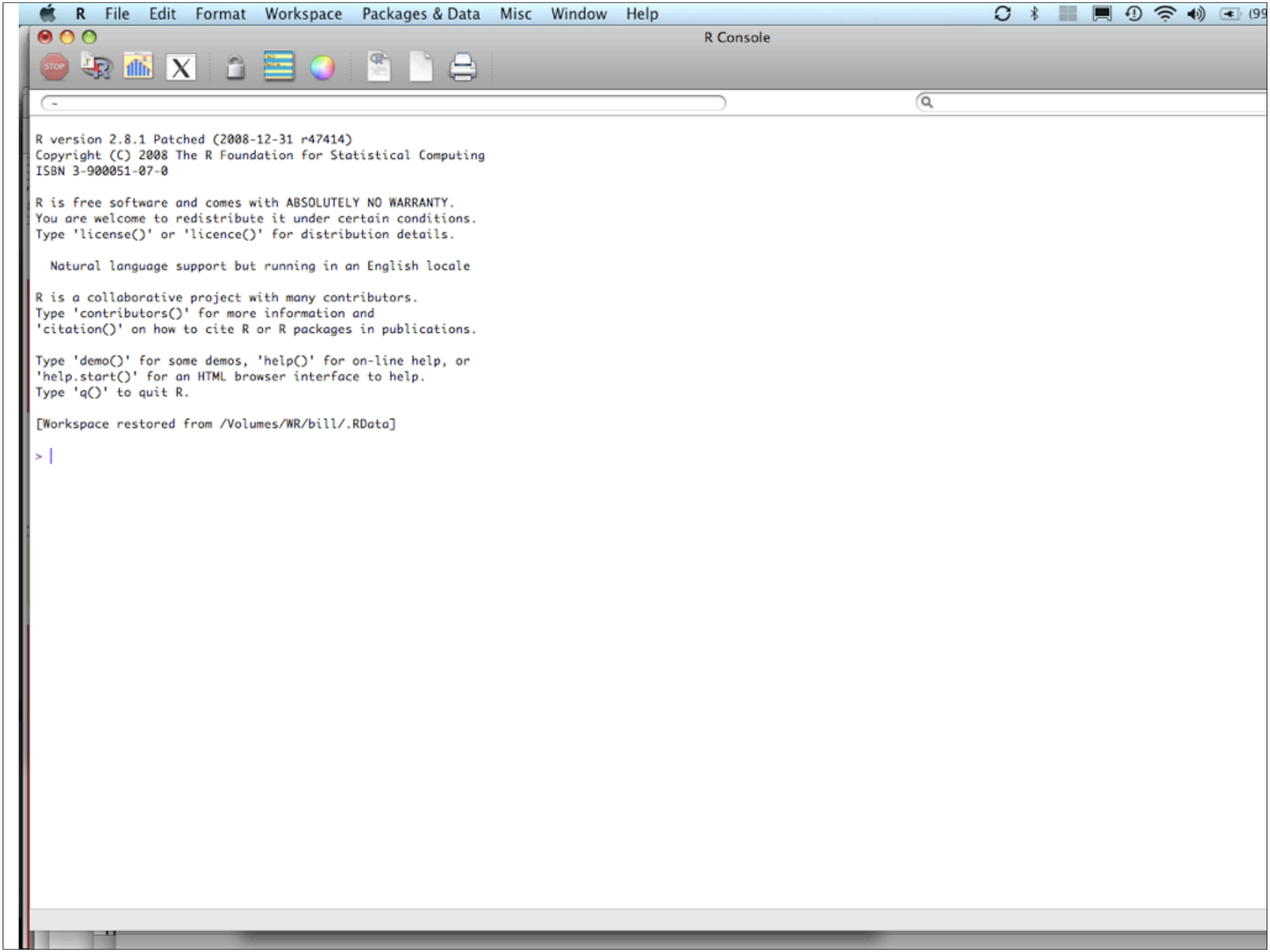
```
install.packages("ctv")
library("ctv")
```

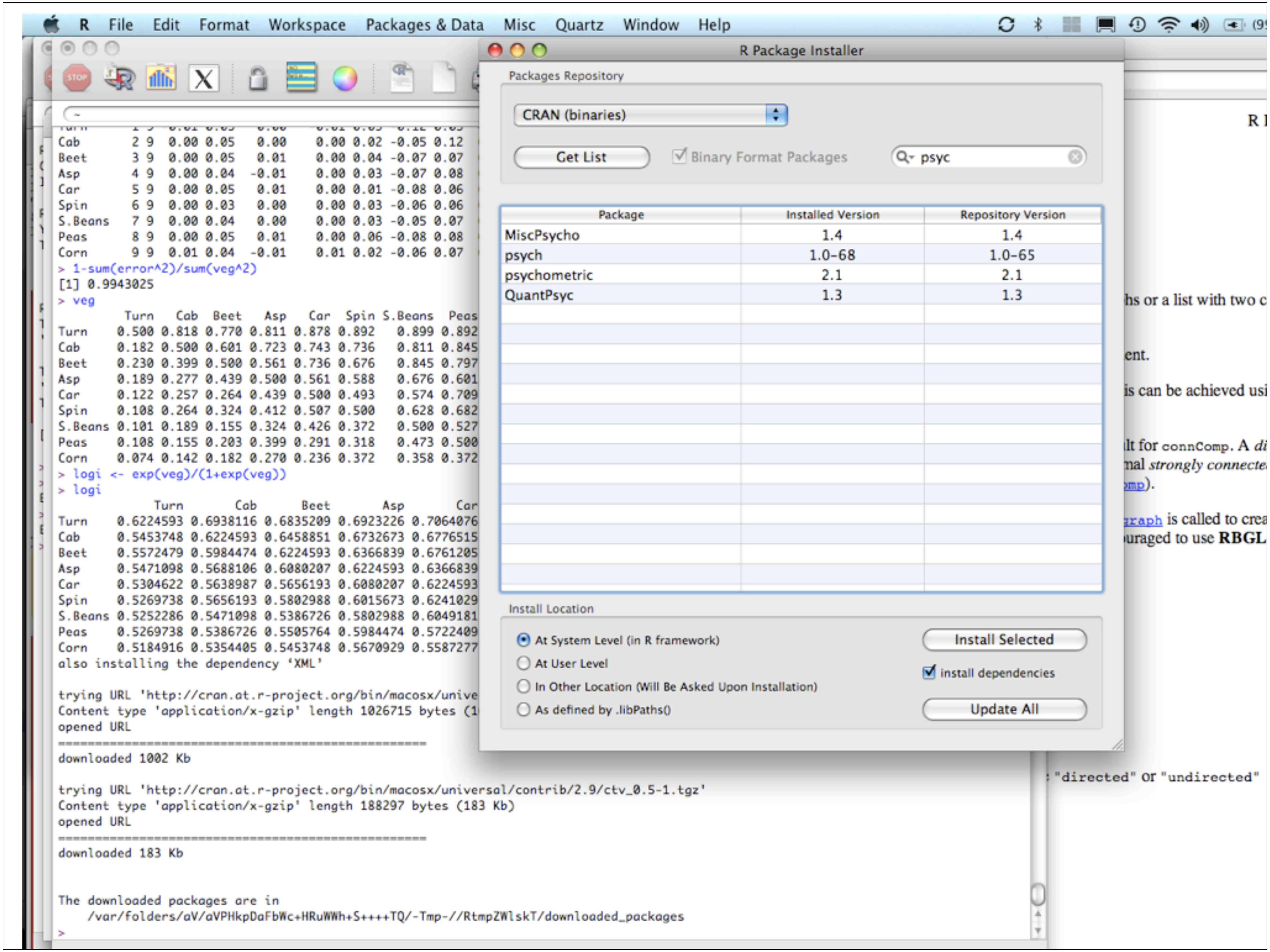
and then the views can be installed via `install.views` or `update.views` (which first assesses which of the packages are already installed and up-to-date), e.g.,

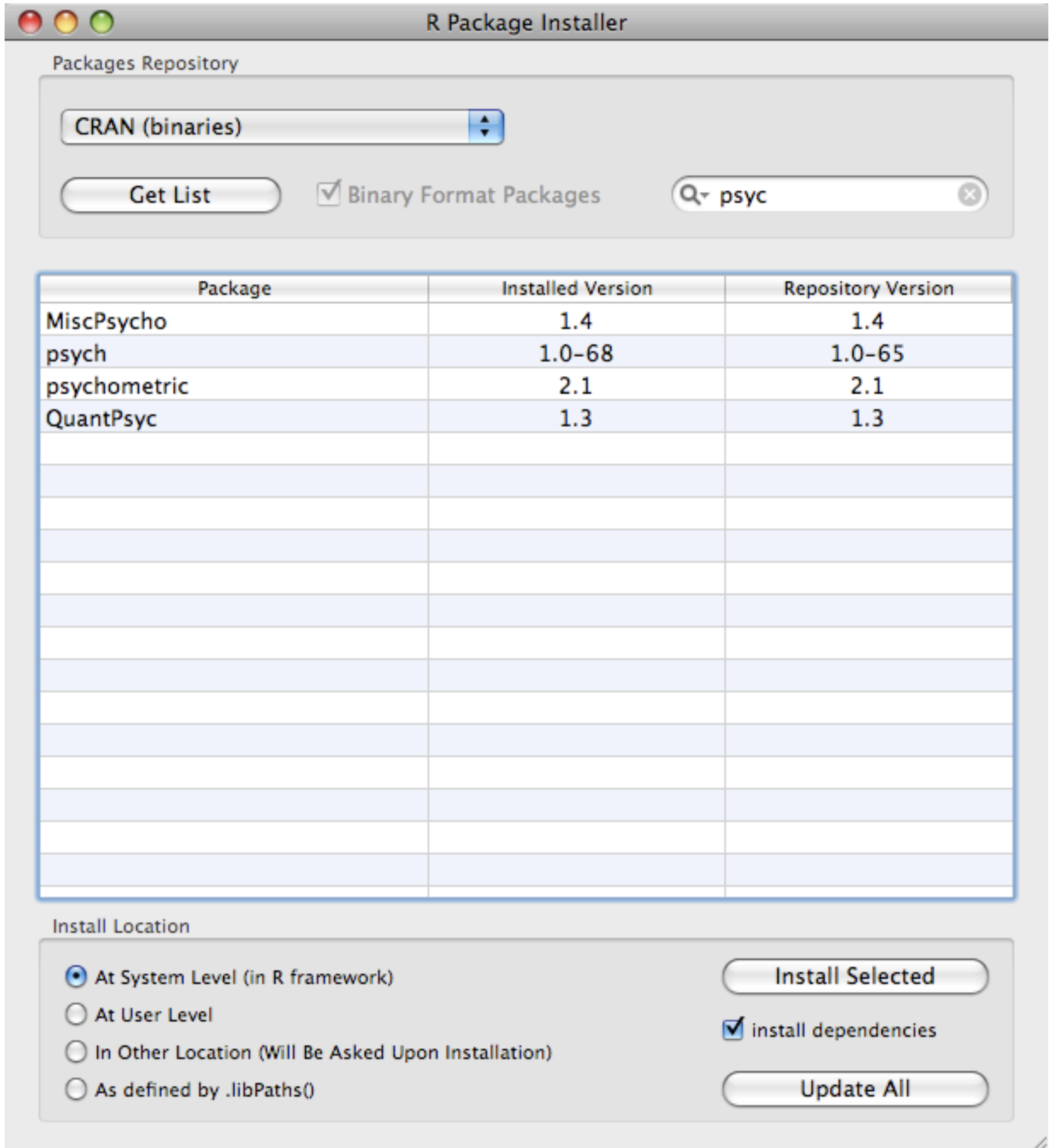
```
install.views("Econometrics")
```

or

```
update.views("Econometrics")
```


R File Edit Format Workspace Packages & Data Misc Window Help
R Console
R version 2.8.1 Patched (2008-12-31 r47414)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace restored from /Volumes/WR/bill/.RData]
>


R File Edit Format Workspace Packages & Data Misc Quartz Window Help
R Package Installer
Packages Repository
CRAN (binaries)
Get List
Binary Format Packages
psyc
Package | Installed Version | Repository Version
MiscPsycho 1.4 1.4
psych 1.0-68 1.0-65
psychometric 2.1 2.1
QuantPsyc 1.3 1.3
Install Location
At System Level (in R framework)
At User Level
In Other Location (Will Be Asked Upon Installation)
As defined by .libPaths()
Install Selected
install dependencies
Update All
Cab 2 9 0.00 0.05 0.00 0.00 0.02 -0.05 0.12
Beet 3 9 0.00 0.05 0.01 0.00 0.04 -0.07 0.07
Asp 4 9 0.00 0.04 -0.01 0.00 0.03 -0.07 0.08
Car 5 9 0.00 0.05 0.01 0.00 0.01 -0.08 0.06
Spin 6 9 0.00 0.03 0.00 0.00 0.03 -0.06 0.06
S.Beans 7 9 0.00 0.04 0.00 0.00 0.03 -0.05 0.07
Peas 8 9 0.00 0.05 0.01 0.00 0.06 -0.08 0.08
Corn 9 9 0.01 0.04 -0.01 0.01 0.02 -0.06 0.07
> 1-sum(error^2)/sum(veg^2)
[1] 0.9943025
> veg
Turn Cab Beet Asp Car Spin S.Beans Peas
Turn 0.500 0.818 0.770 0.811 0.878 0.892 0.899 0.892
Cab 0.182 0.500 0.601 0.723 0.743 0.736 0.811 0.845
Beet 0.230 0.399 0.500 0.561 0.736 0.676 0.845 0.797
Asp 0.189 0.277 0.439 0.500 0.561 0.588 0.676 0.601
Car 0.122 0.257 0.264 0.439 0.500 0.493 0.574 0.709
Spin 0.108 0.264 0.324 0.412 0.507 0.500 0.628 0.682
S.Beans 0.101 0.189 0.155 0.324 0.426 0.372 0.500 0.527
Peas 0.108 0.155 0.203 0.399 0.291 0.318 0.473 0.500
Corn 0.074 0.142 0.182 0.270 0.236 0.372 0.358 0.372
> logi <- exp(veg)/(1+exp(veg))
> logi
Turn Cab Beet Asp Car
Turn 0.6224593 0.6938116 0.6835209 0.6923226 0.7064076
Cab 0.5453748 0.6224593 0.6458851 0.6732673 0.6776515
Beet 0.5572479 0.5984474 0.6224593 0.6366839 0.6761205
Asp 0.5471098 0.5688106 0.6080207 0.6224593 0.6366839
Car 0.5304622 0.5638987 0.5656193 0.6080207 0.6224593
Spin 0.5269738 0.5656193 0.5802988 0.6015673 0.6241029
S.Beans 0.5252286 0.5471098 0.5386726 0.5802988 0.6049181
Peas 0.5269738 0.5386726 0.5505764 0.5984474 0.5722409
Corn 0.5184916 0.5354405 0.5453748 0.5670929 0.5587277
also installing the dependency 'XML'
trying URL 'http://cran.at.r-project.org/bin/macosx/unive
Content type 'application/x-gzip' length 1026715 bytes (1
opened URL
downloaded 1002 Kb
trying URL 'http://cran.at.r-project.org/bin/macosx/universal/contrib/2.9/ctv_0.5-1.tgz'
Content type 'application/x-gzip' length 188297 bytes (183 Kb)
opened URL
downloaded 183 Kb
The downloaded packages are in
/var/folders/aV/aVPHkpDaFbWc+HRuWWh+S++++TQ/-Tmp-//RtmpZWlskT/downloaded_packages
"directed" or "undirected"

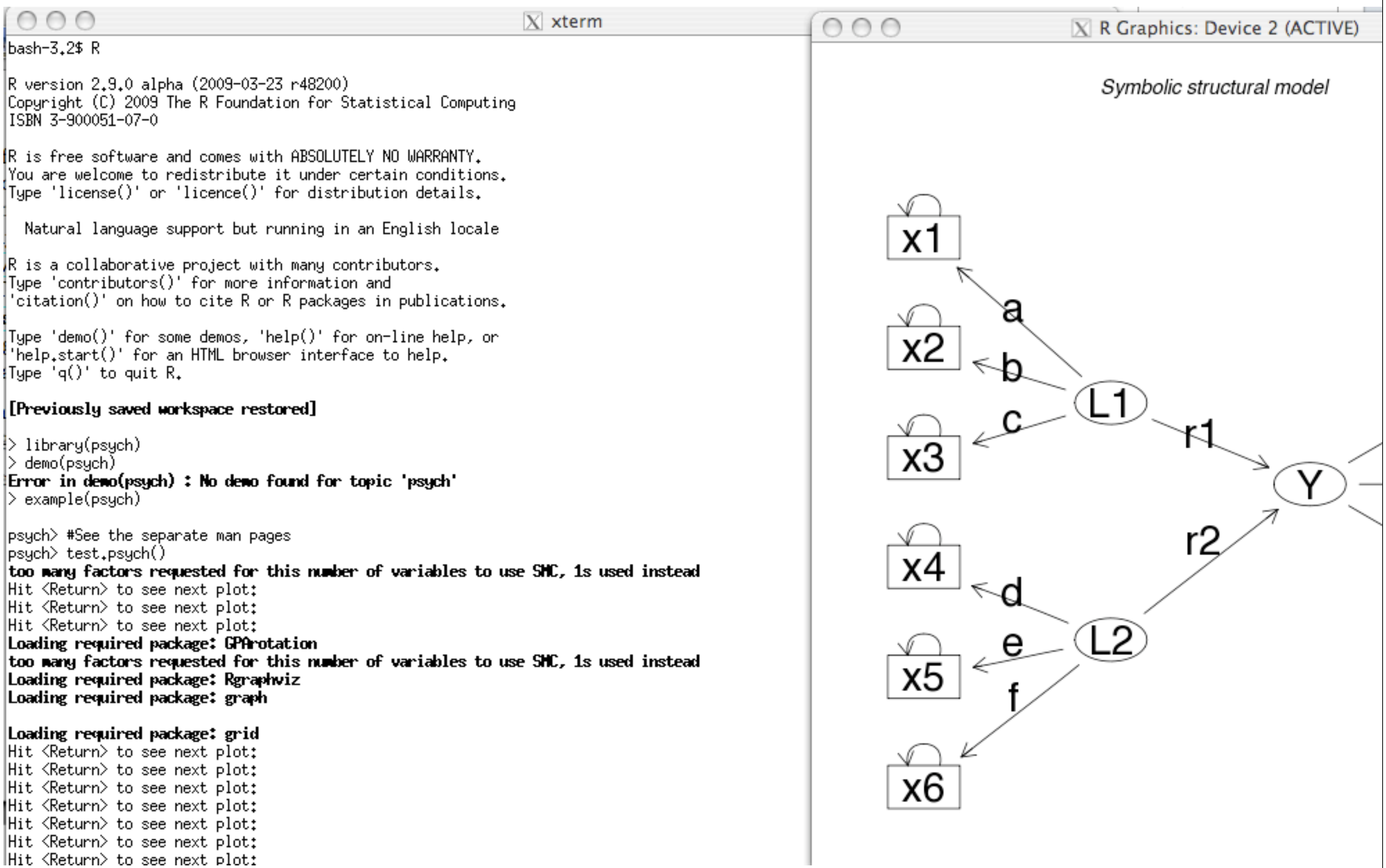

R Package Installer
Packages Repository
CRAN (binaries)
Get List
Binary Format Packages
psyc
Package
Installed Version
Repository Version
MiscPsycho 1.4 1.4
psych 1.0-68 1.0-65
psychometric 2.1 2.1
QuantPsyc 1.3 1.3
Install Location
At System Level (in R framework)
At User Level
In Other Location (Will Be Asked Upon Installation)
As defined by .libPaths()
Install Selected
install dependencies
Update All

Running in X11

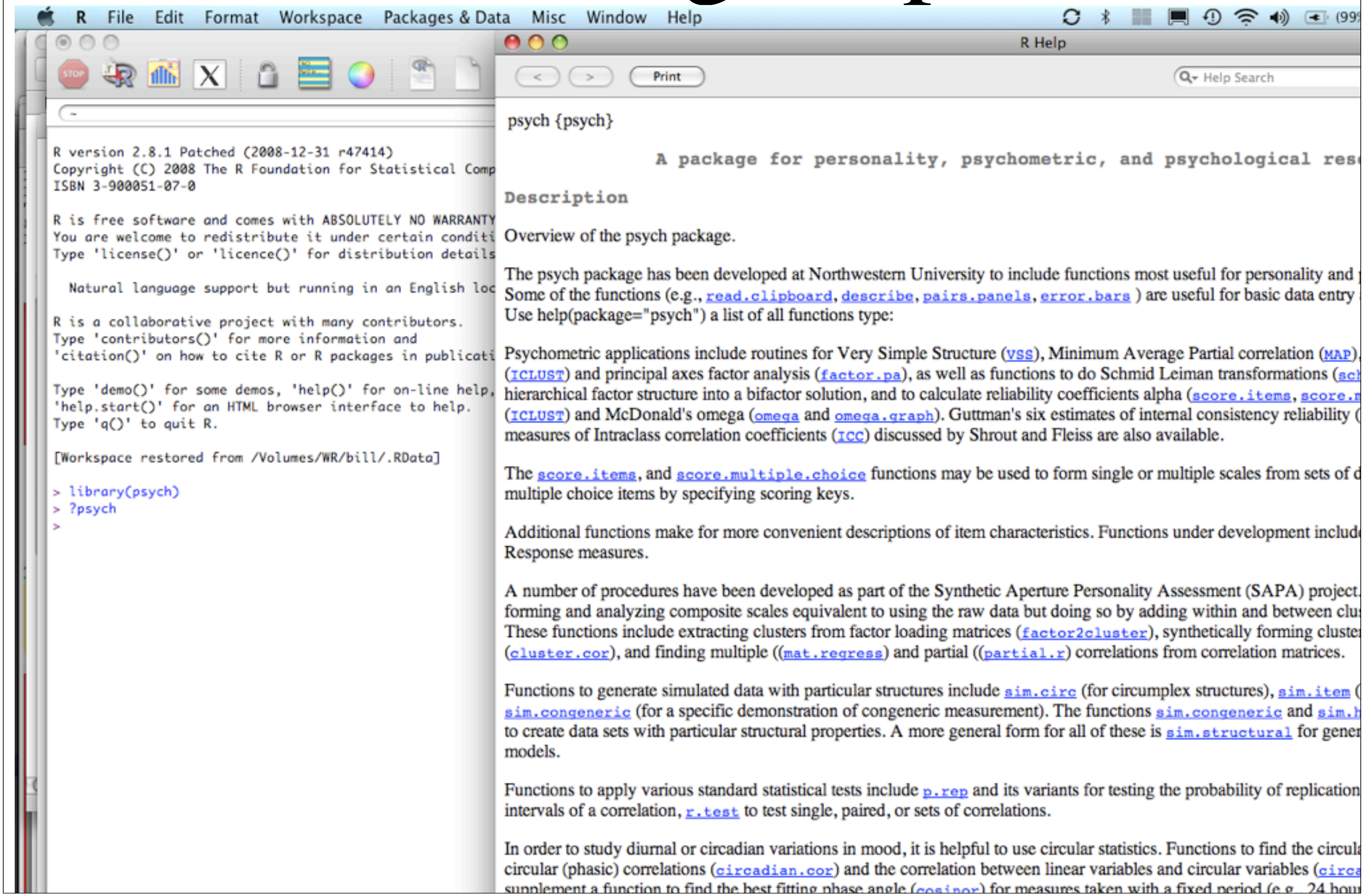
xterm

```
bash-3.2$ R

R version 2.9.0 alpha (2009-03-23 r48200)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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

  Natural language support but running in an English locale

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

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

[Previously saved workspace restored]

> library(psych)
> 
```

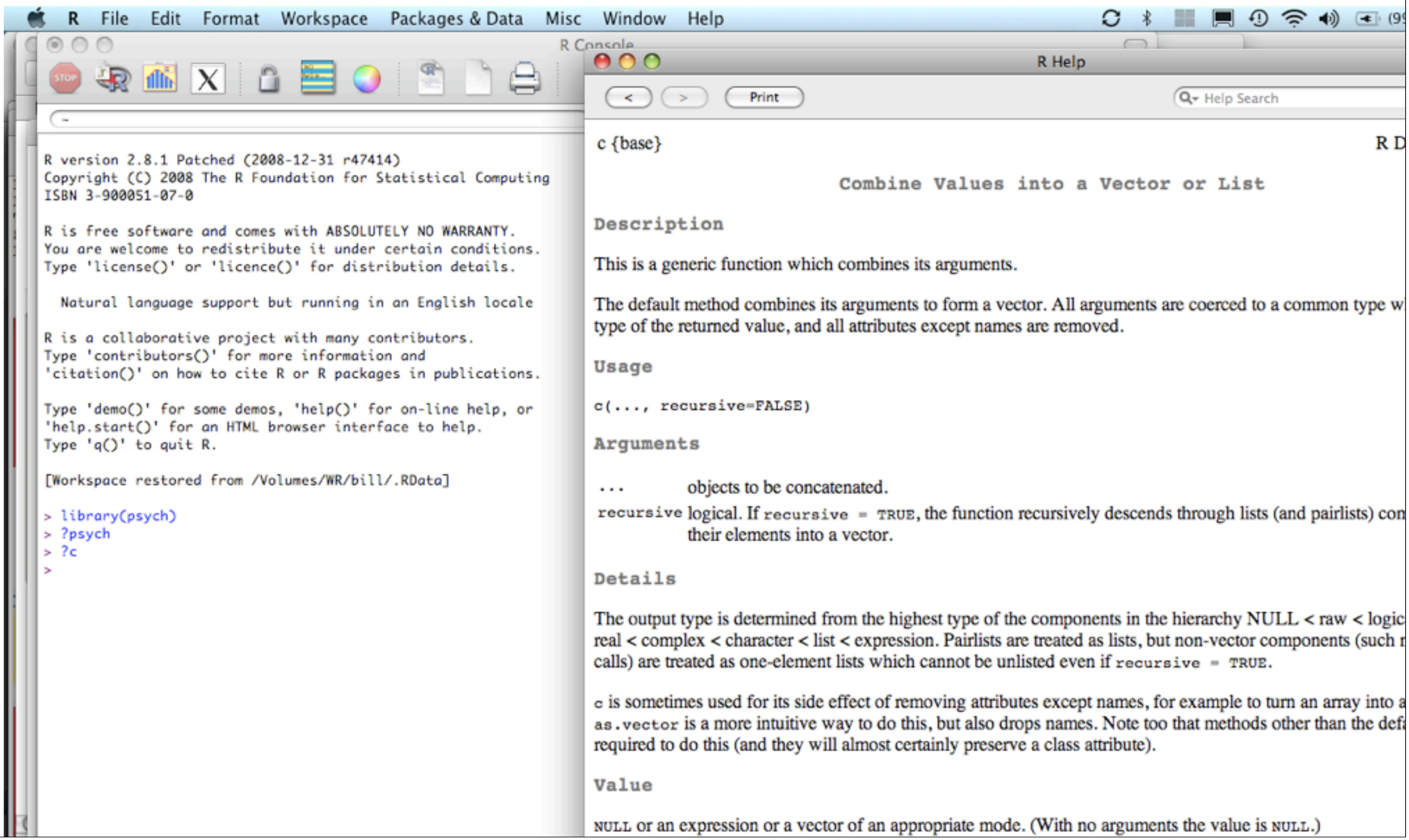
X11


xterm
bash-3.2$ R
R version 2.9.0 alpha (2009-03-23 r48200)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
> library(psych)
> demo(psych)
Error in demo(psych) : No demo found for topic 'psych'
> example(psych)
psych> #See the separate man pages
psych> test.psych()
too many factors requested for this number of variables to use SMC, 1s used instead
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Loading required package: GPArotation
too many factors requested for this number of variables to use SMC, 1s used instead
Loading required package: Rgraphviz
Loading required package: graph
Loading required package: grid
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
R Graphics: Device 2 (ACTIVE)
Symbolic structural model
x1
x2
x3
x4
x5
x6
a
b
c
d
e
f
L1
L2
r1
r2
Y

