Statistics with R

Prepared by José Gama, MSc

Department of Mathematics
Åbo Akademi University

http://web.abo.fi/fak/mnf/mate/kurser/statisticsr/

References:

"A Practical Guide to Geostatistical Mapping of Environmental Variables", Tomislav Hengl, Joint Research Centr, Institute for Environment and Sustainability pp. 56, 80

The R Core Team, "What is R?", R News, Volume 1/1, January 2001

http://cran.r-project.org/doc/html/interface98-paper/paper_2.html

http://en.wikipedia.org/wiki/R_%28programming_language%29

Believe nothing merely because you have been told it.
Do not believe what your teacher tells you merely out of respect for the teacher.
But whatever, after due examination and analysis, you find to be kind, conducive to the good, the benefit, the welfare of all beings - that doctrine believe and cling to, and take it as your guide.

However many holy words you read, however many you speak, what good will they do you if you do not act on upon them?

There are only two mistakes one can make along the road to truth; not going all the way, and not starting.

What we think, we become.

Buddha, spiritual teacher from India

I hear and I forget. I see and I remember. I do and I understand.

Confucius

Chinese philosopher & reformer
(551 BC - 479 BC)

How the classes will be organized

Revision of R/statistical concepts

Examples of reviewed R/statistical concepts

R commands to work on these concepts

Questions, to be solved with R, by the students

Answers and explanations before moving on

Course structure

The course will be 14 lessons of 2 hours each.

The class starts 10 minutes after the scheduled hour.

The grade is pass/fail.

To pass, a student must attend 11 lessons and answer correctly 10 questions from the questionnaire.

Alternatively, a student must answer correctly 15 questions from the questionnaire.

The reading assignments are optional but recommended, in particular for students who will not attend the class.

The online material is enough to learn the basics and answer the questionnaire.

The classes will cover more details and have from 5 to 10 times more examples than the online material. This can be exhausting, but you must ask if you want something explained differently or if you need a break.

The classes will also be more friendly to students not from Computer Science, unlike the online material.

Course structure

R has many strong points, two of which are its great help system and available packages.

However, many students complaint that, after an introductory course on R, they are not comfortable with using the help or packages.

Usually it doesn't matter because they will never use R again or use it only for a few histograms or boxplots, once in a blue moon.

Students from Statistics, Bioinformatics and Environmental Sciences will have to work with many packages and find help on any topic without assistance.

That is why the initial lessons will be long and boring, but necessary to get solid foundations on R.

The R Project for Statistical Computing

“R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.”

“R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes an effective data handling and storage facility,
a suite of operators for calculations on arrays, in particular matrices, a large, coherent, integrated collection of intermediate tools for data analysis, graphical facilities for data analysis and display either on-screen or on hardcopy, and a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities. ”

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

What is R

R is a software environment and programming language for statistical computing and graphics. R is the open source equivalent to the programming language S.
S is very popular on statistical methodology research and was developed by John Chambers and, previously, by Rick Becker and Allan Wilks of Bell Laboratories.
The name "R" comes from the fact that "R" precedes "S" and both authors' names start with "R", Ross Ihaka and Robert Gentleman.
The R basic distribution comes with plenty of statistical procedures such as: linear and generalized linear models, nonlinear regression models, time series analysis, classical parametric and nonparametric tests, clustering and smoothing.
There are many graphical procedures such as: plot, scatterplot, boxplot, distribution-comparison plot, histogram, dotchart, contour lines, 3D surface, etc...
R is extensible with a multitude of packages, some of them for very specialized areas or highly optimized for intensive computations. R is a programming language, allowing object-oriented programming (OOP) and with lexical (static) scoping semantics similar to Scheme (dialect of Lisp).
C, C++, and Fortran code can be linked and called at run time, adding more power and flexibility.

The history of R

R was developed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand.
Ross Ihaka read the book "The Structure and Interpretation of Computer Programs" about the Scheme programming language. Later, he tried to use lexical scope, to obtain own variables, in S, which didn't work because of the differences in the scoping rules of S and Scheme. Years passed, and Robert Gentleman and Ross Ihaka were at the University of Auckland, both working on statistical computing. They decided to create a small Scheme-like interpreter, to be used as a software environment. It was similar to S in syntax because Scheme and S are similar and both authors were familiar with S.
There was a first release in August of 1993. In June of 1995, the source code was distributed under the terms of the Free Software Foundation's GNU general license (GPL).
The interest kept growing and a small mailing list to exchange ideas had to grow to a larger automated mailing list, then to newsgroups and the distribution of code, documentation and binaries expanded to more mirror sites. Finally, the core group of developers had to grow, as well.
In 2001, Robert Gentleman started the project Bioconductor that uses statistical computing, with R, in Computational Biology.

Portable R

R is "perfectly relocatable", that is, after being installed in one machine, the directory can be copied, for example, to a memory stick and it will run from there.
Notes:
Installing packages - download the package from CRAN, use Packages -> Install Package(s) from local zip file(s)
workspace and history can be relocated by copying .Rhistory and .RData

http://my.opera.com/semin/blog/2007/04/02/portable-r

Portable GIS

Runs from a memory stick.
Contents:
•Desktop GIS packages GRASS (windows native version 6.3: does not need cygwin), QGIS (version 0.10 with GRASS plugin) and gvSIG (version 1.1),
•FWTools (GDAL and OGR toolkit, version 2.10)
•XAMPPlite (Apache2/MySQL5/Php5),
•PostgreSQL (version 8.2)/Postgis (version 1.1),
•Mapserver, OpenLayers, Tilecache, Featureserver, and Geoserver web applications.

http://www.archaeogeek.com/blog/portable-gis/

Portable GIMP

The GIMP (GNU Image Manipulation Program), Open Source image editor in a portable version:

http://portableapps.com/apps/graphics_pictures/gimp_portable

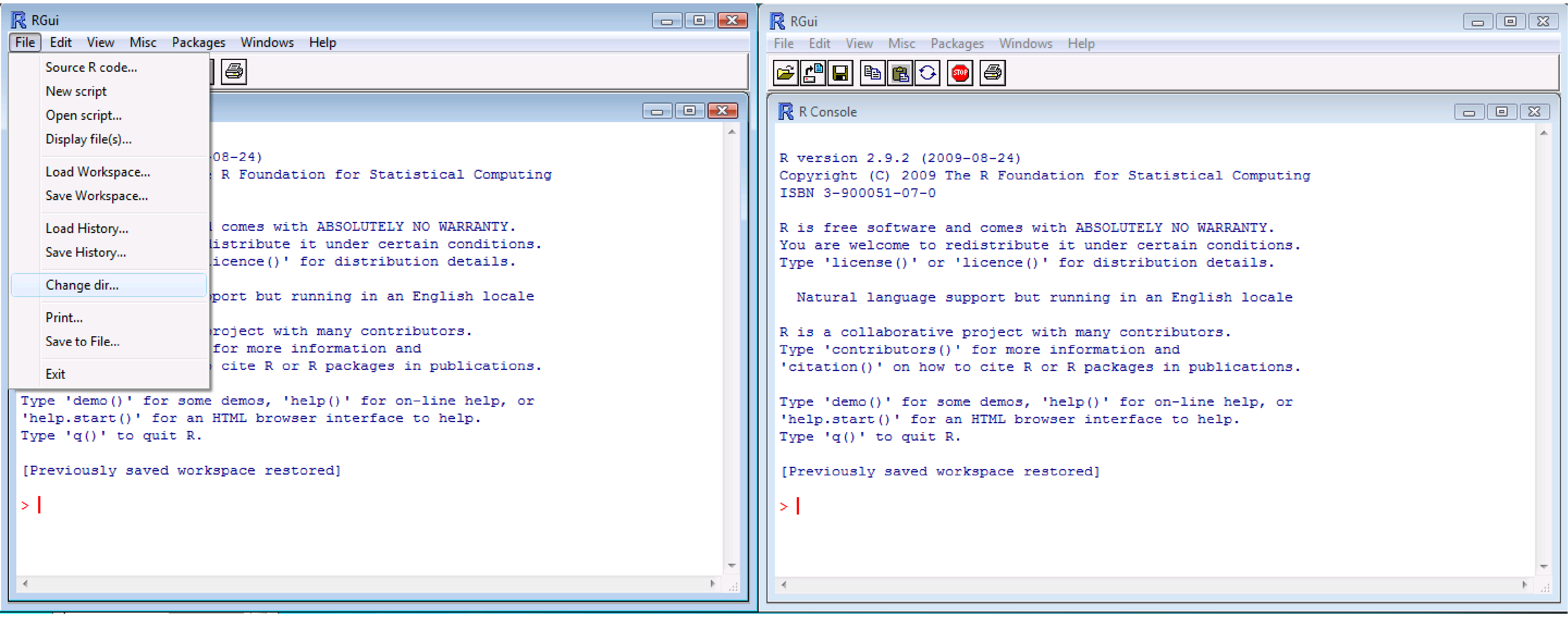
OpenOffice.org Portable

OpenOffice.org Portable is a complete OpenOffice.org office suite, compatible with Microsoft Office, Word Perfect, Lotus and other office applications.
Includes:
•Word processor
•Spreadsheet
•Presentation tool
•Drawing package
•Database

http://portableapps.com/apps/office/openoffice_portable

Two students sharing one computer

Open two R windows and change the working directory on both:

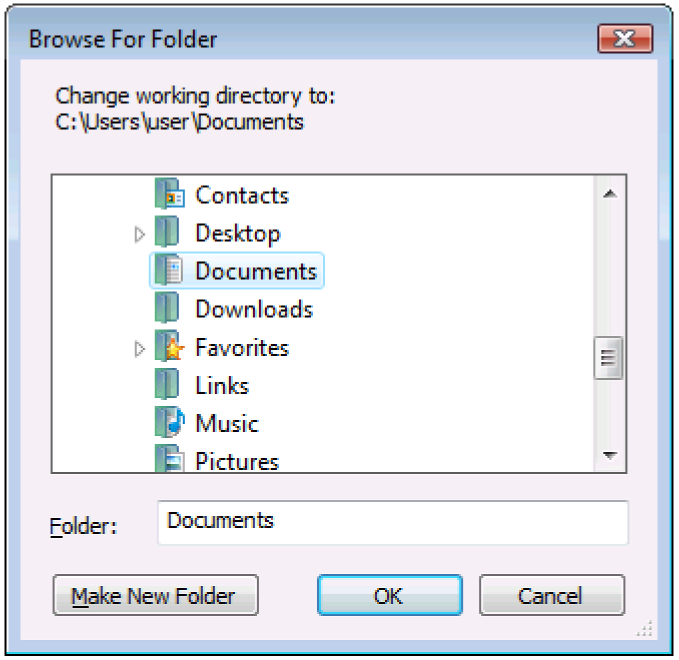


```
08-24)
R Foundation for Statistical Computing

comes with ABSOLUTELY NO WARRANTY.
istribute it under certain conditions.
icence()' for distribution details.

port but running in an English locale

roject with many contributors.
for more information and
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]

>
```

```
R version 2.9.2 (2009-08-24)
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]

>
```

Two students sharing one computer

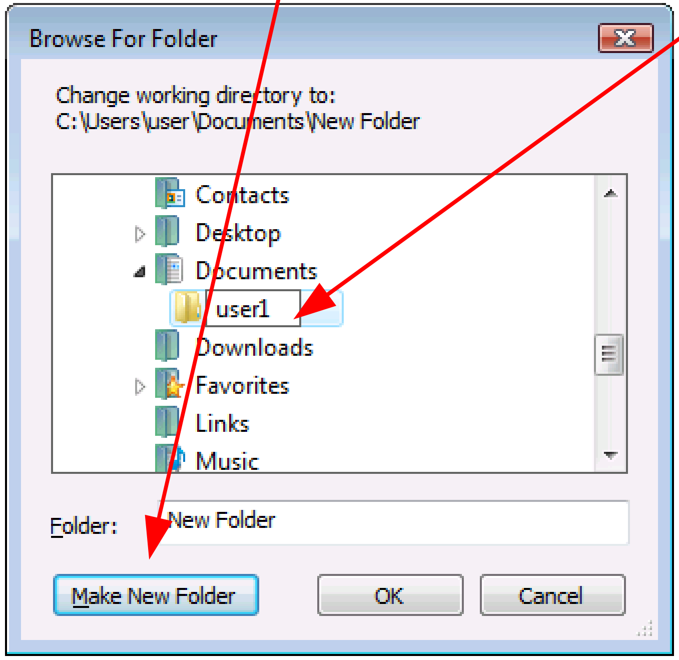
If the default is user\Documents


Browse For Folder

Change working directory to:
C:\Users\user\Documents

Folder: Documents

Click Make New Folder and name it user1


Browse For Folder

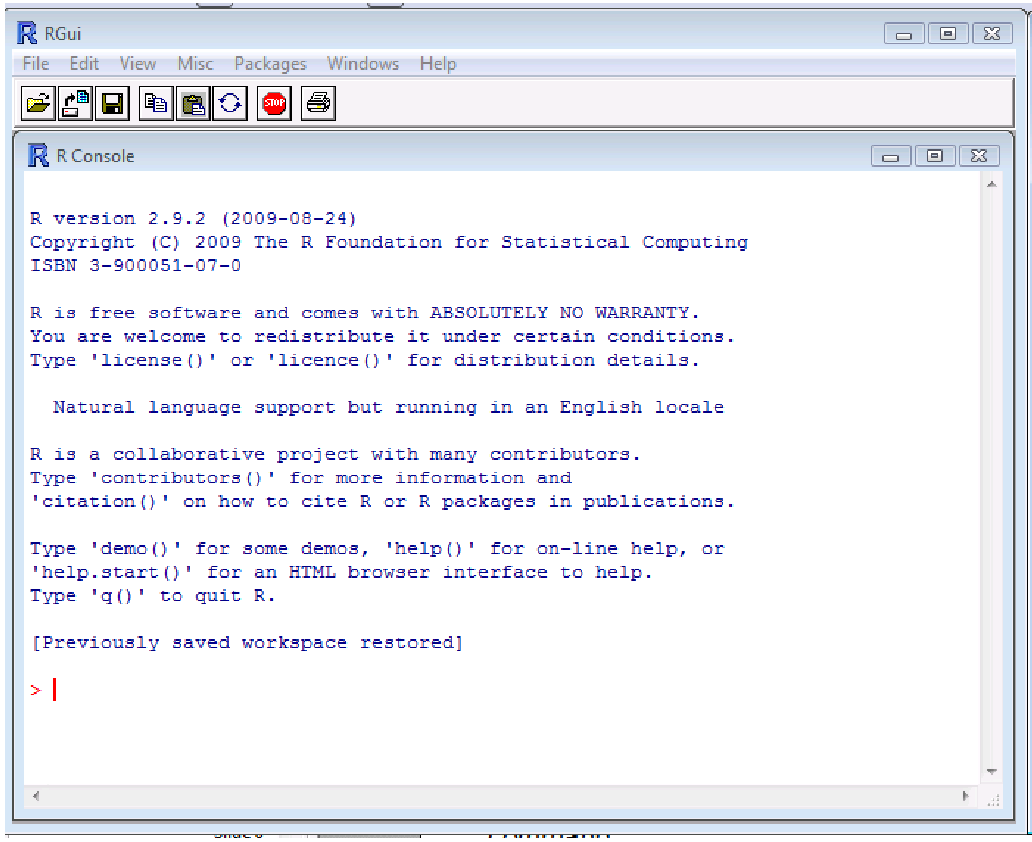
Change working directory to:
C:\Users\user\Documents\New Folder

Folder: New Folder

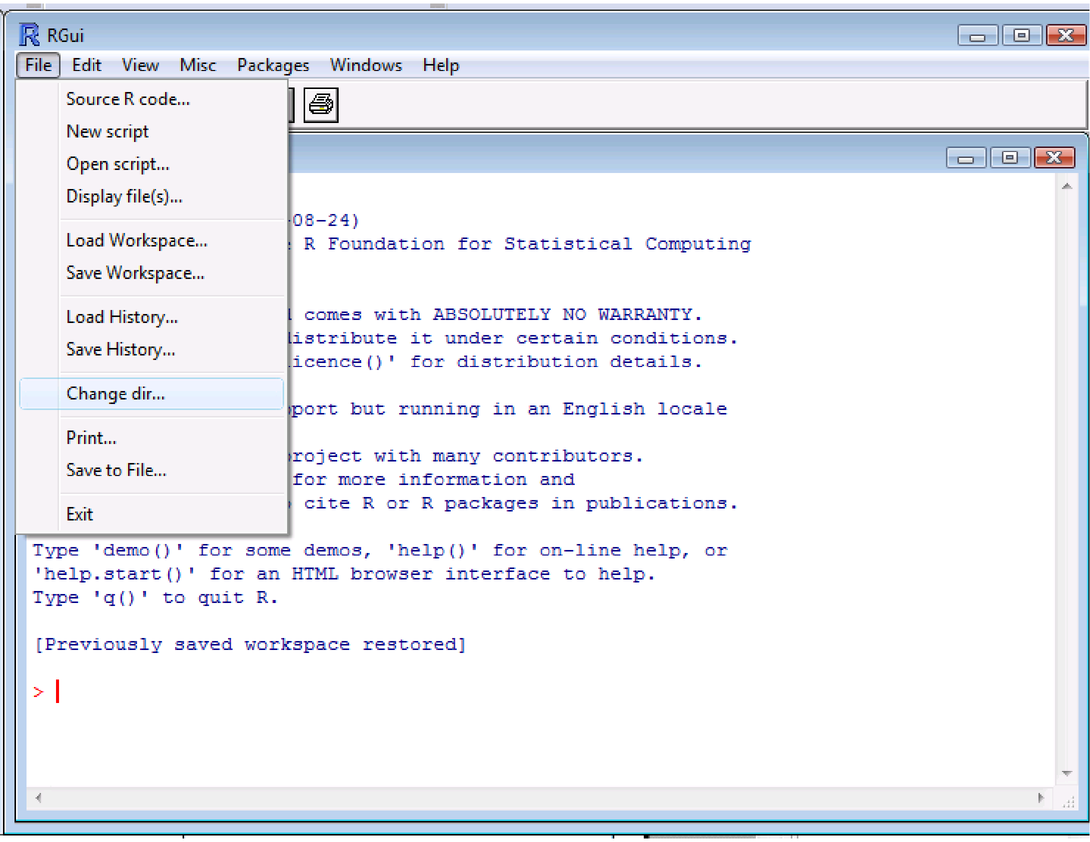
Do the same for the other R window but creating a directory user2

Two students sharing one computer

Change the working directory on the other R window:



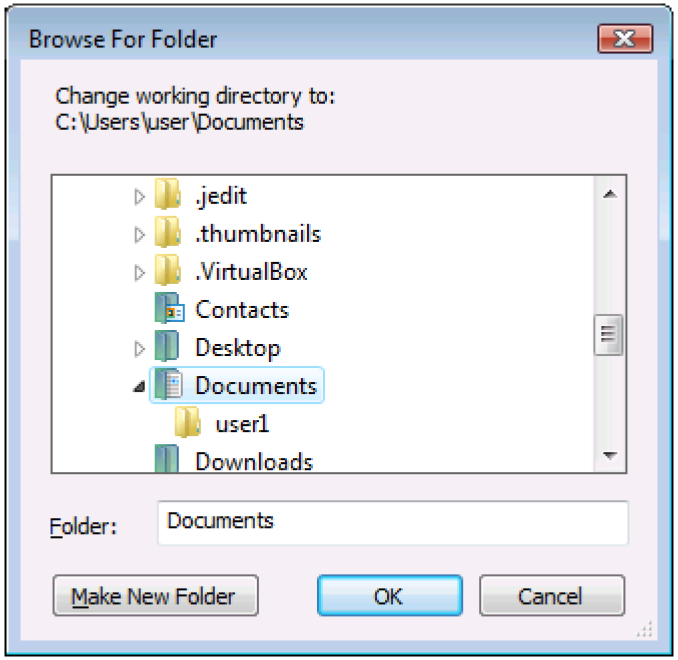
```
R version 2.9.2 (2009-08-24)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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]

>
```



```
                   08-24)
                 R Foundation for Statistical Computing

                  comes with ABSOLUTELY NO WARRANTY.
                 istribute it under certain conditions.
                 icence()' for distribution details.

                 port but running in an English locale

                 roject with many contributors.
                  for more information and
                  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]

>
```

Two students sharing one computer

If the default is user\Documents

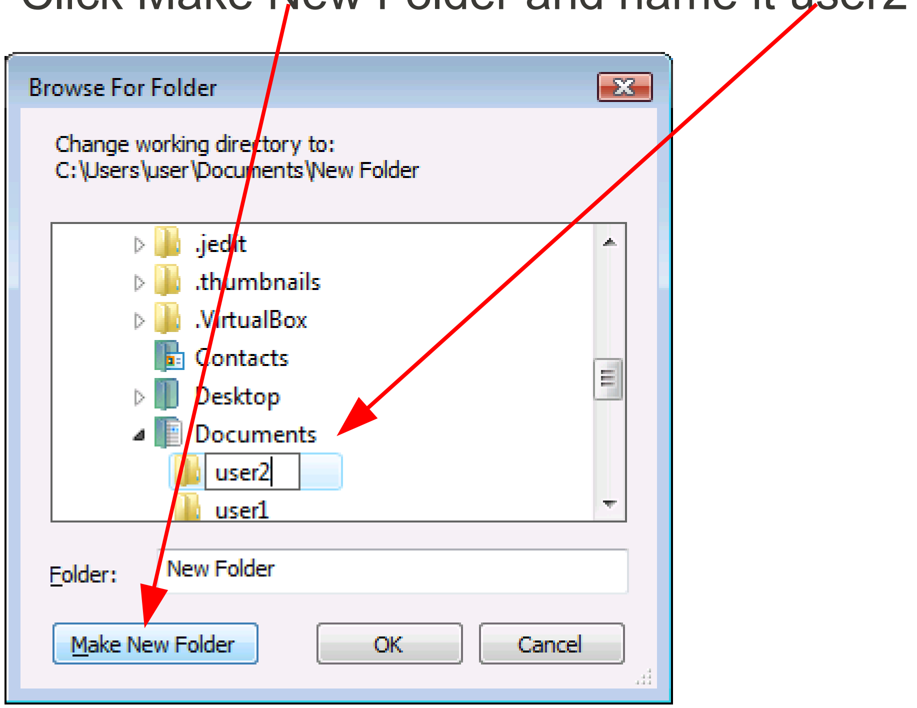


Browse For Folder

Change working directory to:
C:\Users\user\Documents

Folder: Documents

Click Make New Folder and name it user2



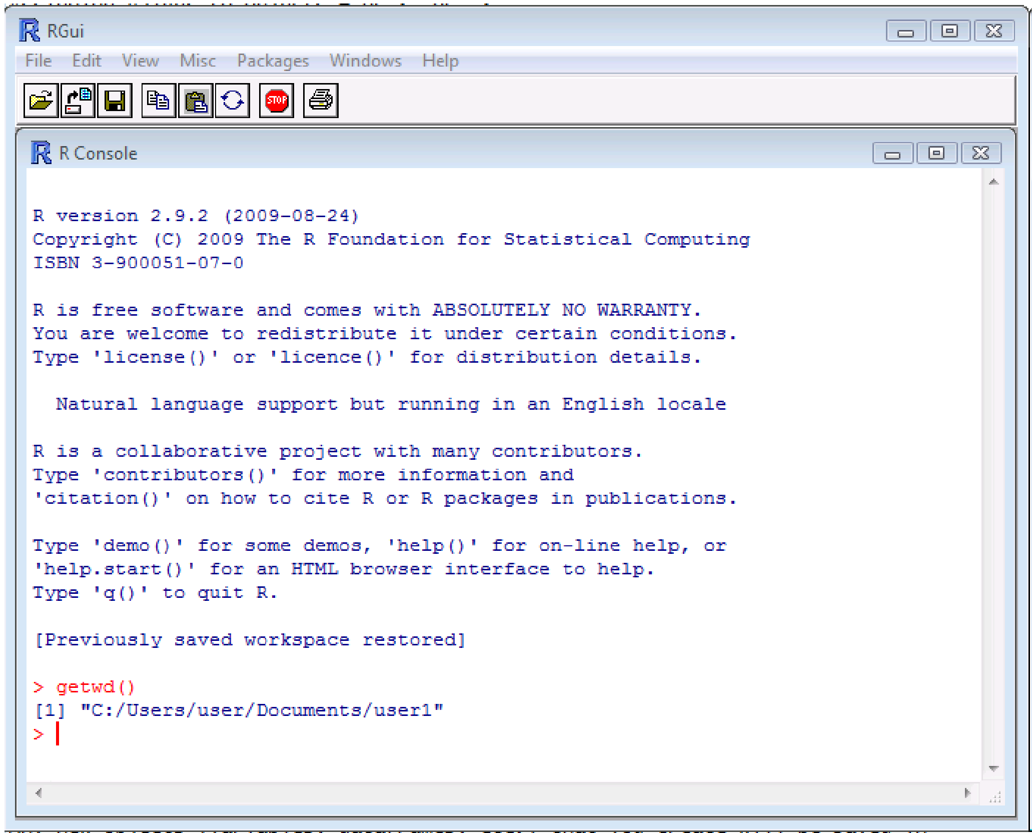
Browse For Folder

Change working directory to:
C:\Users\user\Documents\New Folder

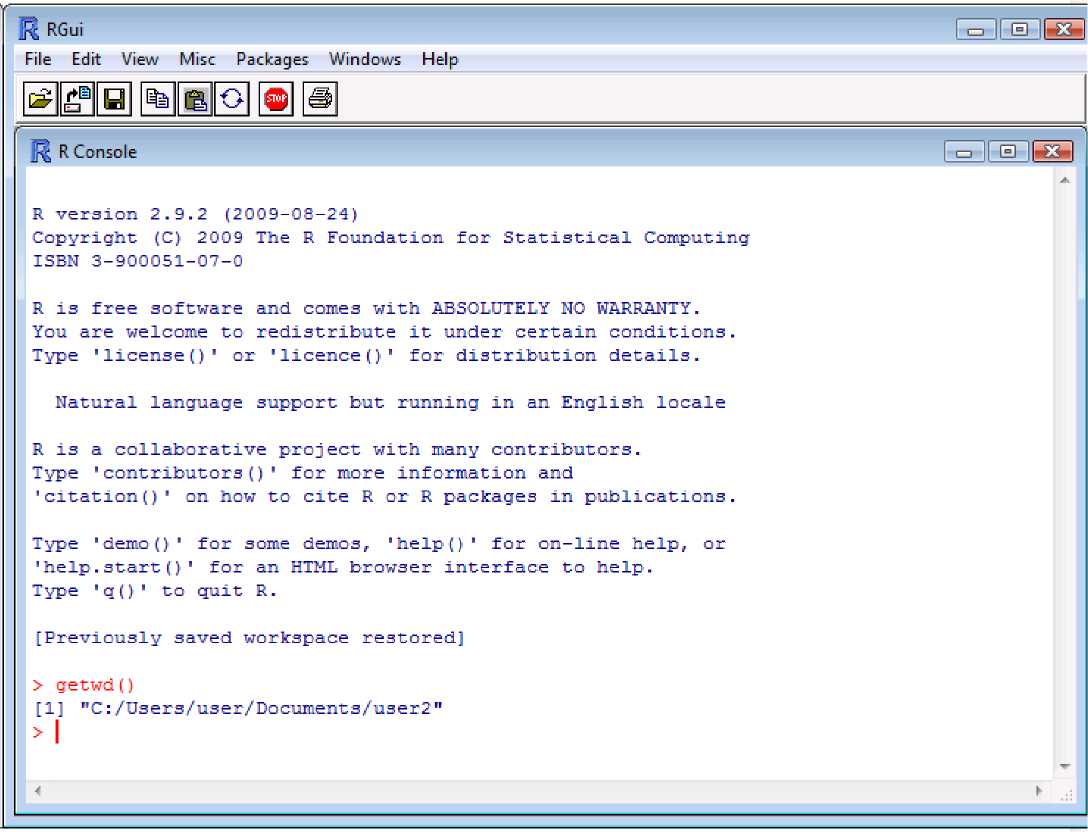
Folder: New Folder

Two students sharing one computer

Use getwd() to check the current working directory:



```
R version 2.9.2 (2009-08-24)
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]

> getwd()
[1] "C:/Users/user/Documents/user1"
> 
```