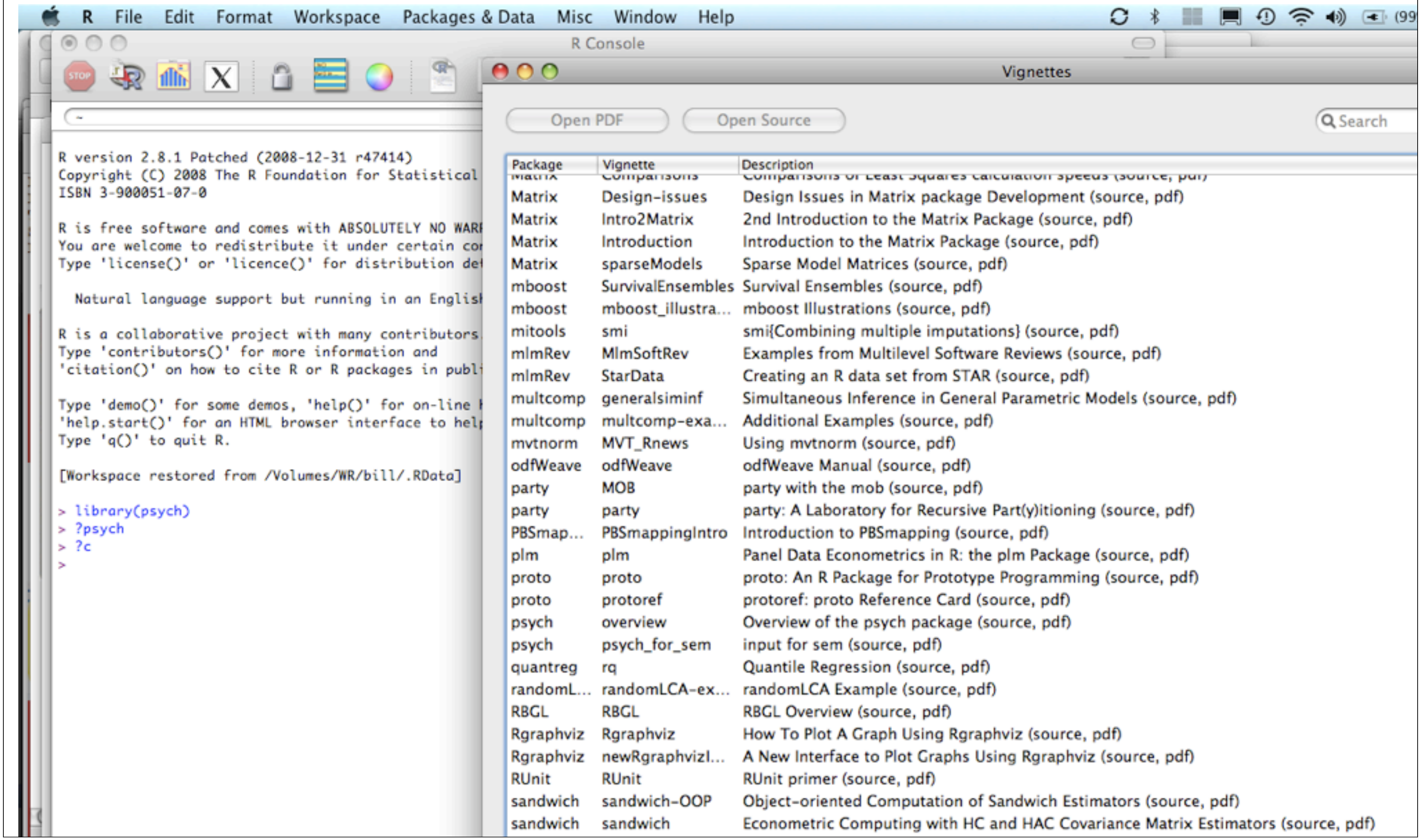
Getting help



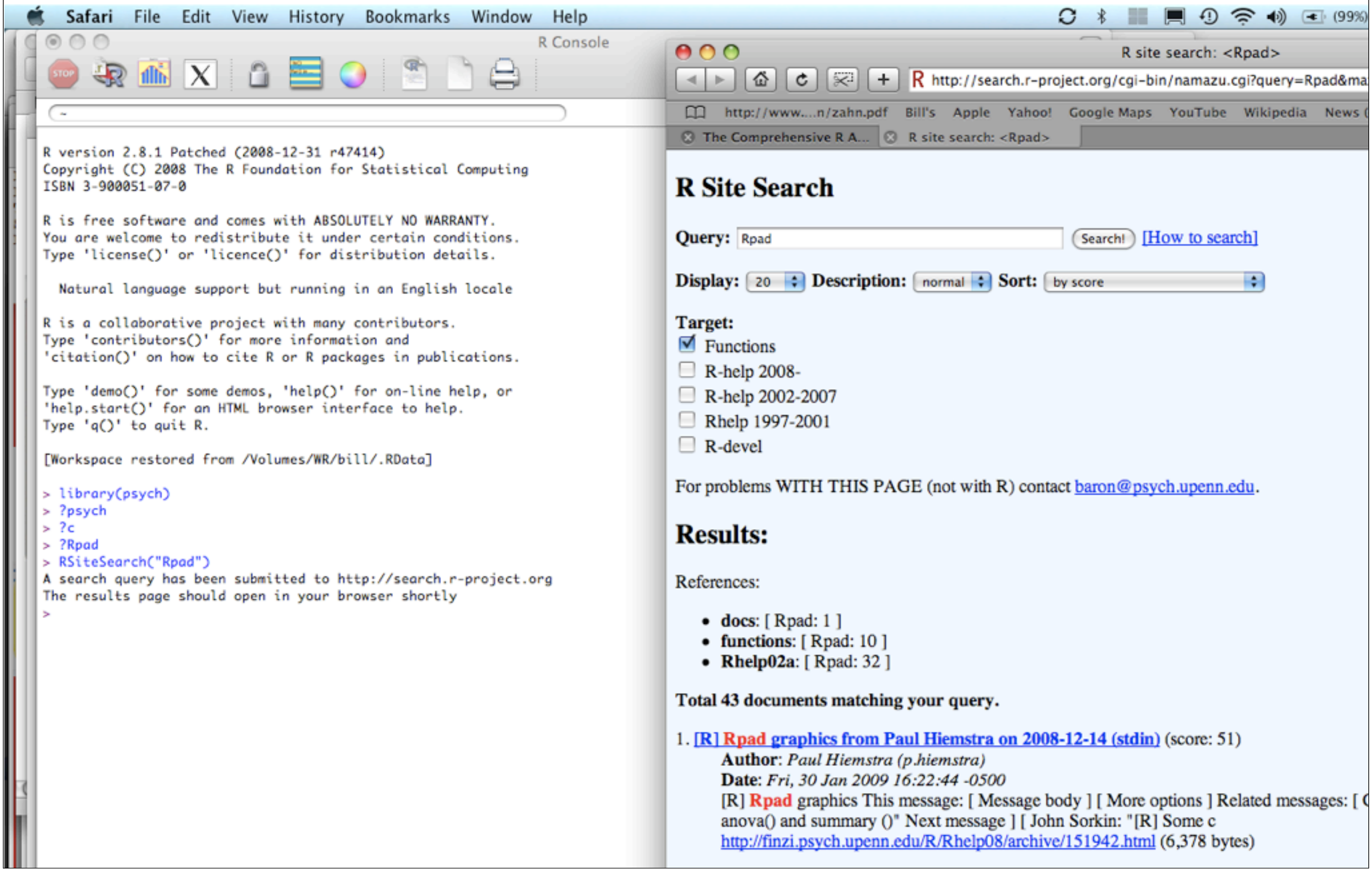
Getting help



Package vignettes



RSiteSearch



Rpad

R/Rpad Reference Card

by Tom Short, EPRI Solutions, Inc., tshort@eprisolutions.com 2005-07-12
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Help and basics

Most R functions have online documentation.
help(topic) documentation on `topic`
?topic id.
help.search("topic") search the help system
apropos("topic") the names of all objects in the search list matching the regular expression "topic"
help.start() start the HTML version of help
str(a) display the internal *str*ucture of an R object
summary(a) gives a "summary" of `a`, usually a statistical summary but it is *generic* meaning it has different operations for different classes of `a`
ls() show objects in the search path; specify `pat="pat"` to search on a pattern
ls.str() str() for each variable in the search path
dir() show files in the current directory
methods(a) shows S3 methods of `a`
methods(class=class(a)) lists all the methods to handle objects of class a
options(...) set or examine many global options; common ones: `width`, `digits`, `error`
library(x) load add-on packages; `library(help=x)` lists datasets and functions in package `x`.
attach(x) database `x` to the R search path; `x` can be a list, data frame, or R data file created with `save`. Use `search()` to show the search path.
detach(x) `x` from the R search path; `x` can be a name or character string of an object previously attached or a package.

Input and output

load() load the datasets written with `save`
data(x) loads specified data sets
read.table(file) reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; use `comment.char=""` to prevent "#" from being interpreted as a comment; use `skip=n` to skip `n` lines before reading data; see the help for options on row naming, NA treatment, and others
read.csv("filename",header=TRUE) id. but with defaults set for reading comma-delimited files
read.delim("filename",header=TRUE) id. but with defaults set

cat(..., file="", sep=" ") prints the arguments after coercing to character; `sep` is the character separator between arguments
print(a, ...) prints its arguments; generic, meaning it can have different methods for different objects
format(x,...) format an R object for pretty printing
write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") prints `x` after converting to a data frame; if `quote` is `TRUE`, character or factor columns are surrounded by quotes ("); `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input
sink(file) output to `file`, until `sink()`
Most of the I/O functions have a `file` argument. This can often be a character string naming a file or a connection. `file=""` means the standard input or output. Connections can include files, pipes, zipped files, and R variables.
On windows, the file connection can also be used with `description = "clipboard"`. To read a table copied from Excel, use
`x <- read.delim("clipboard")`
To write a table to the clipboard for Excel, use
`write.table(x,"clipboard",sep="\t",col.names=NA)`
For database interaction, see packages `RODBC`, `DBI`, `RMySQL`, `RPgSQL`, and `ROracle`. See packages `XML`, `hdf5`, `netCDF` for reading other file formats.

Data creation

c(...) generic function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector
from:to generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"
seq(from,to) generates a sequence `by=` specifies increment; `length=` specifies desired length
seq(along=x) generates `1, 2, ..., length(x)`; useful for `for` loops
rep(x,times) replicate `x` `times`; use `each=` to repeat "each" element of `x` `each` times; `rep(c(1,2,3),2)` is 1 2 3 1 2 3; `rep(c(1,2,3),each=2)` is 1 1 2 2 3 3
data.frame(...) create a data frame of the named or unnamed arguments; `data.frame(v=1:4,ch=c("a","B","c","d"),n=10)`; shorter vectors are recycled to the length of the longest
list(...) create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=3i)`;
array(x,dim=) array with data `x`; specify dimensions like `dim=c(3,4,2)`; elements of `x` recycle if `x` is not long enough
matrix(x,nrow=,ncol=) matrix; elements of `x` recycle
factor(x,levels=) encodes a vector `x` as a factor
gl(n,k,length=n*k,labels=1:n) generate levels (factors) by specifying the pattern of their levels; `k` is the number of levels, and `n` is the number of replications
expand.grid() a data frame from all combinations of the supplied vectors or factors
rbind(...) combine arguments by rows for matrices, data frames, and others

Slicing and extracting data

Indexing lists

`x[n]`	list with elements `n`
`x[[n]]`	n^{th} element of the list
`x[["name"]]`	element of the list named `"name"`
`x$name`	id.

Indexing vectors

`x[n]`	n^{th} element
`x[-n]`	all *but* the n^{th} element
`x[1:n]`	first `n` elements
`x[-(1:n)]`	elements from `n+1` to the end
`x[c(1,4,2)]`	specific elements
`x["name"]`	element named `"name"`
`x[x > 3]`	all elements greater than 3
`x[x > 3 & x < 5]`	all elements between 3 and 5
`x[x %in% c("a","and","the")]`	elements in the given set

Indexing matrices

`x[i,j]`	element at row `i`, column `j`
`x[i,]`	row `i`
`x[,j]`	column `j`
`x[,c(1,3)]`	columns 1 and 3
`x["name",]`	row named `"name"`

Indexing data frames (matrix indexing plus the following)

`x[["name"]]`	column named `"name"`
`x$name`	id.

Variable conversion

as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.complex(x), as.charac
... convert type; for a complete list, use `methods(as)`

Variable information

is.na(x), is.null(x), is.array(x), is.data.fr is.numeric(x), is.complex(x), is.charac
... test for type; for a complete list, use `methods(is)`
length(x) number of elements in `x`
dim(x) Retrieve or set the dimension of an object; `dim(x) <- c`
dimnames(x) Retrieve or set the dimension names of an object
nrow(x) number of rows; `NROW(x)` is the same but treats a vecto row matrix
ncol(x) and **NCOL(x)** id. for columns
class(x) get or set the class of `x`; `class(x) <- "myclass"`
unclass(x) remove the class attribute of `x`
attr(x,which) get or set the attribute `which` of `x`
attributes(obj) get or set the list of attributes of `obj`

Data selection and manipulation

which.max(x) returns the index of the greatest element of x

Data Manipulation
Data Entry

I. from console
II. from clipboard (copied from other programs)
III. from file (text files, csv, SPSS, Excel, MySQL)
IV. from the web

R is a desk calculator

```
> 2+2
[1] 4
> 3^4
[1] 81
> pi
[1] 3.141593
> x <- c(1,3,5,7)
> x
[1] 1 3 5 7
> m <- mean(x)
> m
[1] 4
> mean(x)
[1] 4
> sd(x)
[1] 2.581989
>
```

A very fancy desk calculator

```
> set.seed(42)
> V <- seq(1:5)
> M <- matrix(sample(5,15),replace=TRUE,ncol=3,nrow=5)
Error in matrix(sample(5, 15), replace = TRUE, ncol = 3, nrow = 5) :
  unused argument(s) (replace = TRUE)
> M <- matrix(sample(5,15,replace=TRUE),ncol=3,nrow=5)
> V
[1] 1 2 3 4 5
> M
     [,1] [,2] [,3]
[1,]    5    3    3
[2,]    5    4    4
[3,]    2    1    5
[4,]    5    4    2
[5,]    4    4    3
```

But a calculator none the less

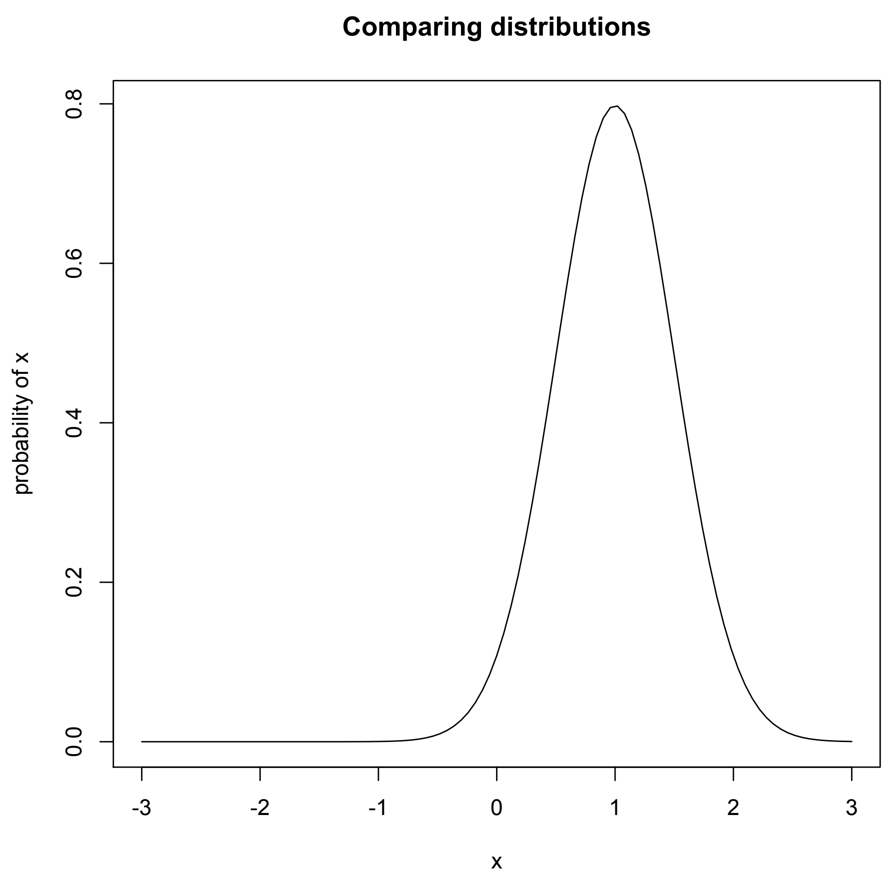
```
> V
[1] 1 2 3 4 5
> M
     [,1] [,2] [,3]
[1,]    5    3    3
[2,]    5    4    4
[3,]    2    1    5
[4,]    5    4    2
[5,]    4    4    3
```

```
> V * M
     [,1] [,2] [,3]
[1,]    5    3    3
[2,]   10    8    8
[3,]    6    3   15
[4,]   20   16    8
[5,]   20   20   15
```

```
> t(M) %*% M
     [,1] [,2] [,3]
[1,]   95   73   67
[2,]   73   58   50
[3,]   67   50   63
```

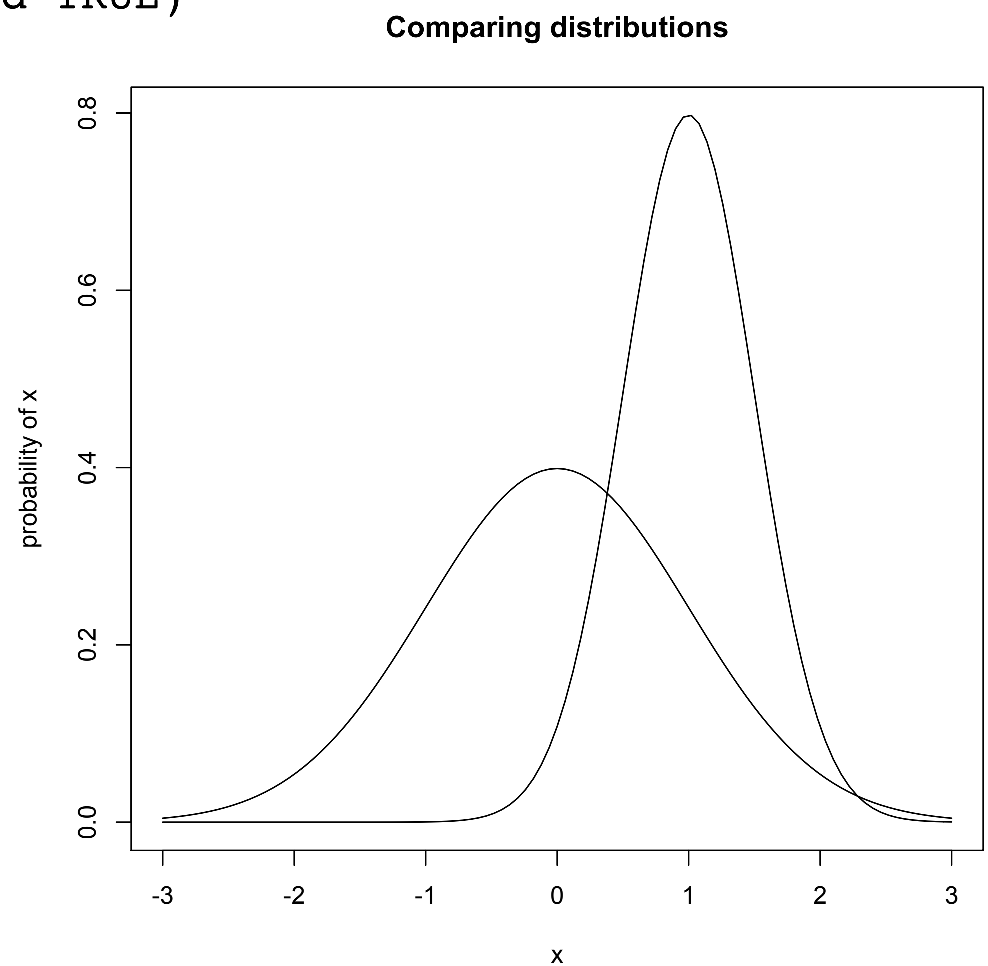
A graphing calculator

```
> curve(dnorm(x,1,0.5),-3,3,ylab="probability of
x",main="Comparing distributions")
```



Add the second line

```
> curve(dnorm(x,1,0.5),-3,3,ylab="probability of x",main="Comparing
distributions")
> curve(dnorm(x,0,1),add=TRUE)
```



R is a stats table

```
> pt(2.0,6) #probability (one tailed of a t > 2.0)
[1] 0.9537868
> pnorm(2.0) #probability of a normal distribution
with value of 2.0)
[1] 0.9772499
> dnorm(-1)  #normal with z value of -1.0
[1] 0.2419707
> pf(3.5,1,20)  #probability of an F statistic
[1] 0.923926
> qf(.95,1,60) # the critical value of an F at the 95
percent value
[1] 4.001191
> qchisq(.95,1) # the critical value for a 1 df #$
\chi^2$
[1] 3.841459
```

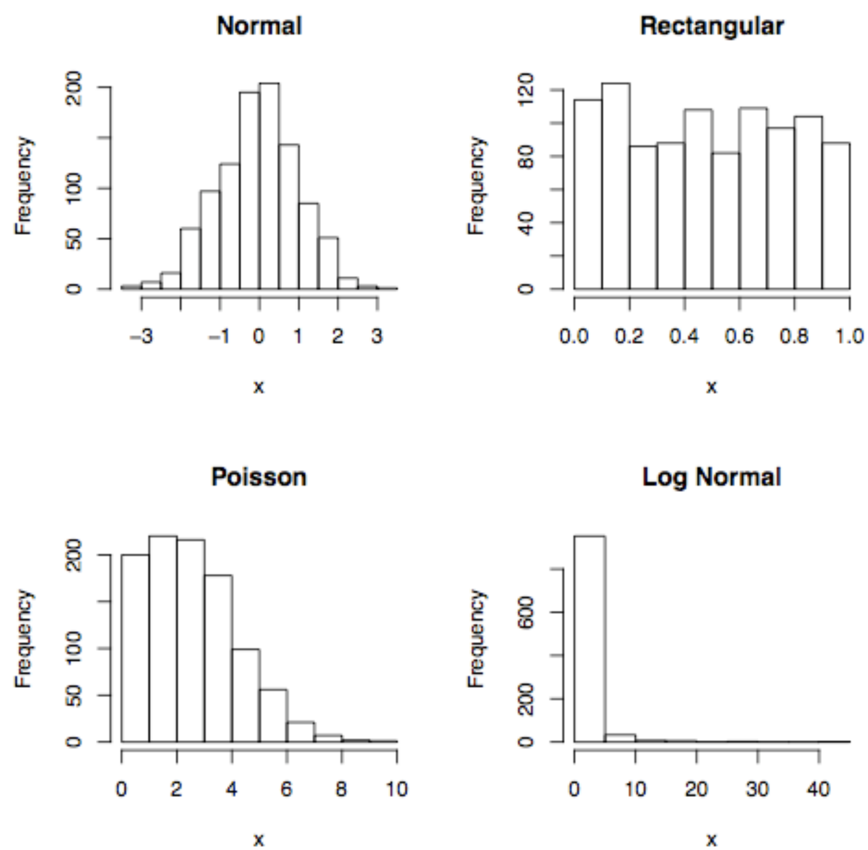
A powerful stats table

```
> z <- seq(0,3,.2)
> z
 [1] 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.
2.8 3.0
> pnorm(z)
 [1] 0.5000000 0.5792597 0.6554217 0.7257469 0.7881446
0.8413447 0.8849303 0.9192433 0.9452007 0.9640697 0.9772499
0.9860966
[13] 0.9918025 0.9953388 0.9974449 0.9986501
> round(data.frame(z,p=pnorm(z)),3)
     z     p
1  0.0 0.500
2  0.2 0.579
3  0.4 0.655
4  0.6 0.726
5  0.8 0.788
6  1.0 0.841
7  1.2 0.885
8  1.4 0.919
```

Random data

Distribution	base name	P 1	P 2	P 3	example application
Normal	norm	mean	sigma		Most data
Multivariate normal	mvnorm	mean	r	sigma	Most data
Log Normal	lnorm	log mean	log sigma		income or reaction time
Uniform	unif	min	max		rectangular distributions
Binomial	binom	size	prob		Bernuilli trials (e.g. coin flips)
Student's t	t	df		nc	Finding significance of a t-test
Multivariate t	mvt	df	corr	nc	Multivariate applications
Fisher's F	f	df1	df2	nc	Testing for significance of F test
χ^2	chisq	df		nc	Testing for significance of χ^2
Beta	beta	shape1	shape2	nc	distribution theory
Cauchy	cauchy	location	scale		Infinite variance distribution
Exponential	exp	rate			Exponential decay
Gamma	gamma	shape	rate	scale	distribution theoryh
Hypergeometric	hyper	m	n	k	
Logistic	logis	location	scale		Item Response Theory
Poisson	pois	lambda			Count data
Weibull	weibull	shape	scale		Reaction time distributions

Graphics of random data



Graphing commands

```
op <- par(mfrow=c(2,2))

n <- 1000
x <- rnorm(n)
hist(x,main="Normal")
x <- runif(n)
hist(x,main="Rectangular")
x <- rpois(n,3)
hist(x,main="Poisson")
x <- rlnorm(n)
hist(x,main="Log Normal")

op <- par(mfrow=c(1,1))
```

Data Entry example

```
> 5/2              #it is a calculator
[1] 2.5
> 2^8            #still a calculator
[1] 256
> a <- c("A","short","list")  #store this value
> a                             #now show it
[1] "A"     "short" "list"
>
```

Data entry

```
> A <- 1:25  # make a sequence
> A              #show it
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25
> B <- matrix(A,ncol=5) #convert the vector to a matrix
> B                # show it
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
> c <- seq(25,5,by=-5)   #seq can do more interesting stuff
> c
[1] 25 20 15 10  5
```

From the clipboard

Copy this text

```
   V1 V2 V3 V4 V5
S1  1  6 11 16 21
S2  2  7 12 17 22
S3  3  8 13 18 23
S4  4  9 14 19 24
S5  5 10 15 20 25
```

library(psych)

> A <- read.clipboard()

> A   #show it

```
   V1 V2 V3 V4 V5
S1  1  6 11 16 21
S2  2  7 12 17 22
S3  3  8 13 18 23
S4  4  9 14 19 24
S5  5 10 15 20 25
```

From the clipboard (no headers)

```
1  6 11 16 21
 2  7 12 17 22
 3  8 13 18 23
 4  9 14 19 24
 5 10 15 20 25
```

> C <- read.clipboard(header=FALSE)

> C #automatically adds column names!

```
  V1 V2 V3 V4 V5
1  1  6 11 16 21
2  2  7 12 17 22
3  3  8 13 18 23
4  4  9 14 19 24
5  5 10 15 20 25
```

From built in datasets

```
> data() #show all data sets

> data(sat.act) # get this one

> head(sat.act) #show the first lines

> dim(sat.act)  #show dimensions

[1] 700   6
```

```
      gender education age ACT SATV SATQ
29442      2         3  19  24  500  500
29457      2         3  23  35  600  500
29498      2         3  20  21  480  470
29503      1         4  27  26  550  520
29504      1         2  33  31  600  550
29518      1         5  26  28  640  640
```

From files or the web

```
Fn <- file.choose()  #uses the system commands to find it

My.data <- read.table(Fn)

> My.data <- read.table(Fn,header=TRUE)

> head(My.data)     #show the first 6 lines
```

	epiE	epiS	epiImp	epilie	epiNeur	bfagree	bfcon	bfext	bfneur	bfopen	bdi	traitanx	stateanx
1	18	10	7	3	9	138	96	141	51	138	1	24	22
2	16	8	5	1	12	101	99	107	116	132	7	41	40
3	6	1	3	2	5	143	118	38	68	90	4	37	44
4	12	6	4	3	15	104	106	64	114	101	8	54	40
5	14	6	5	3	2	115	102	103	86	118	8	39	67
6	6	4	2	5	15	110	113	61	54	149	5	51	38

Getting data from the web

```
> datafilename <- "http://personality-project.org/r/datasets/maps.mixx.epi.bfi.data"
> my.data(datafilename)
Error: could not find function "my.data"
> my.data <- read(datafilename)
Error: could not find function "read"
> my.data <- read.table(datafilename,header=TRUE) #read the data file
>
> dim(my.data)
[1] 231  13
> headtail(mydata)
Error in inherits(x, "data.frame") : object "mydata" not found
> headtail(my.data)
    epiE epiS epiImp epilie epiNeur bfagree bfcon bfext bfneur bfopen bdi traitanx stateanx
1     18   10      7      3       9     138    96   141     51    138   1       24       22
2     16    8      5      1      12     101    99   107    116    132   7       41       40
3      6    1      3      2       5     143   118    38     68     90   4       37       44
4     12    6      4      3      15     104   106    64    114    101   8       54       40
...  ...  ...    ...    ...     ...     ...   ...   ...    ...    ... ...      ...      ...
228   12    7      4      3      15     155   129   127     88    110   9       35       34
229   19   10      7      2      11     162   152   163    104    164   1       29       47
230    4    1      1      2      10      95   111    75    123    138   5       39       58
231    8    6      3      2      15      85    62    90    131     96  24       58       58
```

Looking at the data

> headtail(My.data) #a psych function demonstrating the passing of parameters to functions

```
> > headtail(My.data,5,3)
    epiE epiS epiImp epilie epiNeur bfagree bfcon bfext bfneur bfopen bdi traitanx stateanx
1     18   10      7      3       9     138    96   141     51    138   1       24       22
2     16    8      5      1      12     101    99   107    116    132   7       41       40
3      6    1      3      2       5     143   118    38     68     90   4       37       44
4     12    6      4      3      15     104   106    64    114    101   8       54       40
5     14    6      5      3       2     115   102   103     86    118   8       39       67
...  ...  ...    ...    ...     ...     ...   ...   ...    ...    ... ...      ...      ...
229   19   10      7      2      11     162   152   163    104    164   1       29       47
230    4    1      1      2      10      95   111    75    123    138   5       39       58
231    8    6      3      2      15      85    62    90    131     96  24       58       58
```

Editing the data

Fix(My.data) # edits the object

A <- edit(My.data) #edits and then returns the new version

(Or just edit in your favorite editor)

Why R? Graphics

I. Sample graphics taken from

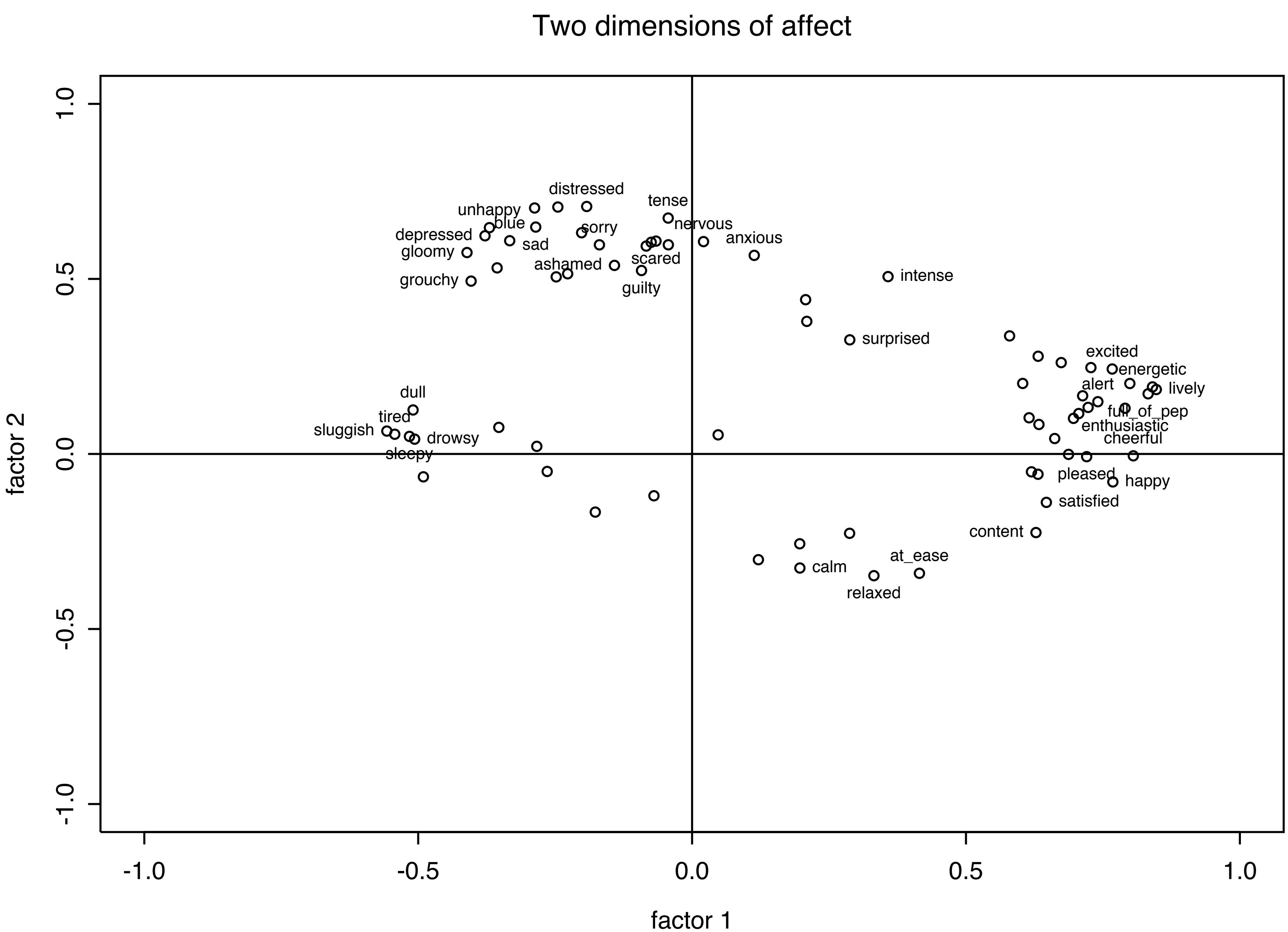
   A. http:personality-project.org/r/

      1. showing what can be done by an amateur

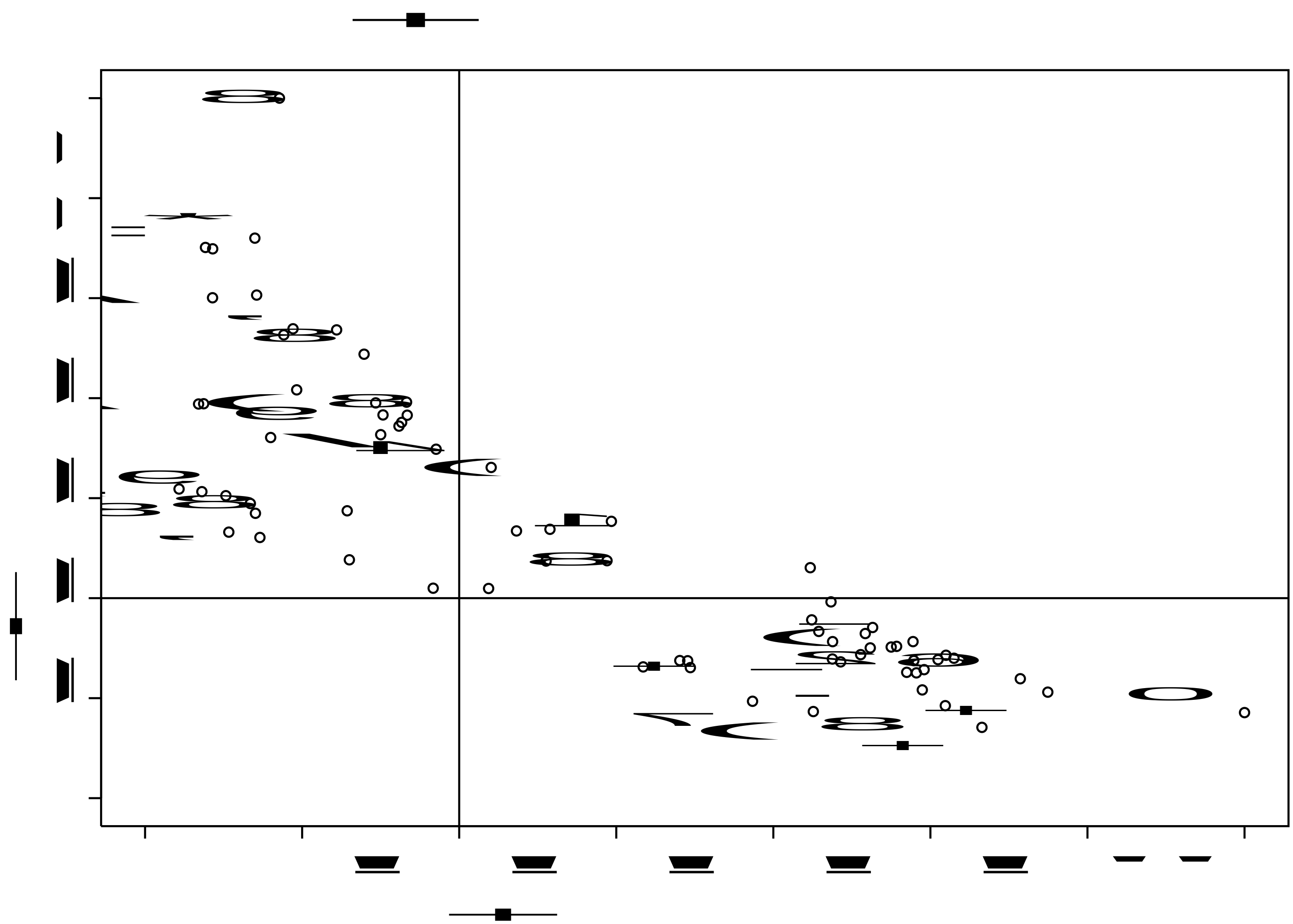
   B. http://addictedtor.free.fr/graphiques/

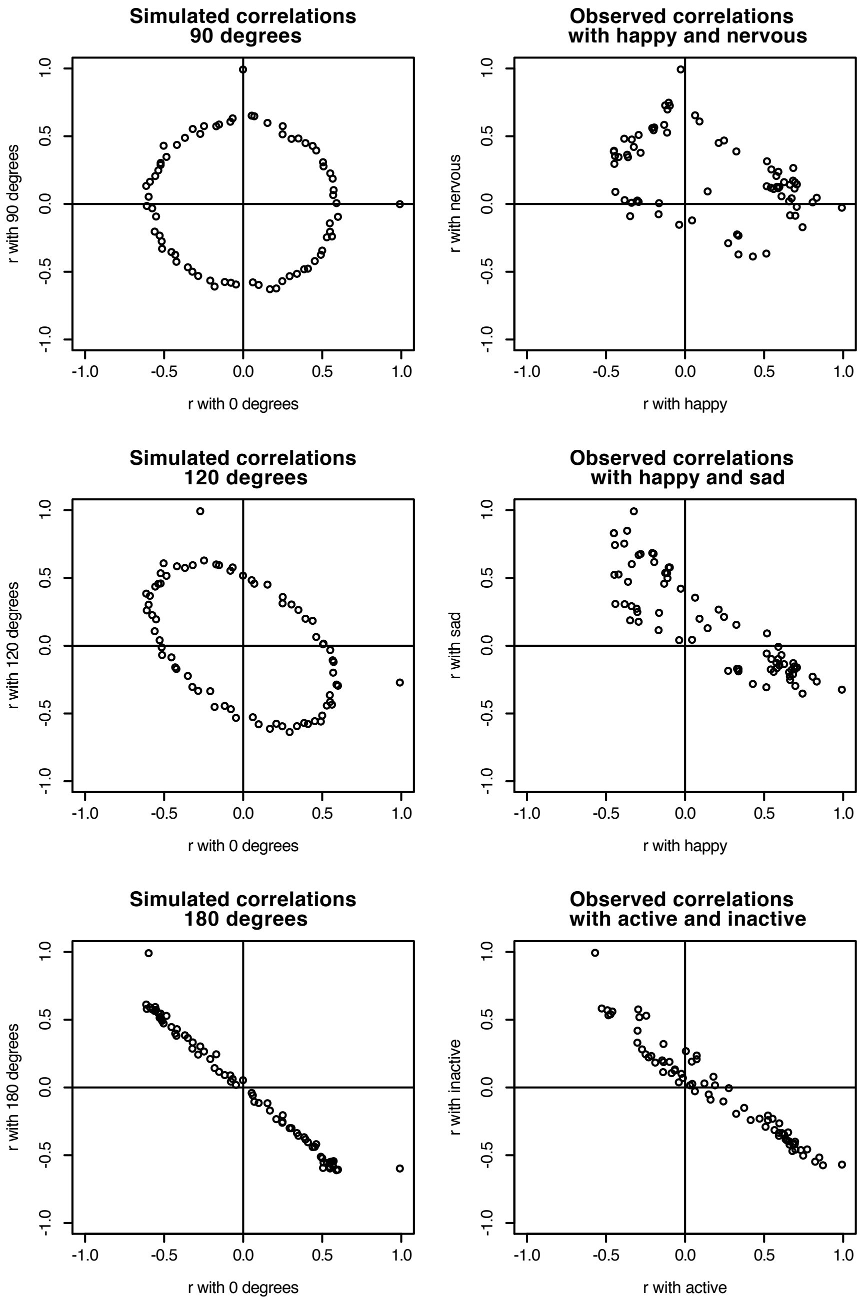
      1. showing some most impressive graphs

Standard Plots of factor loadings



Data points can be dynamically identified

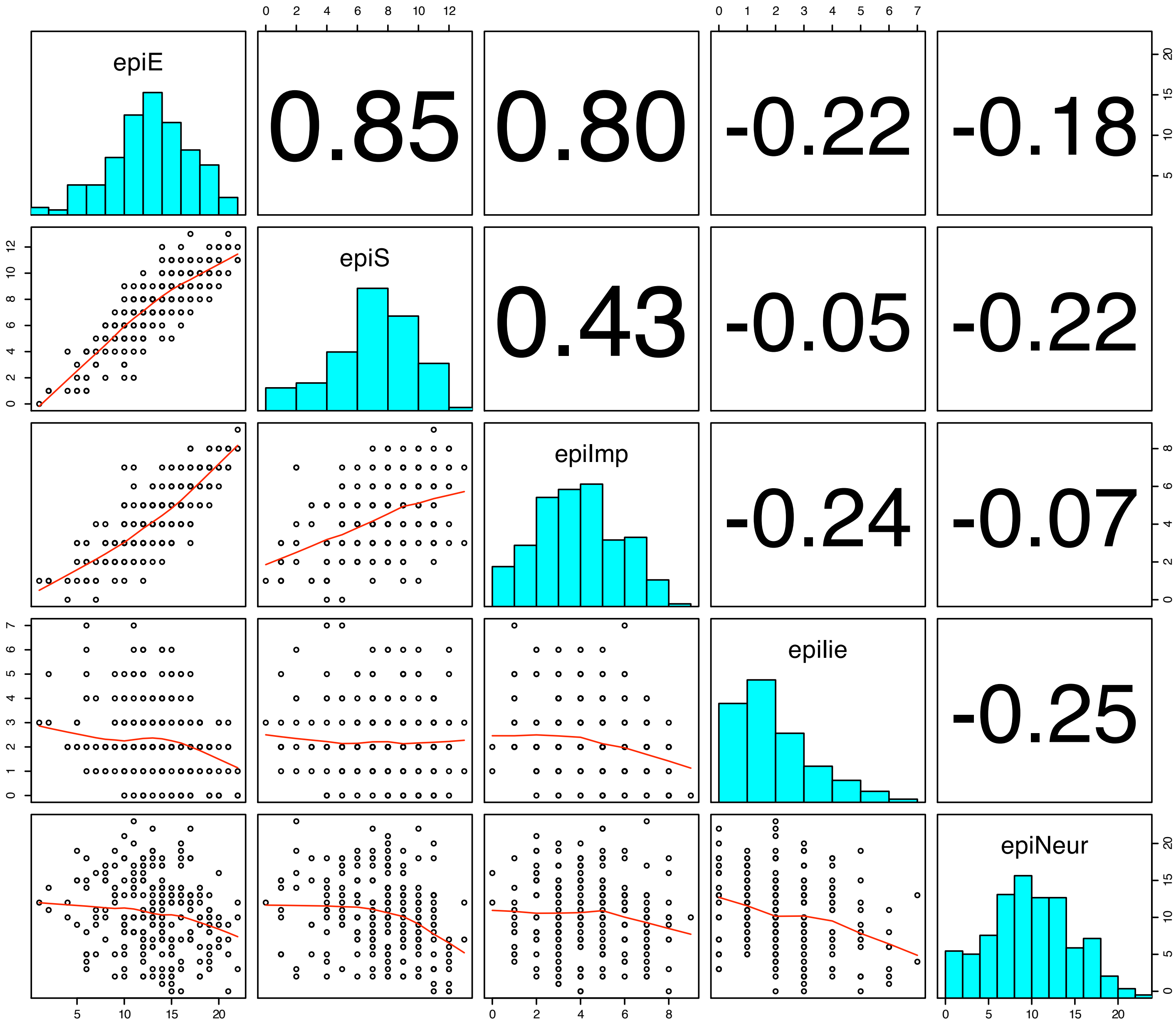




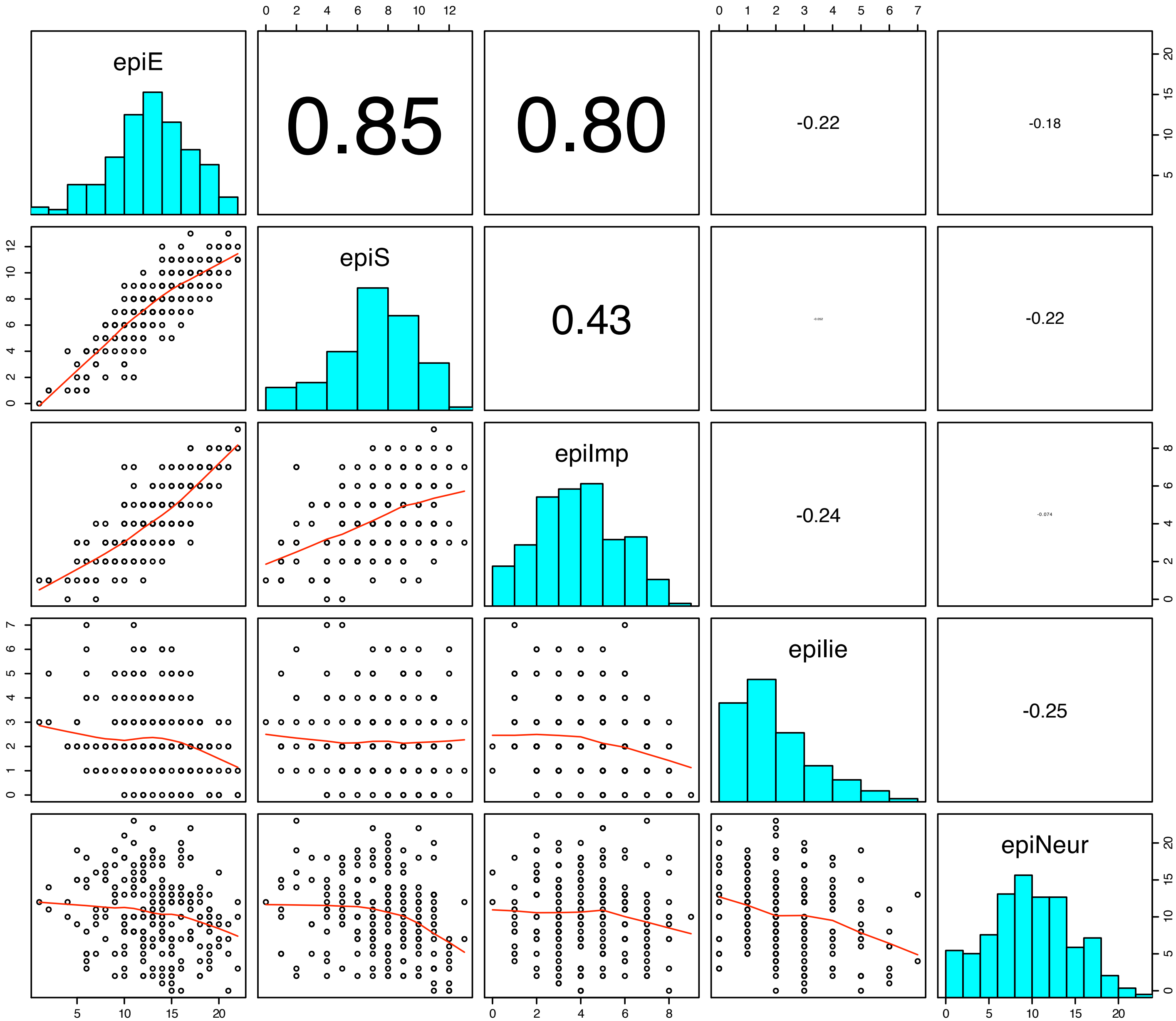
Multi-panel graphs can be labeled separately and organized vertically or horizontally

Simulated data can be generated to fit normal, rectangular, binomial, poissen, exponential, etc. distributions

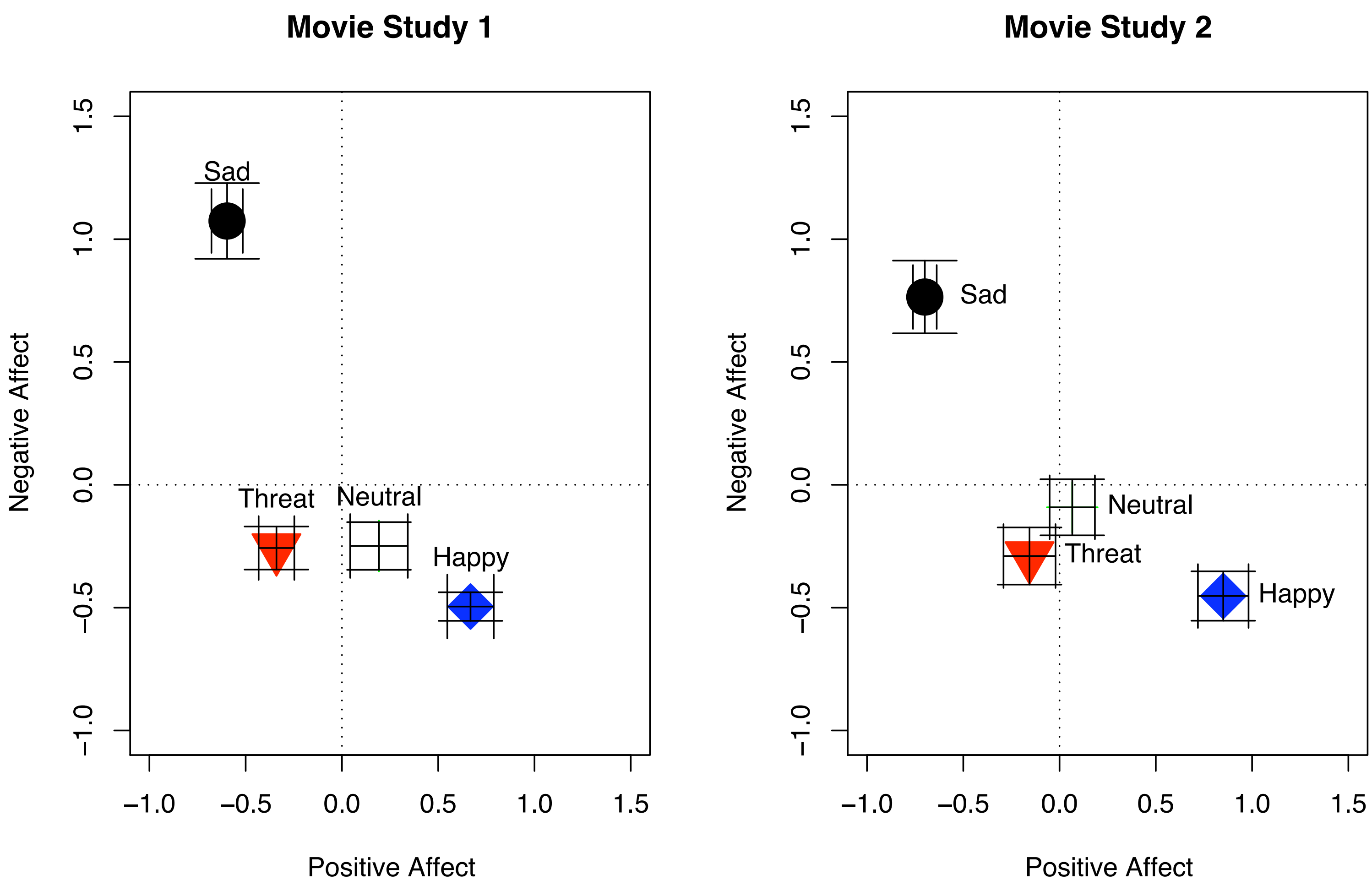
Scatter Plot Matrices can show smoothed fits