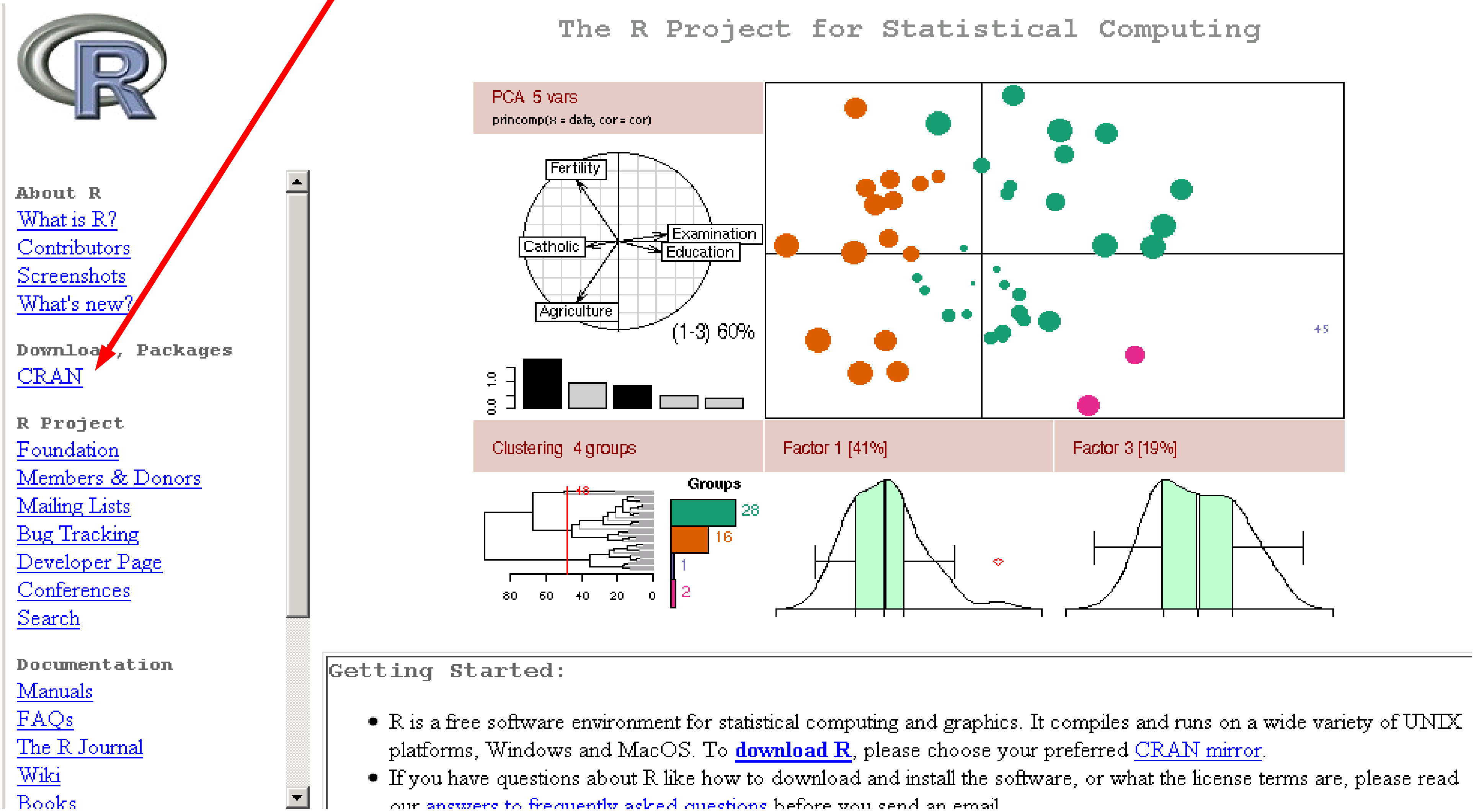
```
R version 2.9.2 (2009-08-24)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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]

> getwd()
[1] "C:/Users/user/Documents/user2"
> 
```

Both students can take turns on the computer, using their own R window and saving their work to separate workspaces.

Downloading and Installing R

The R Project for Statistical Computing Homepage:
http://www.r-project.org/

That page has a link to anther page with the CRAN mirrors



The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

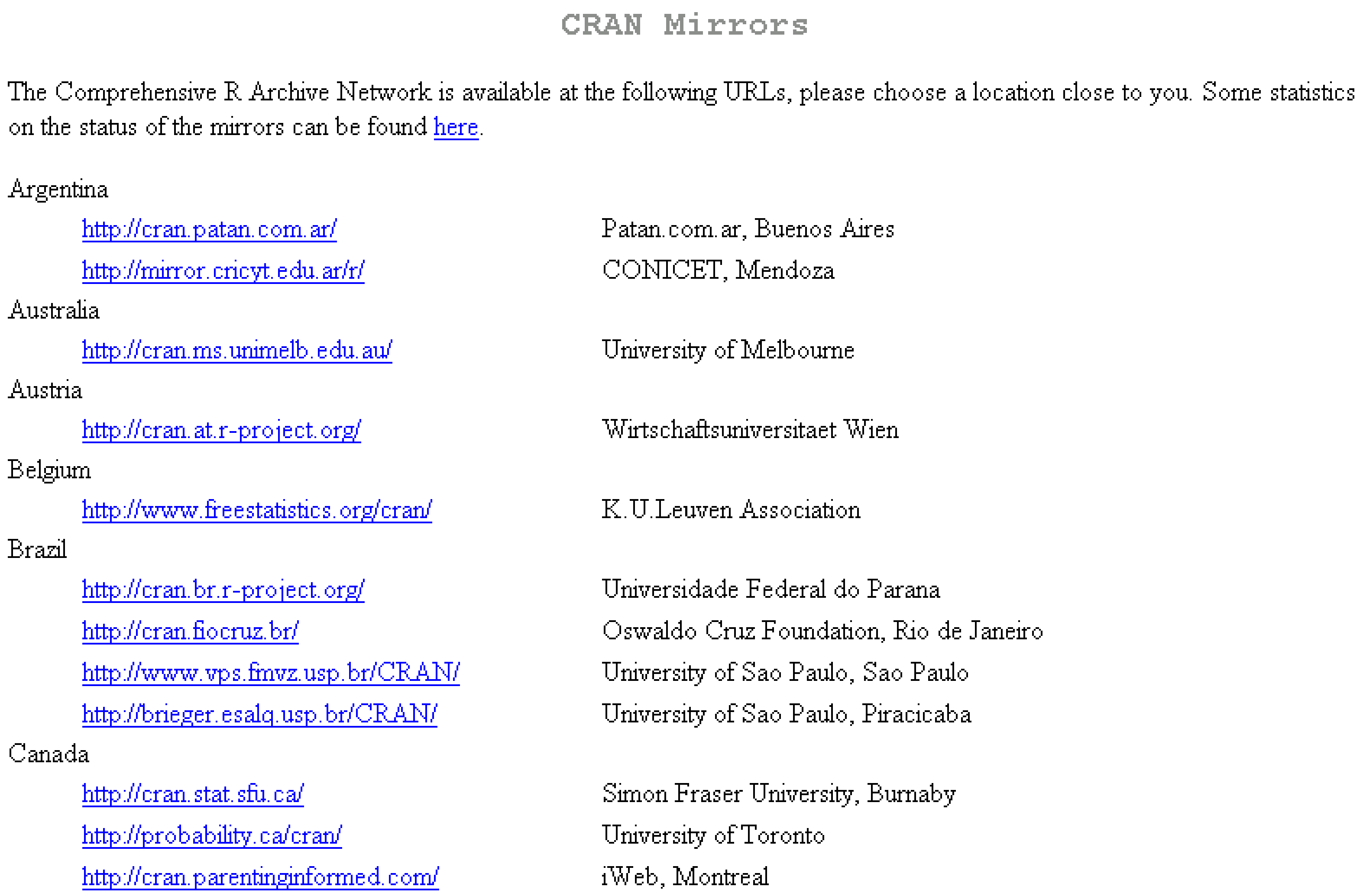
R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- 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.

Scrolling down the list, there are links to Sweden, UK and the US, among many others

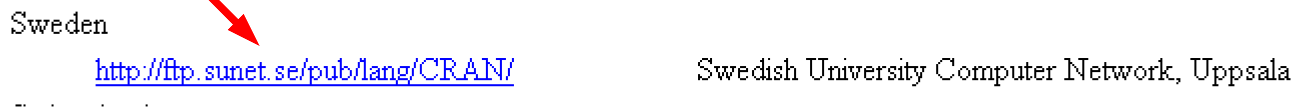


CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here.

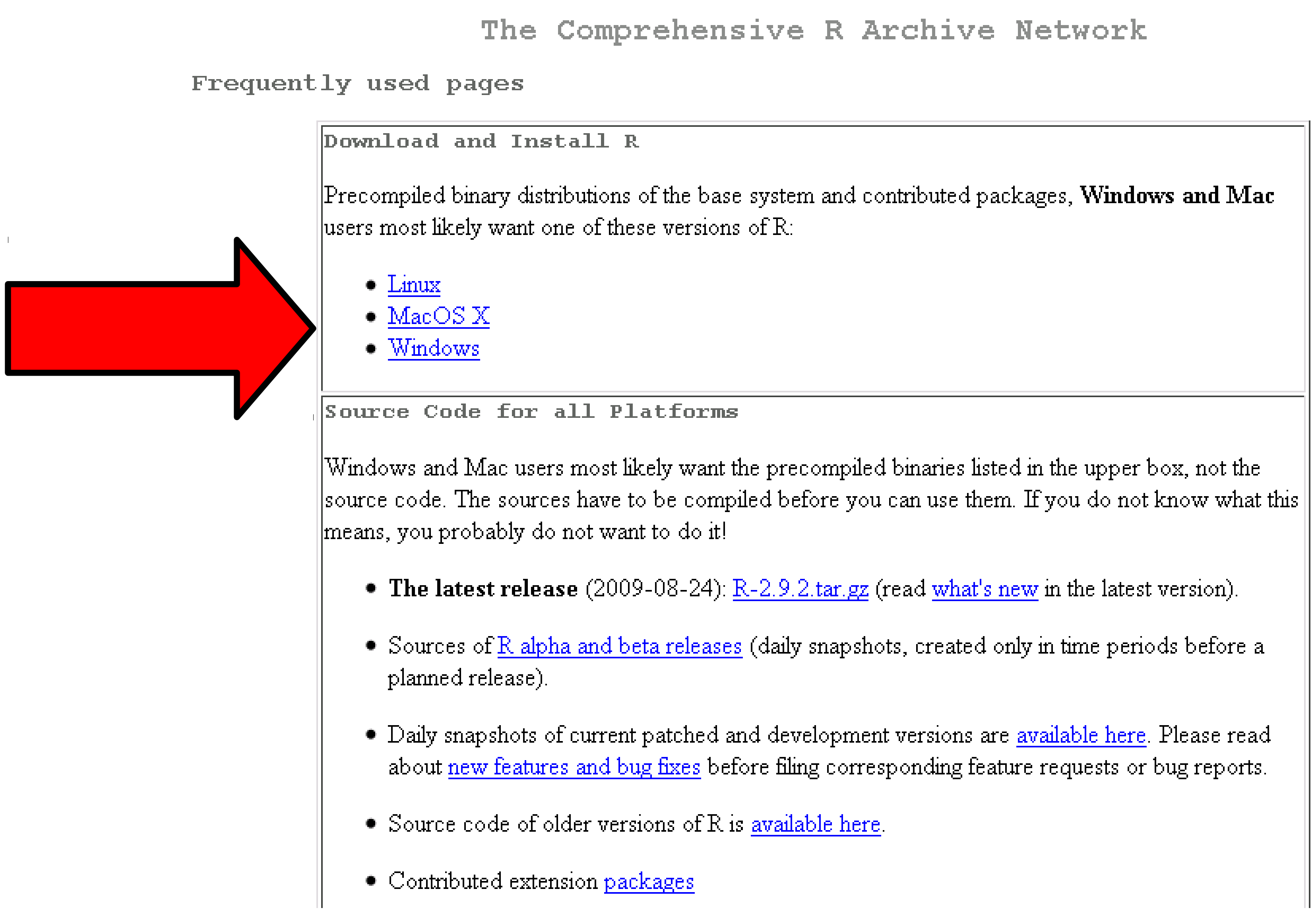
Argentina	
http://cran.patan.com.ar/	Patan.com.ar, Buenos Aires
http://mirror.cricyt.edu.ar/r/	CONICET, Mendoza
Australia	
http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	
http://cran.at.r-project.org/	Wirtschaftsuniversitaet Wien
Belgium	
http://www.freestatistics.org/cran/	K.U.Leuven Association
Brazil	
http://cran.br.r-project.org/	Universidade Federal do Parana
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Canada	
http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://probability.ca/cran/	University of Toronto
http://cran.parentinginformed.com/	iWeb, Montreal

Scroll down to Sweden



Sweden	
http://ftp.sunet.se/pub/lang/CRAN/	Swedish University Computer Network, Uppsala

Links to the different platforms



The Comprehensive R Archive Network

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:

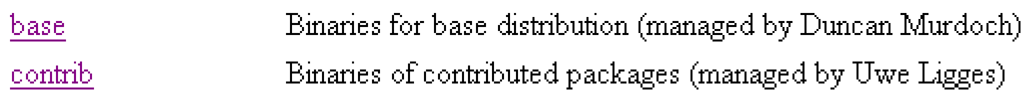
- Linux
- MacOS X
- Windows

Source Code for all Platforms

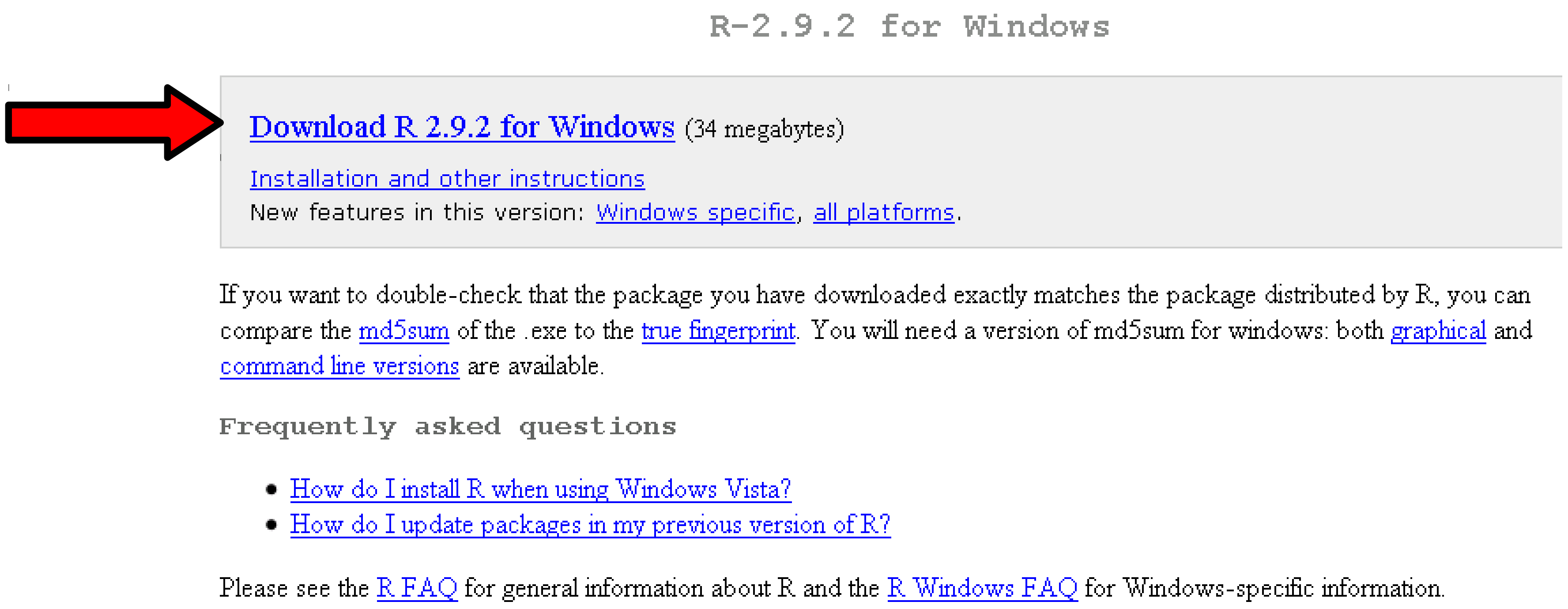
Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2009-08-24): R-2.9.2.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

Let's download the Windows version (base)



base	Binaries for base distribution (managed by Duncan Murdoch)
contrib	Binaries of contributed packages (managed by Uwe Ligges)



R-2.9.2 for Windows

Download R 2.9.2 for Windows (34 megabytes)

Installation and other instructions
New features in this version: Windows specific, all platforms.

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the md5sum of the .exe to the true fingerprint. You will need a version of md5sum for windows: both graphical and command line versions are available.

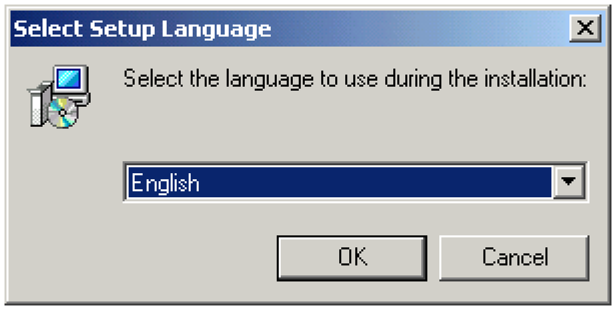
Frequently asked questions

- How do I install R when using Windows Vista?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Installing the Windows version (base)

Download R-2.9.2-win32.exe (the 2.9.2 is the version number, it might be different) and execute it. There are several languages that can be used during the installation, which is very straightforward.

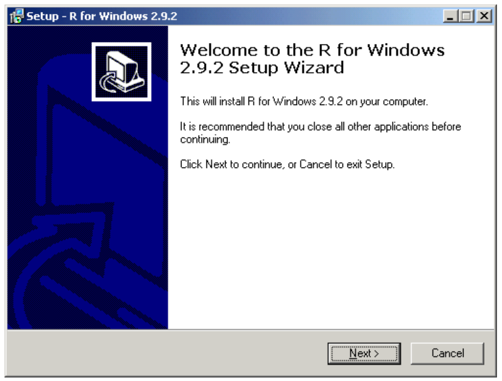


Select Setup Language

Select the language to use during the installation:

English

OK | Cancel



Setup - R for Windows 2.9.2

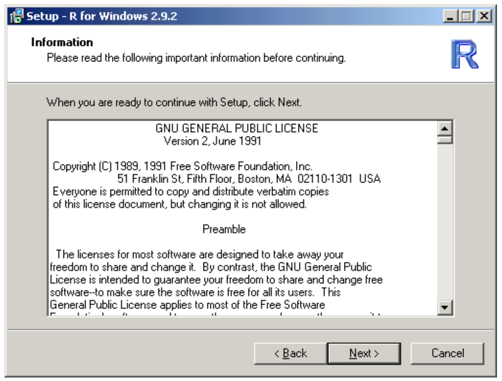
Welcome to the R for Windows 2.9.2 Setup Wizard

This will install R for Windows 2.9.2 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Next > | Cancel



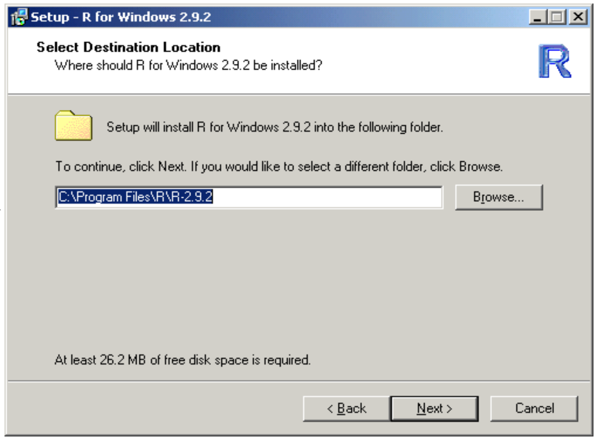
Setup - R for Windows 2.9.2

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

```
                GNU GENERAL PUBLIC LICENSE
                   Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                        Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
```

< Back | Next > | Cancel



Setup - R for Windows 2.9.2

Select Destination Location
Where should R for Windows 2.9.2 be installed?

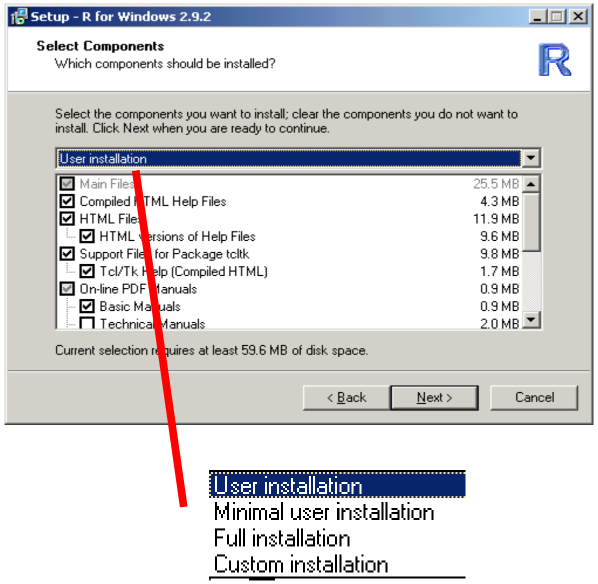
Setup will install R for Windows 2.9.2 into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\R\R-2.9.2 | Browse...

At least 26.2 MB of free disk space is required.

< Back | Next > | Cancel



Setup - R for Windows 2.9.2

Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

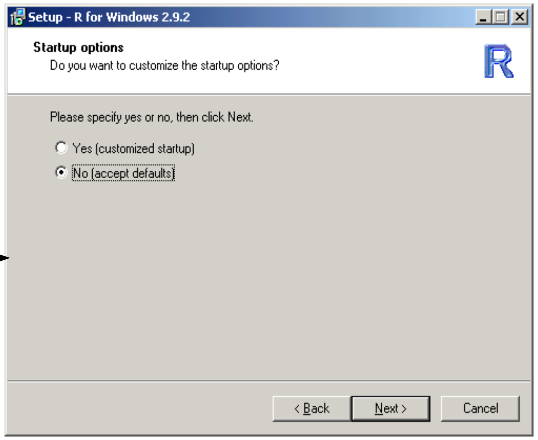
User installation

✓ Main Files	25.5 MB
✓ Compiled HTML Help Files	4.3 MB
✓ HTML Files	11.9 MB
✓ HTML versions of Help Files	9.6 MB
✓ Support Files for Package tcltk	9.8 MB
✓ Tcl/Tk Help (Compiled HTML)	1.7 MB
✓ On-line PDF Manuals	0.9 MB
✓ Basic Manuals	0.9 MB
☐ Technical Manuals	2.0 MB

Current selection requires at least 59.6 MB of disk space.

< Back | Next > | Cancel

User installation
Minimal user installation
Full installation
Custom installation



Setup - R for Windows 2.9.2

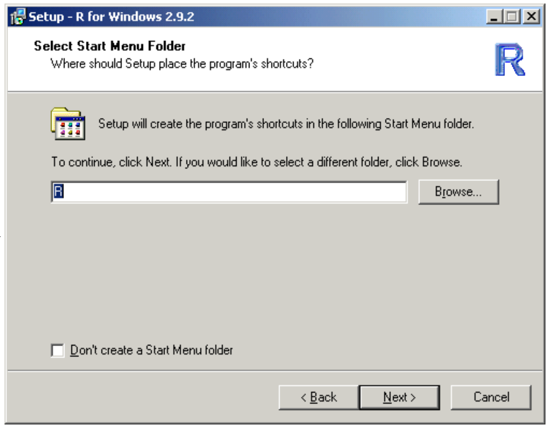
Startup options
Do you want to customize the startup options?

Please specify yes or no, then click Next.

○ Yes (customized startup)
● No (accept defaults)

< Back | Next > | Cancel

Installing the Windows version (base)


Setup - R for Windows 2.9.2

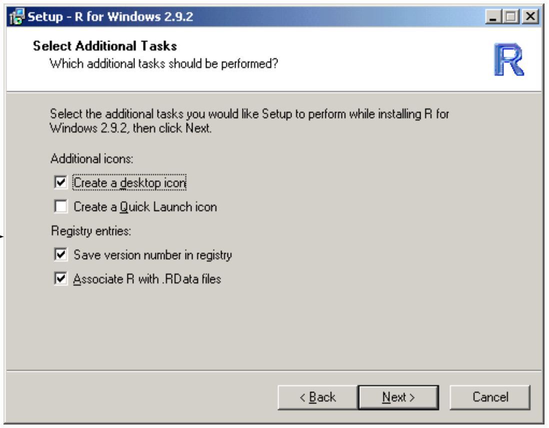
Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

R

☐ Don't create a Start Menu folder

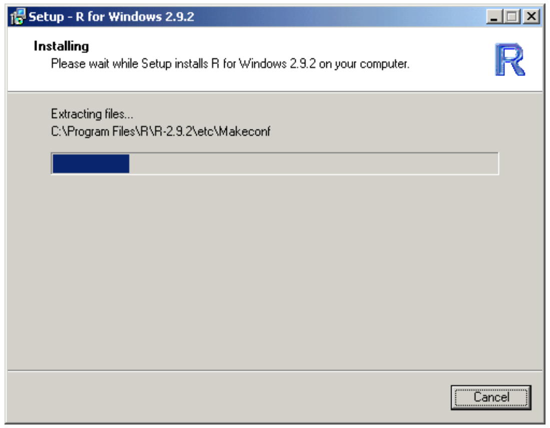

Setup - R for Windows 2.9.2

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing R for Windows 2.9.2, then click Next.

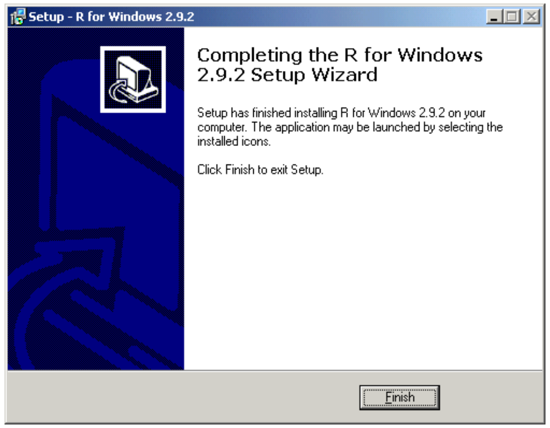
Additional icons:
☑ Create a desktop icon
☐ Create a Quick Launch icon

Registry entries:
☑ Save version number in registry
☑ Associate R with .RData files


Setup - R for Windows 2.9.2

Installing
Please wait while Setup installs R for Windows 2.9.2 on your computer.

Extracting files...
C:\Program Files\R\R-2.9.2\etc\Makeconf


Setup - R for Windows 2.9.2

Completing the R for Windows 2.9.2 Setup Wizard

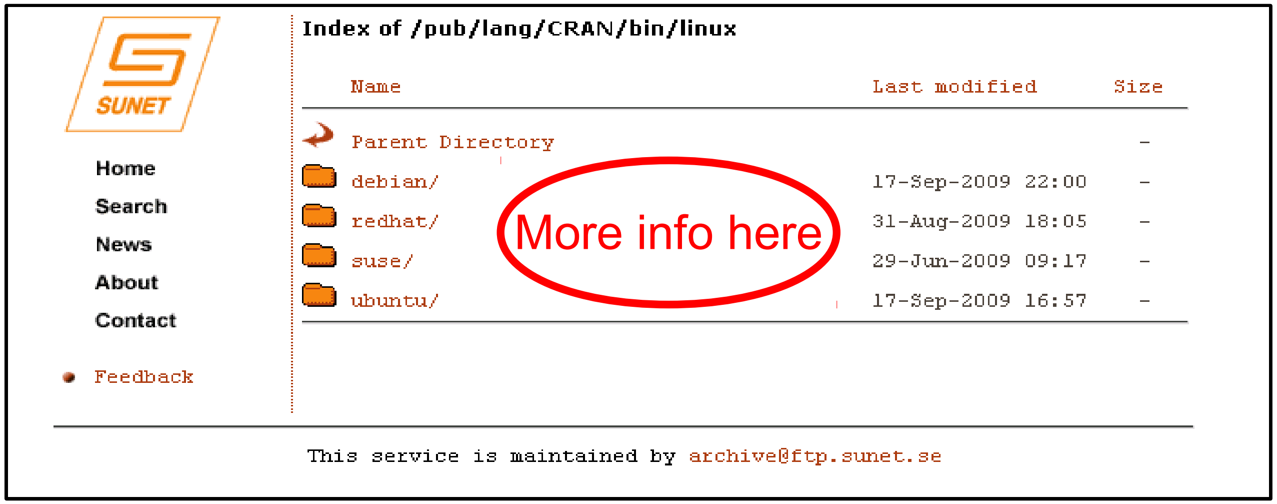
Setup has finished installing R for Windows 2.9.2 on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

Installing the Linux version

On Fedora 10, as root:
yum install R

On Fedora 8 or 9, as root:
yum install R R-devel



Index of /pub/lang/CRAN/bin/linux

```
    Name                                        Last modified      Size
    Parent Directory                                                  -
    debian/                                     17-Sep-2009 22:00     -
    redhat/                                     31-Aug-2009 18:05     -
    suse/                                       29-Jun-2009 09:17     -
    ubuntu/                                     17-Sep-2009 16:57     -
```

This service is maintained by archive@ftp.sunet.se

Installing the Linux version

On Ubuntu:

```
gpg --keyserver subkeys.pgp.net --recv-key E2A11821
gpg -a --export E2A11821 | sudo apt-key add -

sudo gedit /etc/apt/sources.list
```

Add this line to the bottom of the sources.list file:

```
deb http://rh-mirror.linux.iastate.edu/CRAN/bin/linux/ubuntu hardy/
```

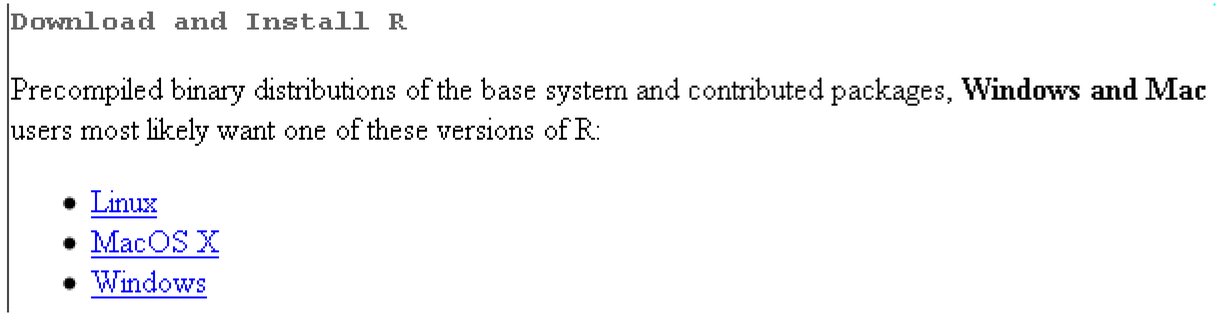
Use your own: feisty or jaunty, etc... instead of hardy
Save the file and go back to the Bash terminal.

```
sudo apt-get update

sudo apt-get install r-base r-base-dev
```

From: https://stat.ethz.ch/pipermail/r-help/2009-February/187644.html

Installing the MacOSX version

First, download "R-2.9.2.dmg" from the "bin/macosx" directory of a CRAN site

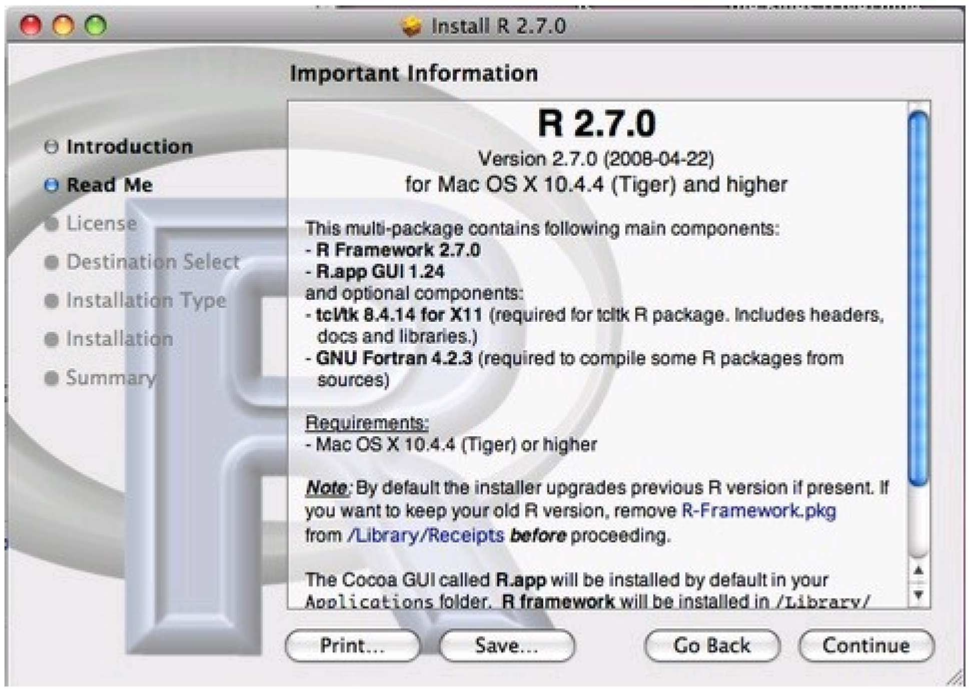


Download and Install R

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

Double-click on the icon to mount the disk image file



R 2.7.0
Version 2.7.0 (2008-04-22)
for Mac OS X 10.4.4 (Tiger) and higher

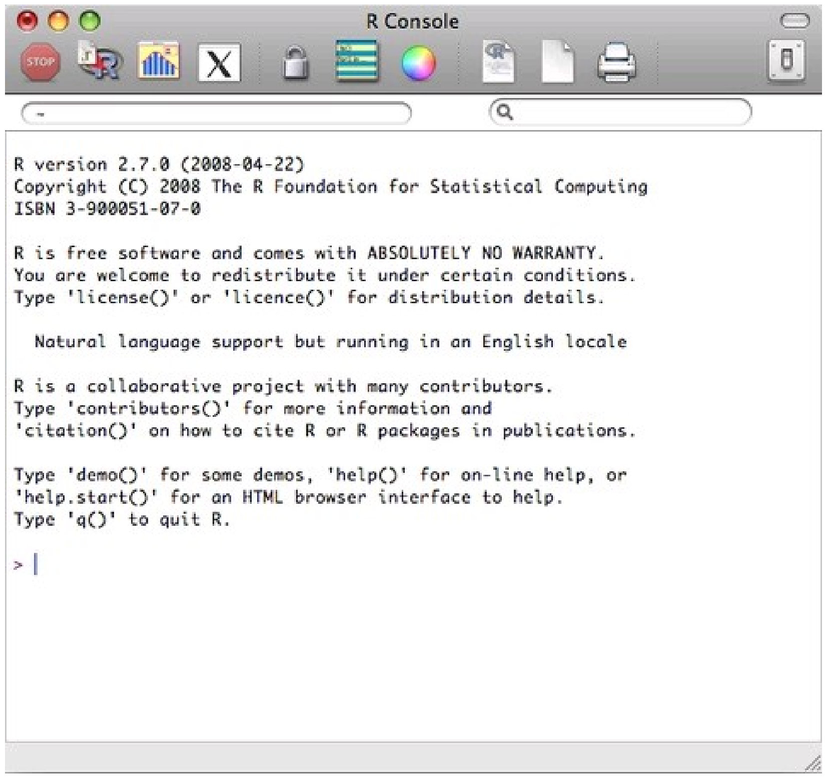
This multi-package contains following main components:
- **R Framework 2.7.0**
- **R.app GUI 1.24**

and optional components:
- **tcl/tk 8.4.14 for X11** (required for tcltk R package. Includes headers, docs and libraries.)
- **GNU Fortran 4.2.3** (required to compile some R packages from sources)

Requirements:
- Mac OS X 10.4.4 (Tiger) or higher

Note: By default the installer upgrades previous R version if present. If you want to keep your old R version, remove R-Framework.pkg from /Library/Receipts ***before*** proceeding.

The Cocoa GUI called **R.app** will be installed by default in your Applications folder. **R framework** will be installed in /Library/



```
R version 2.7.0 (2008-04-22)
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.

> |
```

http://blogs.oreilly.com/digitalmedia/2008/06/free-statistics-package-for-yo.html

Installing R

References/to learn more:

The R book
Michael J. Crawley pp 1
2007 John Wiley & Sons Ltd

Basic statistics using R pp. 8
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics with R
Vincent Zoonekynd, pp 3
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/introduccion-al-analisis-de-datos-y-al-lenguaje-s

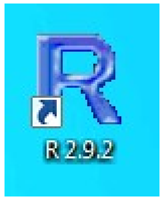
Geographic Data Analysis
Pat Bartlein
http://geography.uoregon.edu/bartlein/courses/geog417/exercises/ex1.htm

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/RGetToKnow.html

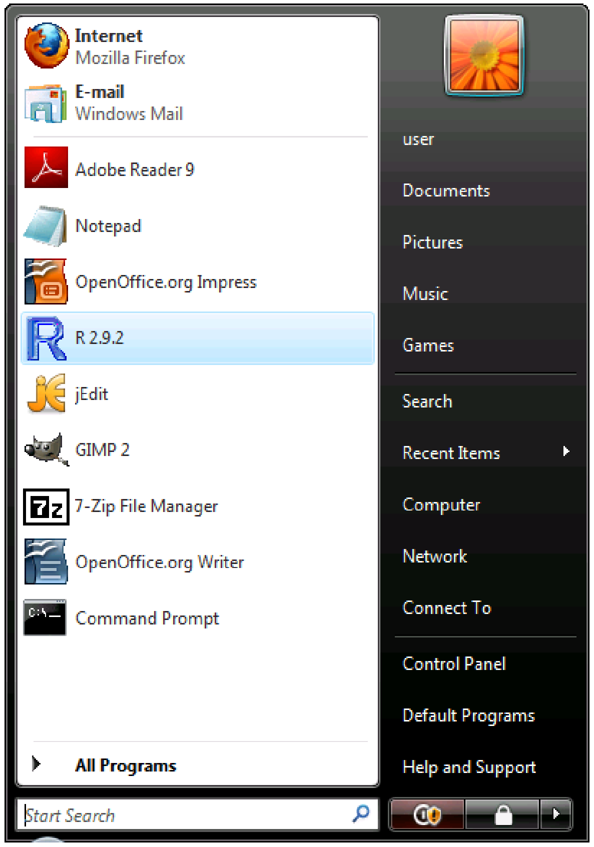
Chem 351 Archives Page
David Harvey
http://fs6.depauw.edu:50080/~harvey/Chem%20351/PDF%20Files/Handouts/RDocs/Obtaining%20and%20Installing%20R.pdf

Starting R

On Windows, if there is a shortcut on the desktop:

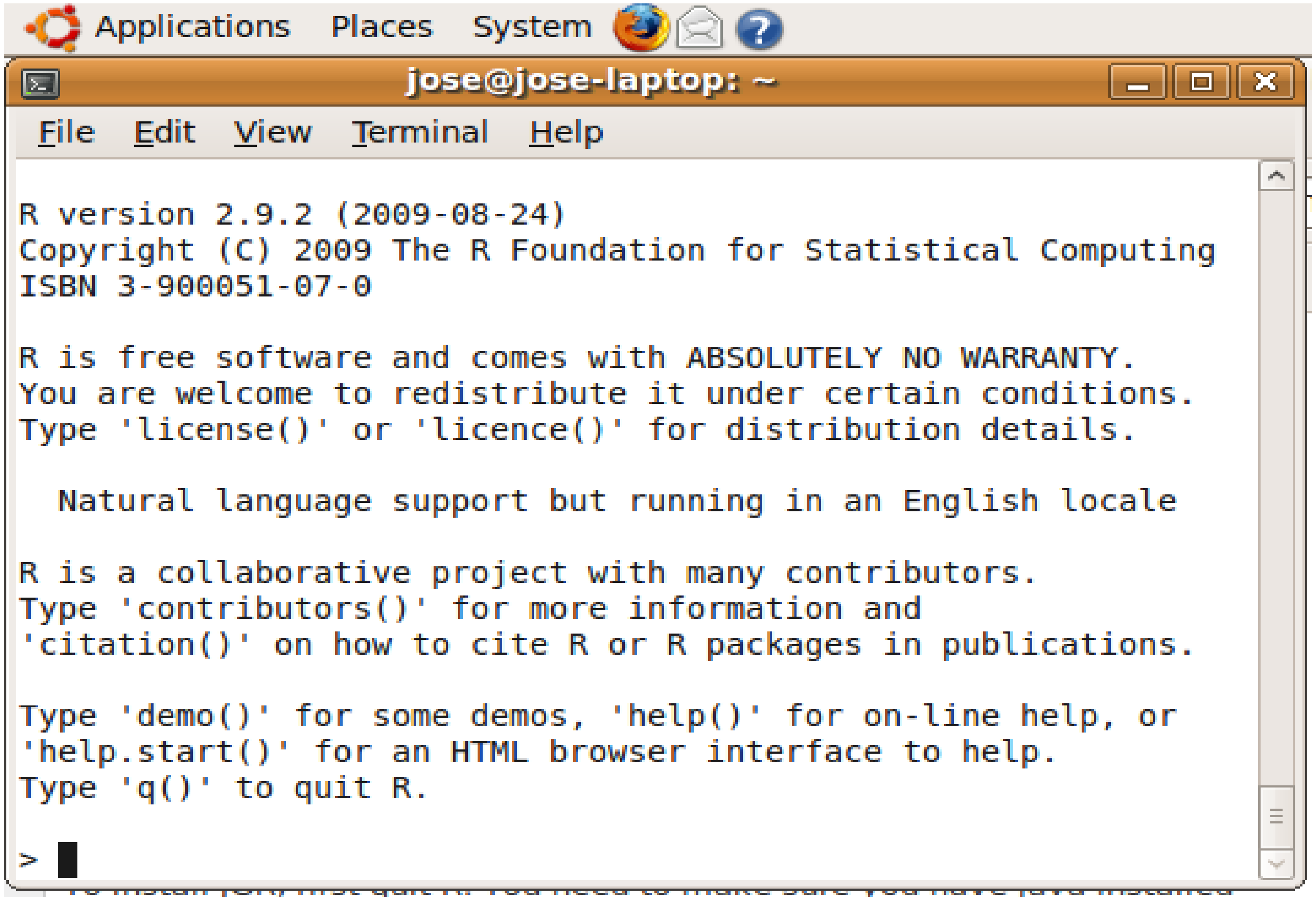


Or on the Start menu:



Starting R

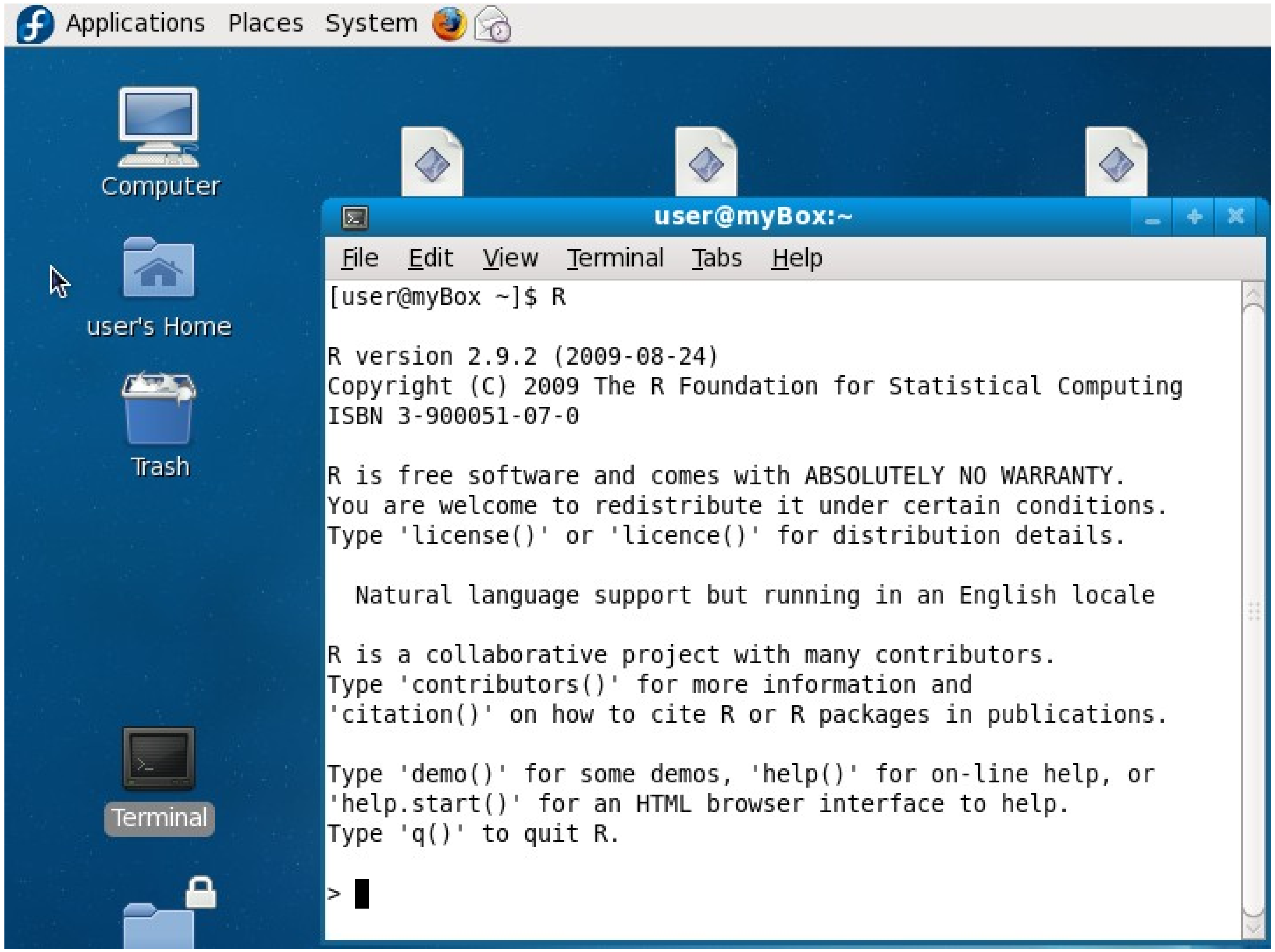
On Ubuntu, type R at the prompt



```
R version 2.9.2 (2009-08-24)
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.

> 
```