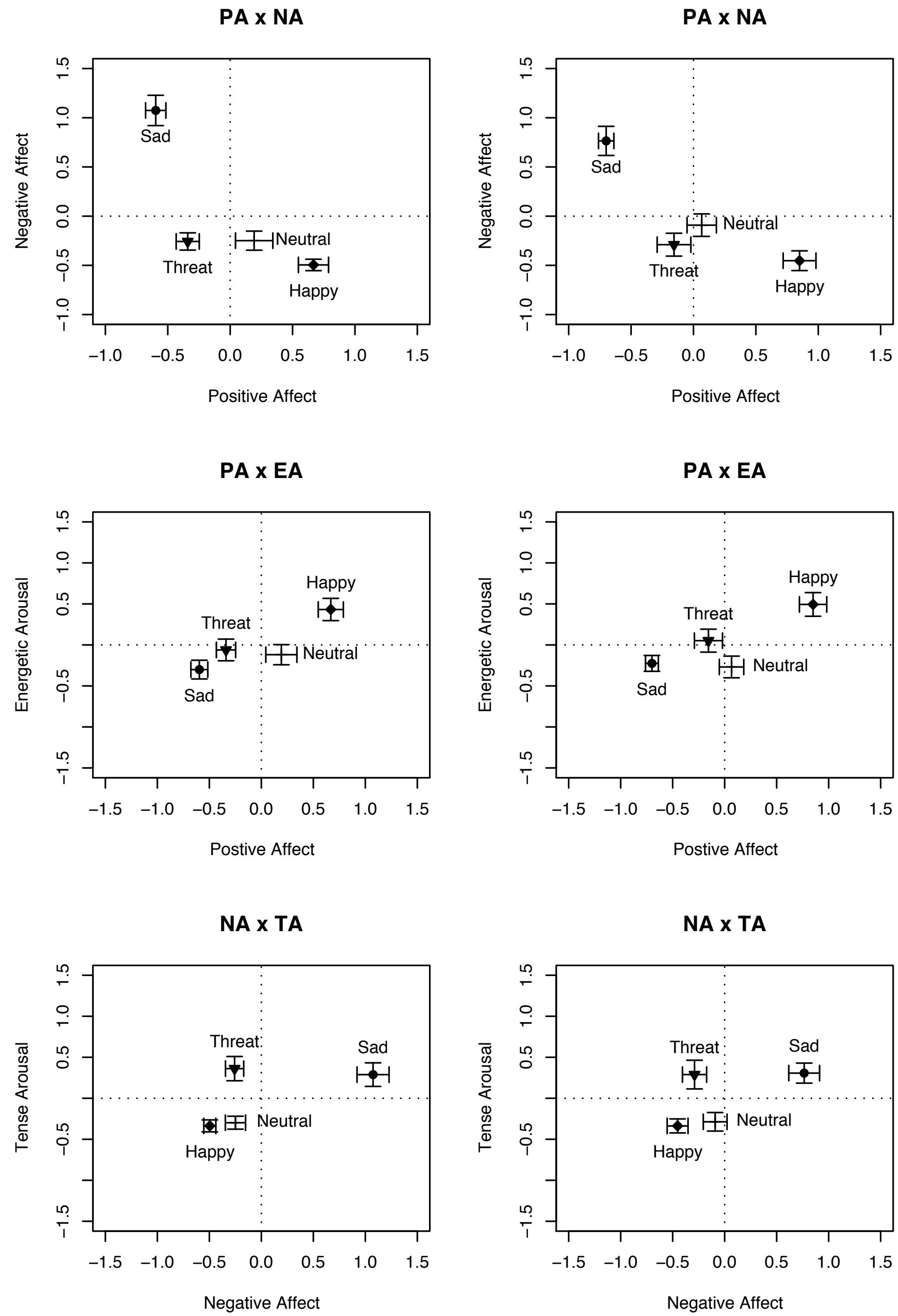


Can scale font size of correlations by absolute size of r



Error bars on two dimensions

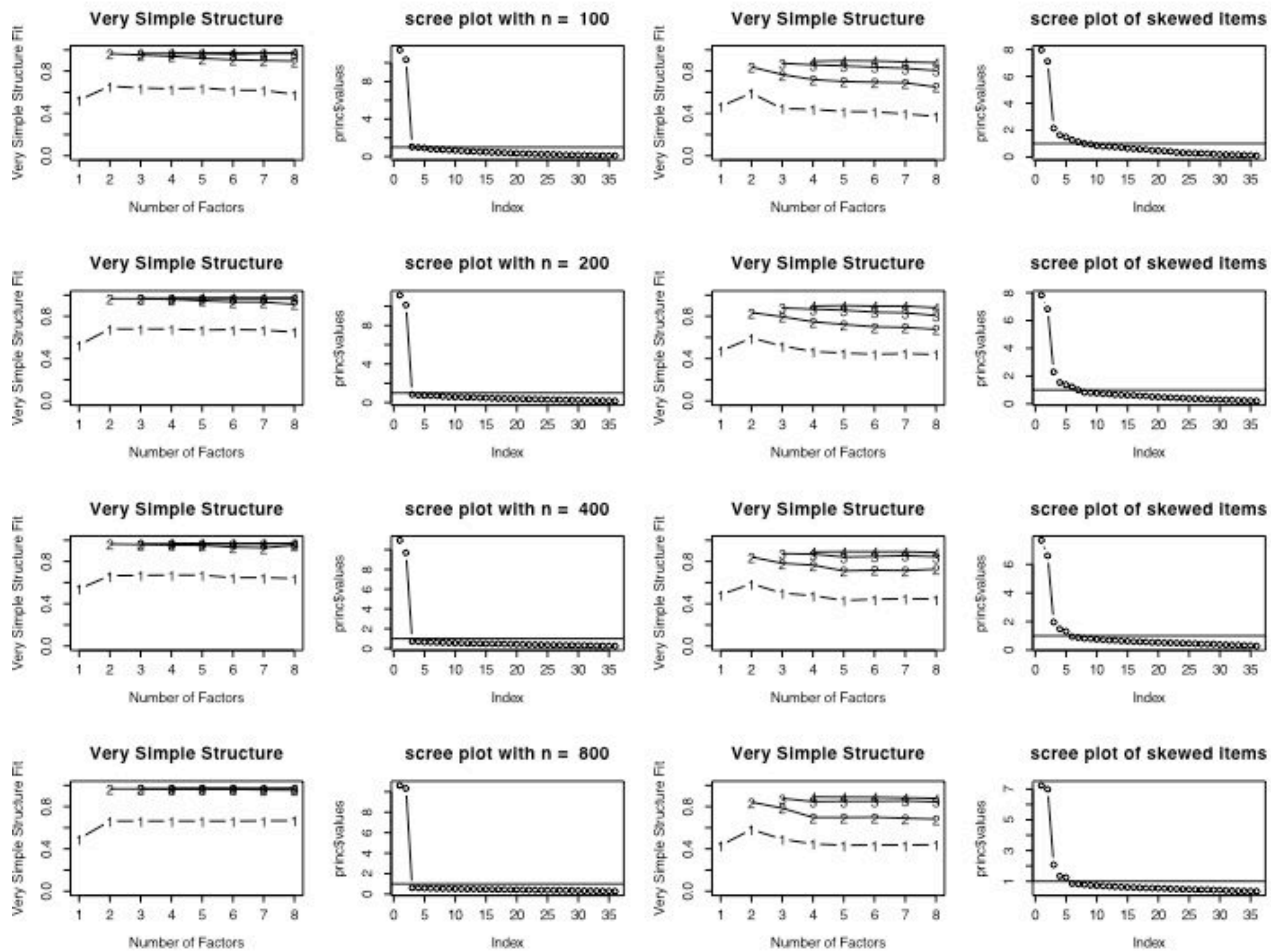




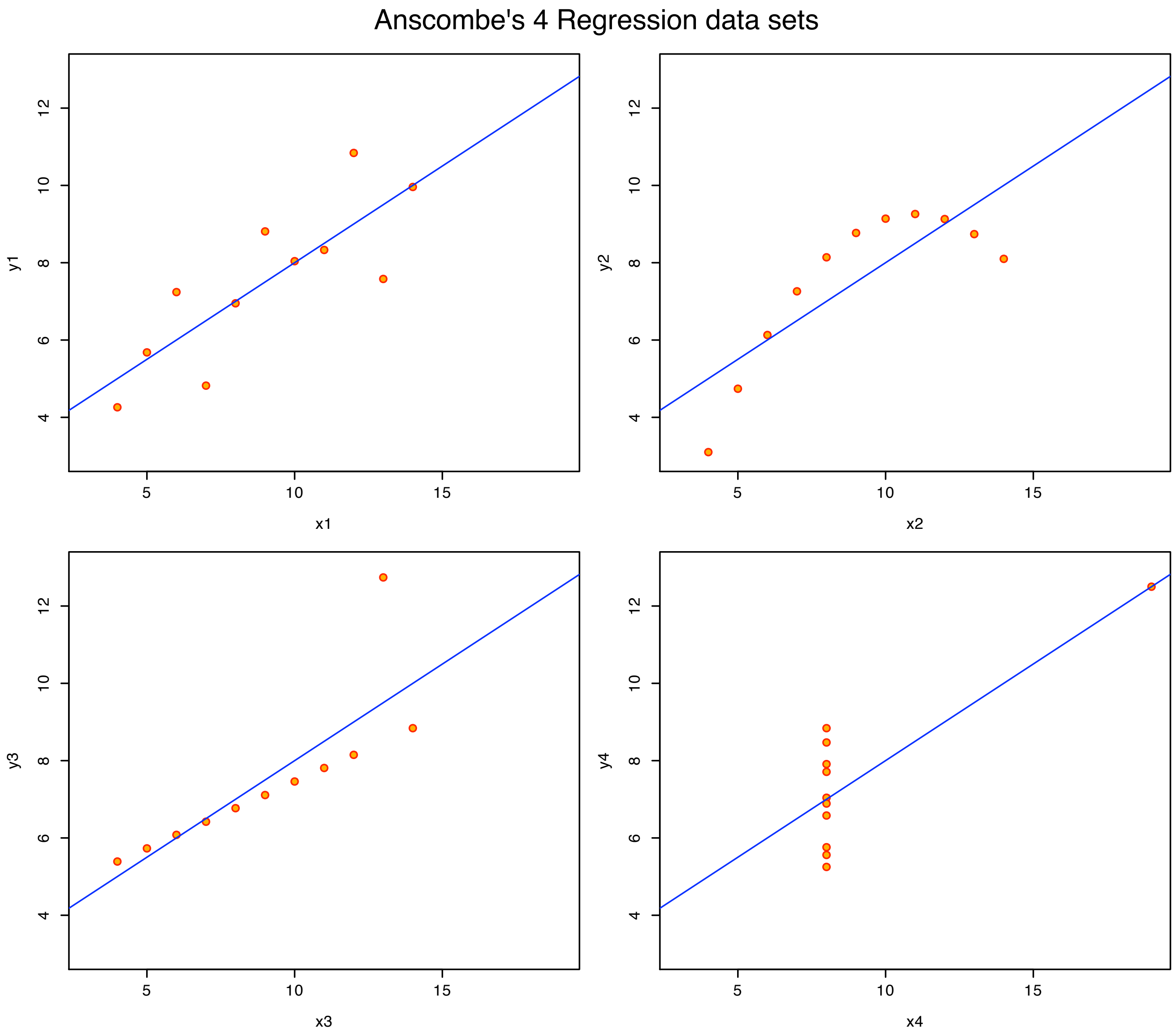
Presentation graphics scale to fit page and produce output for pdf presentations

Window or page size is controllable

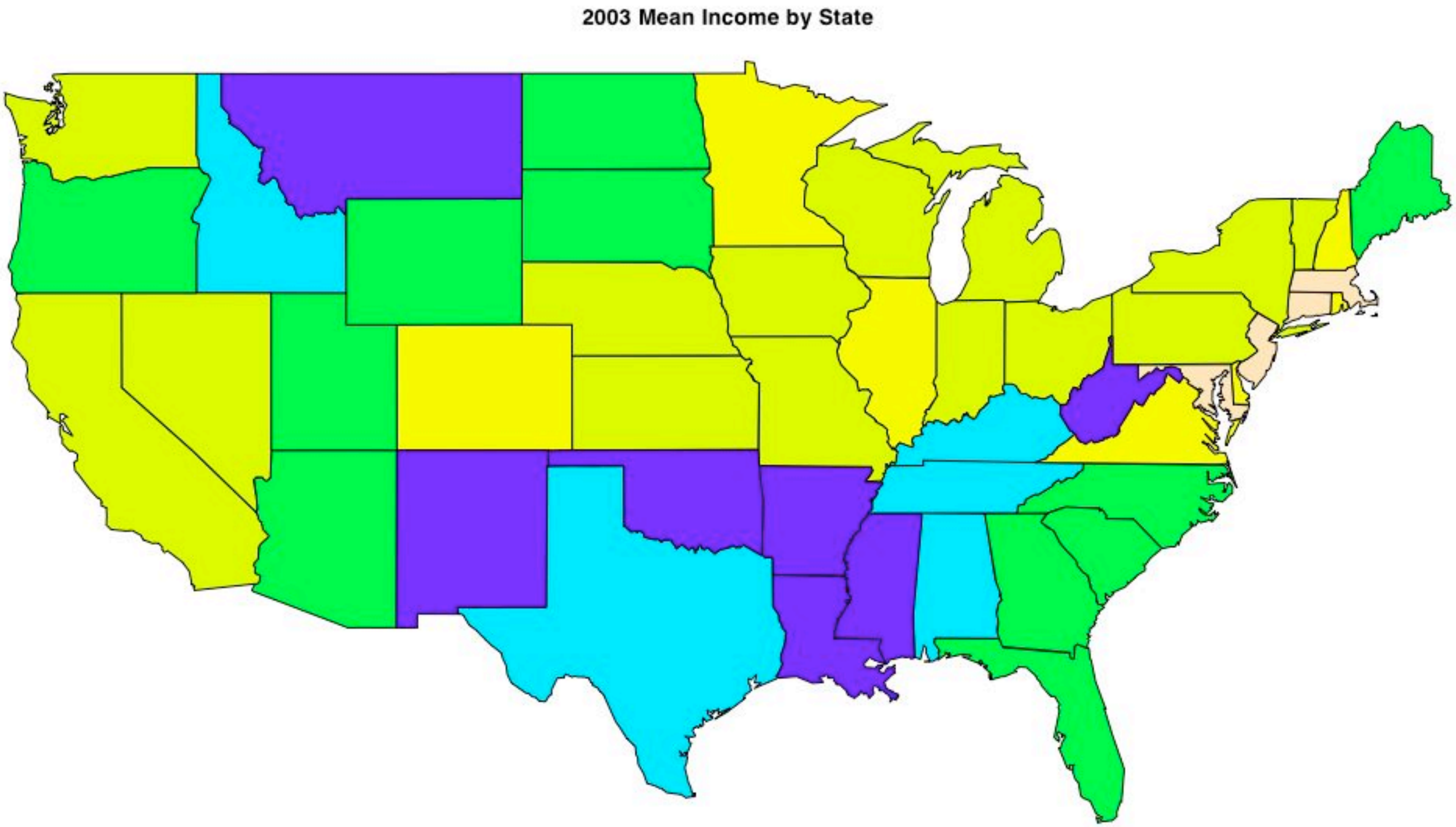
Graphic output can be taken from multiple programs (e.g. Very Simple Structure, factor analysis



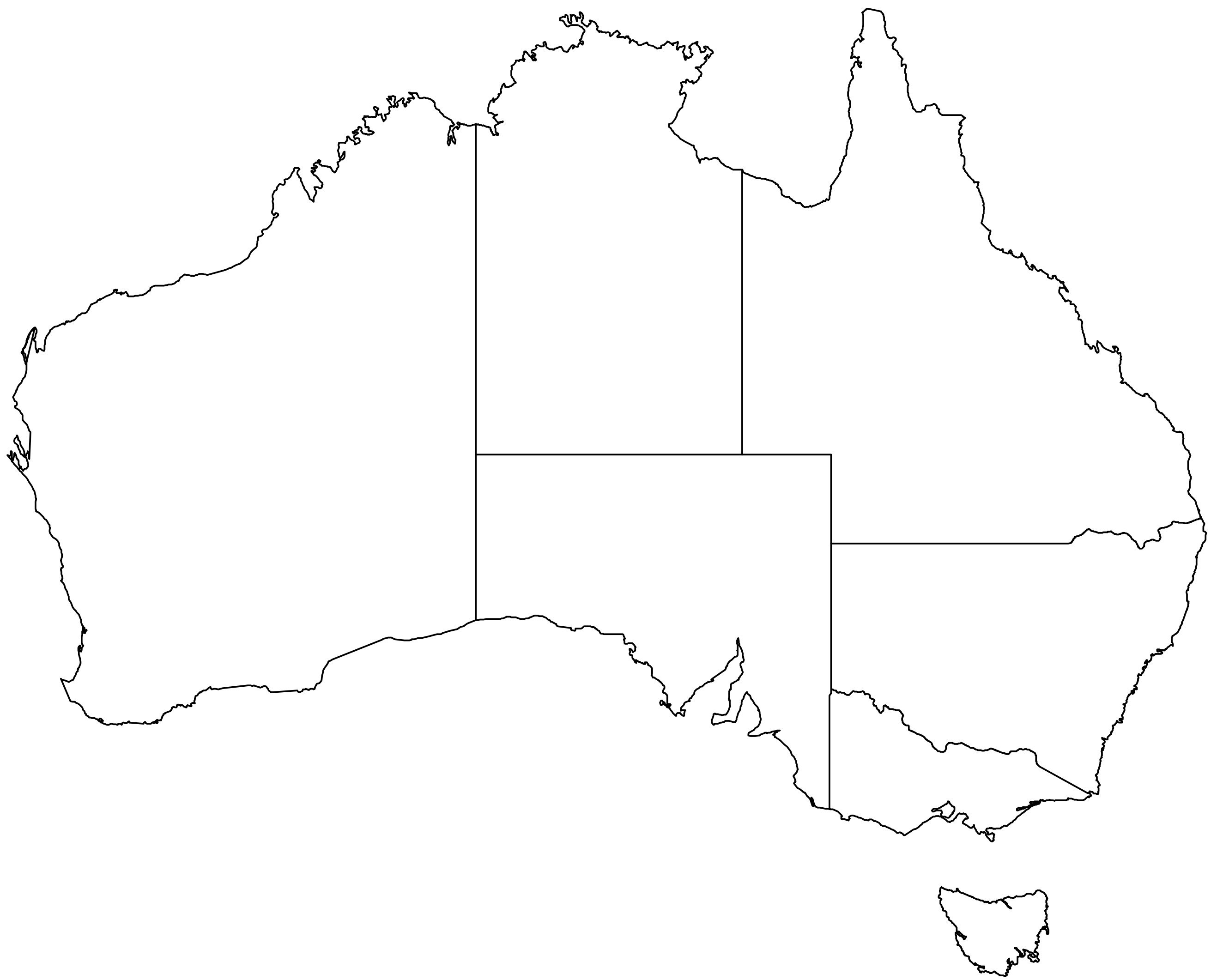
Built-in data sets provide useful demonstrations of stats



Mapping data (GIS) may be combined with descriptive data

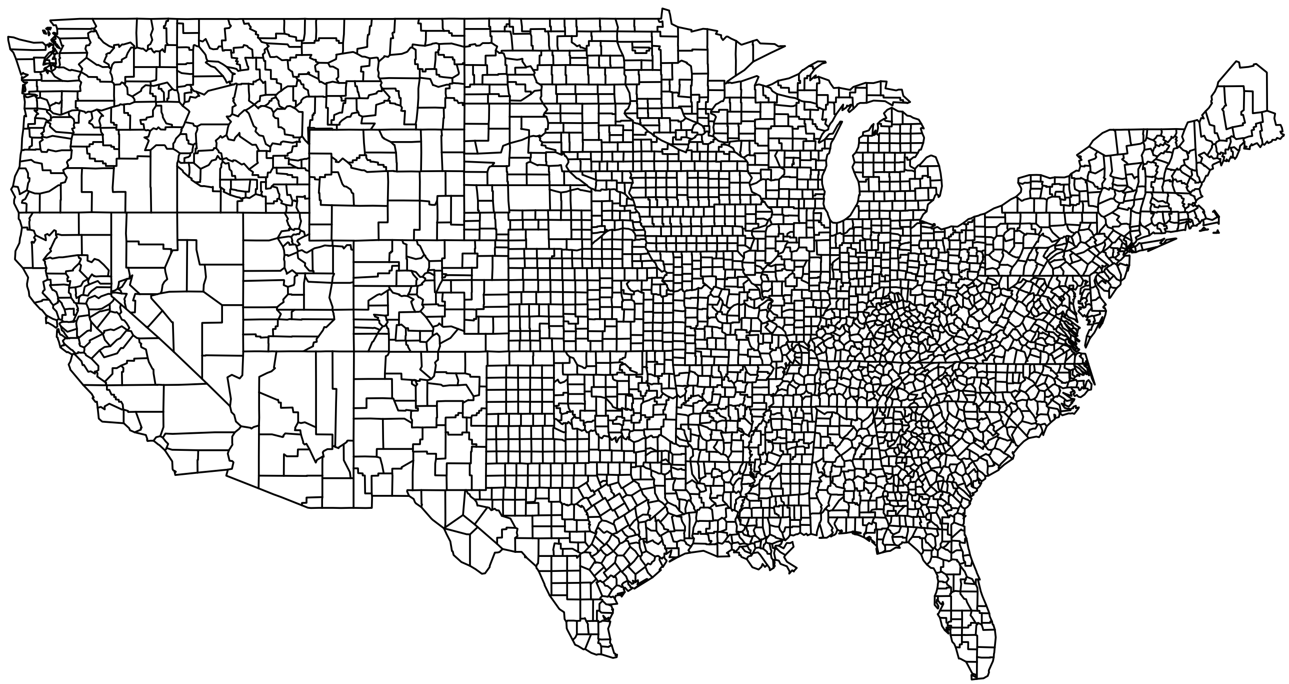


Many GIS map files are available for download from ESRI



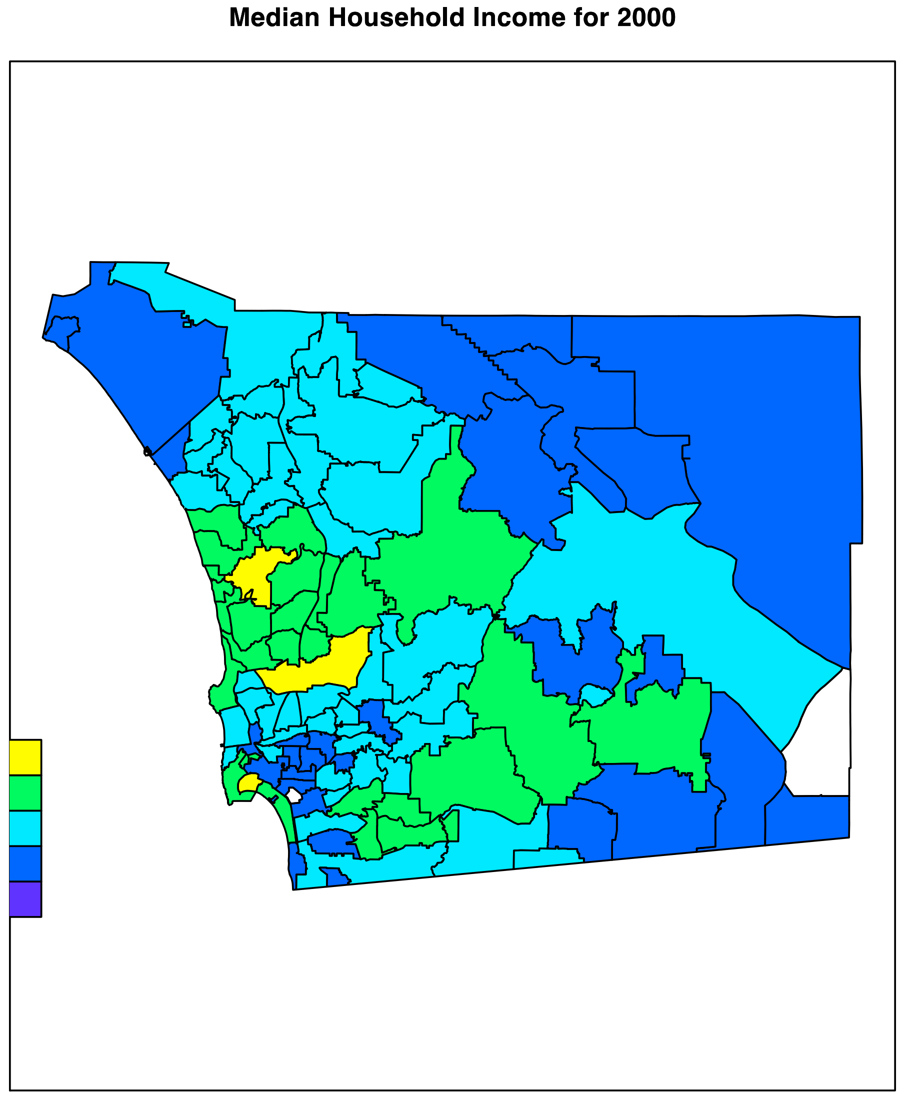
GIS maps detail regional boundaries

US counties



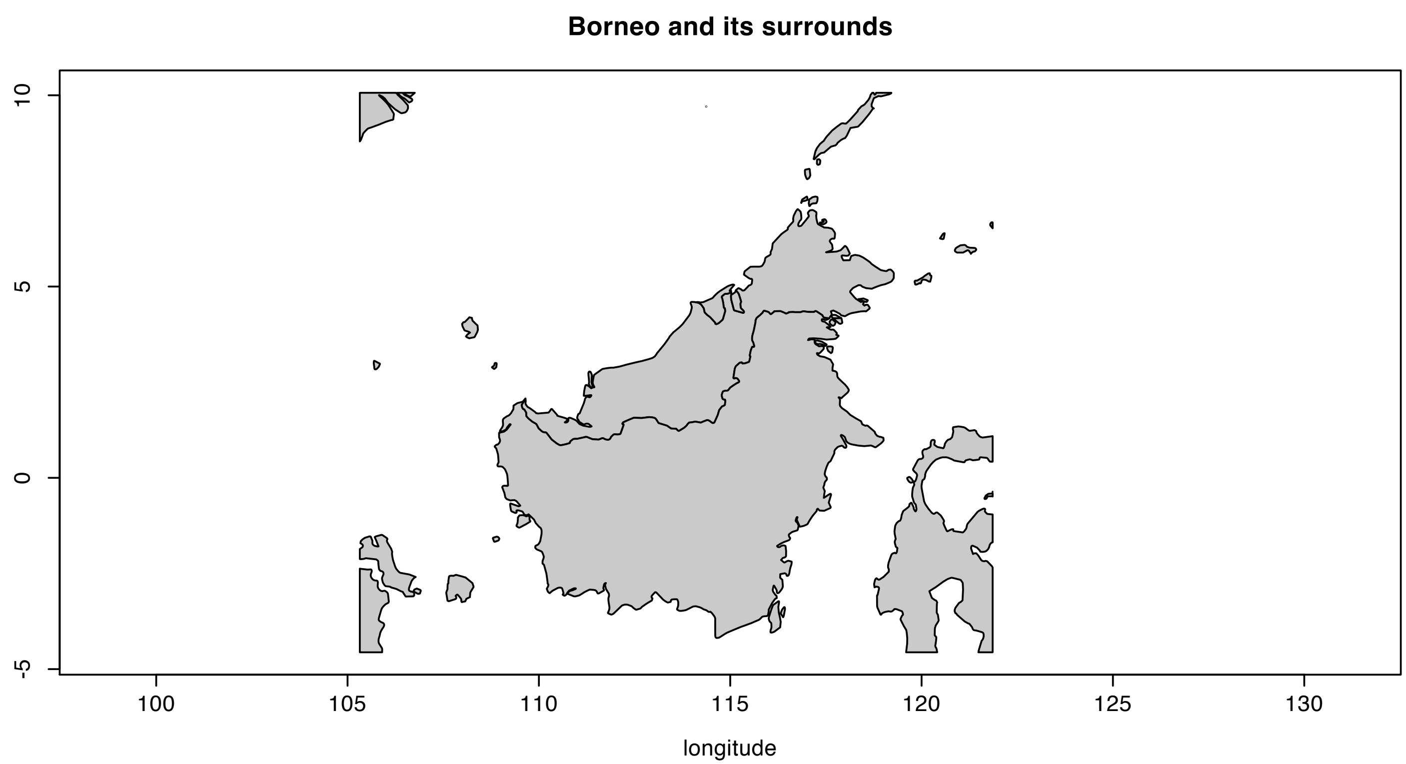
Counties of California

Combine income data from 2000 census with zipcode map of San Diego County

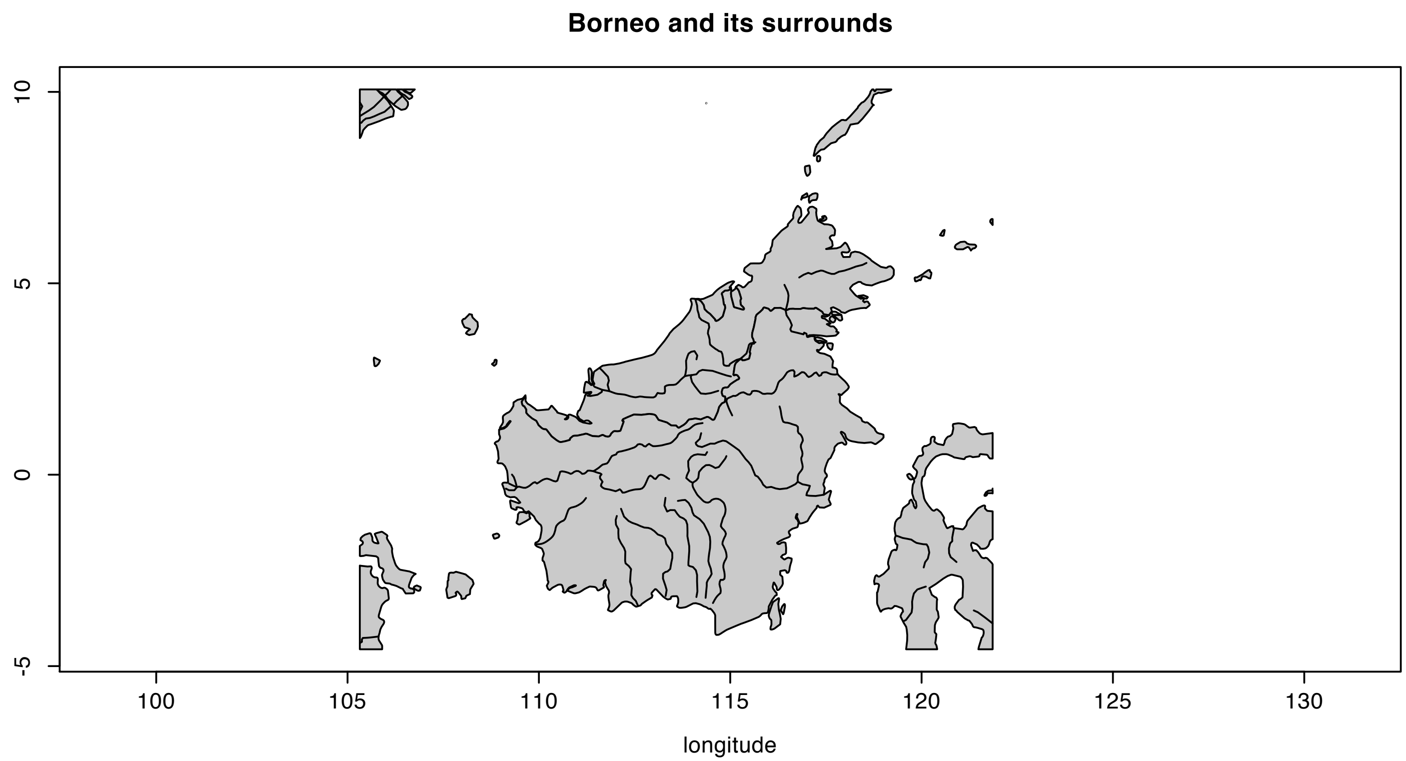


2000 median household income for zip codes

GIS files of Borneo can show country boundaries (e.g., Malaysia, Brunei, Indonesia)