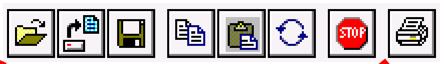
Starting R

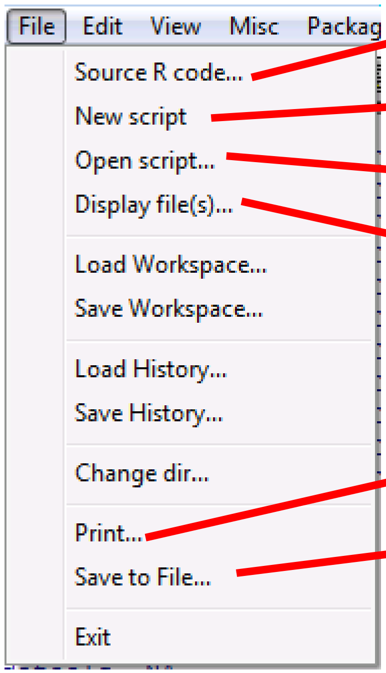
On Fedora, type R at the prompt



```
[user@myBox ~]$ R

R version 2.9.2 (2009-08-24)
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.

>
```

Rgui





Load R functions

Open the R editor

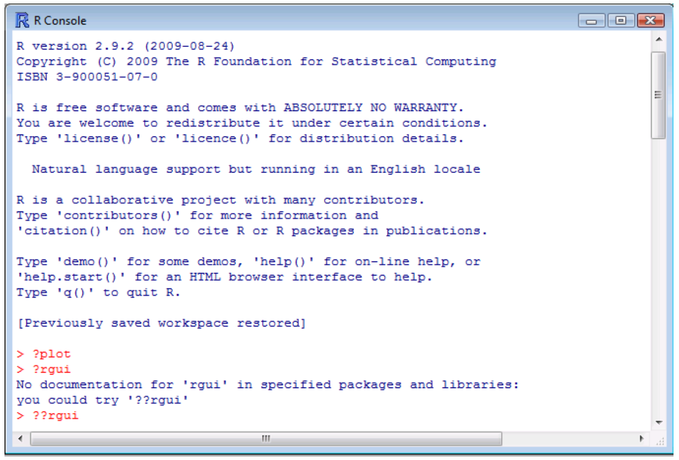
Open a file on the R editor

Display text file(s)

Print the History

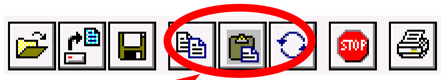
Save the History as text

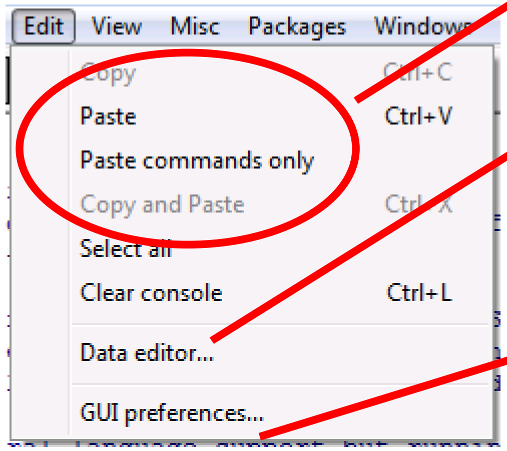
History is the text on the console:



```
R version 2.9.2 (2009-08-24)
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]

> ?plot
> ?rgui
No documentation for 'rgui' in specified packages and libraries:
you could try '??rgui'
> ??rgui
```

Rgui

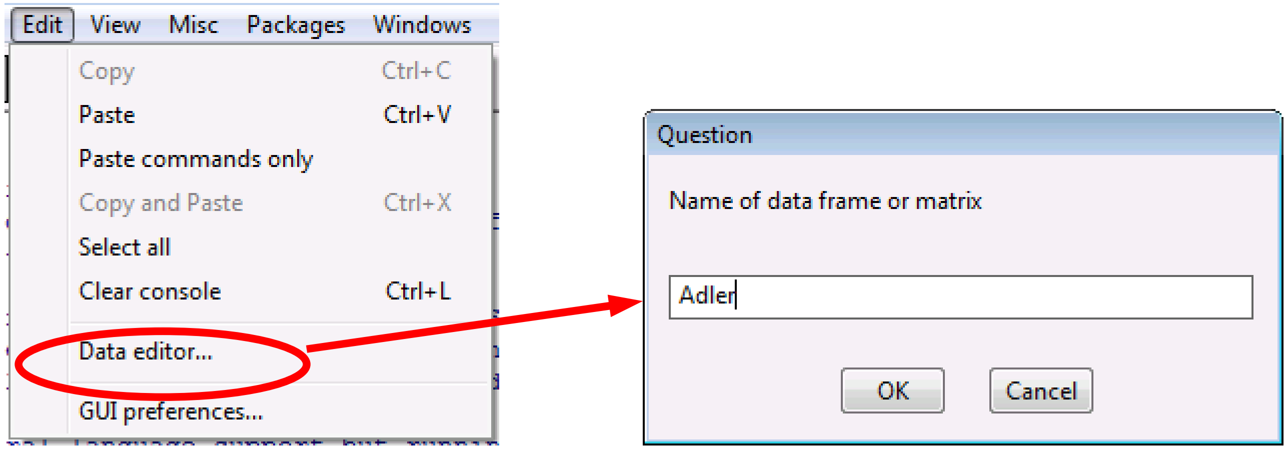


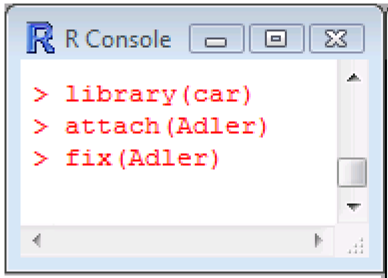


For editing a matrix or dataframe from the current session

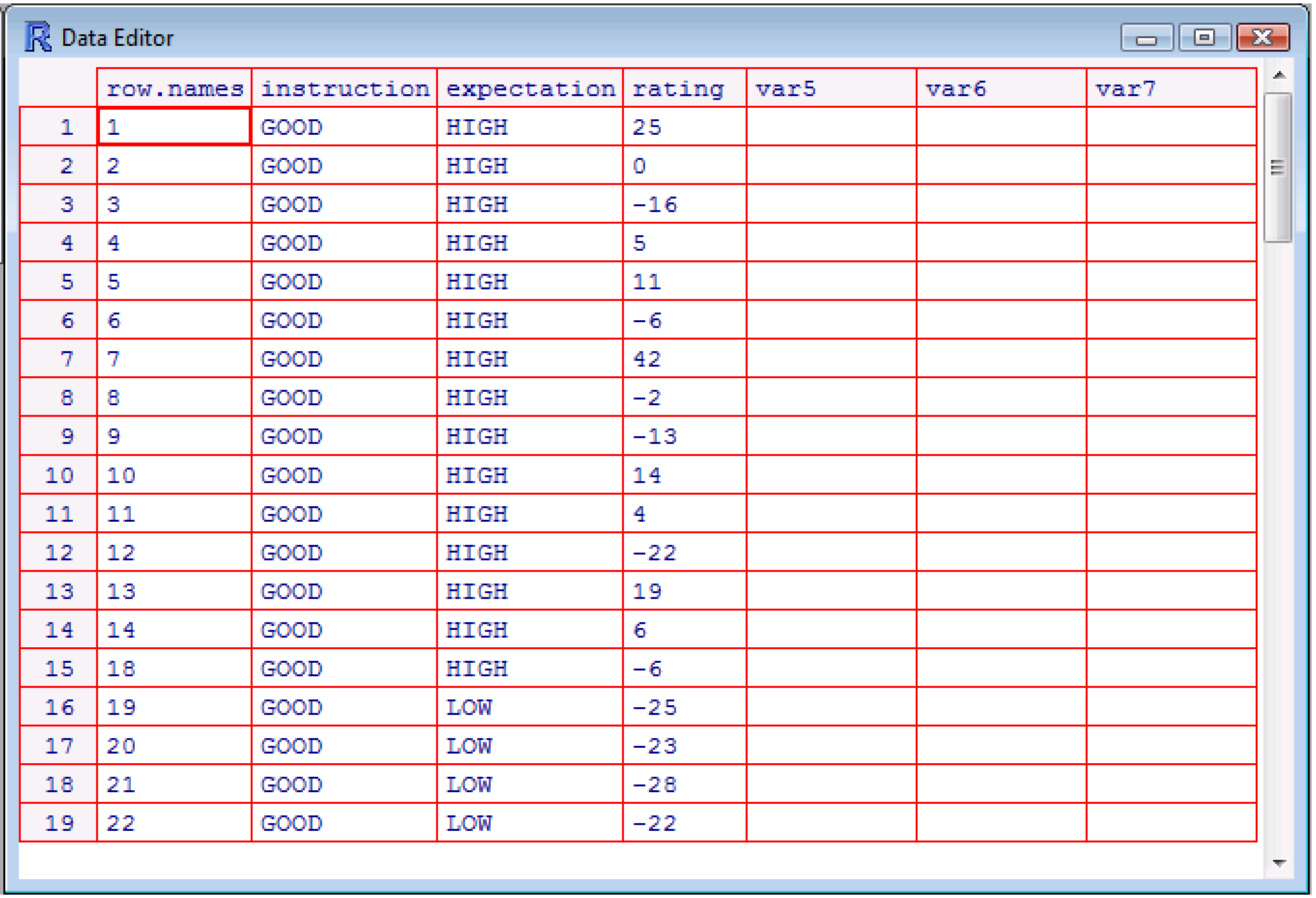
For customizing the GUI

Rgui





```
> library(car)
> attach(Adler)
> fix(Adler)
```



	row.names	instruction	expectation	rating	var5	var6	var7
1	1	GOOD	HIGH	25			
2	2	GOOD	HIGH	0			
3	3	GOOD	HIGH	-16			
4	4	GOOD	HIGH	5			
5	5	GOOD	HIGH	11			
6	6	GOOD	HIGH	-6			
7	7	GOOD	HIGH	42			
8	8	GOOD	HIGH	-2			
9	9	GOOD	HIGH	-13			
10	10	GOOD	HIGH	14			
11	11	GOOD	HIGH	4			
12	12	GOOD	HIGH	-22			
13	13	GOOD	HIGH	19			
14	14	GOOD	HIGH	6			
15	18	GOOD	HIGH	-6			
16	19	GOOD	LOW	-25			
17	20	GOOD	LOW	-23			
18	21	GOOD	LOW	-28			
19	22	GOOD	LOW	-22			

Data Editor

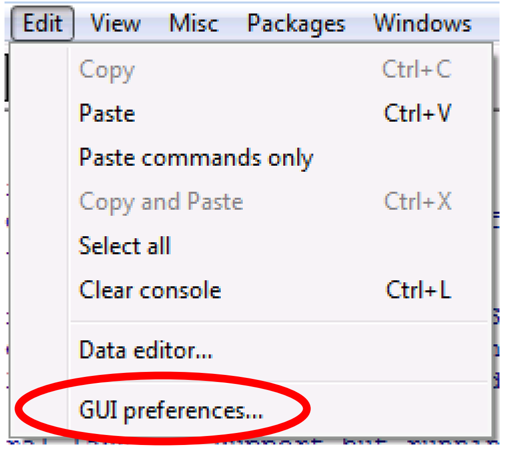
For editing a matrix or dataframe from the current session

Same as the fix function

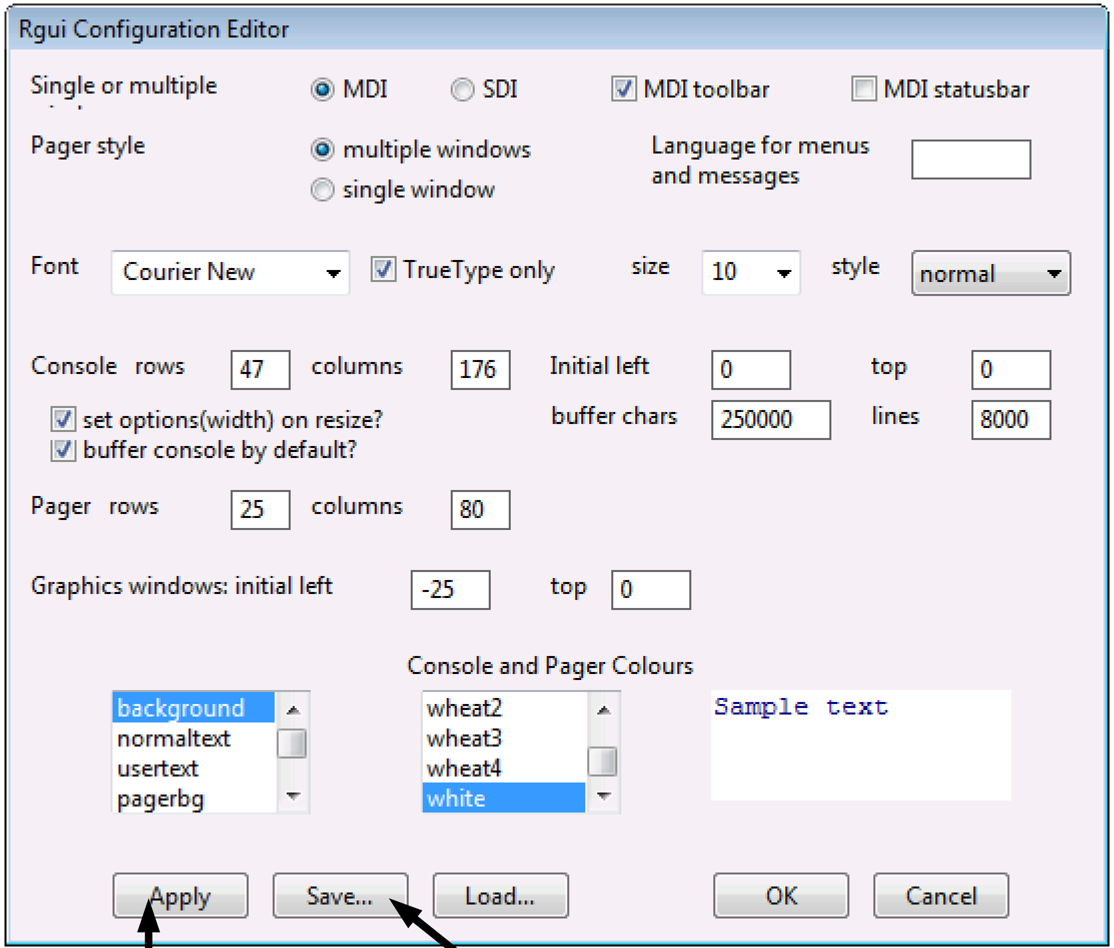
The cells are editable, like a spreadsheet

Rgui

Customizing the GUI


Edit View Misc Packages Windows

Copy Ctrl+C
Paste Ctrl+V
Paste commands only
Copy and Paste Ctrl+X
Select all
Clear console Ctrl+L
Data editor...
GUI preferences...


Rgui Configuration Editor

Single or multiple: [✓ MDI] SDI [✓ MDI toolbar] MDI statusbar

Pager style: [✓ multiple windows] single window

Language for menus and messages:

Font: Courier New [✓ TrueType only] size: 10 style: normal

Console rows: 47 columns: 176 Initial left: 0 top: 0

[✓ set options(width) on resize?]

[✓ buffer console by default?]

buffer chars: 250000 lines: 8000

Pager rows: 25 columns: 80

Graphics windows: initial left: -25 top: 0

Console and Pager Colours

background, normaltext, usertext, pagerbg — selected: background

wheat2, wheat3, wheat4, white — selected: white

```
Sample text
```

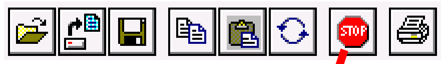
Some editors will only work with MDI

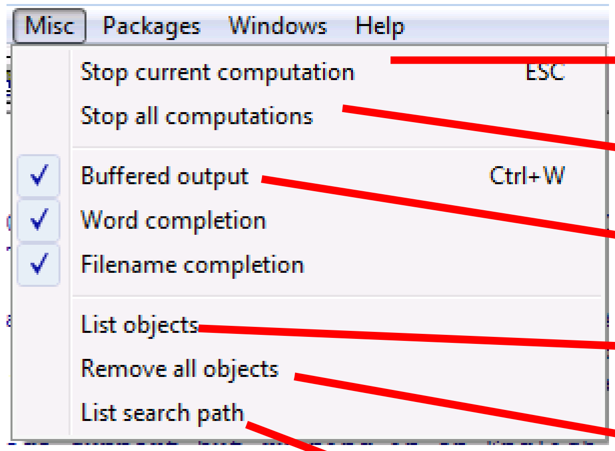
Buffer chars and Lines can be increased if it is necessary to work with a long History and there is an error because there is no space for it

This will apply the changes to the current session

To make changes permanent, that is, for every session,they must be saved. The default file Rconsole is loaded when a session starts

Rgui





Stop current computation ESC
Stop all computations
✓ Buffered output Ctrl+W
✓ Word completion
✓ Filename completion
List objects
Remove all objects
List search path

???

Stop computation on the current window

Stop computation on all the windows

Uncheck to get output immediately on the console*

Lists the names of the objects on the workspace

Deletes all the objects on the workspace

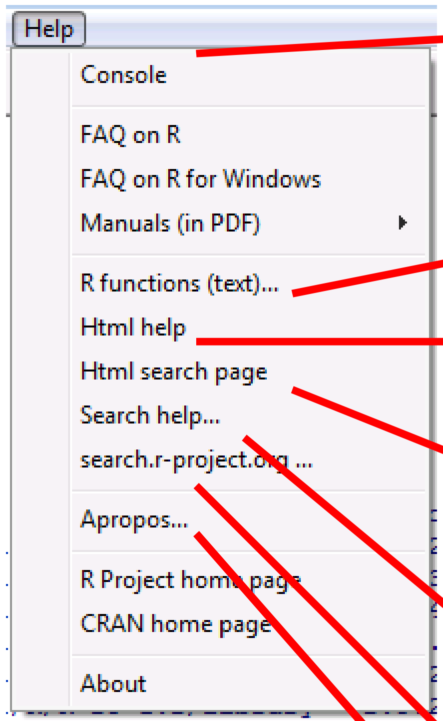
Lists attached packages and R objects

```
ls()
rm(list=ls(all=TRUE))
search()
```

* for example, to call some code and see its progress on the screen

R help

From the console:



Help about the console controls

Help about a known function name

help("plot")
?plot

R HTML manuals and references

help.start()

R HTML search engine for keywords, function and data names and text in help page titles

Search a reference from the manual

help.search("test")
??test

Use the online R Site Search

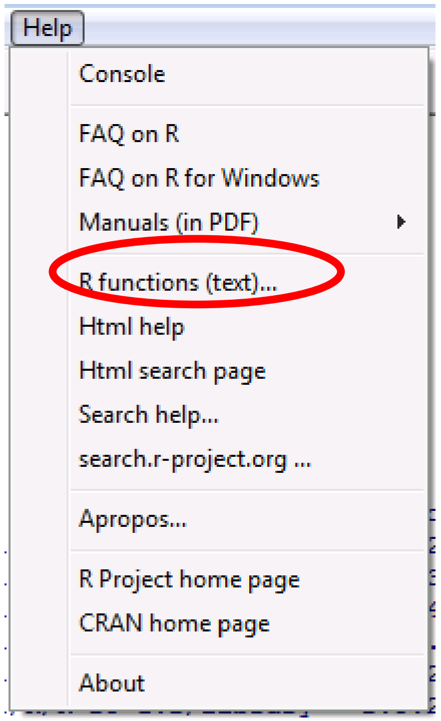
RSiteSearch("test")

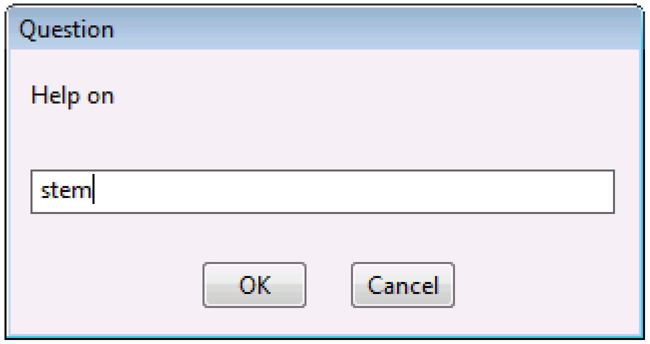
Look for a function name, partially known

apropos(“test”)

R help

Search for function “stem”





On the console:

help("stem")

Which could be called directly..

A help file will open on a new window.

R help

Function stem was found

Description of what the function does

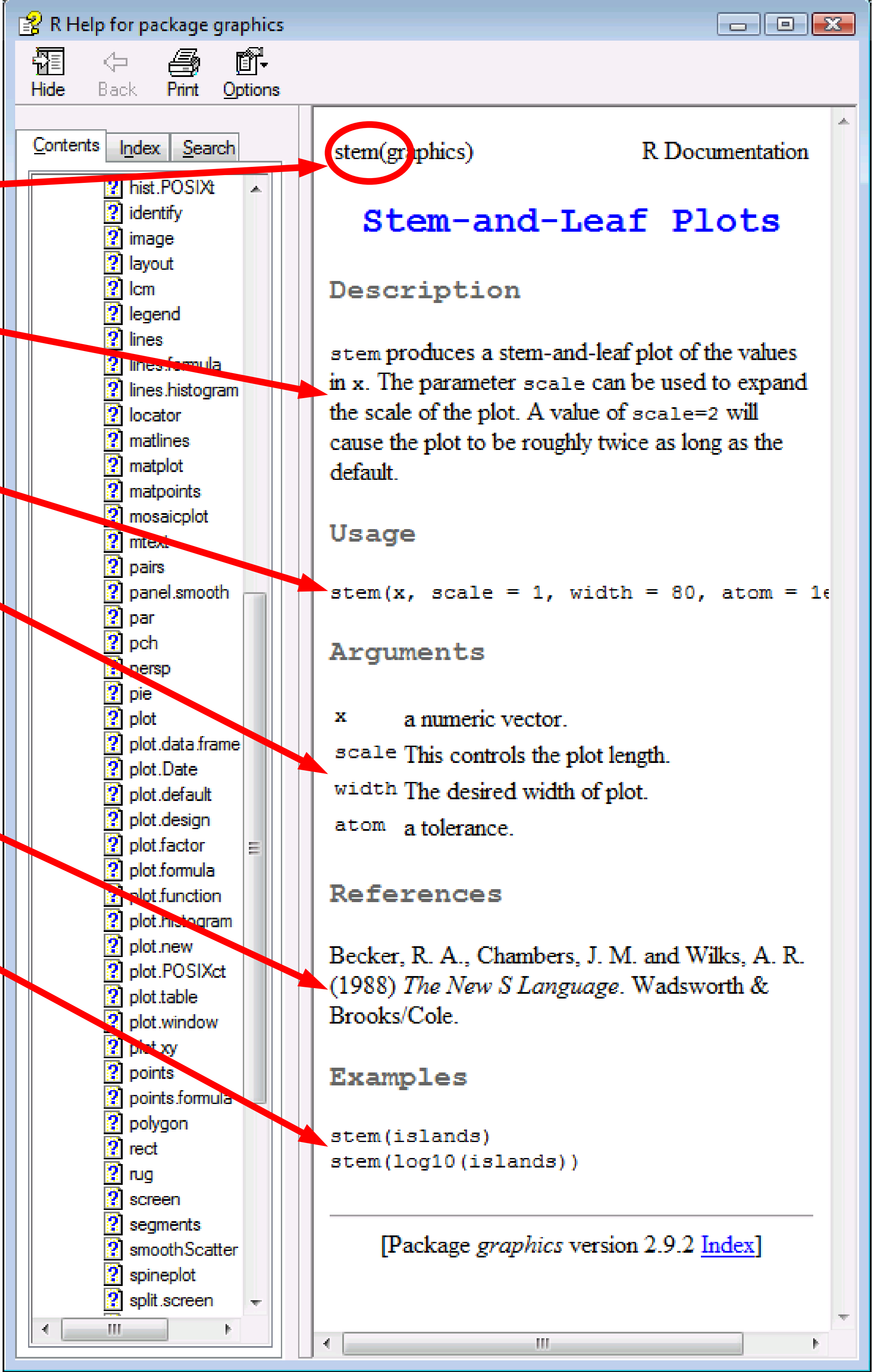
Function call with arguments

Description of arguments, sometimes with links to related objects

References to literature

How to use the function, the two examples provided will run automatically:

example("stem")



stem(graphics) R Documentation

Stem-and-Leaf Plots

Description

`stem` produces a stem-and-leaf plot of the values in `x`. The parameter `scale` can be used to expand the scale of the plot. A value of `scale=2` will cause the plot to be roughly twice as long as the default.

Usage

```
stem(x, scale = 1, width = 80, atom = 1e
```

Arguments

`x`	a numeric vector.
`scale`	This controls the plot length.
`width`	The desired width of plot.
`atom`	a tolerance.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

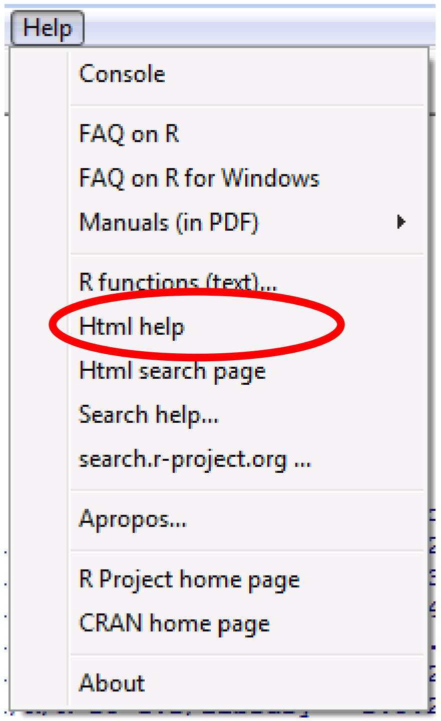
Examples

```
stem(islands)
stem(log10(islands))
```

[Package *graphics* version 2.9.2 Index]

R help

R HTML manuals and references



It doesn't show on the console but the equivalent command is:

help.start()

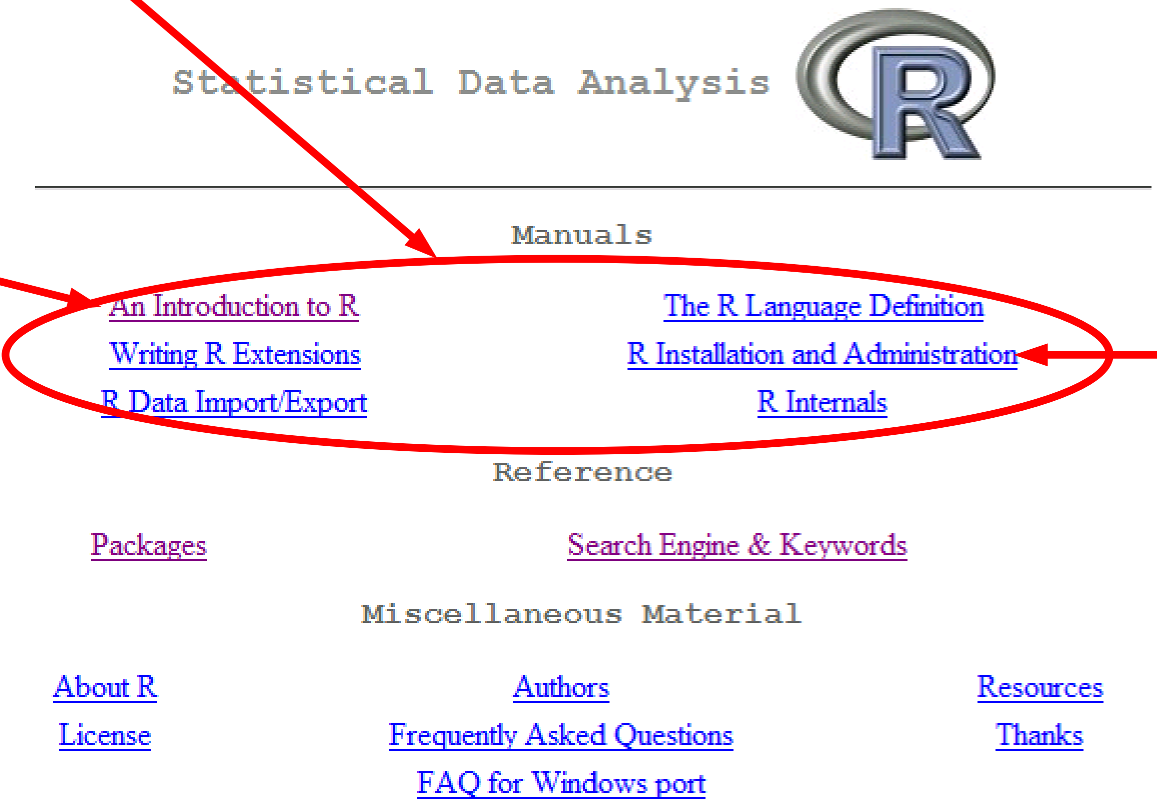
A new tab will open on the browser with the HTML help page.

R help

Manuals in HTML format

Very complete introduction to R

Installing and customizing R



Statistical Data Analysis

Manuals

An Introduction to R | The R Language Definition
Writing R Extensions | R Installation and Administration
R Data Import/Export | R Internals

Reference

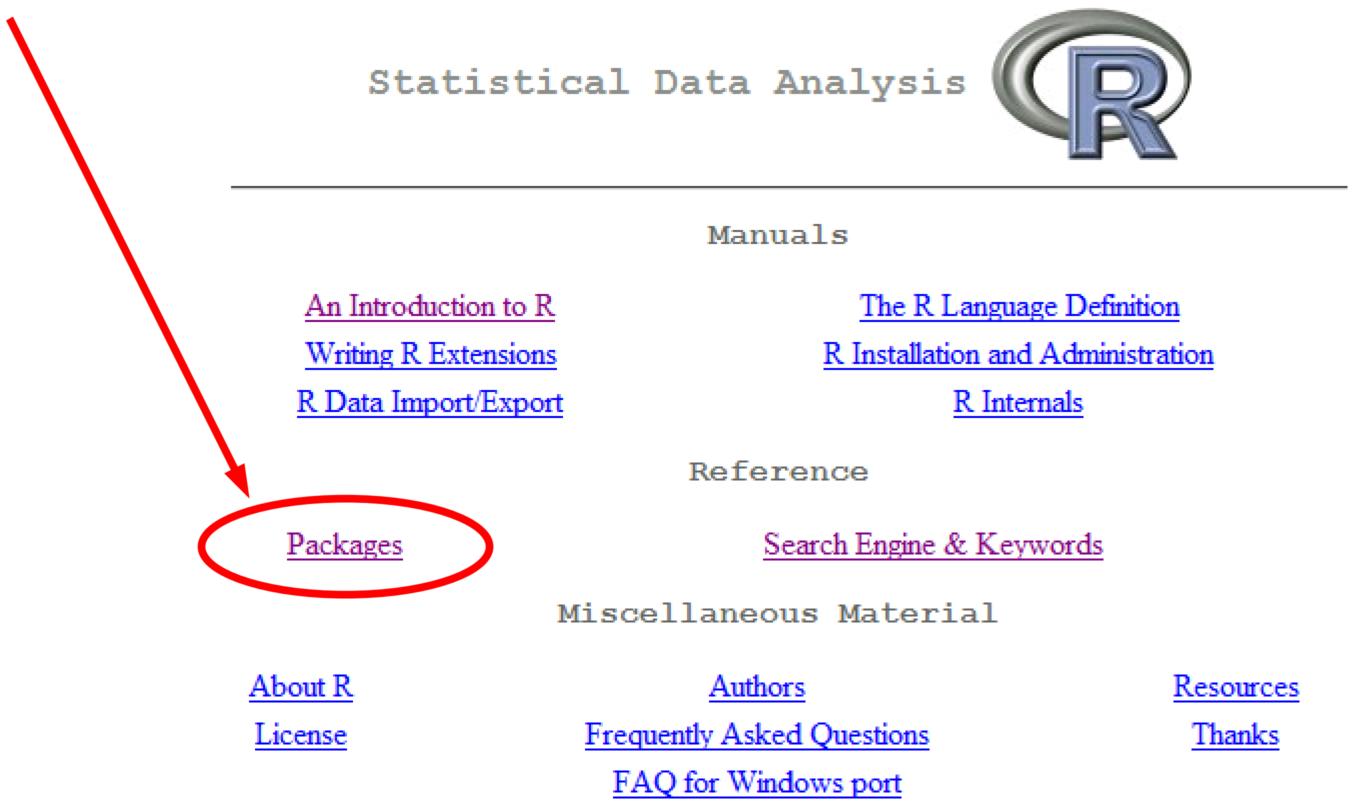
Packages | Search Engine & Keywords

Miscellaneous Material

About R | Authors | Resources
License | Frequently Asked Questions | Thanks
FAQ for Windows port

R help

List of installed packages:



Statistical Data Analysis

Manuals

An Introduction to R
The R Language Definition
Writing R Extensions
R Installation and Administration
R Data Import/Export
R Internals

Reference

Packages
Search Engine & Keywords

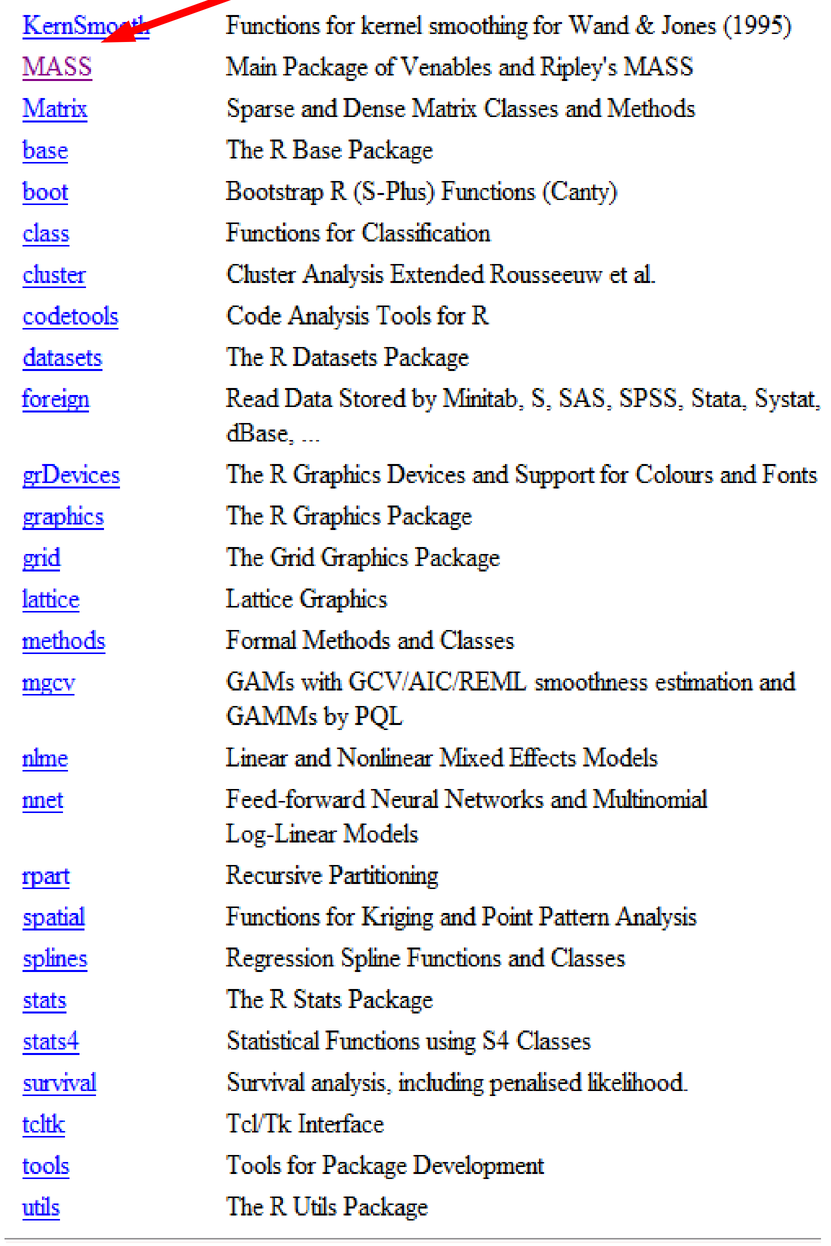
Miscellaneous Material

About R
Authors
Resources
License
Frequently Asked Questions
Thanks
FAQ for Windows port

R help

Click on MASS

List of installed packages:



KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)
MASS	Main Package of Venables and Ripley's MASS
Matrix	Sparse and Dense Matrix Classes and Methods
base	The R Base Package
boot	Bootstrap R (S-Plus) Functions (Canty)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
datasets	The R Datasets Package
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
grDevices	The R Graphics Devices and Support for Colours and Fonts
graphics	The R Graphics Package
grid	The Grid Graphics Package
lattice	Lattice Graphics
methods	Formal Methods and Classes
mgcv	GAMs with GCV/AIC/REML smoothness estimation and GAMMs by PQL
nlme	Linear and Nonlinear Mixed Effects Models
nnet	Feed-forward Neural Networks and Multinomial Log-Linear Models
rpart	Recursive Partitioning
spatial	Functions for Kriging and Point Pattern Analysis
splines	Regression Spline Functions and Classes
stats	The R Stats Package
stats4	Statistical Functions using S4 Classes
survival	Survival analysis, including penalised likelihood.
tcltk	Tcl/Tk Interface
tools	Tools for Package Development
utils	The R Utils Package

Alternatively:

```
> installed.packages()
           Package
base       "base"
boot       "boot"
class      "class"
cluster    "cluster"
codetools  "codetools"
datasets   "datasets"
foreign    "foreign"
graphics   "graphics"
grDevices  "grDevices"
grid       "grid"
KernSmooth "KernSmooth"
lattice    "lattice"
MASS       "MASS"
Matrix     "Matrix"
methods    "methods"
mgcv       "mgcv"
nlme       "nlme"
nnet       "nnet"
rpart      "rpart"
spatial    "spatial"
splines    "splines"
stats      "stats"
stats4     "stats4"
survival   "survival"
tcltk      "tcltk"
tools      "tools"
utils      "utils"
           Imports
```

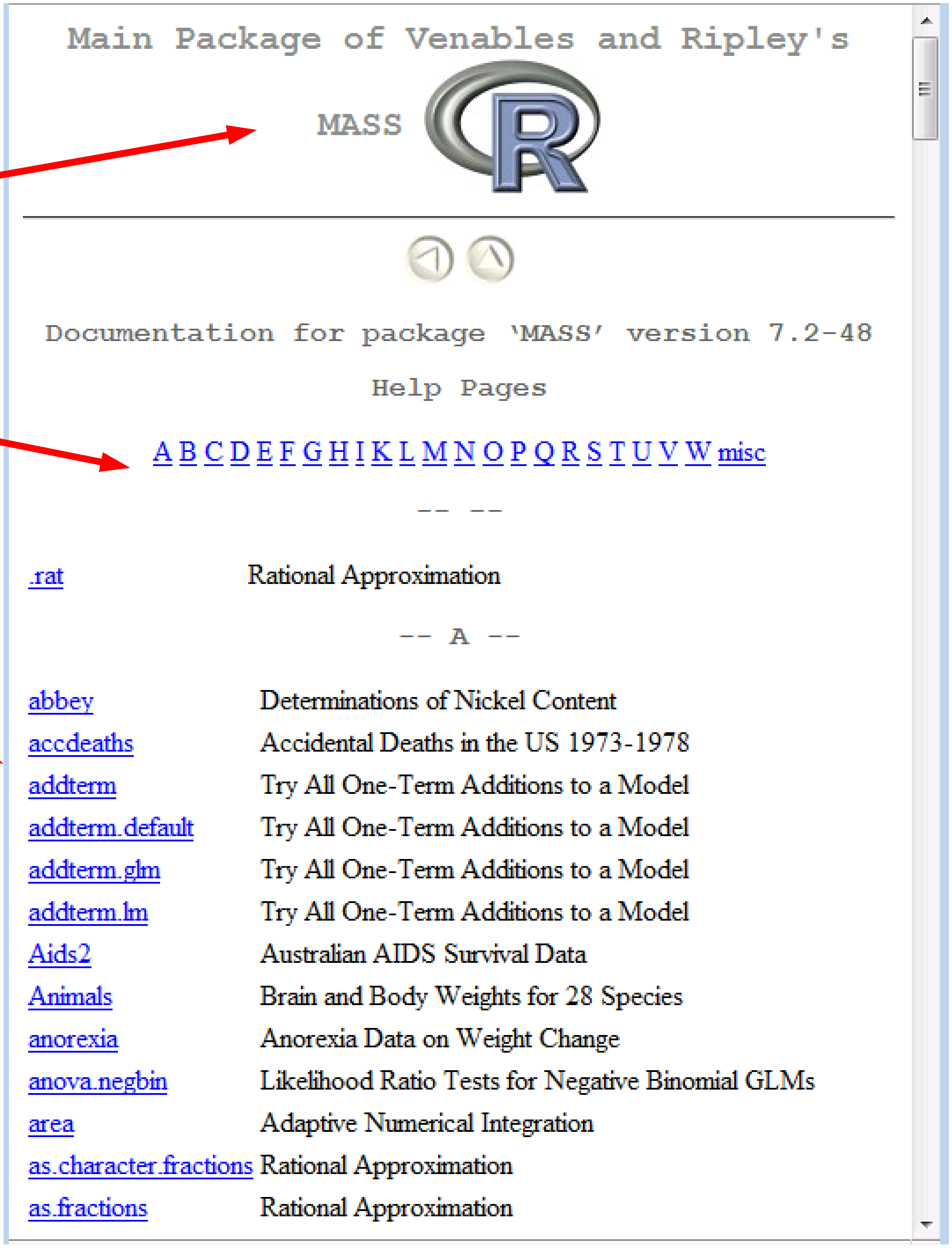
This is a faster way to list the packages but without links to help

R help

Package MASS

Links to function names by their first character

Function names and a simple description



Main Package of Venables and Ripley's

MASS

Documentation for package 'MASS' version 7.2-48

Help Pages

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

-- --

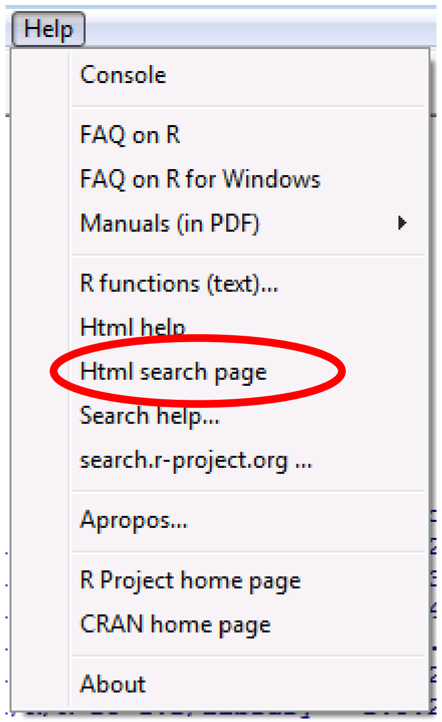
.rat	Rational Approximation

-- A --

abbey	Determinations of Nickel Content
accdeaths	Accidental Deaths in the US 1973-1978
addterm	Try All One-Term Additions to a Model
addterm.default	Try All One-Term Additions to a Model
addterm.glm	Try All One-Term Additions to a Model
addterm.lm	Try All One-Term Additions to a Model
Aids2	Australian AIDS Survival Data
Animals	Brain and Body Weights for 28 Species
anorexia	Anorexia Data on Weight Change
anova.negbin	Likelihood Ratio Tests for Negative Binomial GLMs
area	Adaptive Numerical Integration
as.character.fractions	Rational Approximation
as.fractions	Rational Approximation

R help

R HTML search engine



It doesn't show on the console but the equivalent command would be:

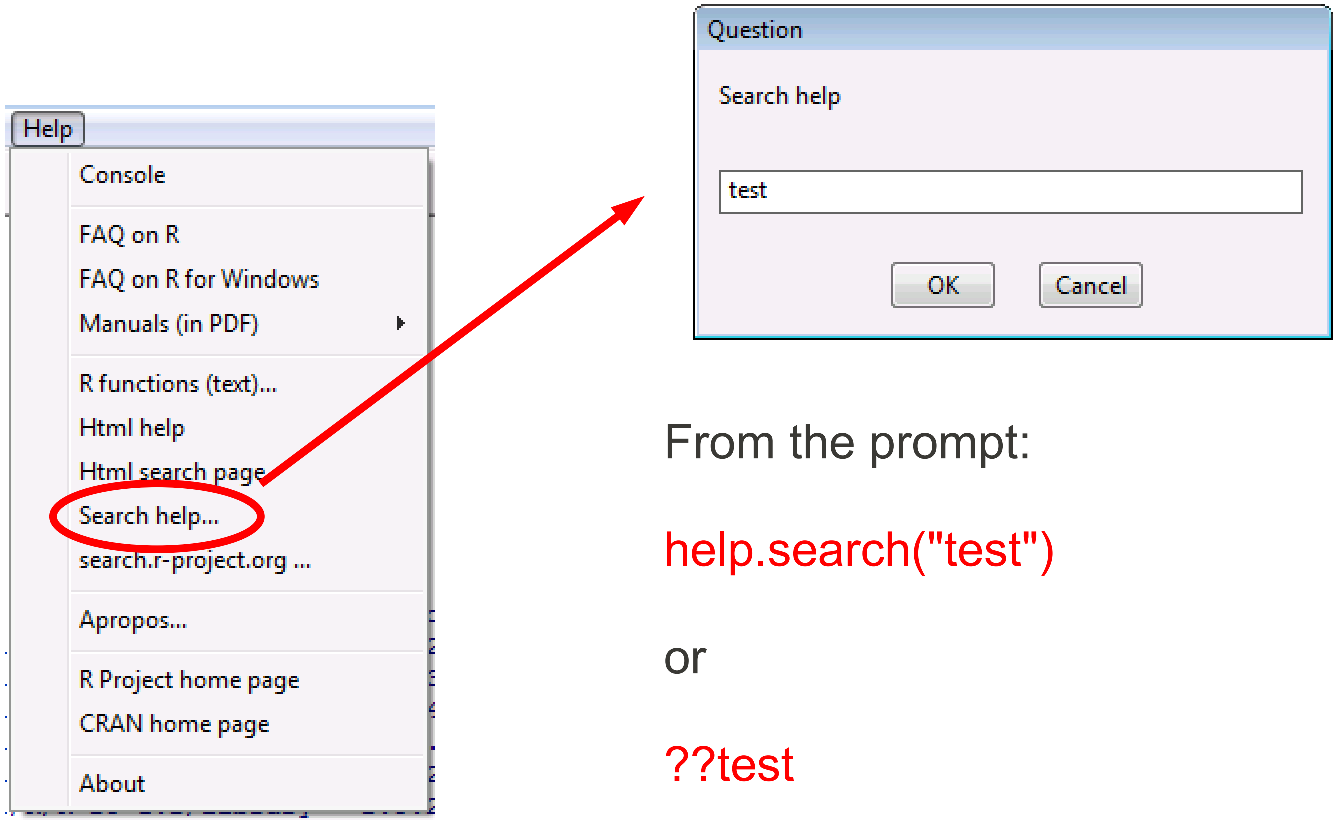
help.start()

Followed by clicking the link

Search Engine & Keywords

R help

Search for a reference from the manual on the keyword “test”



From the prompt:

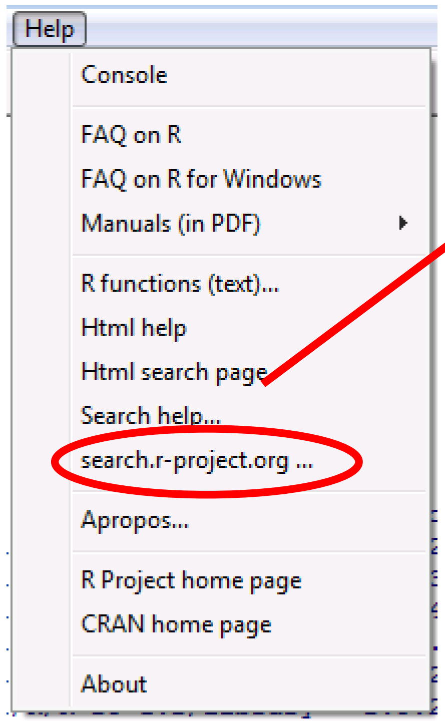
help.search("test")

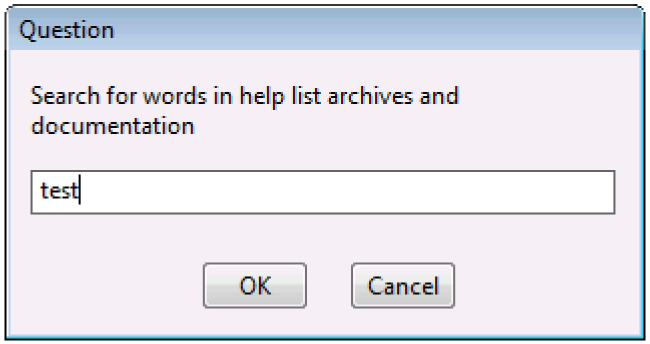
or

??test

R help

Use the online R Site Search





Search for words in help list archives and documentation

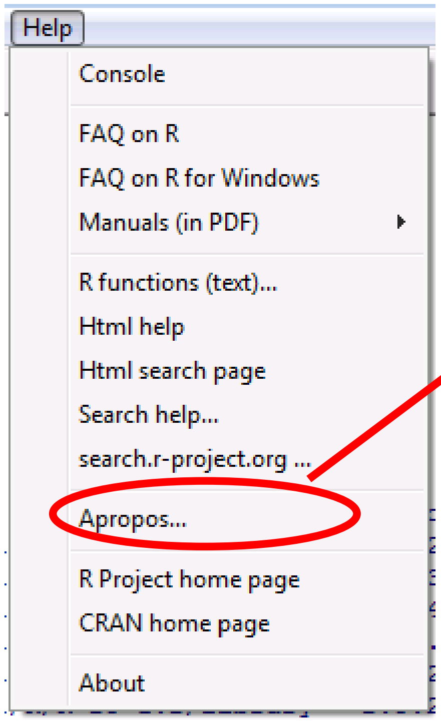
test

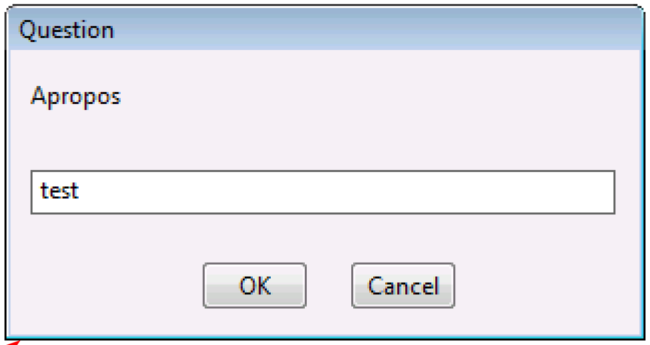
From the prompt:

RsiteSearch("test")

R help

Look for a function name,
partially known





From the prompt:

apropos(“test”)

Try this:

```
apropos("test")
apropos(".test")
apropos("[\\.]test")
apropos("[^\\.]test")
apropos("^test")
apropos("([^\\.]test)|(^test)")
```

R help

apropos("test")
apropos(".test")
apropos("[\\.]test")
apropos("[^\\.]test")
apropos("^test")
apropos("([^\\.]test)|(^test)")

1. “test” is anywhere within the function name
2. find “test” preceded by any character
3. find “.test”
4. find “test” preceded by any character, other than “.”
5. find “test”, only if at the end of the name
6. both 4. and 5.