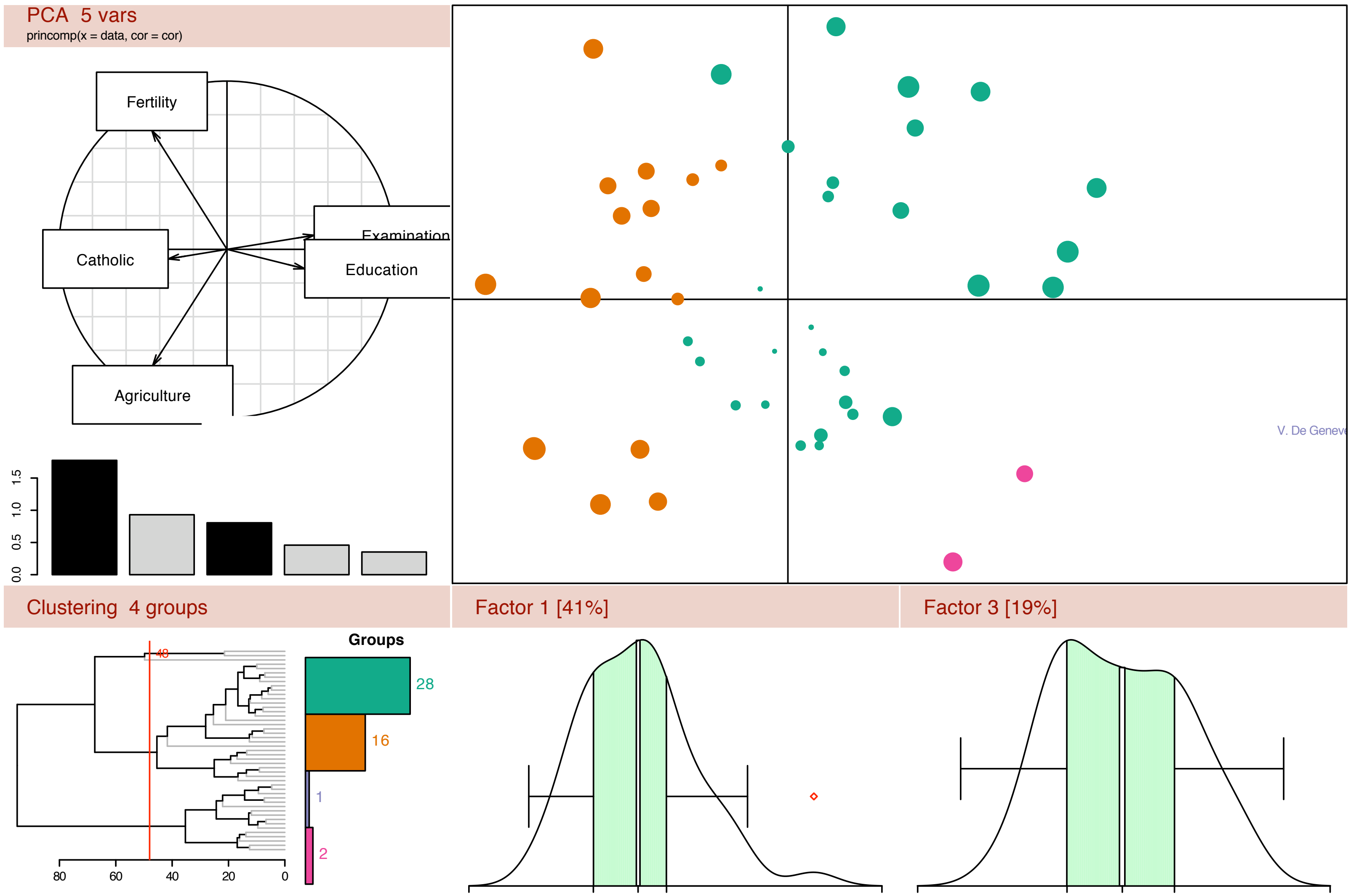
Public access GIS files include rivers and roads



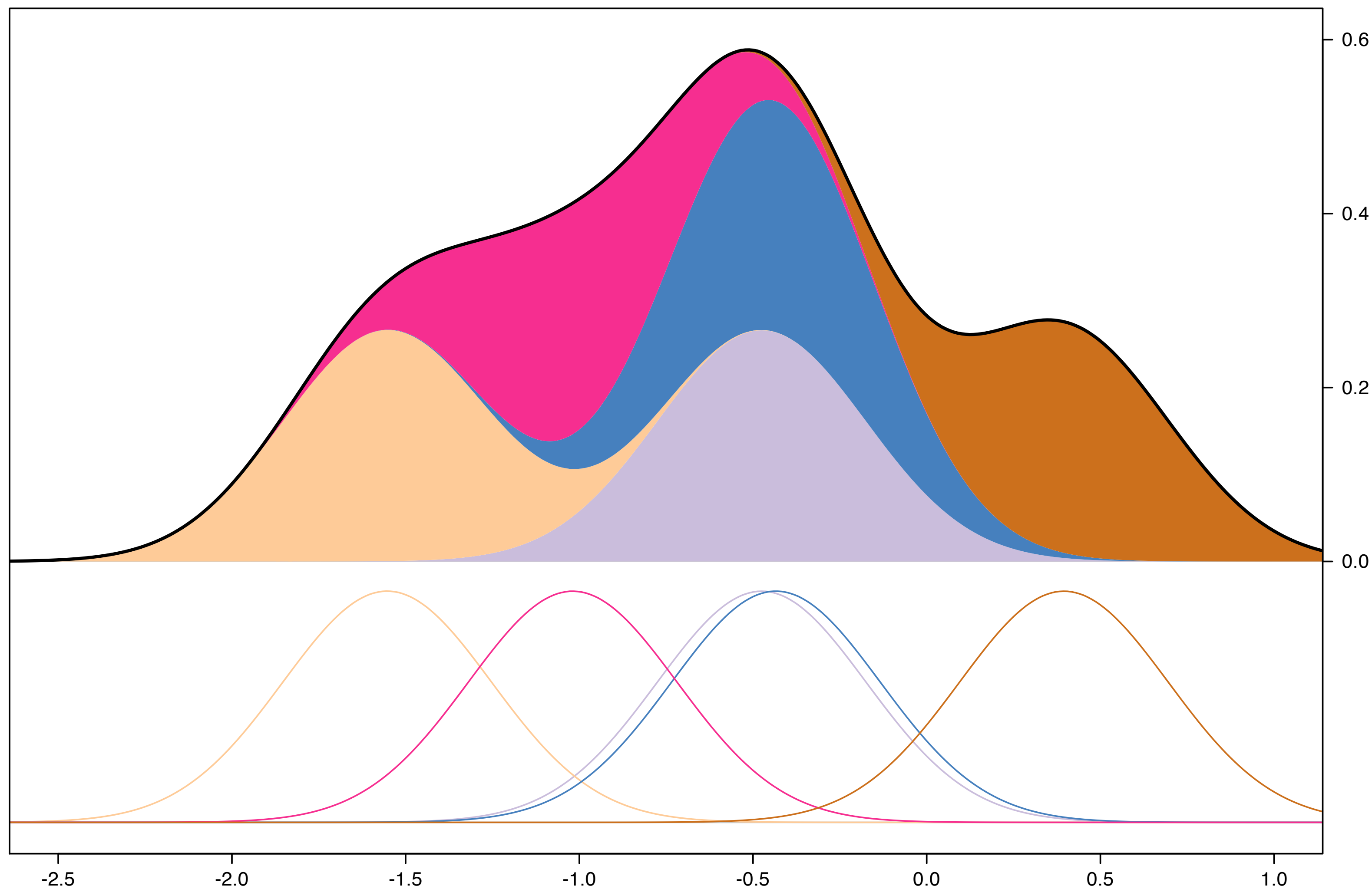
Even more graphics

- Taken from a collection of R demonstrations and graphics
  - http://addictedtor.free.fr/graphiques/

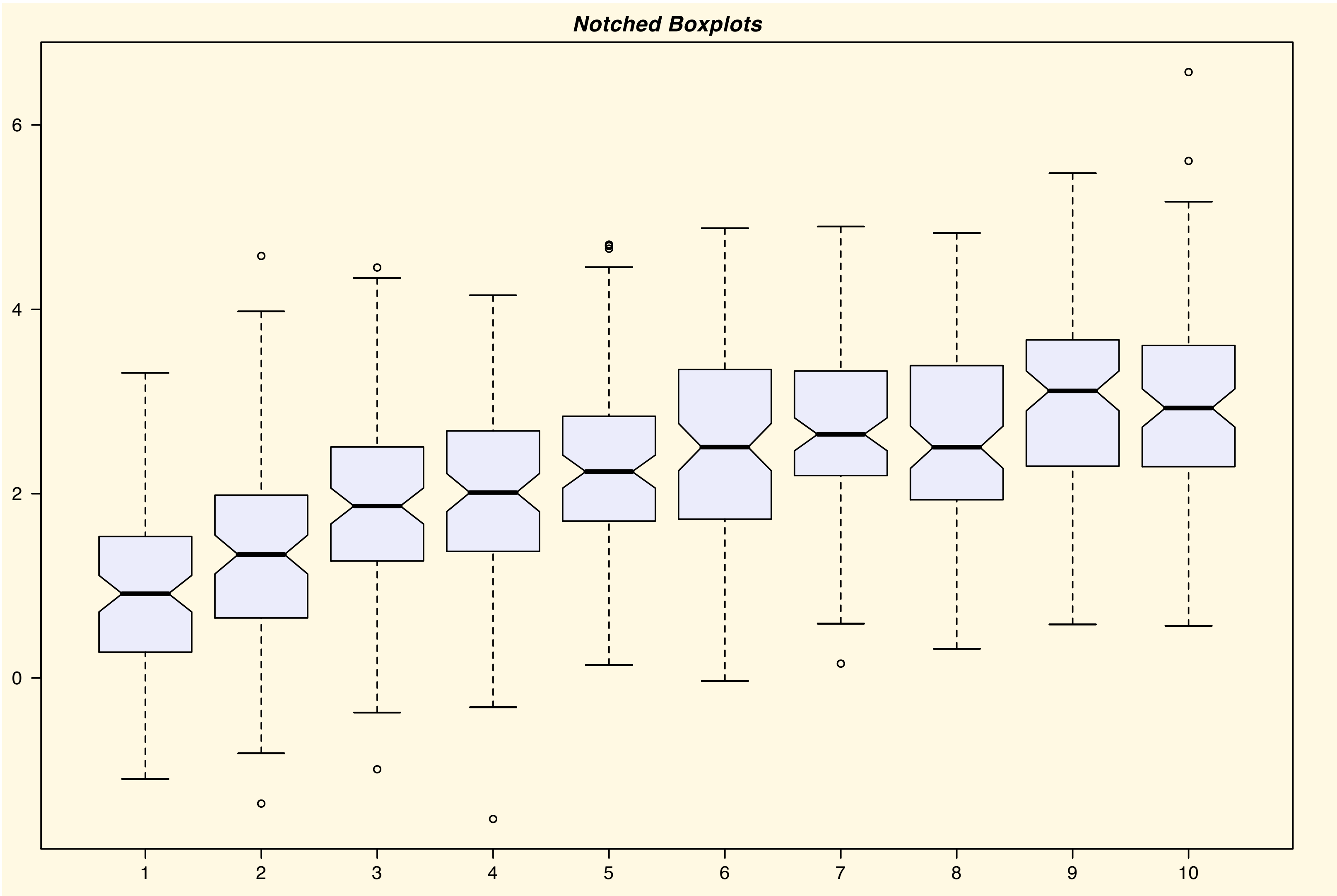
Principal components and clustering of sources of variance in USA arrest data



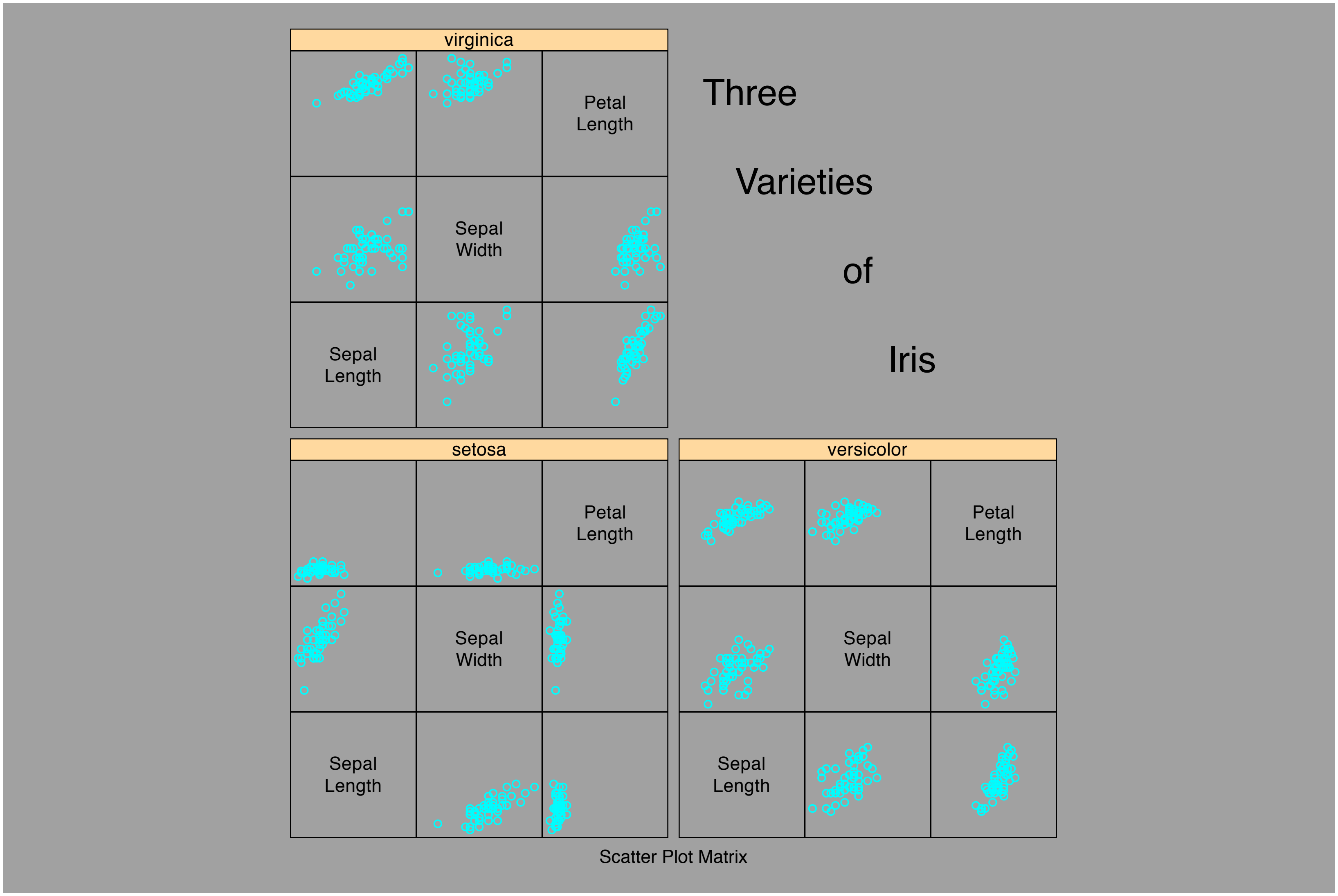
Mixture models



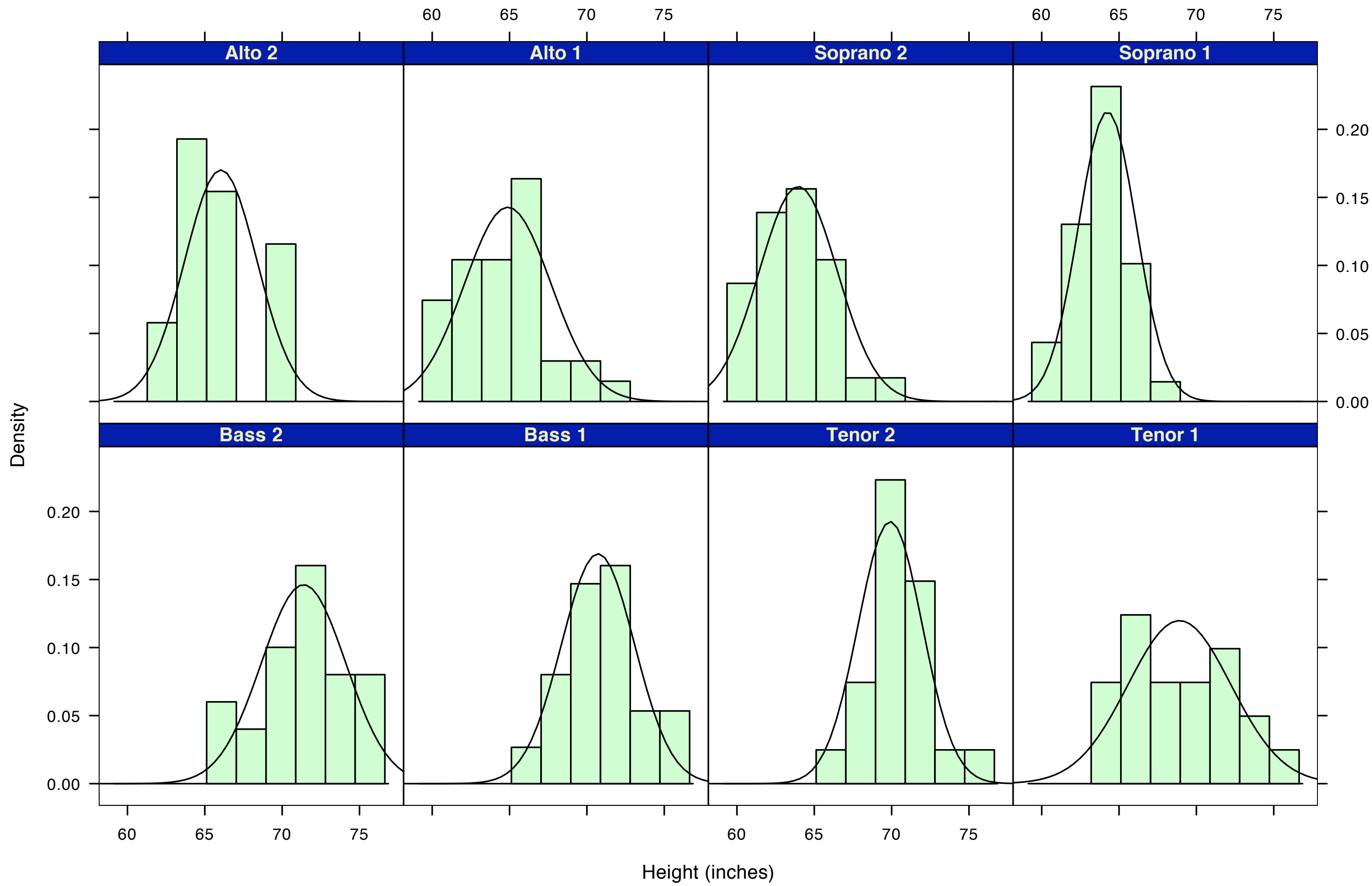
Notched Boxplots show confidence regions



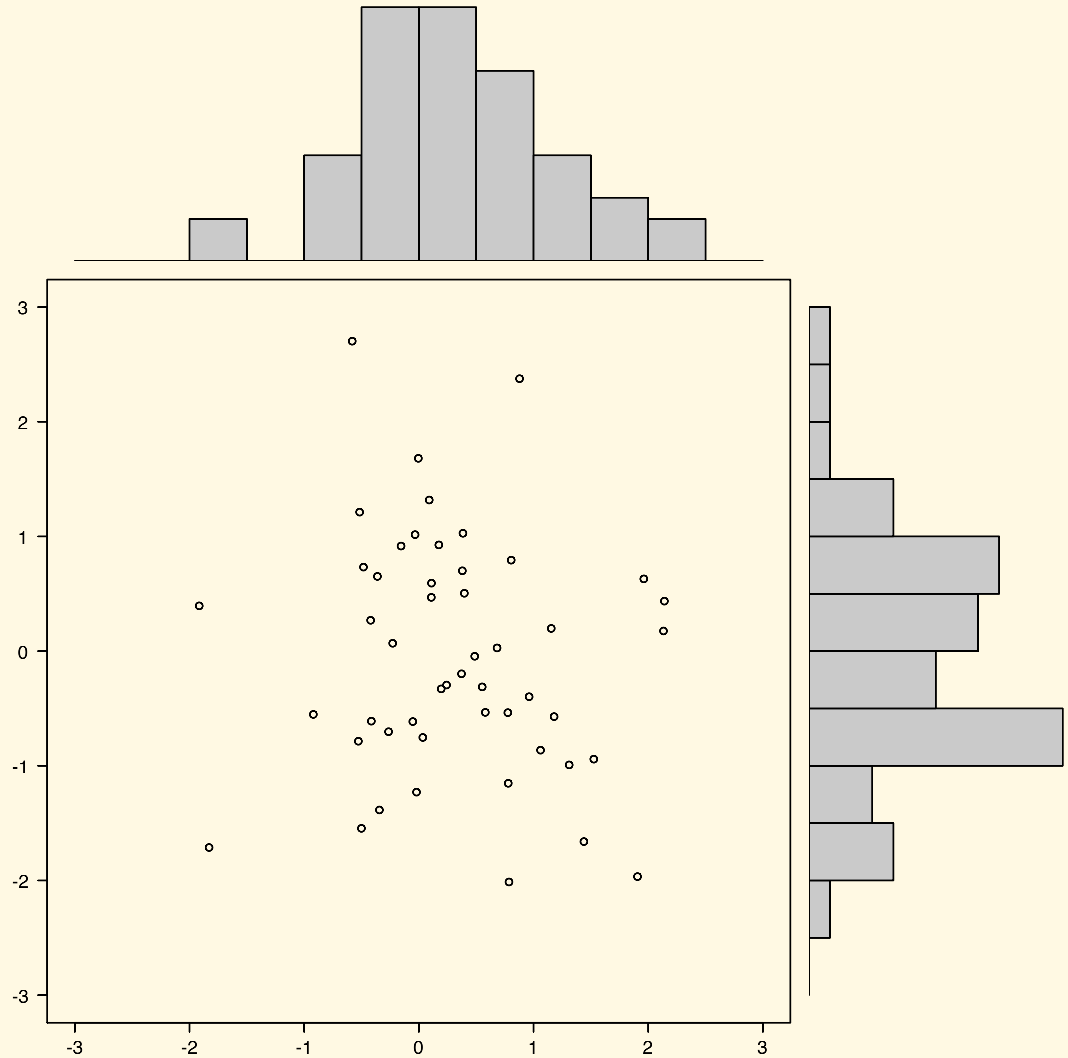
Multipanel graphs



Histograms and fitted distributions



Combine scatter plot with histograms



Why R?