Remember:
Apropos uses regular expressions for searches

```
> apropos("test")
 [1] ".valueClassTest"         "ansari.test"             "bartlett.test"           "binom.test"              "Box.test"                "chisq.test"
 [7] "cor.test"                "file_test"               "fisher.test"             "fligner.test"            "friedman.test"           "kruskal.test"
[13] "ks.test"                 "mantelhaen.test"         "mauchley.test"           "mauchly.test"            "mcnemar.test"            "mood.test"
[19] "oneway.test"             "pairwise.prop.test"      "pairwise.t.test"         "pairwise.wilcox.test"    "poisson.test"            "power.anova.test"
[25] "power.prop.test"         "power.t.test"            "PP.test"                 "prop.test"               "prop.trend.test"         "quade.test"
[31] "shapiro.test"            "t.test"                  "testInheritedMethods"    "testPlatformEquivalence" "testVirtual"             "var.test"
[37] "wilcox.test"
> apropos(".test")
 [1] ".valueClassTest"      "ansari.test"          "bartlett.test"        "binom.test"           "Box.test"             "chisq.test"           "cor.test"
 [8] "file_test"            "fisher.test"          "fligner.test"         "friedman.test"        "kruskal.test"         "ks.test"              "mantelhaen.test"
[15] "mauchley.test"        "mauchly.test"         "mcnemar.test"         "mood.test"            "oneway.test"          "pairwise.prop.test"   "pairwise.t.test"
[22] "pairwise.wilcox.test" "poisson.test"         "power.anova.test"     "power.prop.test"      "power.t.test"         "PP.test"              "prop.test"
[29] "prop.trend.test"      "quade.test"           "shapiro.test"         "t.test"               "var.test"             "wilcox.test"
> apropos("[\\.]test")
 [1] "ansari.test"          "bartlett.test"        "binom.test"           "Box.test"             "chisq.test"           "cor.test"             "fisher.test"
 [8] "fligner.test"         "friedman.test"        "kruskal.test"         "ks.test"              "mantelhaen.test"      "mauchley.test"        "mauchly.test"
[15] "mcnemar.test"         "mood.test"            "oneway.test"          "pairwise.prop.test"   "pairwise.t.test"      "pairwise.wilcox.test" "poisson.test"
[22] "power.anova.test"     "power.prop.test"      "power.t.test"         "PP.test"              "prop.test"            "prop.trend.test"      "quade.test"
[29] "shapiro.test"         "t.test"               "var.test"             "wilcox.test"
> apropos("[^\\.]test")
[1] ".valueClassTest" "file_test"
> apropos("^test")
[1] "testInheritedMethods"    "testPlatformEquivalence" "testVirtual"
> apropos("([^\\.]test)|(^test)")
[1] ".valueClassTest"         "file_test"               "testInheritedMethods"    "testPlatformEquivalence" "testVirtual"
```

R help

How to use help

To show the documentation

```
help() or ?help
```

To find the documentation about the function "plot"

```
?plot
help("plot")
```

To find the documentation about reserved words or non-alphanumeric commands

```
?"for"
?"[["
?"[<-.data.frame"
```

Use double quotes

To find all the installed help files (packages) that have an alias, concept or title named "plot"

```
??plot
help.search("plot")
```

Package "graphics" has a function "plot", let's examine it:

```
?graphics::plot
```

Package "lattice" has a function "xyplot", let's examine it:

```
?lattice::xyplot
```

To get a short description of a package:

```
library(help = graphics)
```

R help

How to use help

When not sure about the function name (on the search path), but it contains "plot"
apropos("plot")

To search R the web site and the R-help mailing list (http://search.r-project.org)
RSiteSearch("plot")

To run the examples from a help topic
example(topic)

To find where there are some demos for the loaded packages
demo()

To find where there are some demos for all the packages
demo(package = .packages(all.available = TRUE))

To show the demo "graphics" from package "graphics", pausing between pages
demo(graphics, package="graphics", ask=TRUE)

To show the demo "graphics" from package "graphics", whithout pausing between pages
demo(graphics, package="graphics", ask=FALSE)

R help

Other sources of help

R Project search engine
http://www.r-seek.org

mailing lists which are used by R users and developers. See
http://www.R-project.org/mail.html

Bug-tracking system
R has a bug-tracking system (or perhaps a bug-filing system is a more precise description) available on the net at
http://bugs.R-project.org/

The R Journal
http://journal.r-project.org/

Journal of Statistics Education
http://www.amstat.org/PUBLICATIONS/JSE/

Technology Innovations in Statistics Education
http://repositories.cdlib.org/uclastat/cts/tise/

Journal of Statistical Software
http://www.jstatsoft.org

R help

Exercise

How to get random numbers in R?

Use only the help tools discussed today

R help

```
?random # no results...

??random

base::RNG              Random Number Generation
base::sample           Random Samples and Permutations
datasets::randu        Random Numbers from Congruential Generator RANDU

?base::RNG # Random Number Generation
?base::sample # Random Samples and Permutations
?datasets::randu # Random Numbers from Congruential Generator RANDU ("widely
considered to be one of the most ill-conceived random number generators designed",
Wikipedia)
```

The command-line editor

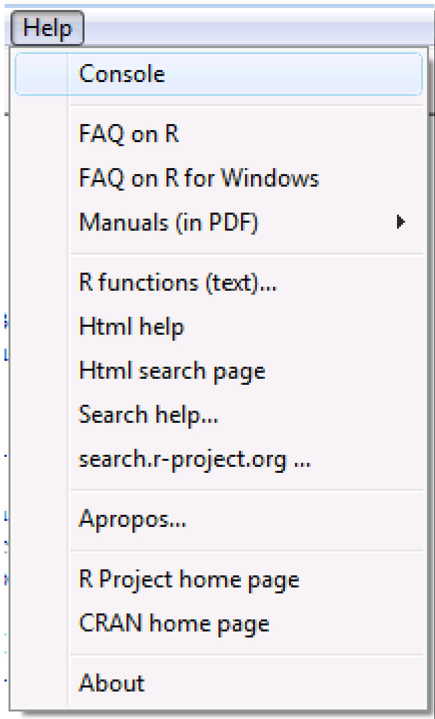
Recall and correction of previous commands

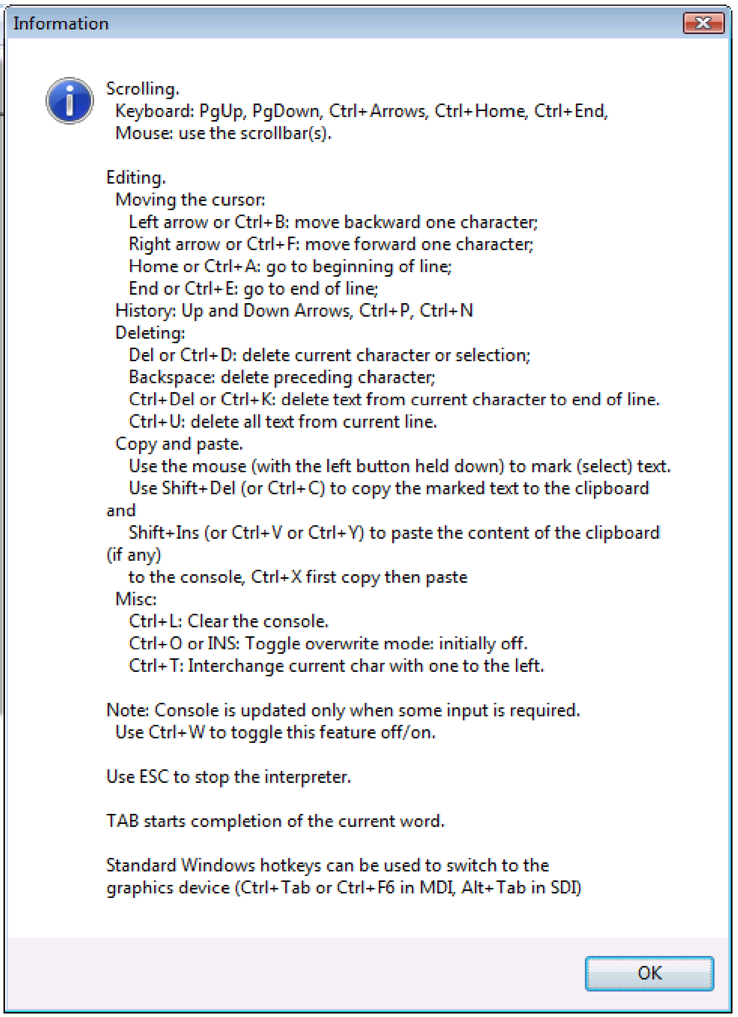
R keeps a command history, a list of the commands executed at the prompt.

Enter will execute the current line of text, at the prompt.

Cursor keys:
Arrow up - show previous command
Arrow down - show next command
Arrows left and right - move around the current line of text, at the prompt.

Editor comands:




Information

Scrolling.
 Keyboard: PgUp, PgDown, Ctrl+Arrows, Ctrl+Home, Ctrl+End,
 Mouse: use the scrollbar(s).

Editing.
 Moving the cursor:
  Left arrow or Ctrl+B: move backward one character;
  Right arrow or Ctrl+F: move forward one character;
  Home or Ctrl+A: go to beginning of line;
  End or Ctrl+E: go to end of line;
 History: Up and Down Arrows, Ctrl+P, Ctrl+N
 Deleting:
  Del or Ctrl+D: delete current character or selection;
  Backspace: delete preceding character;
  Ctrl+Del or Ctrl+K: delete text from current character to end of line.
  Ctrl+U: delete all text from current line.
 Copy and paste.
  Use the mouse (with the left button held down) to mark (select) text.
  Use Shift+Del (or Ctrl+C) to copy the marked text to the clipboard
and
  Shift+Ins (or Ctrl+V or Ctrl+Y) to paste the content of the clipboard
(if any)
  to the console, Ctrl+X first copy then paste
 Misc:
  Ctrl+L: Clear the console.
  Ctrl+O or INS: Toggle overwrite mode: initially off.
  Ctrl+T: Interchange current char with one to the left.

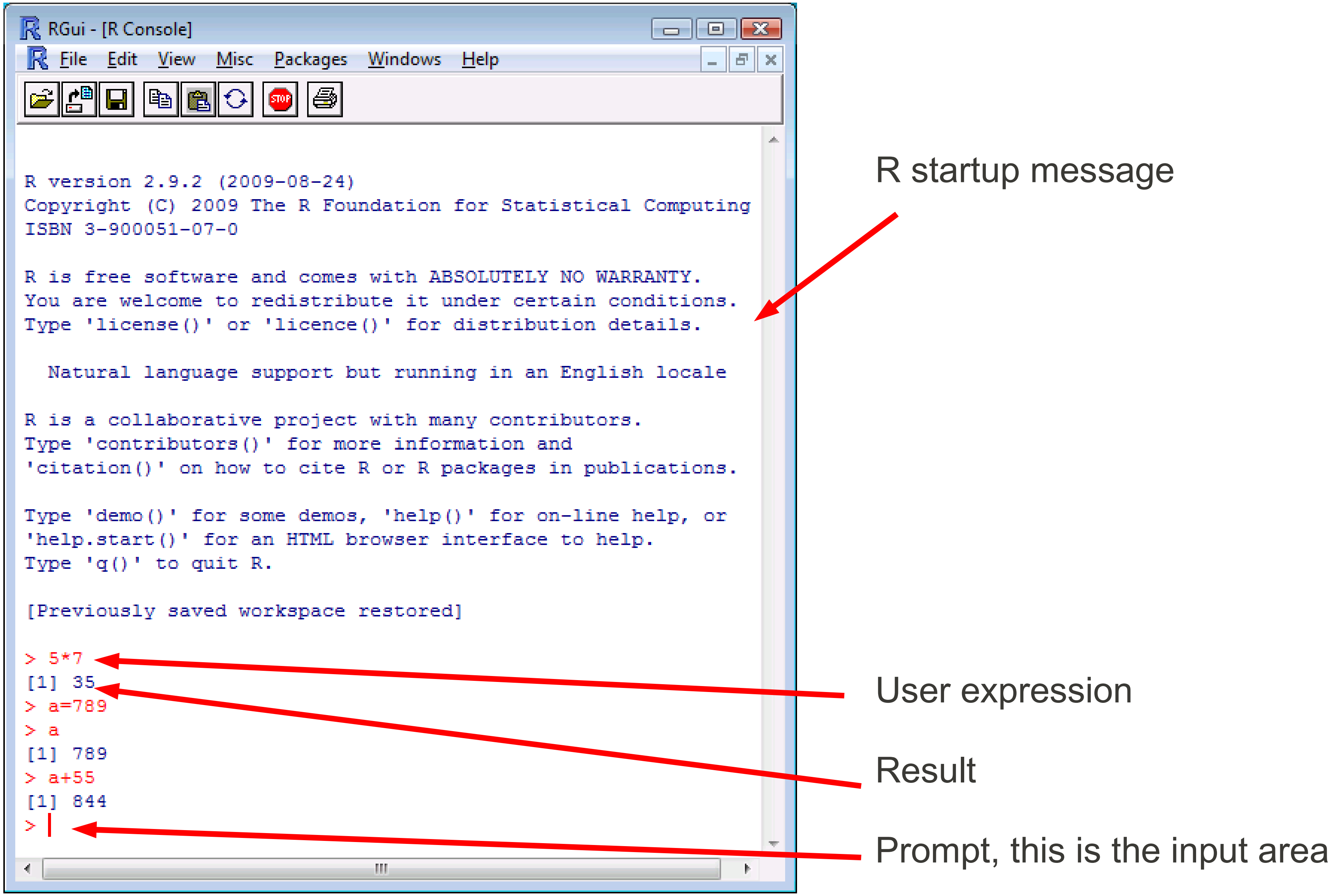
Note: Console is updated only when some input is required.
 Use Ctrl+W to toggle this feature off/on.

Use ESC to stop the interpreter.

TAB starts completion of the current word.

Standard Windows hotkeys can be used to switch to the
graphics device (Ctrl+Tab or Ctrl+F6 in MDI, Alt+Tab in SDI)

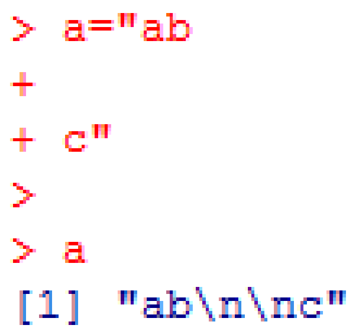
The command-line editor


RGui - [R Console]
File Edit View Misc Packages Windows Help
R startup message
User expression
Result
Prompt, this is the input area

```
R version 2.9.2 (2009-08-24)
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]

> 5*7
[1] 35
> a=789
> a
[1] 789
> a+55
[1] 844
> |
```

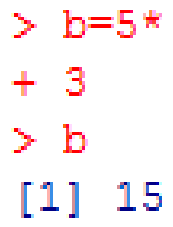
The command-line editor

Incomplete expressions will result on an annoying + that will disappear once the expression is completed.



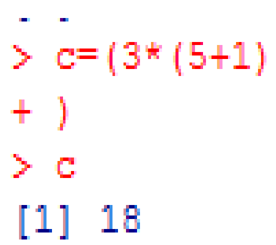
```
> a="ab
+
+ c"
>
> a
[1] "ab\n\nc"
```

A string must be within enclosing double quotes but, pressing enter, will cause a newline character to be part of the string.



```
> b=5*
+ 3
> b
[1] 15
```

An expression is incomplete if it ends with an operator. There are no side effects, once the expression is completed.



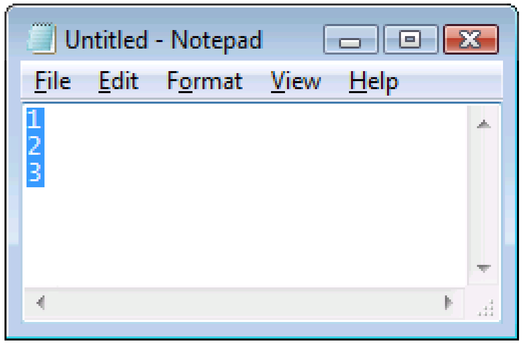
```
> c=(3*(5+1)
+ )
> c
[1] 18
```

An expression with parenthesis will not work, until all the parenthesis are paired. There are no side effects, once the expression is completed.

The command-line editor

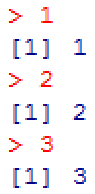
The console will accept multiple commands, if pasted, and execute one line at a time.

For example, copying from Notepad:



```
1
2
3
```

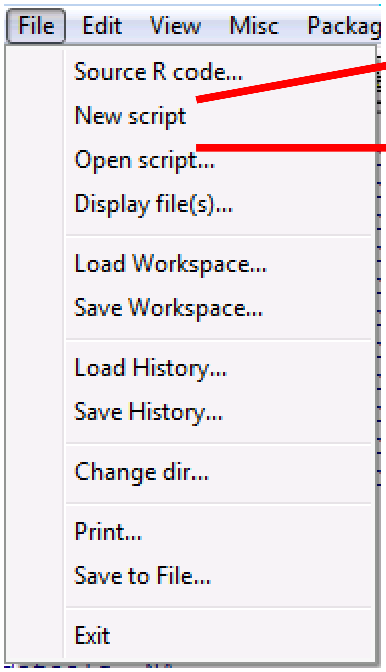
And pasting on R:



```
> 1
[1] 1
> 2
[1] 2
> 3
[1] 3
```

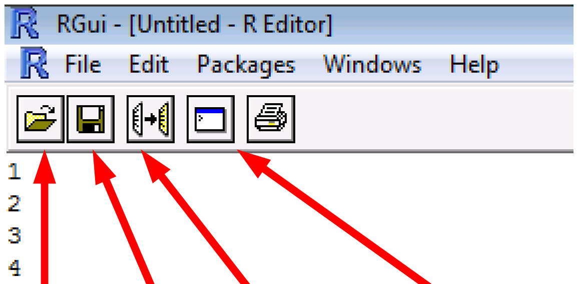
This is unnecessary because R has its own text editor, the R Editor

The R Editor



Open the R editor

Open a file on the R editor



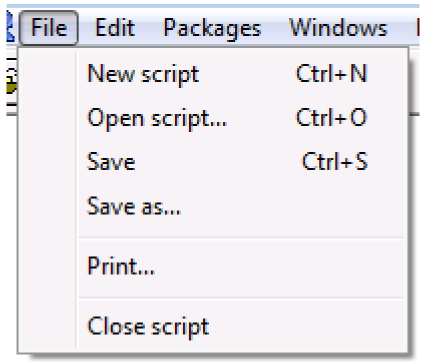
Open R file

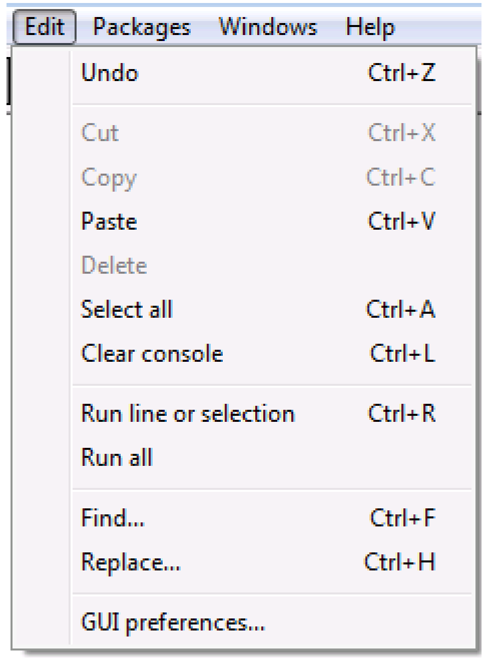
Save R file

Run current line or selected code

Run all the code

Menu changes:





The R Editor

The R Editor has all the capabilities of a basic text editor, just like notepad or pico.

The R Editor can be an alternative to the console because it can execute code, one line at a time, a selection of lines or even a selected portion of code within a larger expression. The code can be saved and loaded as a text file with the extension .R.

On Rgui on the menu go to File/New script
Type this:

```
myvec <- seq(1,by=3, length.out=9)
mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

Edit/Run all

The R Editor

Position the cursor on any line and press ctrl-r, the line of code will execute on the R console and the cursor will mode down to the next line. It is possible to follow the execution of code by pressing ctrl-r continuously.

```
myvec <- seq(1,by=3, length.out=9)
|mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

The cursor is on this line, ctrl-r will execute it

Position the cursor at the beginning of any line and use shift+cursor keys or keep the left-click button on the mouse pressed and move the cursor, to select a few lines of code and press ctrl-r, the line of code will execute on the R console.

```
myvec <- seq(1,by=3, length.out=9)
mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

The R Editor

Position the cursor at the beginning of an expression and use shift+cursor keys or keep the left-click button on the mouse pressed and move the cursor, to select a valid expression and press ctrl-r, the expression will execute on the R console.