I. Data manipulation including statistics as data

   A. the output of any function may be input for any other

II. Graphics for data exploration and interpretation

III. Statistical analysis

   A. Standard univariate and multivariate generalizations of the linear model

   B. Multivariate-structural extensions

Data Manipulation: Data Structures

I. Data types: integer, real, logical, character, string

II. Vectors of any data type

III. Matrices of any data type

IV. Data Frames (similar to matrix of mixed type)

V. Lists of any mixture of types

VI. All operations are functions and the returned values may be used in any data structure (e.g., as an element of a data frame or of a list)

Data structures

I. Elements (integers, reals, logicals, strings)

II. Vectors (ordered sets of similar elements)

III. Matrices (ordered sets of vectors of the same length)

IV. Data Frames (ordered sets of vectors where the vectors can be different, but all the same length)

V. Lists (ordered sets of anything, can be different lengths)

Structure examples

```
> x <- c(1,2,4)
> y <- c(letters[1:6],LETTERS[1:4])
> z <- seq(10,28,2)
> X <- matrix(1:20,ncol=4)
> Y <- matrix(c(11,22,44,4,15,42),ncol=3,byrow=TRUE)
> yz.df <-data.frame(A = y,b=z)
> L <- list(a=x,b=y,c=z,d=X,e=Y,f =yz.df)
> x
[1] 1 2 4
> y
 [1] "a" "b" "c" "d" "e" "f" "A" "B" "C" "D"
> z
 [1] 10 12 14 16 18 20 22 24 26 28
> X
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

```
> Y
     [,1] [,2] [,3]
[1,]   11   22   44
[2,]    4   15   42
> yz.df
   A  b
1  a 10
2  b 12
3  c 14
4  d 16
5  e 18
6  f 20
7  A 22
8  B 24
9  C 26
10 D 28
```

Structure list

```
> L
$a
[1] 1 2 4

$b
 [1] "a" "b" "c" "d" "e" "f" "A" "B" "C" "D"

$c
 [1] 10 12 14 16 18 20 22 24 26 28

$d
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

```
$e
     [,1] [,2] [,3]
[1,]   11   22   44
[2,]    4   15   42
$f
   A  b
1  a 10
2  b 12
3  c 14
4  d 16
5  e 18
6  f 20
7  A 22
8  B 24
9  C 26
10 D 28
```

Structure of lists

```
> str(L)
List of 6
 $ a: num [1:3] 1 2 4
 $ b: chr [1:10] "a" "b" "c" "d" ...
 $ c: num [1:10] 10 12 14 16 18 20 22 24 26 28
 $ d: int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
 $ e: num [1:2, 1:3] 11 4 22 15 44 42
 $ f:'data.frame': 10 obs. of  2 variables:
  ..$ A: Factor w/ 10 levels "a","A","b","B",..: 1 3 5 7 9 10 2 4 
6 8
  ..$ b: num [1:10] 10 12 14 16 18 20 22 24 26 28
```

Accessing elements

```
> L$d
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> L[[4]]
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

Manipulating data

I. Consider scoring a multiple choice test
   A. 10 items, 100 subjects
   B. create the scoring key
   C. score it

Scoring a multiple score test

```
> set.seed(42)
> my.items <- matrix(sample(5,500,replace=TRUE),ncol=5)
> my.key <- c(1,2,3,2,4)
> my.scores <- t(t(my.items)==my.key[])  #these are TRUE or FALSE
> my.scores <- t(t(my.items)==my.key[]) +0 #these are 1s or 0s
> my.total <- rowSums(my.scores)           #total score
> describe(my.total)
  var   n mean   sd median trimmed  mad min max range skew kurtosis   se
1   1 100 0.92 0.82      1    0.84 1.48   0   3     3 0.68    -0.03 0.08
> describe(my.scores)
  var   n mean   sd median trimmed mad min max range skew kurtosis   se
1   1 100 0.19 0.39      0    0.11   0   0   1     1 1.56     0.43 0.04
2   2 100 0.14 0.35      0    0.05   0   0   1     1 2.04     2.20 0.03
3   3 100 0.25 0.44      0    0.19   0   0   1     1 1.14    -0.71 0.04
4   4 100 0.22 0.42      0    0.15   0   0   1     1 1.33    -0.23 0.04
5   5 100 0.12 0.33      0    0.02   0   0   1     1 2.30     3.34 0.03
```

How does that work

```
> my.key
[1] 1 2 3 2 4
> dim(my.items)
[1] 100   5
> dim(t(my.items))
[1]   5 100
> t(my.items)[,1:10]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    5    5    2    5    4    3    4    1    4     4
[2,]    4    2    2    2    5    5    4    4    3     1
[3,]    5    3    5    3    1    3    5    3    2     2
[4,]    3    3    1    2    5    5    2    1    2     2
[5,]    1    3    4    3    5    1    5    2    1     2
> x <- t(my.items) == my.key
> x[,1:10]
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
[1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[2,] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[3,] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
[4,] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE
[5,] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

> x0 <- x+0
> x0[,1:10]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    0    1    0     0
[2,]    0    1    1    1    0    0    0    0    0     0
[3,]    0    1    0    1    0    1    0    1    0     0
[4,]    0    0    0    1    0    0    1    0    1     1
[5,]    0    0    1    0    0    0    0    0    0     0
```

More on multiple choice

```
> my.scores <- t(t(my.items)==my.key[])  #these are TRUE or FALSE
> head(my.scores)
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] FALSE FALSE FALSE FALSE FALSE
[2,] FALSE  TRUE  TRUE FALSE FALSE
[3,] FALSE  TRUE FALSE FALSE  TRUE
[4,] FALSE  TRUE  TRUE  TRUE FALSE
[5,] FALSE FALSE FALSE FALSE FALSE
[6,] FALSE FALSE  TRUE FALSE FALSE
> my.scores <- t(t(my.items)==my.key[]) +0  #these are 1s or 0s
> my.scores
       [,1] [,2] [,3] [,4] [,5]
  [1,]    0    0    0    0    0
  [2,]    0    1    1    0    0
  [3,]    0    1    0    0    1
  [4,]    0    1    1    1    0
  [5,]    0    0    0    0    0
  [6,]    0    0    1    0    0
...
> my.total <- rowSums(my.scores)
> head(my.total)
[1] 0 2 2 3 0 1
```

Or, just use a function

```
> data(iqitems)
> iq.keys <- c(4,4,3,1,4,3,2,3,1,4,1,3,4,3)
> score.multiple.choice(iq.keys,iqitems)
$item.stats
     key    0    1    2    3    4    5    6    r    n mean   sd  skew kurtosis   se
iq1    4 0.04 0.01 0.03 0.09 0.80 0.02 0.01 0.59 1000 0.80 0.40 -1.51     0.27 0.01
iq8    4 0.03 0.10 0.01 0.02 0.80 0.01 0.04 0.39 1000 0.80 0.40 -1.49     0.22 0.01
iq10   3 0.10 0.22 0.09 0.37 0.04 0.13 0.04 0.35 1000 0.37 0.48  0.53    -1.72 0.02
iq15   1 0.03 0.65 0.16 0.15 0.00 0.00 0.00 0.35 1000 0.65 0.48 -0.63    -1.60 0.02
iq20   4 0.03 0.02 0.03 0.03 0.85 0.02 0.01 0.42 1000 0.85 0.35 -2.00     2.01 0.01
iq44   3 0.03 0.10 0.06 0.64 0.02 0.14 0.01 0.42 1000 0.64 0.48 -0.61    -1.64 0.02
iq47   2 0.04 0.08 0.59 0.06 0.11 0.07 0.05 0.51 1000 0.59 0.49 -0.35    -1.88 0.02
iq2    3 0.07 0.08 0.31 0.32 0.15 0.05 0.02 0.26 1000 0.32 0.46  0.80    -1.37 0.01
iq11   1 0.04 0.87 0.03 0.01 0.01 0.01 0.04 0.54 1000 0.87 0.34 -2.15     2.61 0.01
iq16   4 0.05 0.05 0.08 0.07 0.74 0.01 0.00 0.56 1000 0.74 0.44 -1.11    -0.77 0.01
iq32   1 0.04 0.54 0.02 0.14 0.10 0.04 0.12 0.50 1000 0.54 0.50 -0.17    -1.97 0.02
iq37   3 0.07 0.10 0.09 0.26 0.13 0.02 0.34 0.23 1000 0.26 0.44  1.12    -0.74 0.01
iq43   4 0.04 0.07 0.04 0.02 0.78 0.03 0.00 0.50 1000 0.78 0.41 -1.35    -0.18 0.01
iq49   3 0.06 0.27 0.09 0.32 0.14 0.08 0.05 0.28 1000 0.32 0.47  0.79    -1.38 0.01

$alpha
         Averages
Averages     0.63

$av.r
         Averages
Averages     0.11
```

Examine the structure

```
> iq.scores <- score.multiple.choice(iq.keys,iqitems,short=FALSE)
> str(iq.scores)
List of 4
 $ scores    : num [1:1000, 1] 0.429 0.357 0.571 0.571 0.571 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "72" "95" "100" "136" ...
  .. ..$ : chr "Averages"
 $ item.stats:'data.frame':   14 obs. of  15 variables:
  ..$ key     : num [1:14] 4 4 3 1 4 3 2 3 1 4 ...
  ..$ 0       : num [1:14] 0.035 0.028 0.1 0.032 0.028 0.029 0.045 0.071 0.036 0.047 ...
  ..$ 1       : num [1:14] 0.01 0.104 0.223 0.651 0.024 0.097 0.08 0.078 0.866 0.054 ...
  ..$ 2       : num [1:14] 0.034 0.006 0.088 0.163 0.027 0.055 0.586 0.308 0.027 0.079 ...
  ..$ 3       : num [1:14] 0.088 0.016 0.371 0.153 0.034 0.645 0.063 0.315 0.011 0.07 ...
  ..$ 4       : num [1:14] 0.801 0.799 0.044 0.001 0.854 0.019 0.106 0.154 0.009 0.743 ...
  ..$ 5       : num [1:14] 0.024 0.009 0.133 0 0.019 0.145 0.067 0.053 0.008 0.007 ...
  ..$ 6       : num [1:14] 0.008 0.038 0.041 0 0.014 0.01 0.053 0.021 0.043 0 ...
  ..$ r       : num [1:14] 0.591 0.395 0.346 0.35 0.418 ...
  ..$ n       : num [1:14] 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...
  ..$ mean    : num [1:14] 0.801 0.799 0.371 0.651 0.854 0.645 0.586 0.315 0.866 0.743 ...
  ..$ sd      : num [1:14] 0.399 0.401 0.483 0.477 0.353 ...
  ..$ skew    : num [1:14] -1.506 -1.49 0.533 -0.633 -2.002 ...
  ..$ kurtosis: num [1:14] 0.267 0.22 -1.717 -1.601 2.01 ...
  ..$ se      : num [1:14] 0.0126 0.0127 0.0153 0.0151 0.0112 ...
 $ alpha     : num [1, 1] 0.63
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "Averages"
  .. ..$ : chr "Averages"
 $ av.r      : num [1, 1] 0.11
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "Averages"
  .. ..$ : chr "Averages"
```

Use the relevant part of a list

```
> dim(iq.scores)
NULL
> length(iq.scores)
[1] 4
> dim(iq.scores$scores)
[1] 1000    1
> describe(iq.scores$scores)
         var    n mean   sd median trimmed  mad min  max range  skew kurtosis   se
Averages   1 1000 0.61 0.18   0.64    0.63 0.11   0 0.93  0.93 -1.07     1.44 0.01
```

Data Manipulation

I. standard arithmetic and logical operations

II. matrix operations including transpose, inner product, outer product, diagonal, trace, invert

III. searching, sorting, merging

IV. data cleaning by logical commands

Basic data description

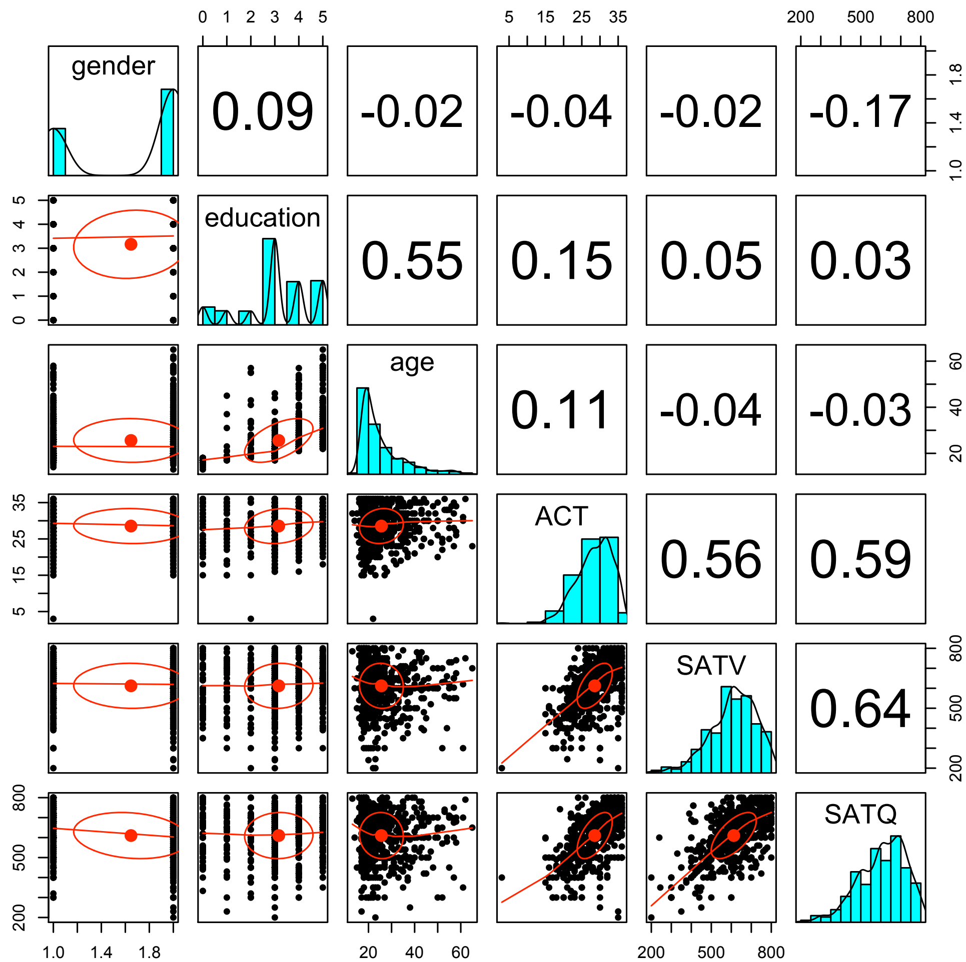
```
summary(My.data)
      epiE            epiS            epiImp           epilie          epiNeur          bfagree           bfcon      
 Min.   : 1.00   Min.   : 0.000   Min.   : 0.000   Min.   :0.000   Min.   : 0.00   Min.   : 74.0   Min.   : 53.0  
 1st Qu.:11.00   1st Qu.: 6.000   1st Qu.: 3.000   1st Qu.:1.000   1st Qu.: 7.00   1st Qu.:112.0   1st Qu.: 99.0  
 Median :14.00   Median : 8.000   Median : 4.000   Median :2.000   Median :10.00   Median :126.0   Median :114.0  
 Mean   :13.68   Mean   : 7.978   Mean   : 4.784   Mean   :2.377   Mean   :10.41   Mean   :125.0   Mean   :113.3  
 3rd Qu.:16.00   3rd Qu.:10.000   3rd Qu.: 6.000   3rd Qu.:3.000   3rd Qu.:14.00   3rd Qu.:136.5   3rd Qu.:128.5  
 Max.   :99.00   Max.   :99.000   Max.   :99.000   Max.   :7.000   Max.   :23.00   Max.   :167.0   Max.   :178.0  
         bfext            bfneur           bfopen            bdi            traitanx        stateanx    
         Min.   :  8.0   Min.   : 34.00   Min.   : 73.0   Min.   : 0.000   Min.   :22.00   Min.   :21.00  
         1st Qu.: 87.5   1st Qu.: 70.00   1st Qu.:110.0   1st Qu.: 3.000   1st Qu.:32.00   1st Qu.:32.00  
         Median :104.0   Median : 90.00   Median :125.0   Median : 6.000   Median :38.00   Median :38.00  
         Mean   :102.2   Mean   : 87.97   Mean   :123.4   Mean   : 6.779   Mean   :39.01   Mean   :39.85  
         3rd Qu.:118.0   3rd Qu.:104.00   3rd Qu.:136.5   3rd Qu.: 9.000   3rd Qu.:44.00   3rd Qu.:46.50  
         Max.   :168.0   Max.   :152.00   Max.   :173.0   Max.   :27.000   Max.   :71.00   Max.   :79.00  

> describe(My.data)
         var   n   mean    sd median trimmed   mad min max range  skew kurtosis   se
epiE       1 231  13.33  4.14     14   13.49  4.45   1  22    21 -0.33    -0.06 0.27
epiS       2 231   7.58  2.69      8    7.77  2.97   0  13    13 -0.57    -0.02 0.18
epiImp     3 231   4.37  1.88      4    4.36  1.48   0   9     9  0.06    -0.62 0.12
epilie     4 231   2.38  1.50      2    2.27  1.48   0   7     7  0.66     0.24 0.10
epiNeur    5 231  10.41  4.90     10   10.39  4.45   0  23    23  0.06    -0.50 0.32
bfagree    6 231 125.00 18.14    126  125.26 17.79  74 167    93 -0.21    -0.27 1.19
bfcon      7 231 113.25 21.88    114  113.42 22.24  53 178   125 -0.02     0.23 1.44
bfext      8 231 102.18 26.45    104  102.99 22.24   8 168   160 -0.41     0.51 1.74
bfneur     9 231  87.97 23.34     90   87.70 23.72  34 152   118  0.07    -0.55 1.54
bfopen    10 231 123.43 20.51    125  123.78 20.76  73 173   100 -0.16    -0.16 1.35
bdi       11 231   6.78  5.78      6    5.97  4.45   0  27    27  1.29     1.50 0.38
traitanx  12 231  39.01  9.52     38   38.36  8.90  22  71    49  0.67     0.47 0.63
stateanx  13 231  39.85 11.48     38   38.92 10.38  21  79    58  0.72    -0.01 0.76
```

R graphics (base)

I. Multiple graphics packages
   A. base graphics
   B. lattice
   C. ggobi
II. Following examples from base graphics

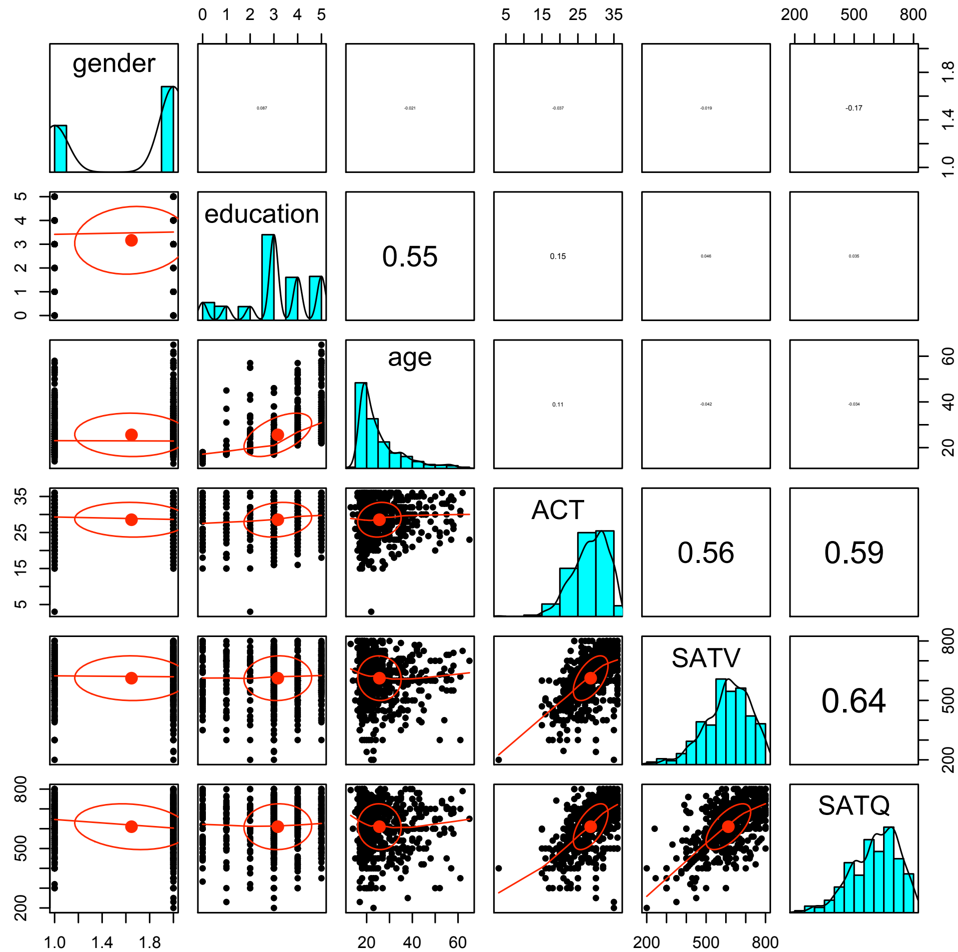
pairs.panels

```
> data(sat.act)
> pairs.panels(sat.act)
```

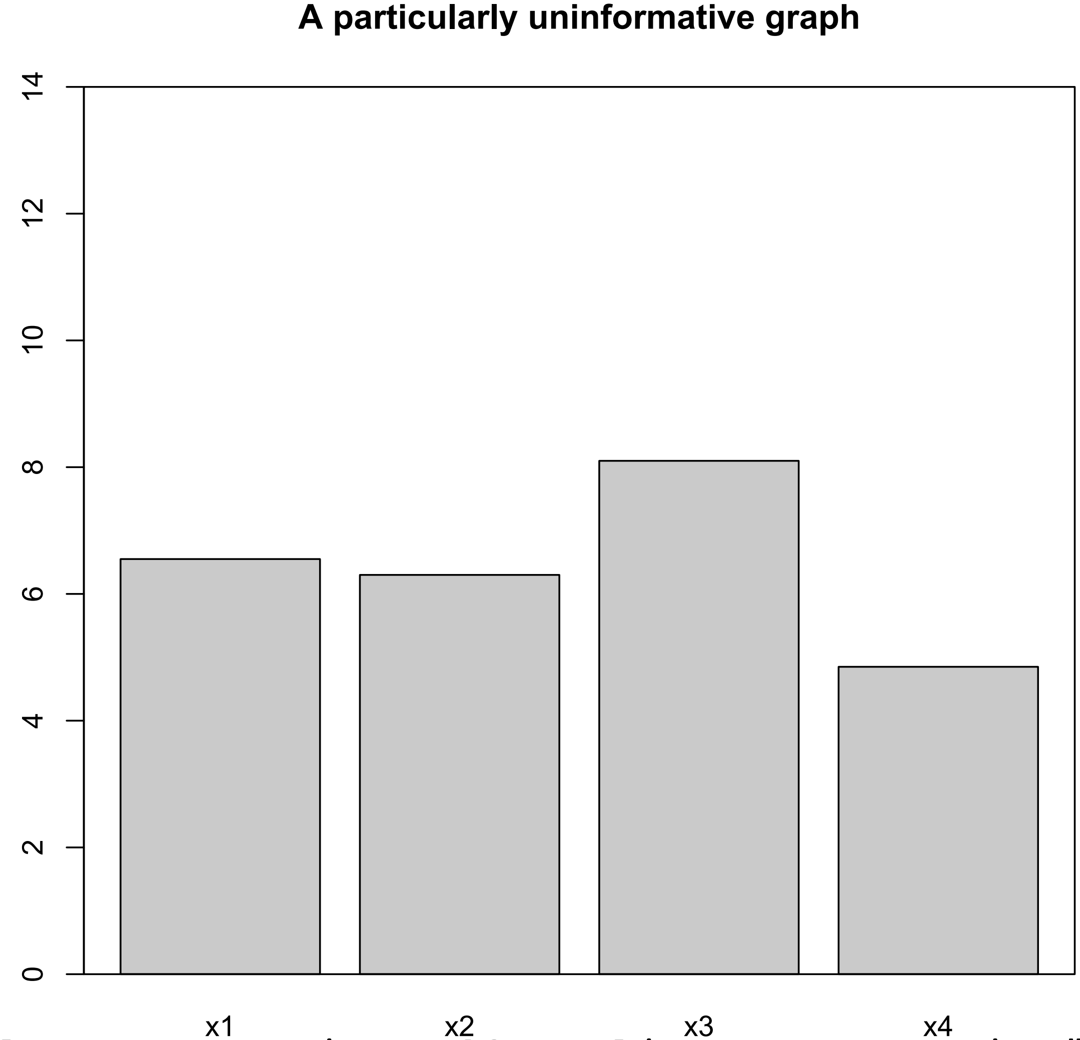


pairs.panels

```
pairs.panels(sat.act,scale=TRUE)
```

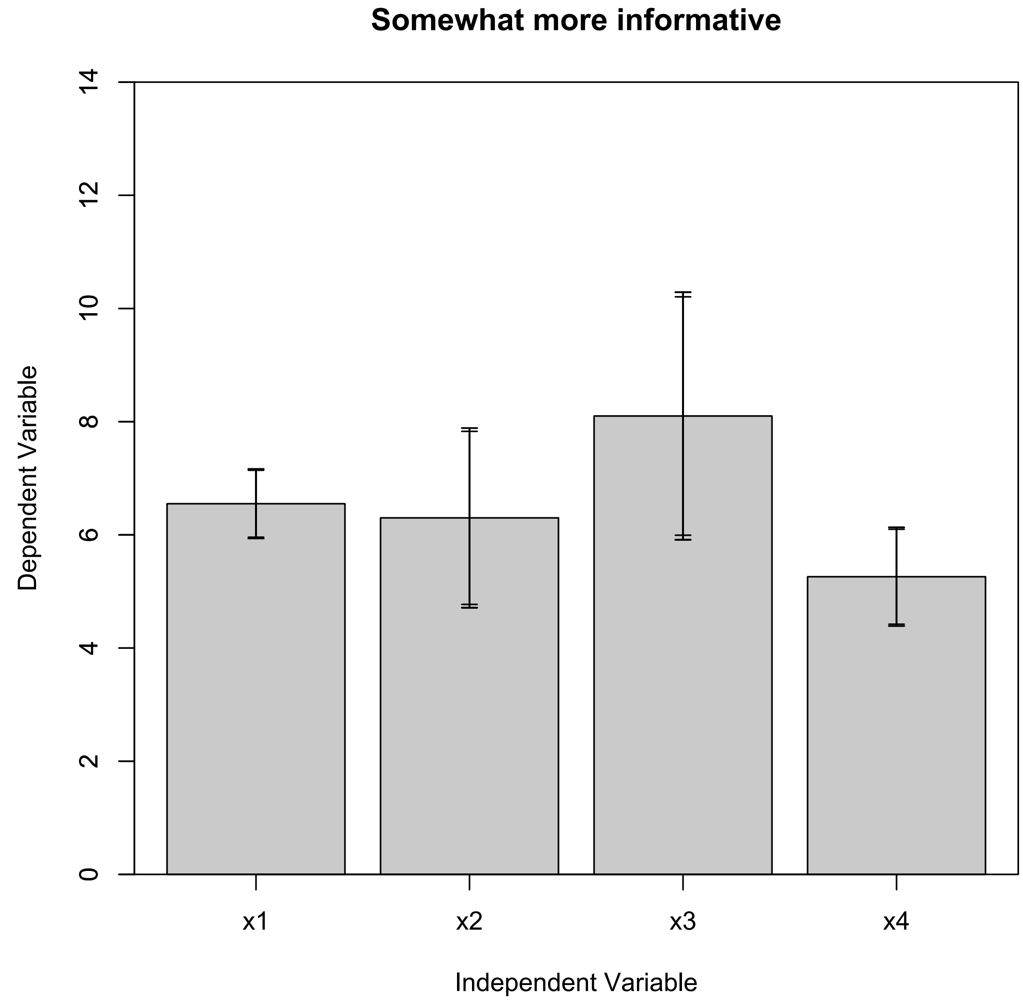


Do not draw bar graphs



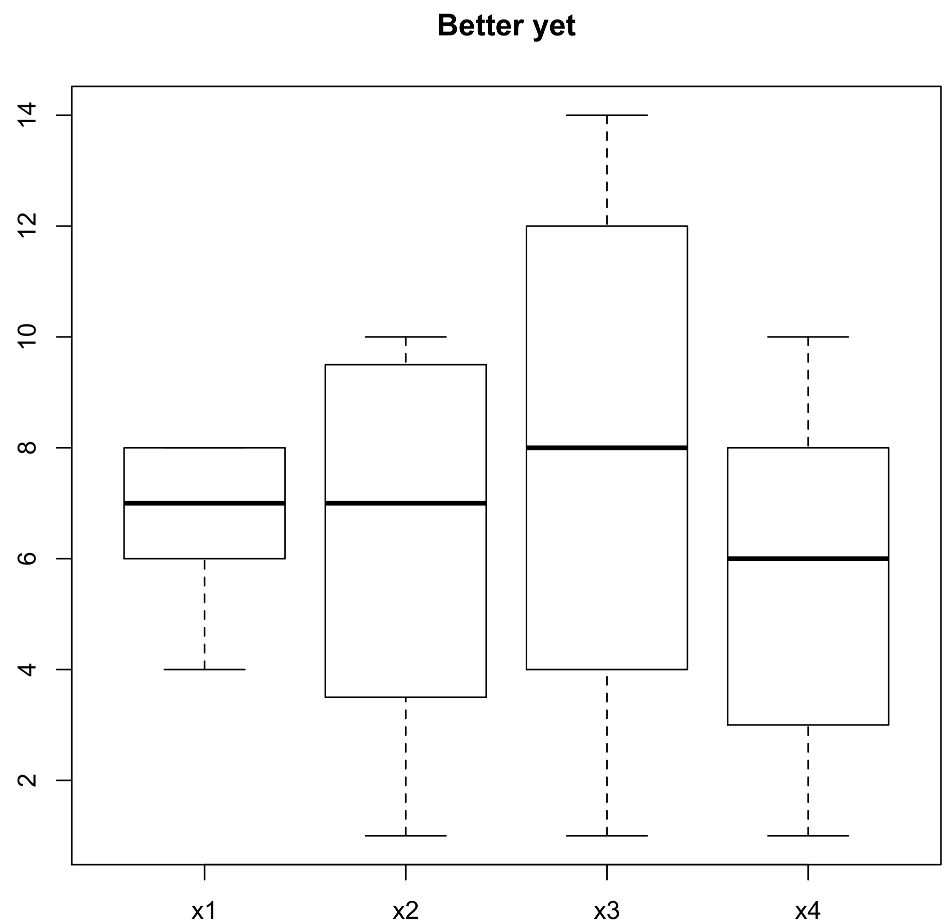

```
> barplot(colMeans(na.omit(X.df)),ylim=c(0,14),main="A particularly
uninformative graph")
> box()
```

Somewhat better



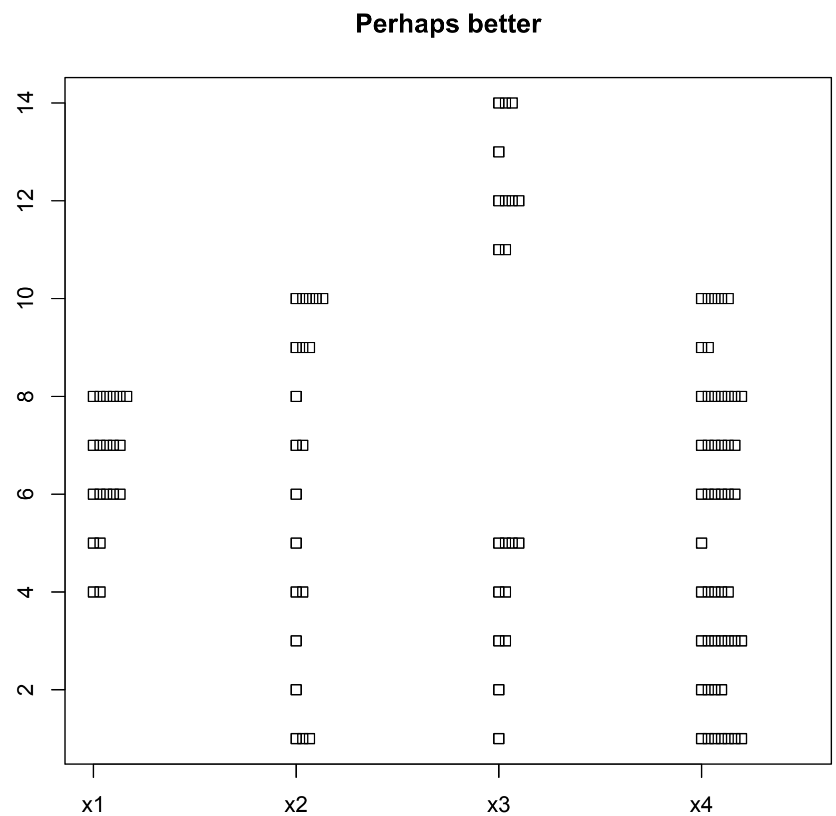

```
error.bars(X.df,bars=TRUE,ylim=c(0,14),
   main="Somewhat more informative")
```

box plot



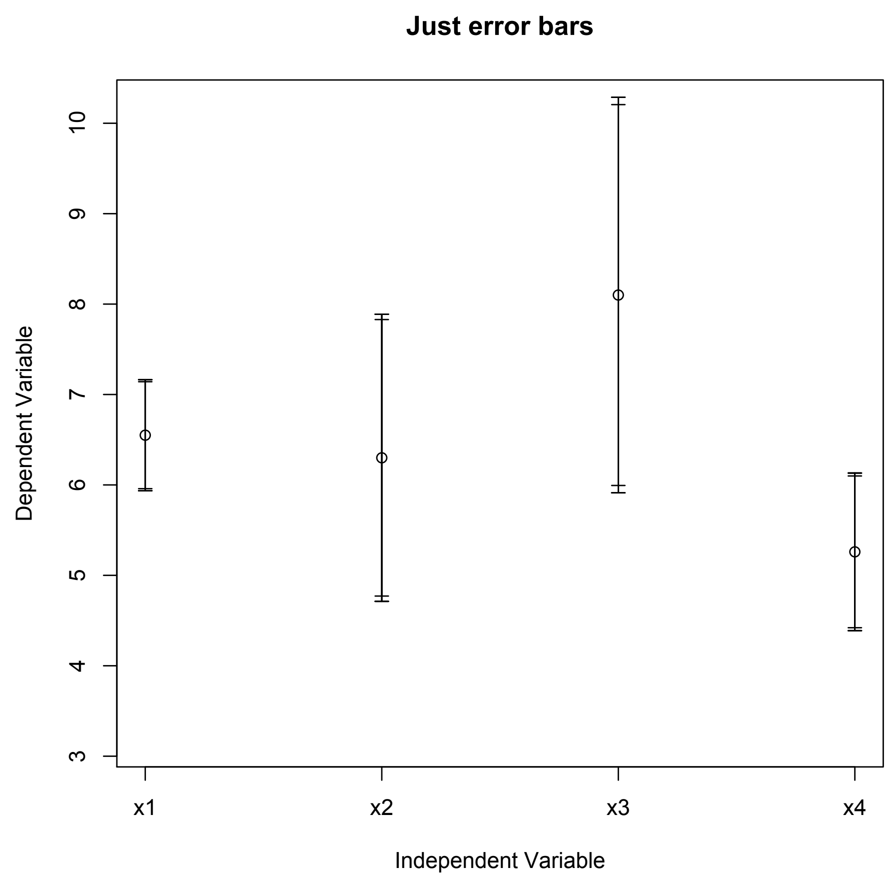

```
boxplot(X.df,main="Better yet")
```

stripchart



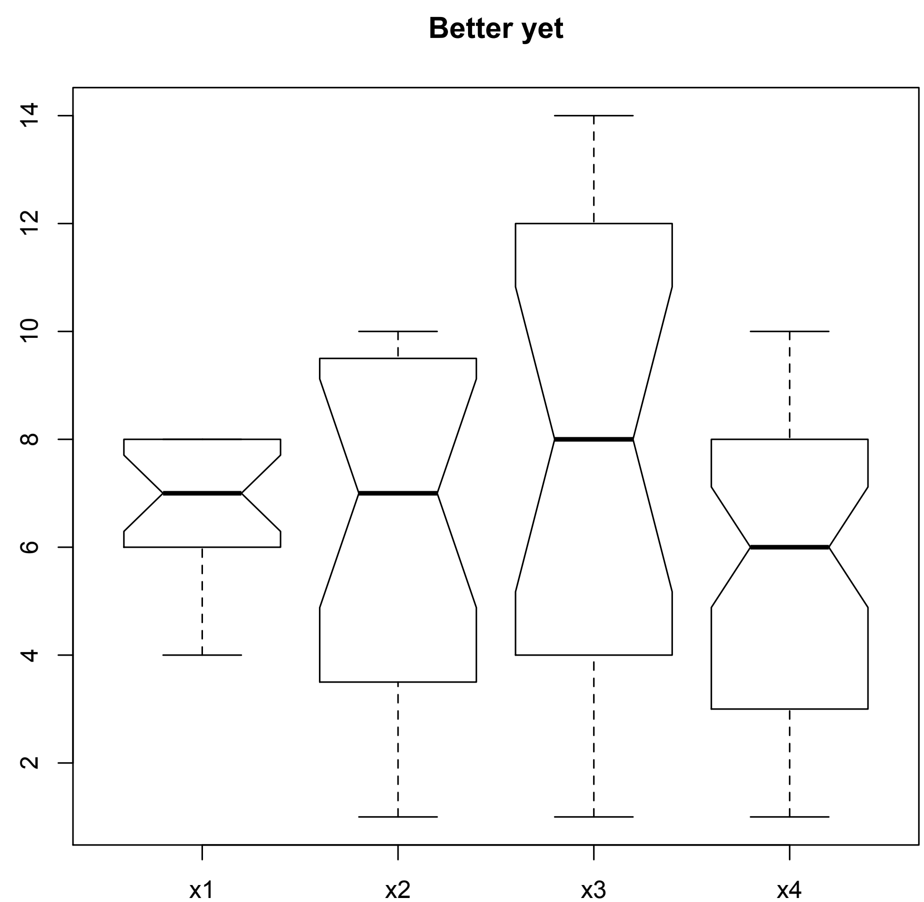

```
stripchart(X.df,method="stack",vertical=TRUE,main="Perhaps
better")
```

error.bars



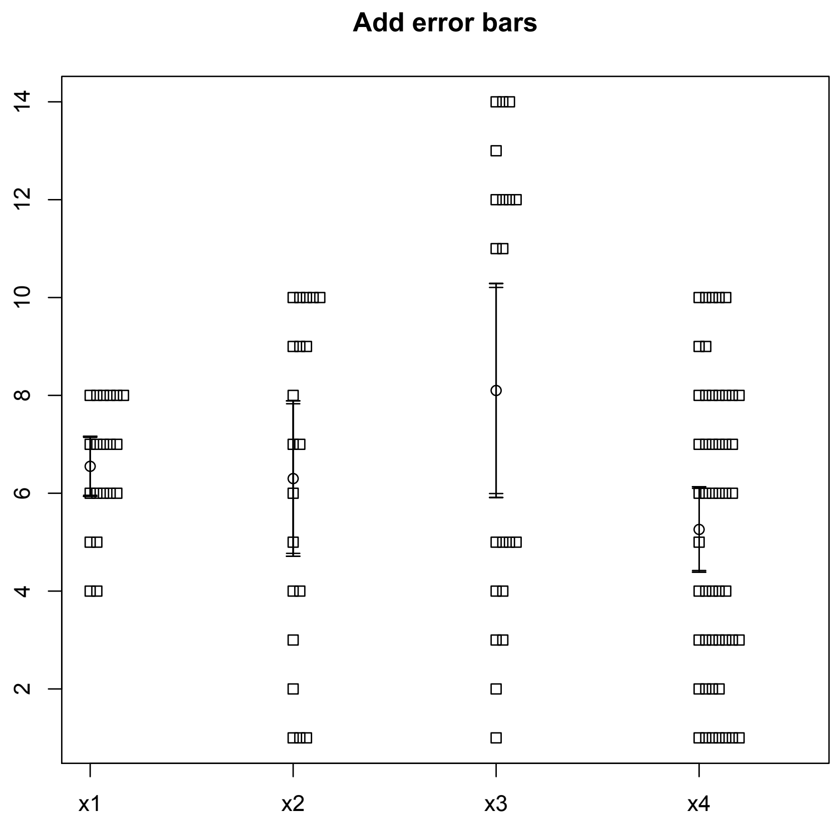

```
> error.bars(X.df,main="Just error bars")
```

notched boxplots



> boxplot(X.df,main="Better yet",notch=TRUE)

stripchart + error.bars

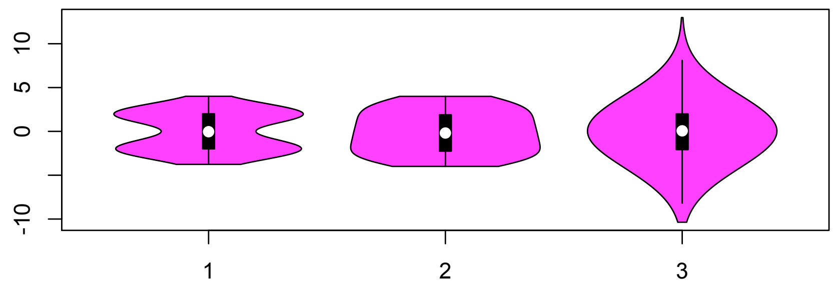


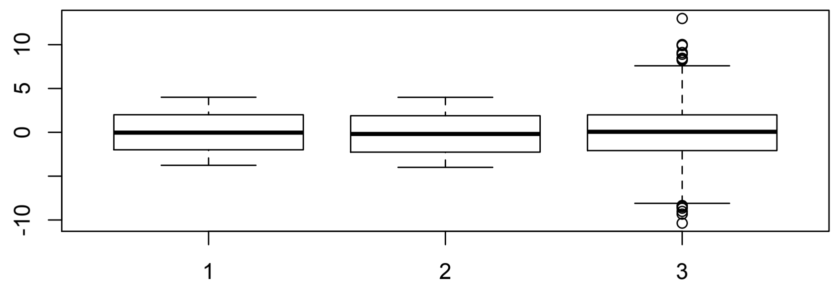

```
> stripchart(X.df,method="stack",vertical=TRUE,main="Add error bars")
> error.bars(X.df,add=TRUE)
```

Alternatives with larger data sets

I. The violin plot shows density distributions
II. Available in the vioplot package

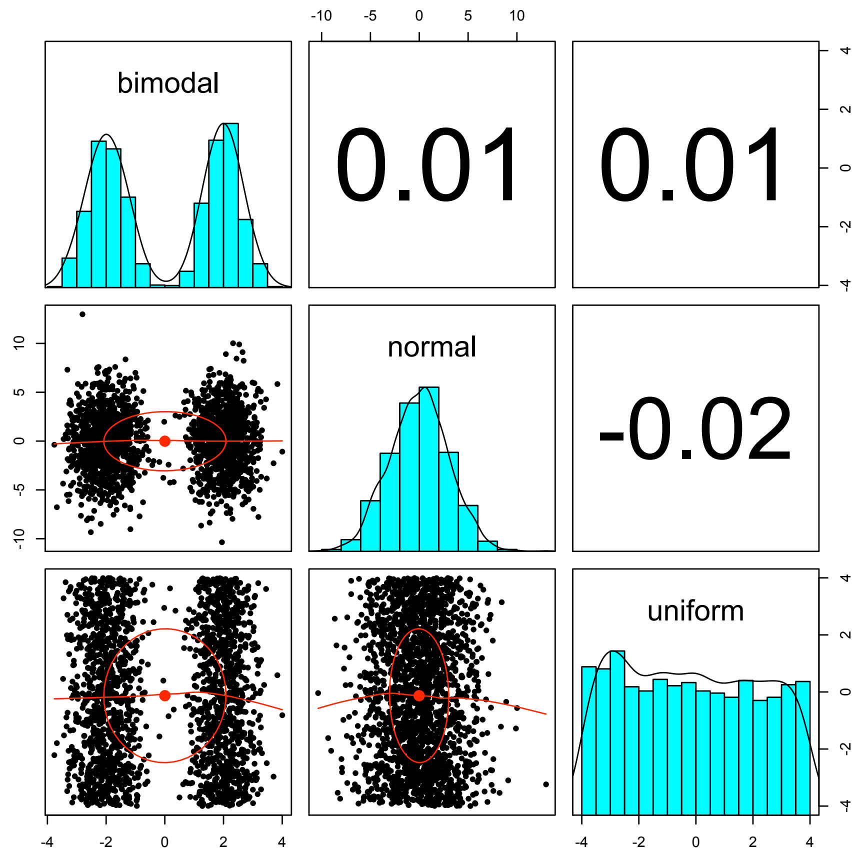
violin plots





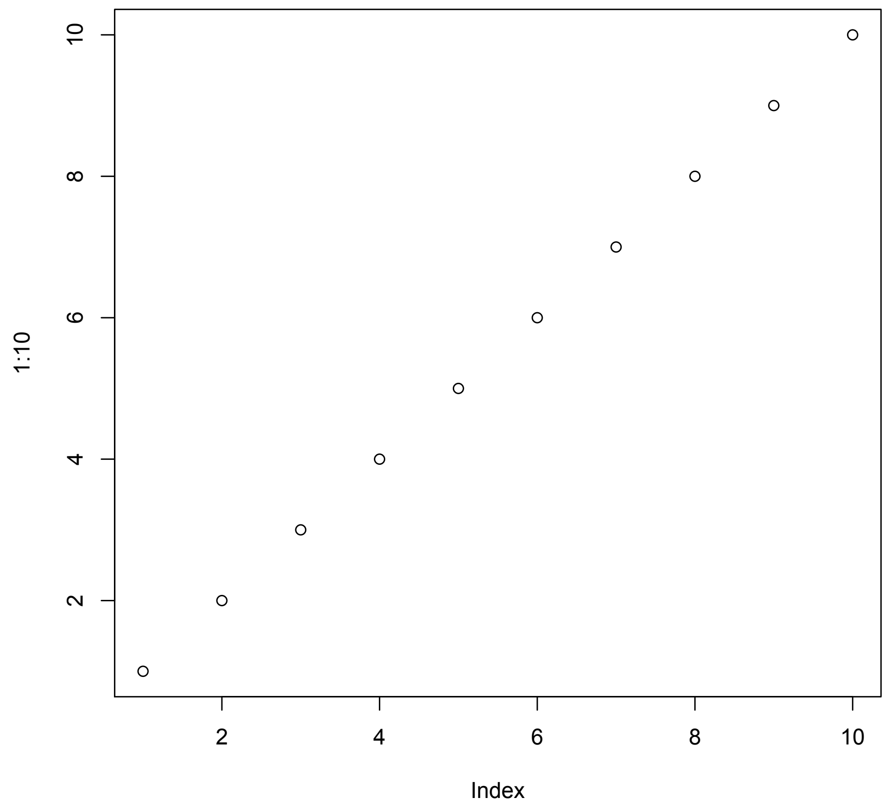

```
mu<-2
  si<-0.6
  bimodal<-c(rnorm(1000,-mu,si),rnorm(1000,mu,si))
  uniform<-runif(2000,-4,4)
  normal<-rnorm(2000,0,3)
  vioplot(bimodal,uniform,normal)
  boxplot(bimodal,uniform,normal)
```

An alternative



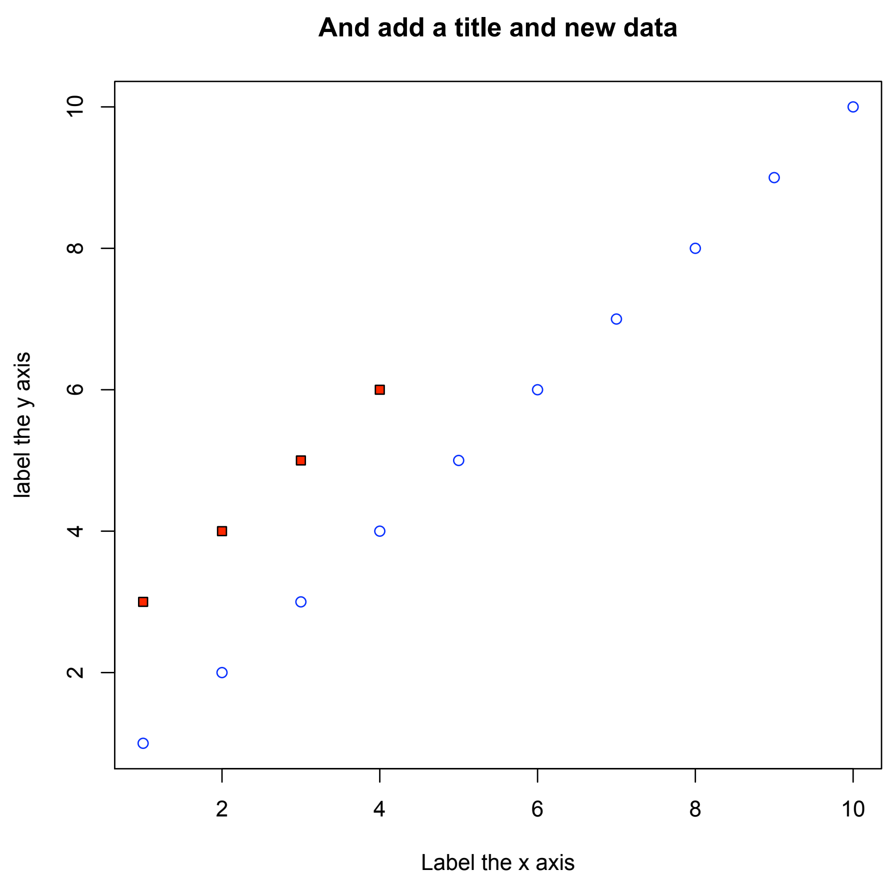

```
> bnu <-data.frame(bimodal,normal,uniform)
> pairs.panels(bnu)
```

Graphics 101



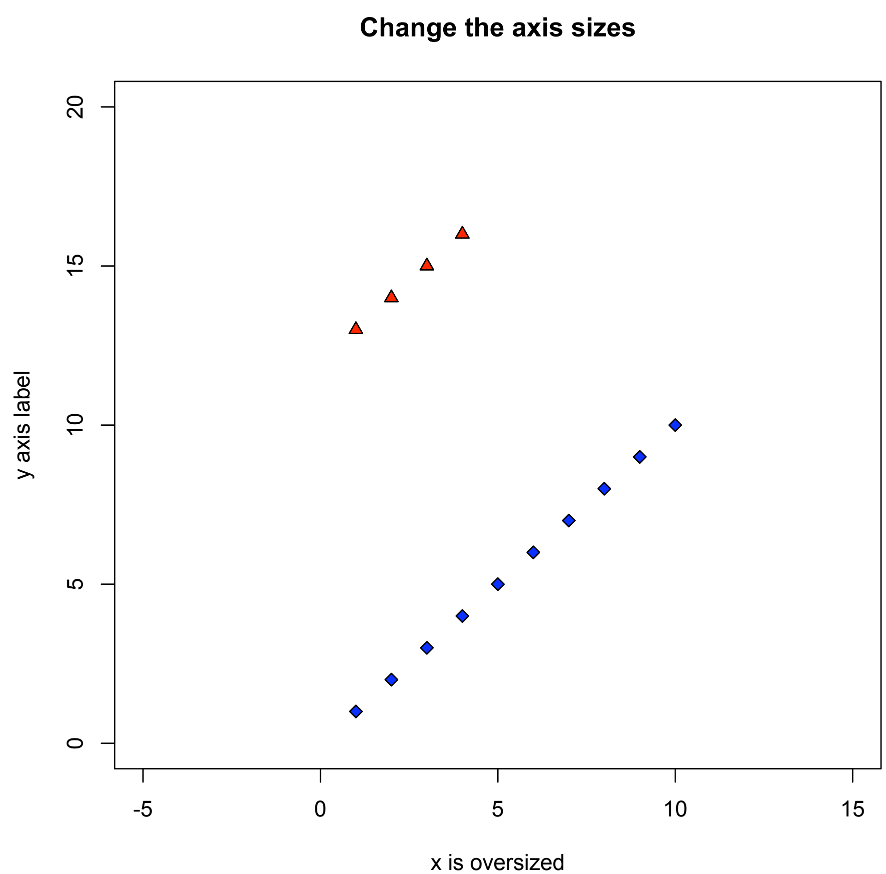

```
plot(1:10)
```

specify labels and title



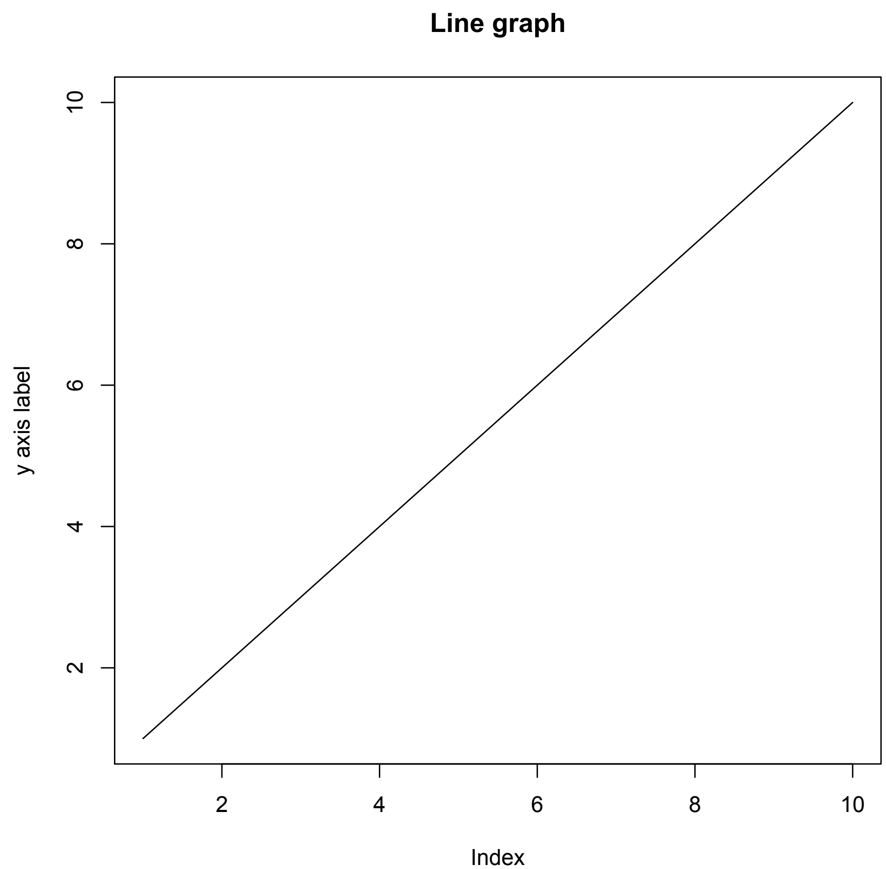