```
myvec <- seq(1,by=3, length.out=9)
mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

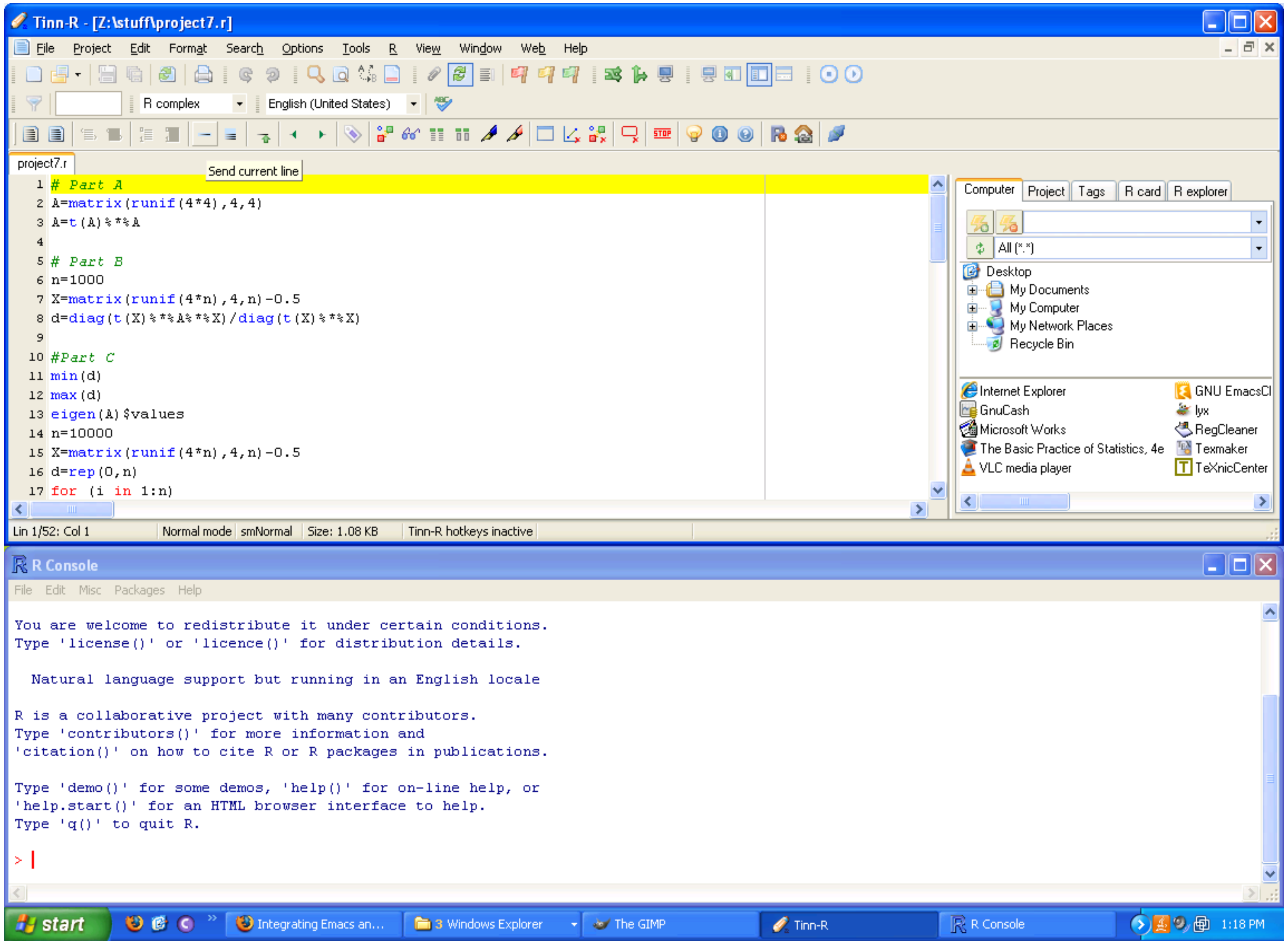
```
myvec <- seq(1,by=3, length.out=9)
mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

```
myvec <- seq(1,by=3, length.out=9)
mymatrix1 <- matrix(myvec,3,3)
mymatrix2 <- matrix(9:1,3,3)
# component-wise multiplication
mymatrix1 * mymatrix2
```

Tinn-R, an editor with more options

Features:

•R console window access from within Tinn-R.
•Incremental execution of R code.
•Integrated R help.
•R Object explorer.
•Line number for a source file.
•Search and Replace.
•Current line highlighting.
Etc...



```
# Part A
A=matrix(runif(4*4),4,4)
A=t(A)%*%A

# Part B
n=1000
X=matrix(runif(4*n),4,n)-0.5
d=diag(t(X)%*%A%*%X)/diag(t(X)%*%X)

#Part C
min(d)
max(d)
eigen(A)$values
n=10000
X=matrix(runif(4*n),4,n)-0.5
d=rep(0,n)
for (i in 1:n)
```

```
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.

> 
```

http://jekyll.math.byuh.edu/other/howto/tinnr/using.shtml

Getting information about R and the system

To get the R version	To get the license info	To learn how to cite R in publications	info about the platform under which R was built	system and user information
R.version	license()	citation()	.Platform	Sys.info()

```
> R.version
               _
platform       i386-pc-mingw32
arch           i386
os             mingw32
system         i386, mingw32
status
major          2
minor          9.2
year           2009
month          08
day            24
svn rev        49384
language       R
version.string R version 2.9.2 (2009-08-24)
> #  this might return a wrong value
> R.version$os
[1] "mingw32"
> # this is correct
> .Platform$OS.type
[1] "windows"
```

```
> Sys.info()[7]
           user
"Administrator"
> Sys.info()["user"]
           user
"Administrator"
```

Getting information about R and the system

To get a list of the installed packages	To get a list new packages available	version information about R and attached or loaded packages	numerical characteristics of the machine	names of open graphics devices
installed.packages()	old.packages()	sessionInfo()	.Machine	.Device

command line+R Editor

References/to learn more:

The R book
Michael J. Crawley pp 9
2008 John Wiley & Sons Ltd

Basic statistics using R pp. 34
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/calculator.r

Chem 351 Archives Page
David Harvey
http://fs6.depauw.edu:50080/~harvey/Chem%20351/PDF%20Files/Handouts/RDocs/Some
%20Basic%20R%20Commands.pdf

Packages

The base distribution of R is the R programming language interpreter and some packages that are loaded by default. Packages, AKA extensions, are libraries that can be installed and used when needed and extend the functionality of R by adding new objects, for example new statistical functions, and their documentation and even data.

A package is a zip file, containing a file with the description of the package and subdirectories with the source code of the package and other information such as documentation, configuration, license, etc... This is described on the manual "Writing R Extensions".
Several projects distribute contributed packages, such as CRAN (The Comprehensive R Archive Network), Bioconductor (Analysis and comprehension of genomic data), OmegaHat (software for S, R and Xlisp-Stat), etc...

There are about 30 default packages, the base package has functions for the R programming language, other packages have functions for data input/output, graphics, utilities and statistical tools.

Packages are one of the strengths of R, with over 2000 packages available, therefore, there are many functions to handle packages.

Packages

How to get and use a package

Two steps for using a package $\left\{ \begin{array}{l} \text{Get/download/}\textbf{install}\text{ the package (get the file into the hard drive)} \\ \\ \text{Use/access/}\textbf{load}\text{ the package (get the file from the hard drive into memory, from R)} \end{array} \right.$

Packages

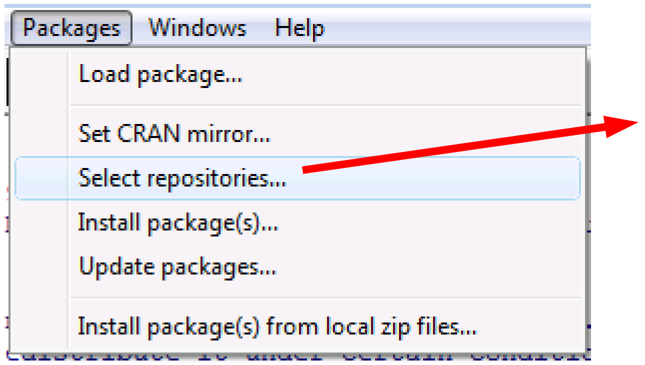
Installing a package

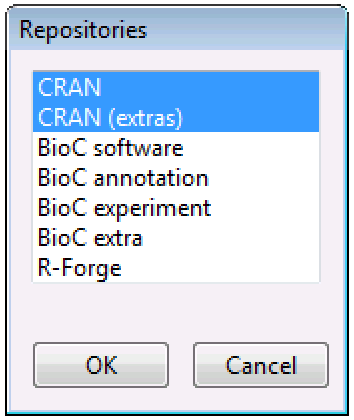
- Select repository (repositories store packages distributed by the main projects), optional
- Set CRAN mirror (there are CRAN mirrors in most countries, allowing fast downloads), optional
- Install package (get the file into the hard drive)
  - Download from the web
  - Copy from a USB stick

Packages

Select repository

Which distributor has the necessary packages





```
> setRepositories()
--- Please select repositories for use in this session ---

1: + CRAN
2: + CRAN (extras)
3:   BioC software
4:   BioC annotation
5:   BioC experiment
6:   BioC extra
7:   R-Forge

Enter one or more numbers separated by spaces
1: |
```

CRAN is the basic R distribution

CRAN (extras) are Contributed R packages

BioC are packages from Bioconductor (bioinformatics/biostatistics, focused on inference using DNA microarrays)

R-forge are packages from Omegahat (umbrella project for S, R and Lisp-stats, focused on statistical tools, with web applications, web services, Java, distribuited computing, etc...)

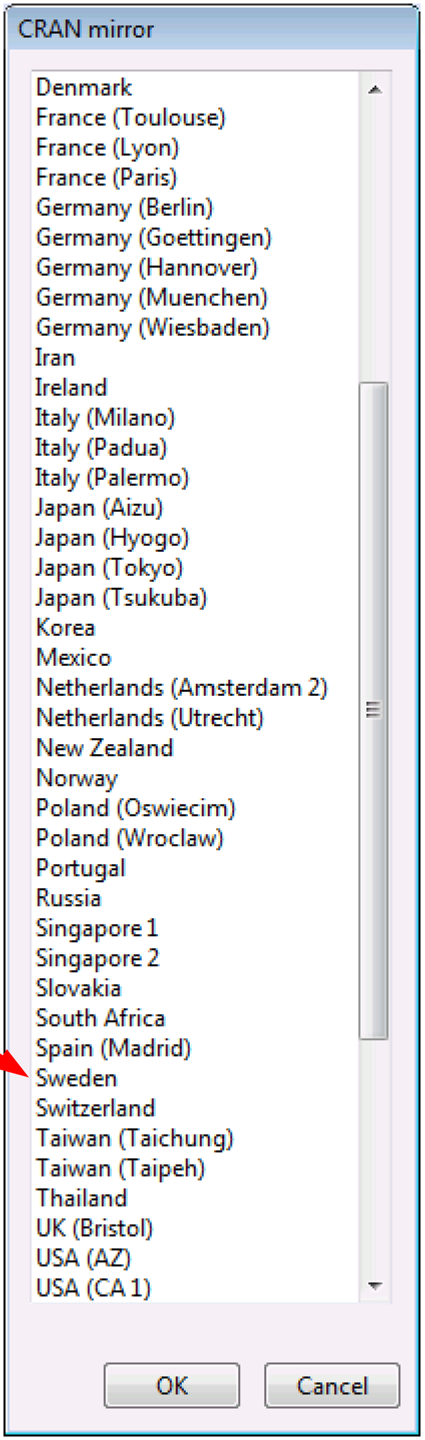
Packages

Set CRAN mirror

Which server is closer or faster/more reliable

Sweden is the closest

```
> library()
> chooseCRANmirror()
```

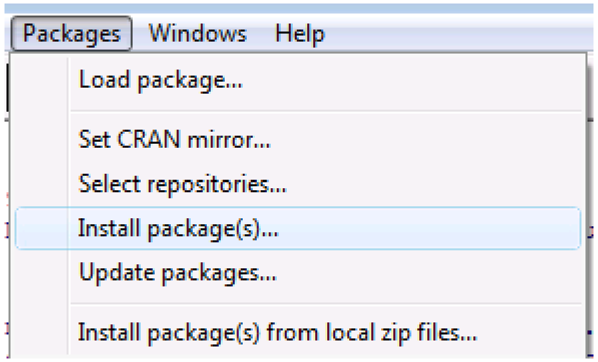


CRAN mirror

Denmark
France (Toulouse)
France (Lyon)
France (Paris)
Germany (Berlin)
Germany (Goettingen)
Germany (Hannover)
Germany (Muenchen)
Germany (Wiesbaden)
Iran
Ireland
Italy (Milano)
Italy (Padua)
Italy (Palermo)
Japan (Aizu)
Japan (Hyogo)
Japan (Tokyo)
Japan (Tsukuba)
Korea
Mexico
Netherlands (Amsterdam 2)
Netherlands (Utrecht)
New Zealand
Norway
Poland (Oswiecim)
Poland (Wroclaw)
Portugal
Russia
Singapore 1
Singapore 2
Slovakia
South Africa
Spain (Madrid)
Sweden
Switzerland
Taiwan (Taichung)
Taiwan (Taipeh)
Thailand
UK (Bristol)
USA (AZ)
USA (CA 1)

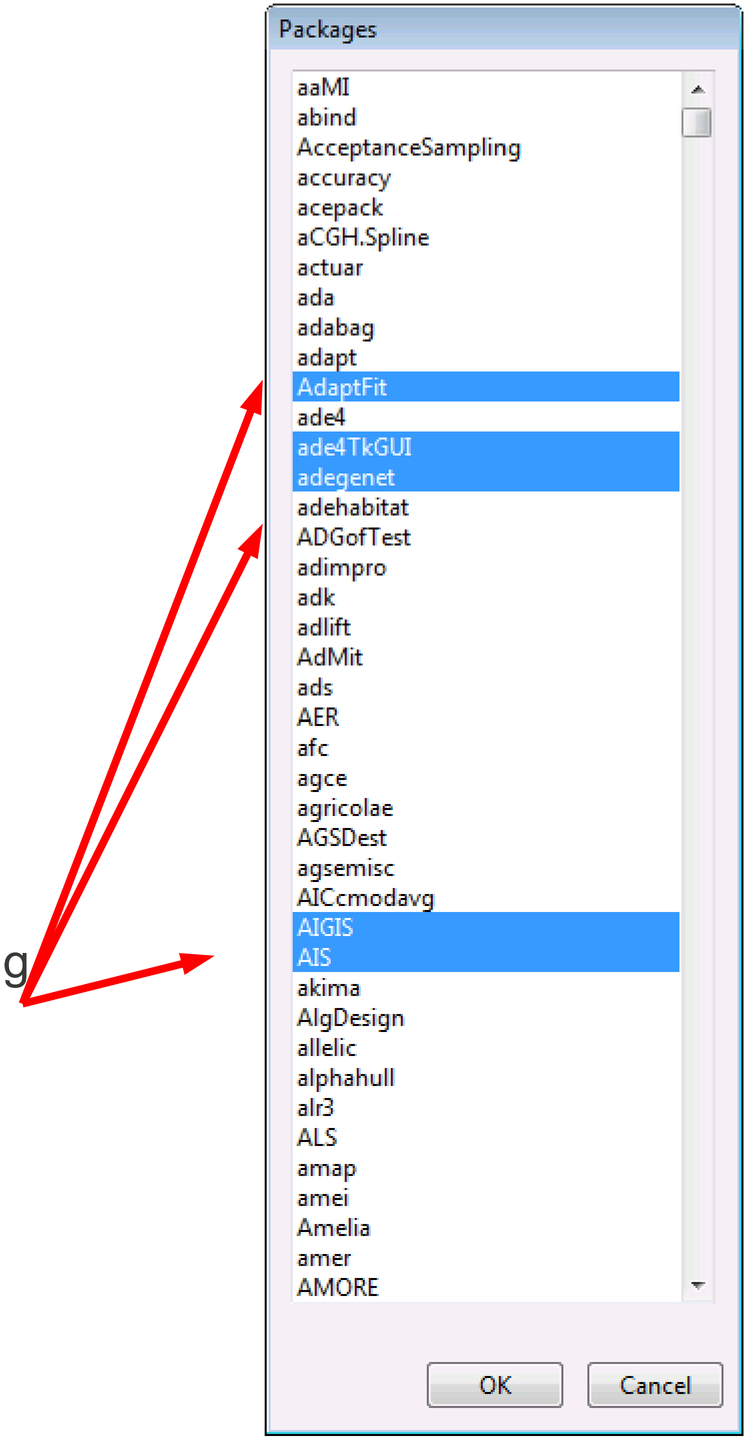
Packages

Install package



If no CRAN mirror was selected in this session, then it will ask for one.

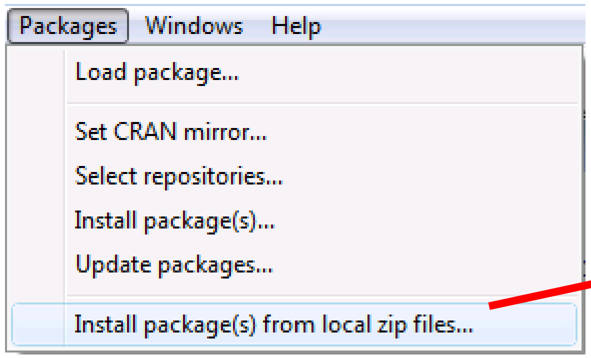
Multiple packages can be chosen by pressing the control key and clicking on the package name.

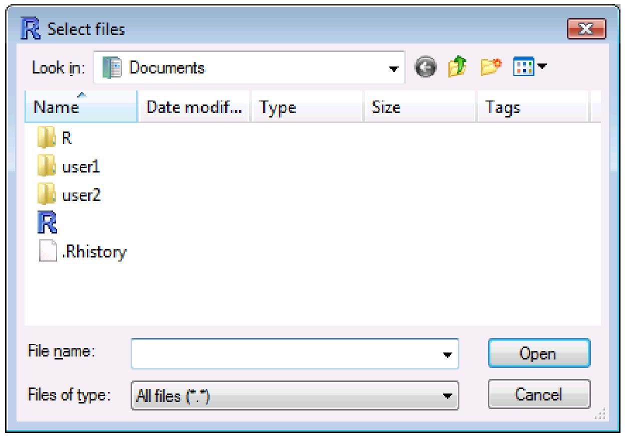


Packages

Install package from local zip file

Instead of from the web, for machines without web access





Packages

Functions to work with packages

available.packages
old.packages
new.packages
download.packages
update.packages
install.packages
remove.packages

available.packages() - packages/bundles available at one or more repositories

old.packages() - packages/bundles that have newer versions on the repositories

new.packages() - uninstalled packages/bundles that are available at the repositories

download.packages() - downloads the newest versions of packages/bundles

update.packages() - the user will be prompted for which packages/bundles with a newer version to update

install.packages() - installs new packages/bundles

remove.packages() - removes installed packages/bundles and updates index information as necessary

Packages

When do I need such functions?

available.packages() - I want a list of all the existing packages!

old.packages() - are there newer versions of the packages/bundles installed?

new.packages() - are there new packages/bundles?

download.packages() - I want to download packages/bundles.

update.packages() - I want to see which packages/bundles have a newer version and decide, interactively, which ones to update.

install.packages() - I want to install packages/bundles.

remove.packages() - I want to remove installed packages/bundles.

Packages

Other functions:

library()	list all available packages
library(lib.loc = .Library)	list all packages in the default library
library(ada)	load package "ada"
require(ada)	load the package "ada" from inside other functions
library(help = ada)	documentation on package 'ada'
search()	list of attached packages and R objects
.packages	information about package availability
.packages(all.available = TRUE)	return all available as character vector
detach("package:ada")	unload package "ada"

Trying to use function "foo" from a package that is not yet loaded will return an error:
Error: could not find function "foo"

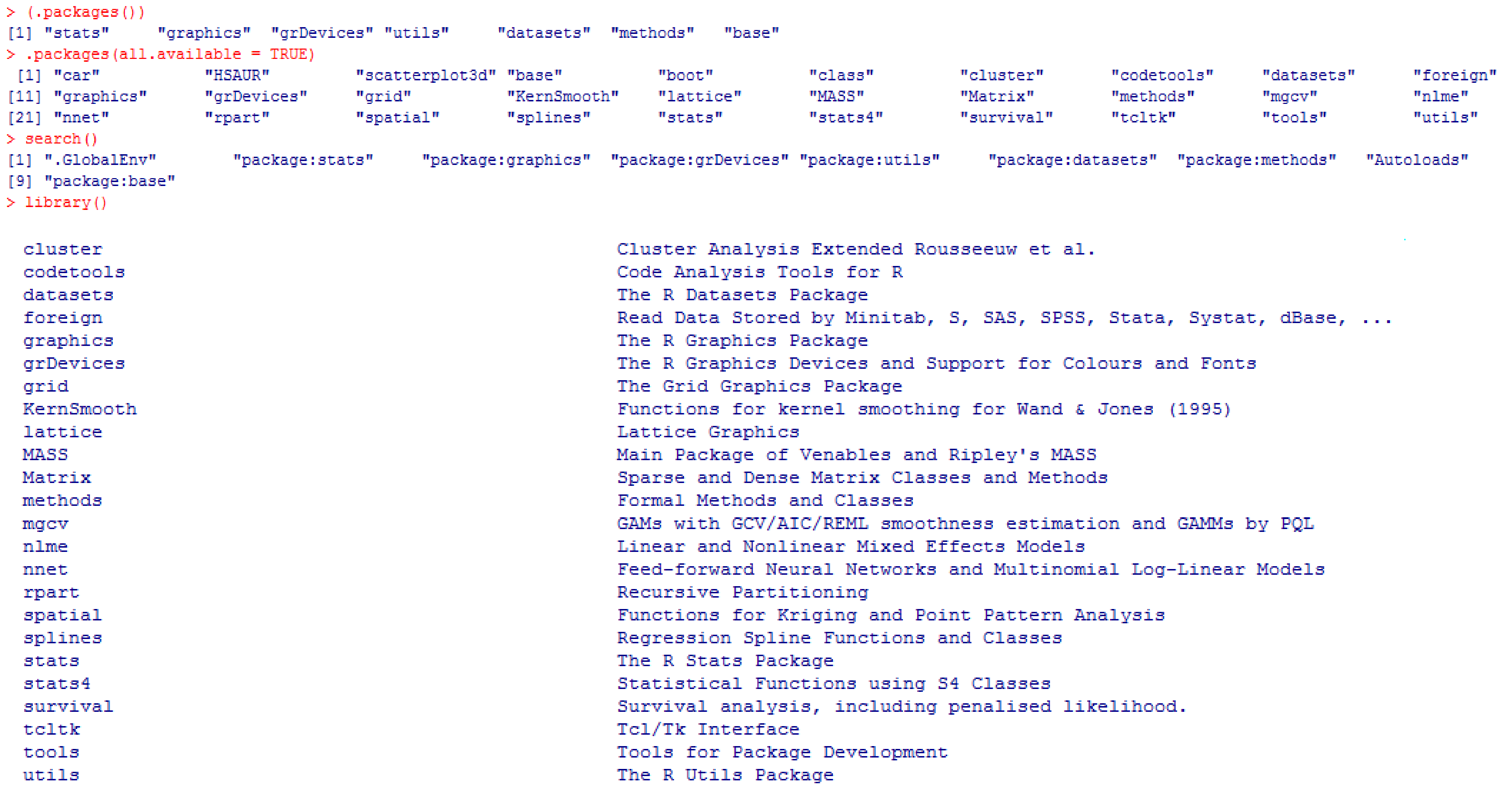
Packages

Loaded packages

search() = .packages() + R objects

Installed packages

library() = .packages(all.available = TRUE) with description

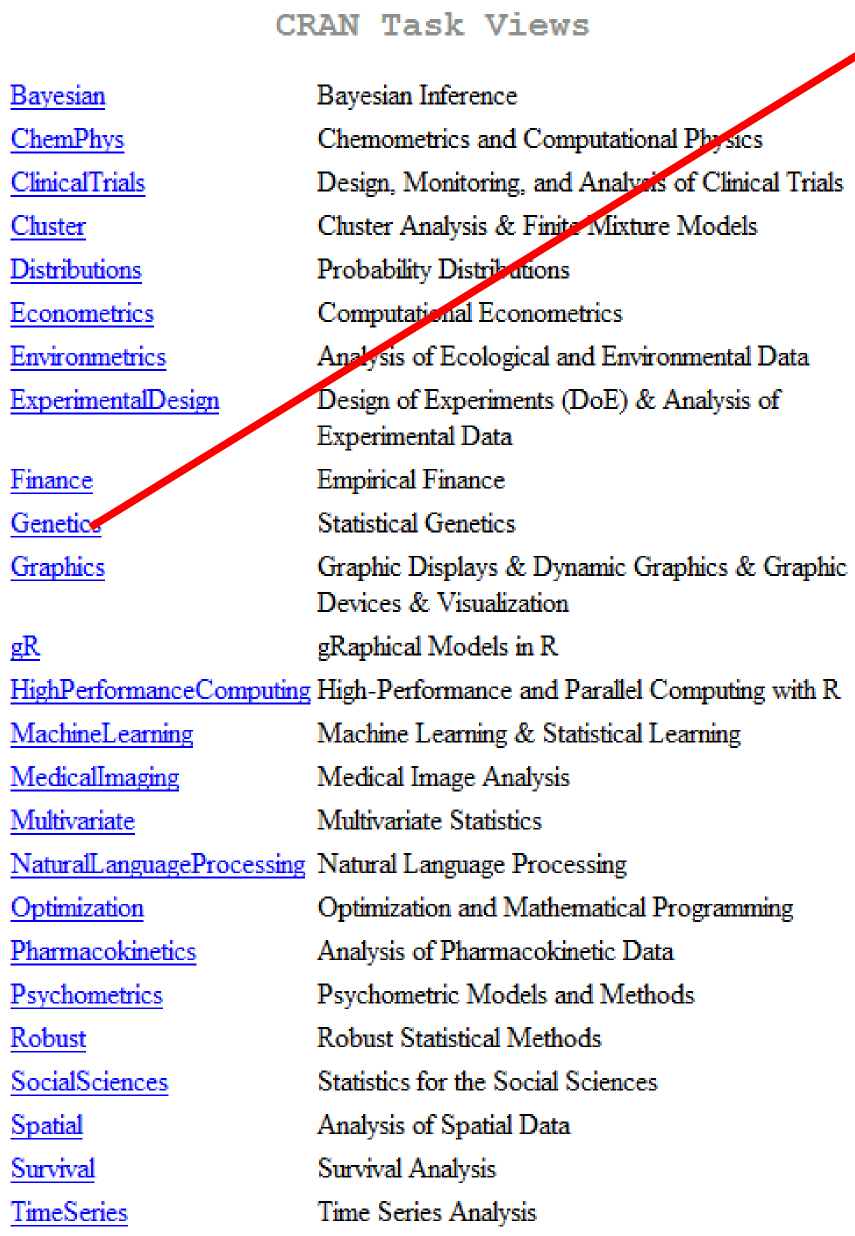