```
> plot(1:10,xlab ="Label the x axis",ylab="label the y axis",
+ main="And add a title and new data",pch=21,col="blue")
> points(1:4,3:6,bg="red",pch=22)
```

Change the ranges



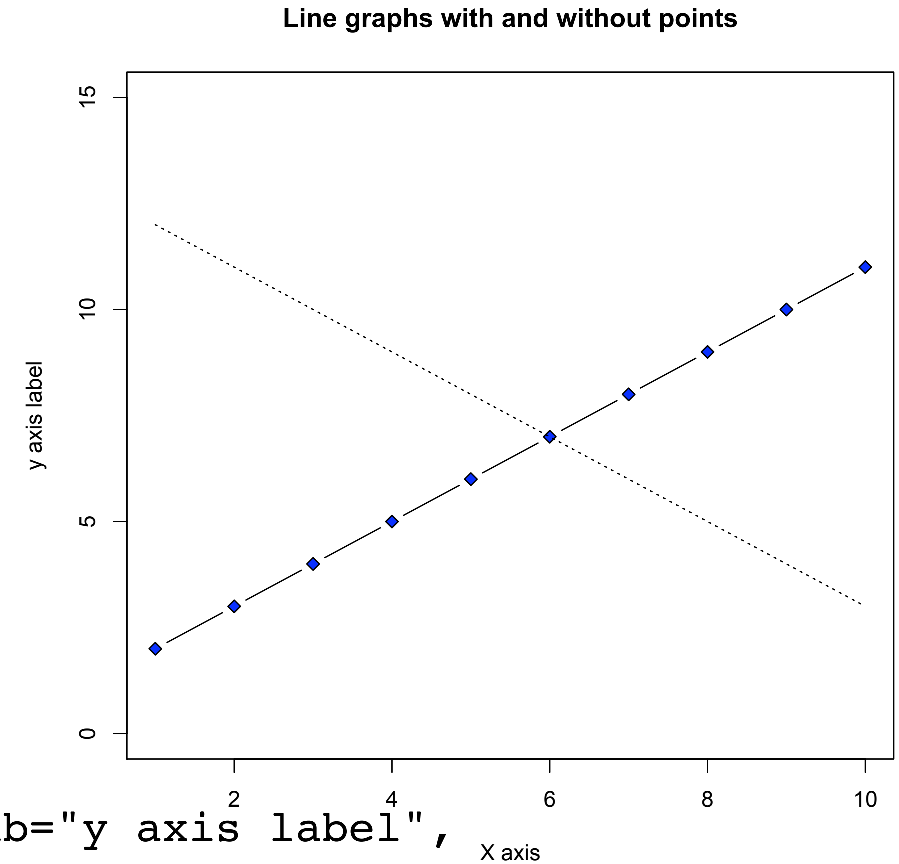

```
> plot(1:10,xlab ="x is oversized",ylab="y axis label",
+ main="Change the axis
sizes",pch=23,bg="blue",xlim=c(-5,15),ylim=c(0,20))
> points(1:4,13:16,bg="red",pch=24)
```

Plot the same data as a line graph



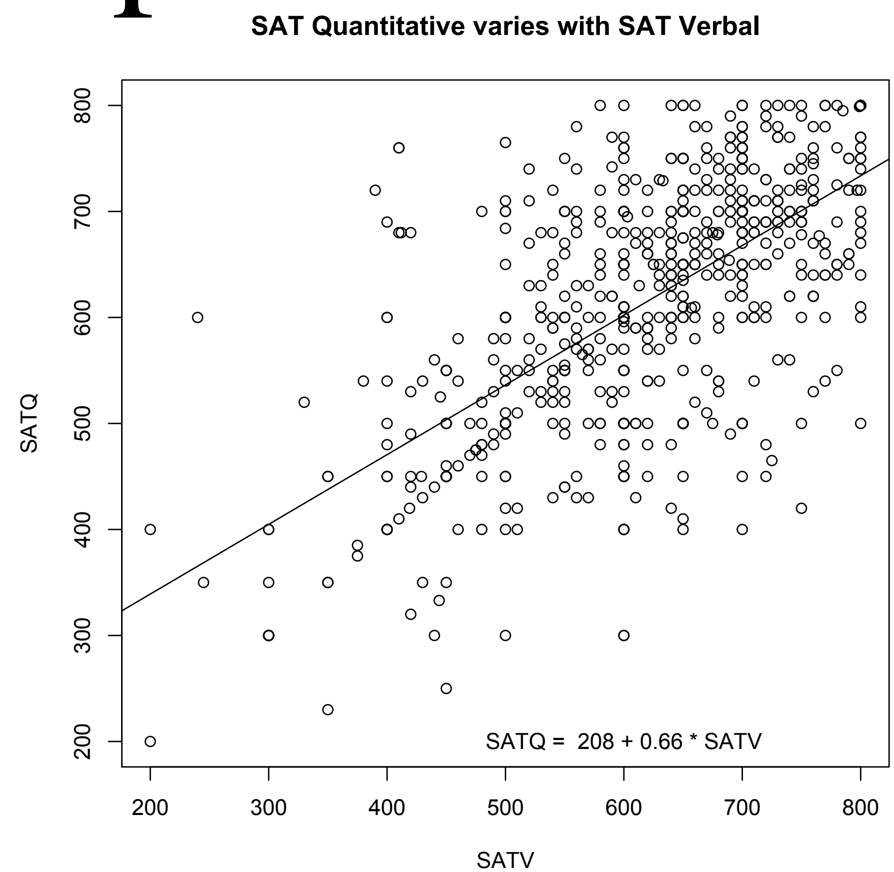

```
> plot(1:10,ylab="y axis label",main="Line
graph",pch=23,bg="blue",type="l")
```

Show the data points, add a line



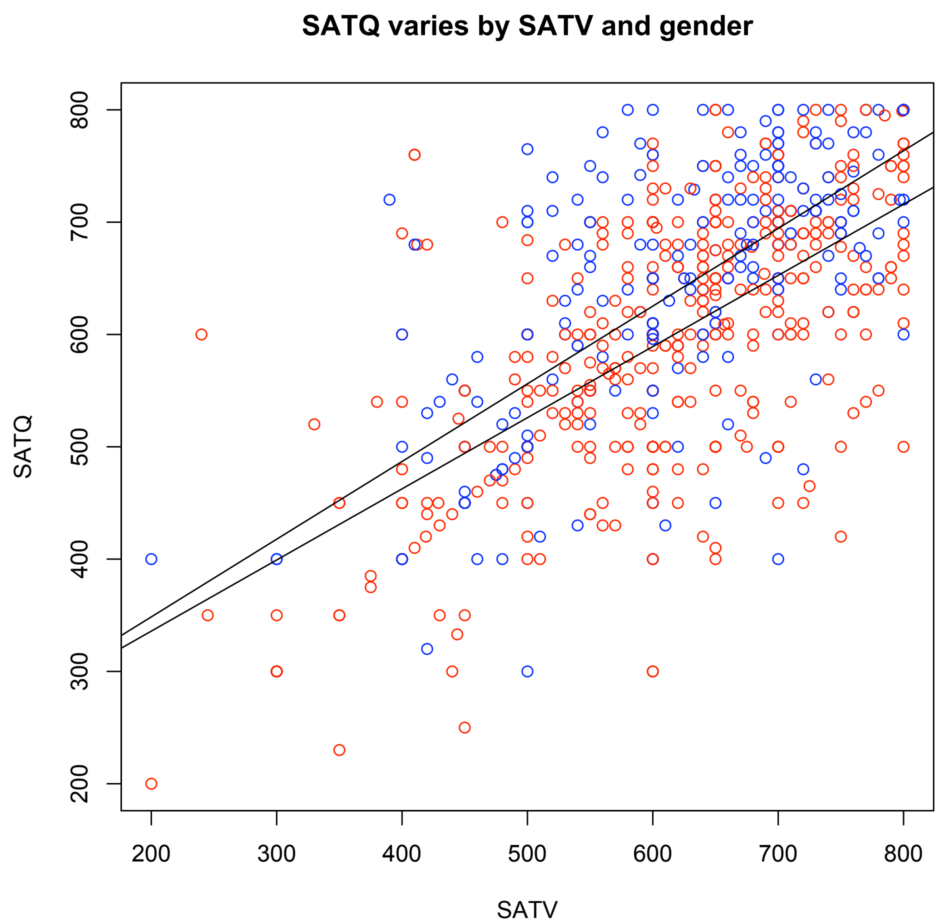

```
> plot(1:10,2:11,xlab="X axis",ylab="y axis label",
+ main="Line graphs with and without
points",pch=23,bg="blue",type="b",ylim=c(0,15))
> points(1:10,12:3,type="l",lty="dotted")
```

Regression plots




```
> data(sat.act)
> with(sat.act,plot(SATQ~SATV,main="SAT Quantitative varies with SAT
Verbal"))
> model = lm(SATQ~SATV,data=sat.act)
> abline(model)
> lab <- paste("SATQ = ",round(model$coef[1]),"+",round(model$coef[2],
2),"* SATV")
> text(600,200,lab)
```

Two groups

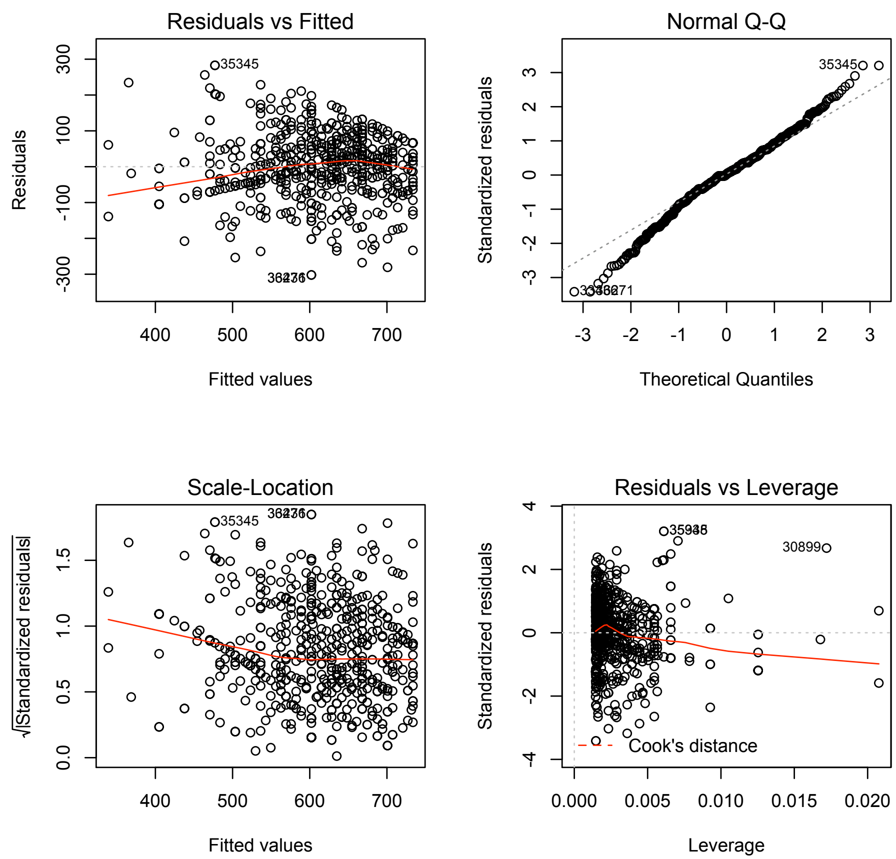



```
> data(sat.act)
> color <- c("blue","red")
> with(sat.act,plot(SATQ~SATV,col=color[gender],main="SATQ varies by
SATV and gender"))
> by(sat.act,sat.act$gender,function(x) abline(lm(SATQ~SATV,data=x)))
```

The actual regression

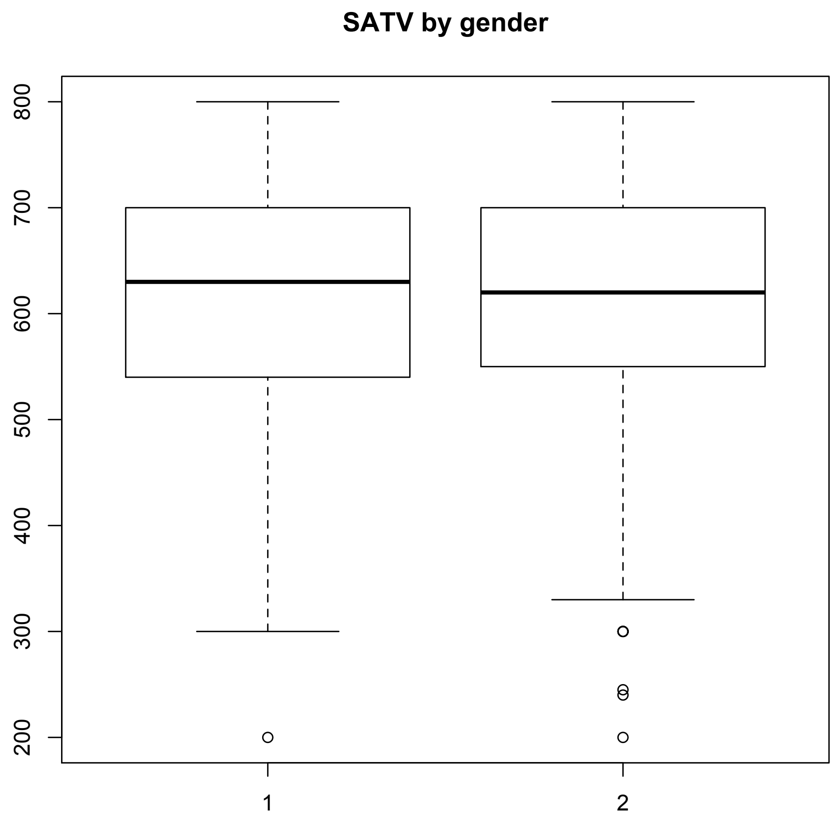
```
> by(sat.act,sat.act$gender, function(x) lm(SATQ~SATV,data=x))
sat.act$gender: 1

Call:
lm(formula = SATQ ~ SATV, data = x)

Coefficients:
(Intercept)         SATV
   210.2046       0.6917

------------------------------------------------------------------------
-
sat.act$gender: 2

Call:
lm(formula = SATQ ~ SATV, data = x)

Coefficients:
(Intercept)         SATV
   209.2093       0.6334
```

Error diagnostics




```
> op <- par(mfrow=c(2,2))
> plot(lm(SATQ~SATV,data=sat.act))
> op <- par(mfrow=c(1,1))
```

Boxplots by group



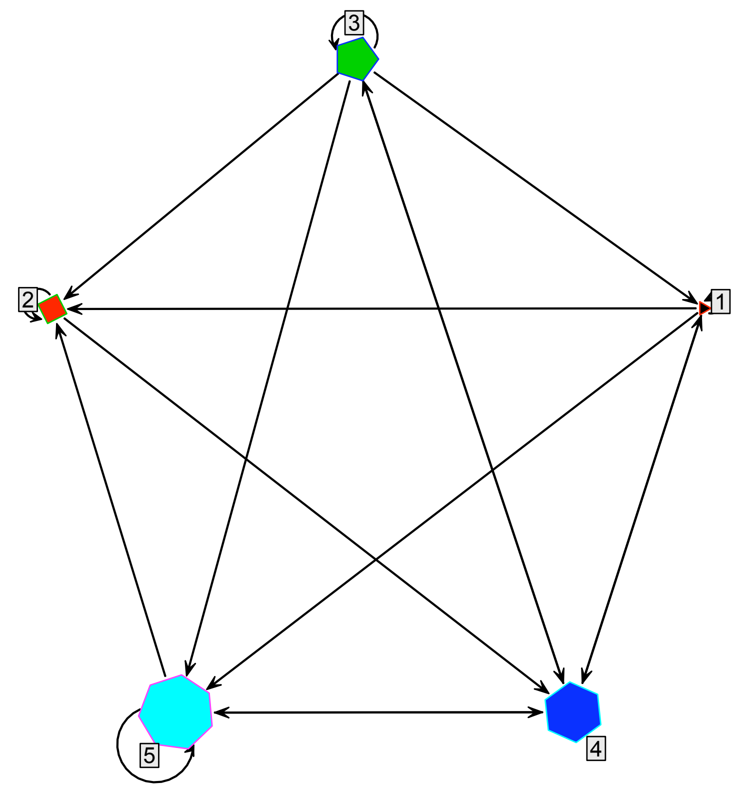

```
boxplot(sat.act$SATV~sat.act$gender,main="SATV by gender")
```

More complex graphs

I. Using the Rgraphviz package (installed from Bioconductor rather than CRAN)

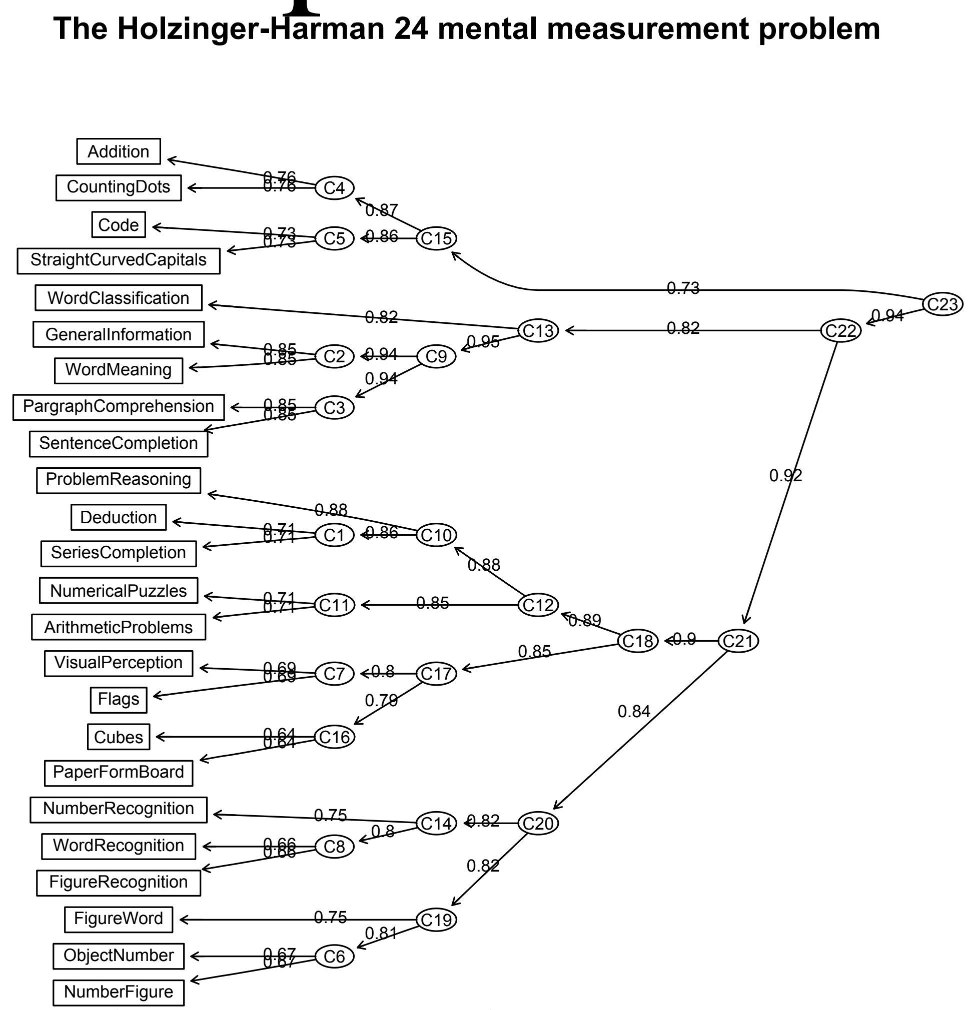
II. Using the Social Network Analysis (sna) package.

sna example



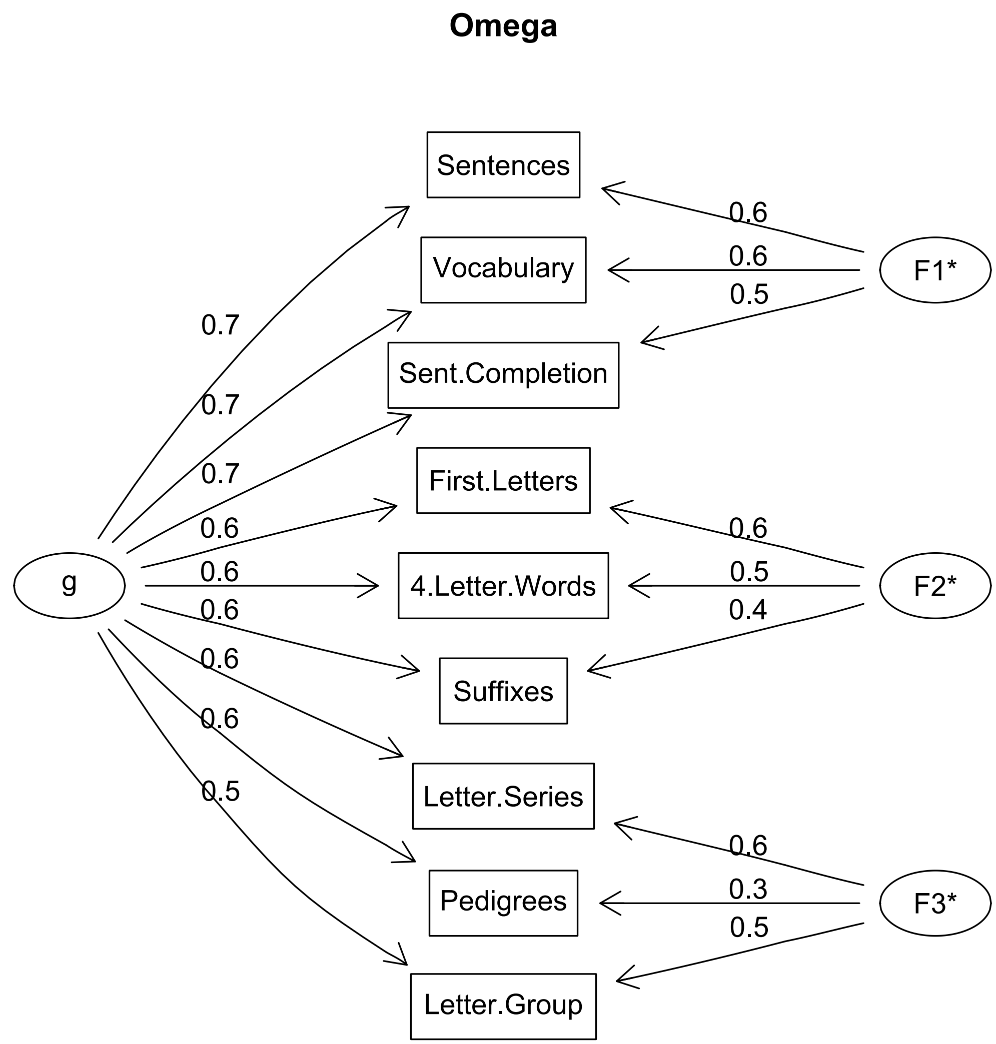

```
#A colorful demonstration...
gplot(rgraph(5,diag=TRUE),diag=TRUE,vertex.cex=1:5,vertex.sides=3:8,
    vertex.col=1:5,vertex.border=2:6,vertex.rot=(0:4)*72,
    displaylabels=TRUE,label.bg="gray90")
```

Cluster analysis with Rgraphviz output




```
> data(Harman74.cor)
> ic<- ICLUST(Harman74.cor$cov,title="The Holzinger-Harman 24 mental measurement problem")
```

A bifactor solution



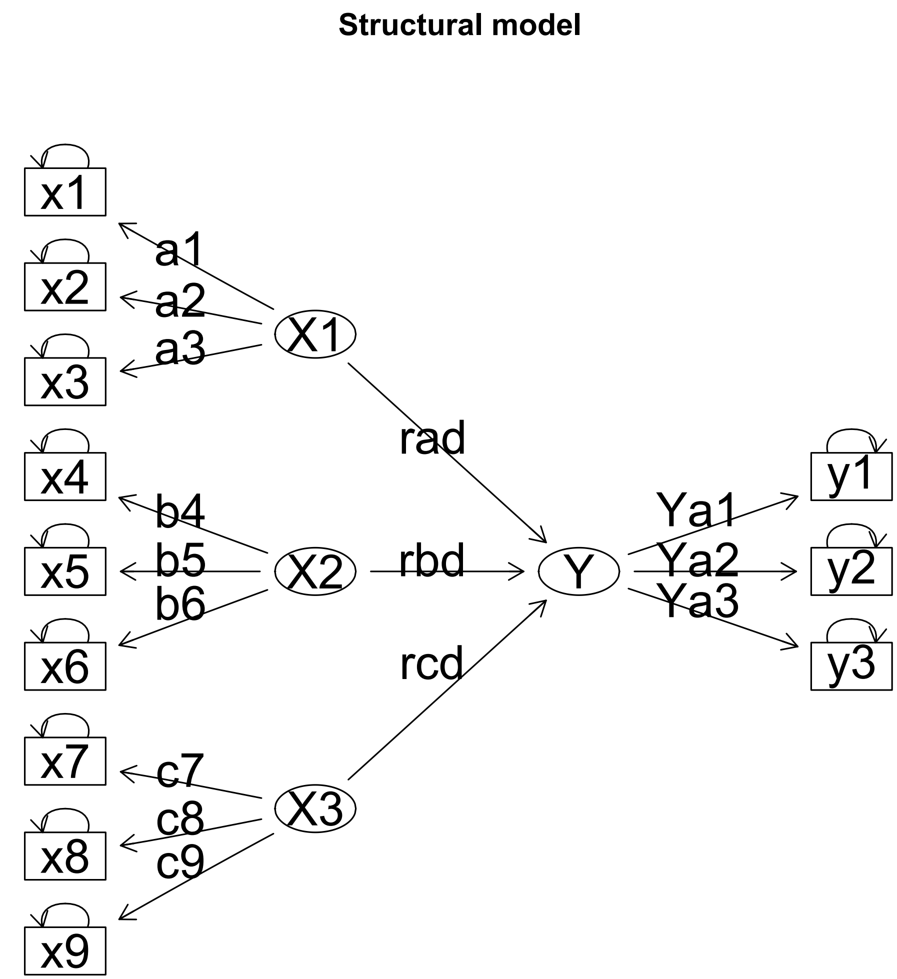

```
> data(bifactor)
> om <- omega(Thurstone,main="A bifactor solution to a Thurstone data
set")
```

Creating matrix input for a graph

```
> fxs <- structure.list(9,list(X1=c(1,2,3), X2 =c(4,5,6),X3 =
c(7,8,9)))
> phi <- phi.list(4,list(F1=c(4),F2=c(4),F3=c(4),F4=c(1,2,3)))
> fyx <- structure.list(3,list(Y=c(1,2,3)),"Y")
>
> fxs
      X1   X2   X3
 [1,] "a1" "0"  "0"
 [2,] "a2" "0"  "0"
 [3,] "a3" "0"  "0"
 [4,] "0"  "b4" "0"
 [5,] "0"  "b5" "0"
 [6,] "0"  "b6" "0"
 [7,] "0"  "0"  "c7"
 [8,] "0"  "0"  "c8"
 [9,] "0"  "0"  "c9"
```

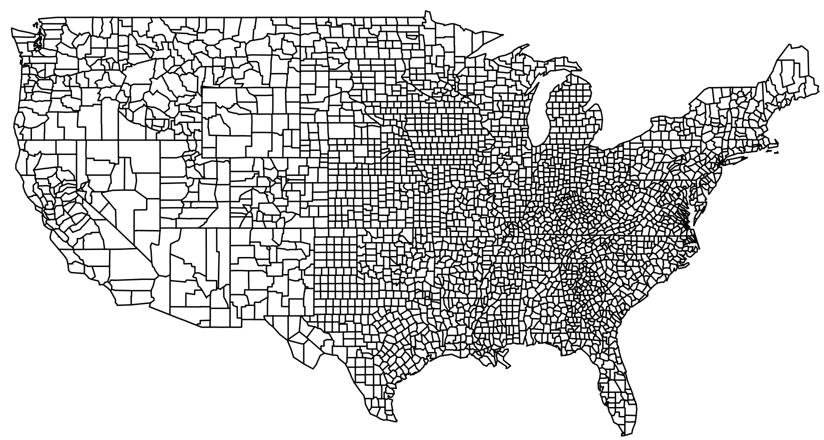
```
> phi
   F1    F2    F3    F4
F1 "1"   "0"   "0"   "rda"
F2 "0"   "1"   "0"   "rdb"
F3 "0"   "0"   "1"   "rdc"
F4 "rad" "rbd" "rcd" "1"
> fyx
      Y
[1,] "Ya1"
[2,] "Ya2"
[3,] "Ya3
```

Showing the matrices




```
> sg3 <- structure.graph(fxs,phi,fyx)
```

Using the maps package



```
> library(maps)
> map("county")
```