```
> (.packages())
[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
> .packages(all.available = TRUE)
 [1] "car"           "HSAUR"         "scatterplot3d" "base"          "boot"          "class"         "cluster"       "codetools"     "datasets"      "foreign"
[11] "graphics"      "grDevices"     "grid"          "KernSmooth"    "lattice"       "MASS"          "Matrix"        "methods"       "mgcv"          "nlme"
[21] "nnet"          "rpart"         "spatial"       "splines"       "stats"         "stats4"        "survival"      "tcltk"         "tools"         "utils"
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"  "package:grDevices" "package:utils"     "package:datasets"  "package:methods"   "Autoloads"
[9] "package:base"
> library()

  cluster                 Cluster Analysis Extended Rousseeuw et al.
  codetools               Code Analysis Tools for R
  datasets                The R Datasets Package
  foreign                 Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
  graphics                The R Graphics Package
  grDevices               The R Graphics Devices and Support for Colours and Fonts
  grid                    The Grid Graphics Package
  KernSmooth              Functions for kernel smoothing for Wand & Jones (1995)
  lattice                 Lattice Graphics
  MASS                    Main Package of Venables and Ripley's MASS
  Matrix                  Sparse and Dense Matrix Classes and Methods
  methods                 Formal Methods and Classes
  mgcv                    GAMs with GCV/AIC/REML smoothness estimation and GAMMs by PQL
  nlme                    Linear and Nonlinear Mixed Effects Models
  nnet                    Feed-forward Neural Networks and Multinomial Log-Linear Models
  rpart                   Recursive Partitioning
  spatial                 Functions for Kriging and Point Pattern Analysis
  splines                 Regression Spline Functions and Classes
  stats                   The R Stats Package
  stats4                  Statistical Functions using S4 Classes
  survival                Survival analysis, including penalised likelihood.
  tcltk                   Tcl/Tk Interface
  tools                   Tools for Package Development
  utils                   The R Utils Package
```

Packages

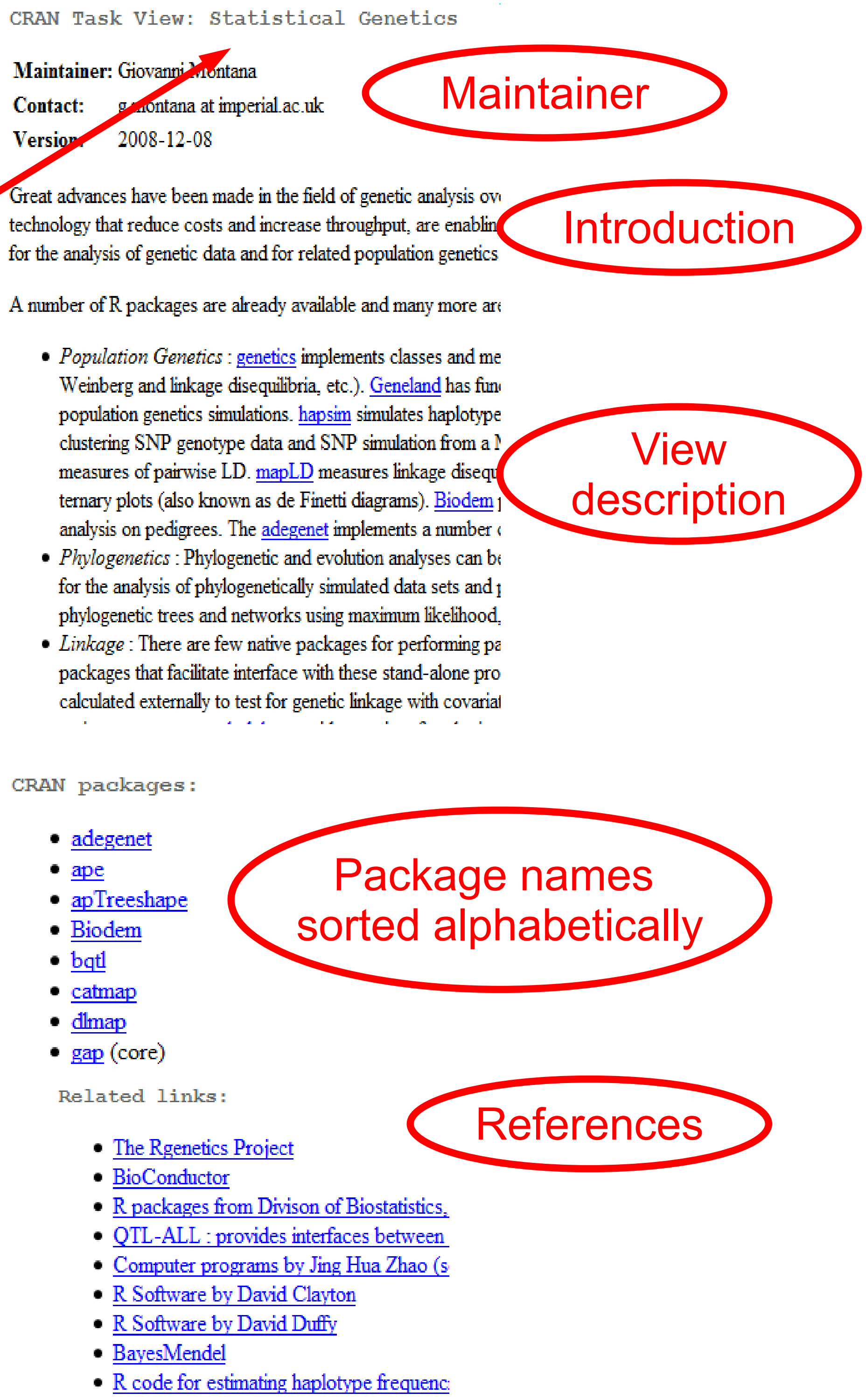
To browse packages by topics (views)

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



CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Design, Monitoring, and Analysis of Clinical Trials
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 with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
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



CRAN Task View: Statistical Genetics

Maintainer: Giovanni Montana
Contact: g.montana at imperial.ac.uk
Version: 2008-12-08

Great advances have been made in the field of genetic analysis ove
technology that reduce costs and increase throughput, are enablin
for the analysis of genetic data and for related population genetics

A number of R packages are already available and many more are

- *Population Genetics* : genetics implements classes and me Weinberg and linkage disequilibria, etc.). Geneland has fun population genetics simulations. hapsim simulates haplotype clustering SNP genotype data and SNP simulation from a N measures of pairwise LD. mapLD measures linkage disequ ternary plots (also known as de Finetti diagrams). Biodem analysis on pedigrees. The adegenet implements a number
- *Phylogenetics* : Phylogenetic and evolution analyses can be for the analysis of phylogenetically simulated data sets and phylogenetic trees and networks using maximum likelihood,
- *Linkage* : There are few native packages for performing pa packages that facilitate interface with these stand-alone pro calculated externally to test for genetic linkage with covariat

CRAN packages:

- adegenet
- ape
- apTreeshape
- Biodem
- bqtl
- catmap
- dlmap
- gap (core)

Related links:

- The Rgenetics Project
- BioConductor
- R packages from Divison of Biostatistics,
- QTL-ALL : provides interfaces between
- Computer programs by Jing Hua Zhao (s
- R Software by David Clayton
- R Software by David Duffy
- BayesMendel
- R code for estimating haplotype frequenc

Packages

CRAN task views are categories of contributed packages with simplified installation:

To automatically install these views, the ctv package needs to be installed:

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

The views can be installed now:

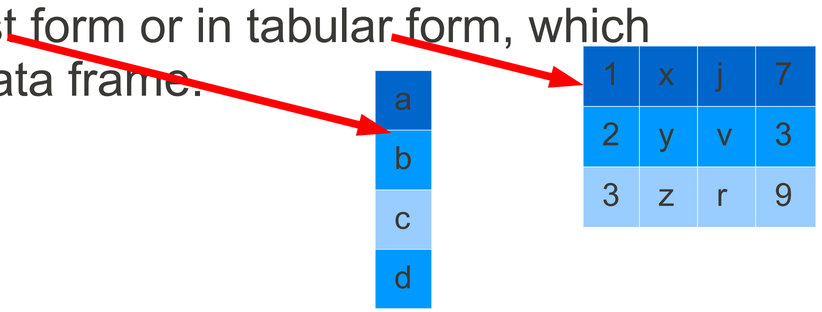
```
install.views("Genetics")
```

or

```
update.views("Genetics")
```

Data sets

A dataset is a collection of data, usually in a list form or in tabular form, which corresponds, on R, to data types vector and data frame.



R loads datasets from:

1. files ending ‘.R’ or ‘.r’ are opened with source()
2. files ending ‘.RData’ or ‘.rda’ are opened with load()
3. files ending ‘.tab’, ‘.txt’ or ‘.TXT’ are read with read.table(..., header = TRUE) into a data frame.
4. files ending ‘.csv’ or ‘.CSV’ are read with read.table(..., header = TRUE, sep = ";") into a data frame.

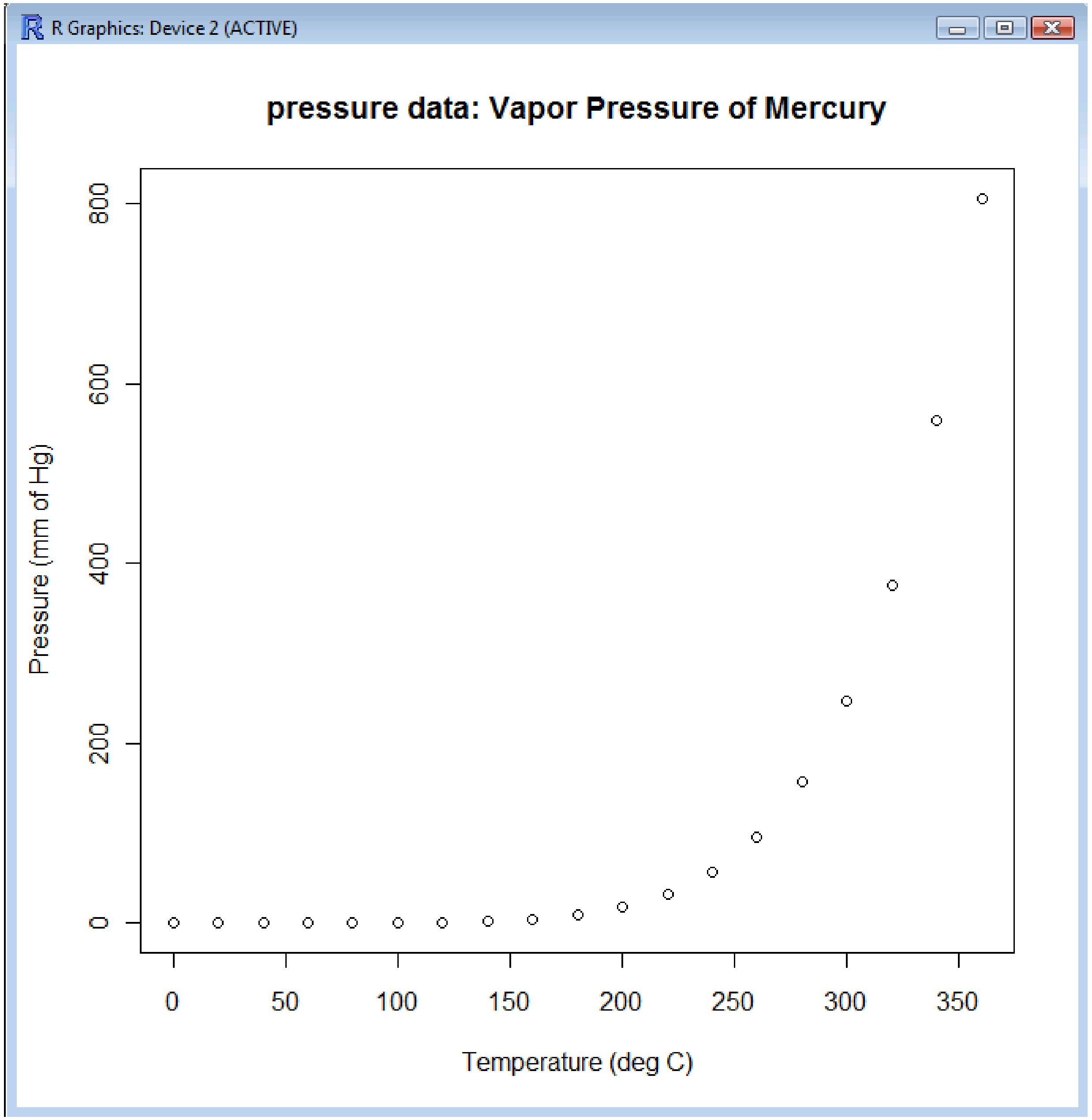
Data set functions:

```
data()                          # list all available data sets
try(data(package = "car") )# list the data sets in the car package
data(car)  # load the data set 'car'
help(car)               # give information on data set 'car'
data(package = .packages(all.available = TRUE)) # lists the data sets in all available packages
```

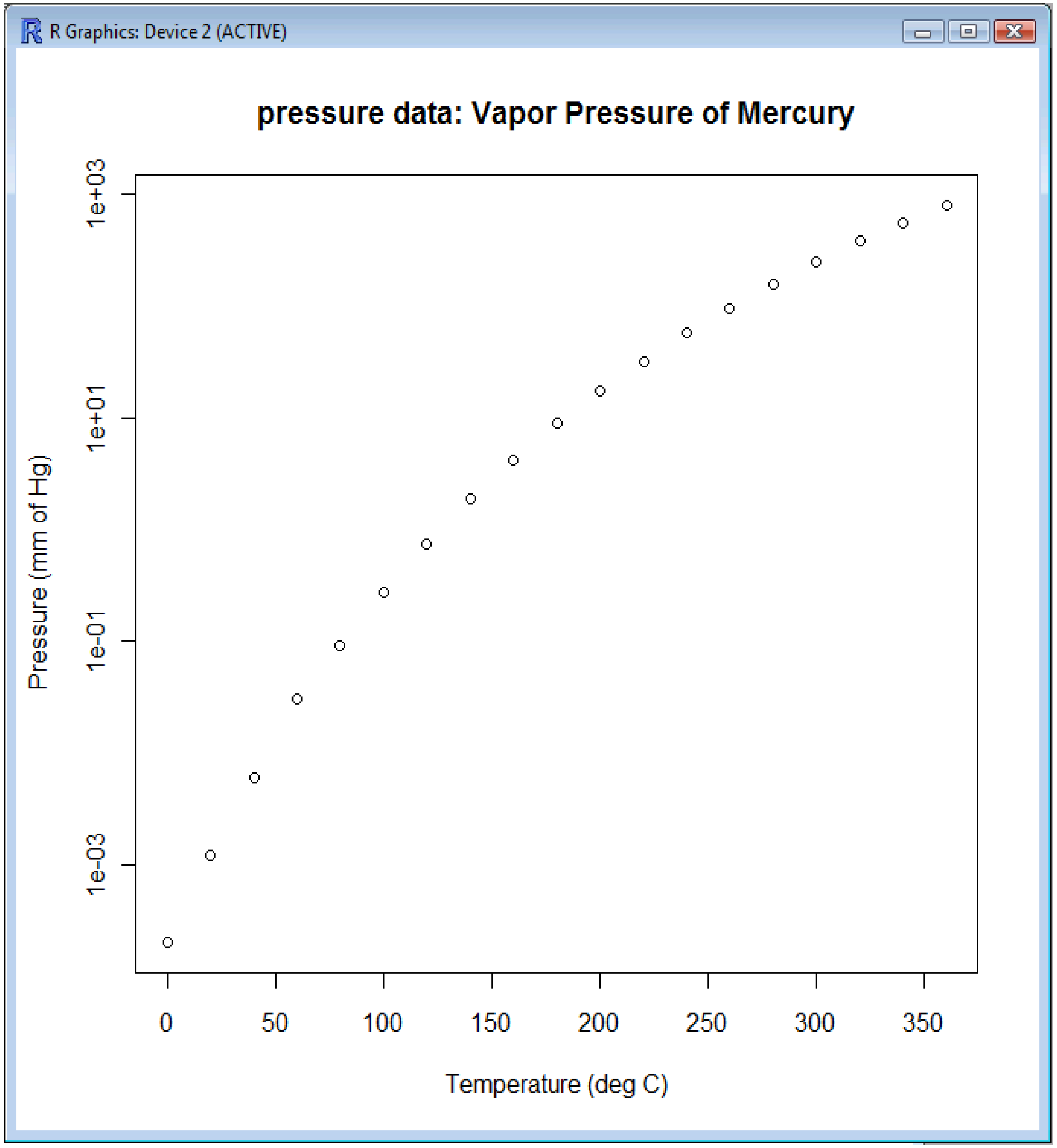
R comes with some datasets already installed, one is pressure and it is the "Vapor Pressure of Mercury as a Function of Temperature".

```
require(graphics) #just to make sure the graphics library is loaded
pressure
?pressure

mean(pressure)
median(pressure)
min(pressure)
max(pressure)
quantile(pressure$pressure)
summary(pressure)

var(pressure)
sd(pressure)
cor(pressure)

boxplot(pressure)
```

```
#decimal scale
plot(pressure, xlab = "Temperature (deg C)",  ylab = "Pressure (mm of Hg)", main = "pressure data: Vapor Pressure of Mercury")
```



```
#log scale
plot(pressure, xlab = "Temperature (deg C)",  log = "y", ylab = "Pressure (mm of Hg)", main =
"pressure data: Vapor Pressure of Mercury")
```



Packages

Assignment: packages and help

1. Is the car package loaded?
2. Is the car package installed?
3. Install the car package
4. Load the car package
5. Is there help for the car package?
6. Find out information about the data frame Angell
7. Find out what the function scatterplot does
8. Run an example of scatterplot
9. Unload the car package
10. Uninstall the car package
11. List packages for epidemiology
12. List packages for environmental sciences

http://cran.r-project.org/web/packages/car/car.pdf

Packages

1. Is the car package loaded?
2. Is the car package installed?

```
> # 1. Is the car package loaded?
> (.packages())
[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
[7] "base"
> # 2. Is the car package installed?
> (.packages(all.available=TRUE))
 [1] "biglm"         "DBI"           "ISwR"          "leaps"
 [5] "RODBC"         "RSQLite"       "scatterplot3d" "base"
 [9] "boot"          "class"         "cluster"       "codetools"
[13] "datasets"      "foreign"       "graphics"      "grDevices"
[17] "grid"          "KernSmooth"    "lattice"       "MASS"
[21] "Matrix"        "methods"       "mgcv"          "nlme"
[25] "nnet"          "rpart"         "spatial"       "splines"
[29] "stats"         "stats4"        "survival"      "tcltk"
[33] "tools"         "utils"
```

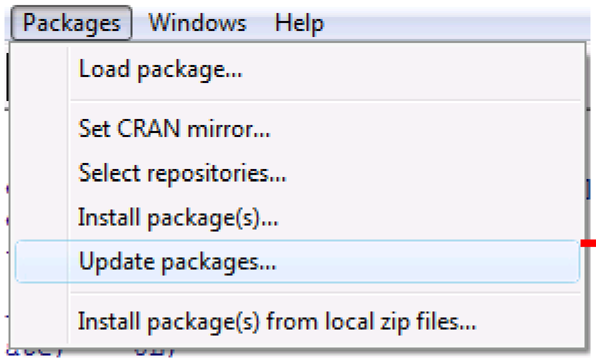
Packages

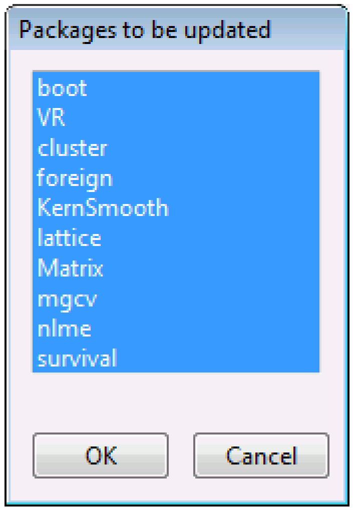
Before installing a package it is advisable to make sure all installed dependencies have their latest versions.

On the console:

update.packages()

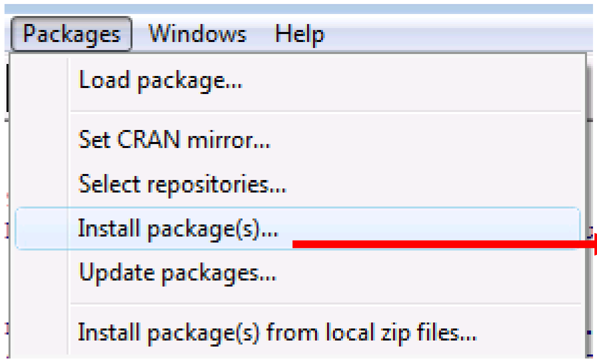
On RGui:

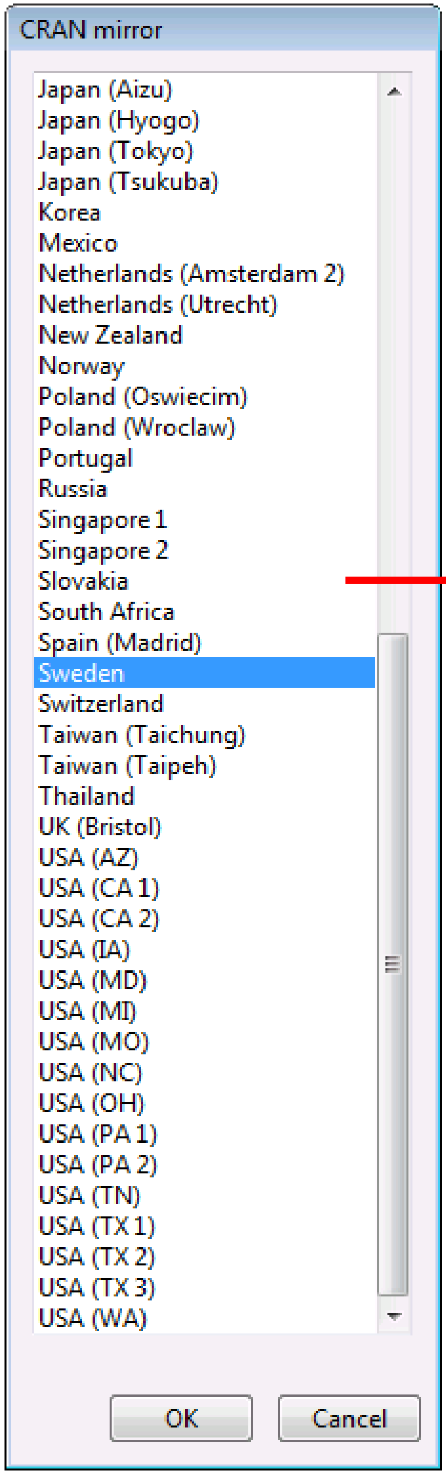


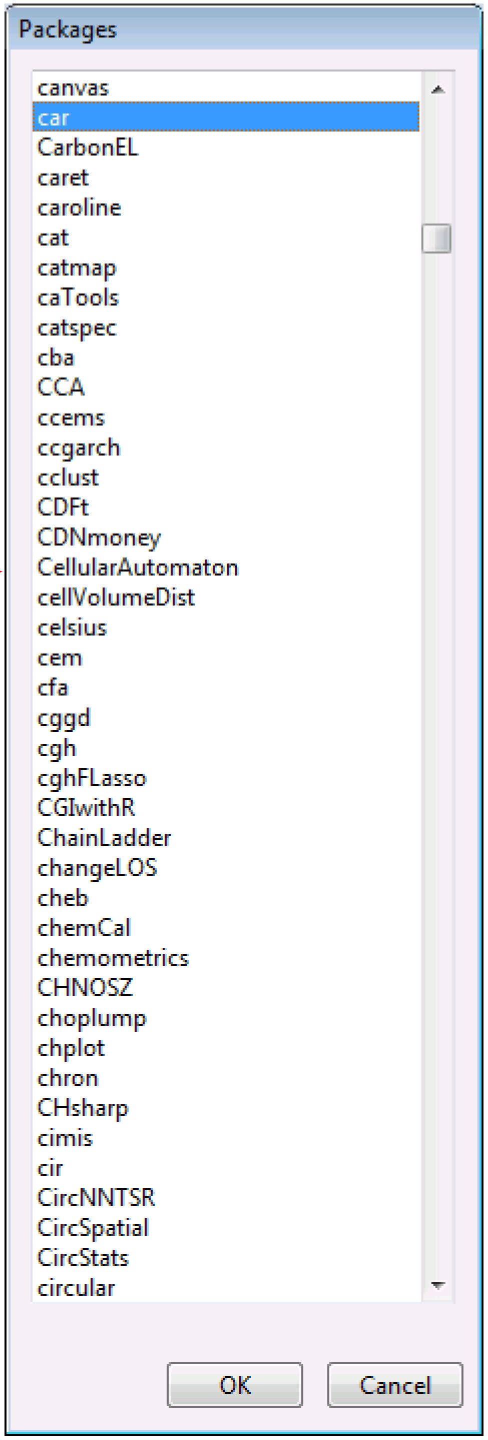


Packages

3. Install package car
with RGui from the web





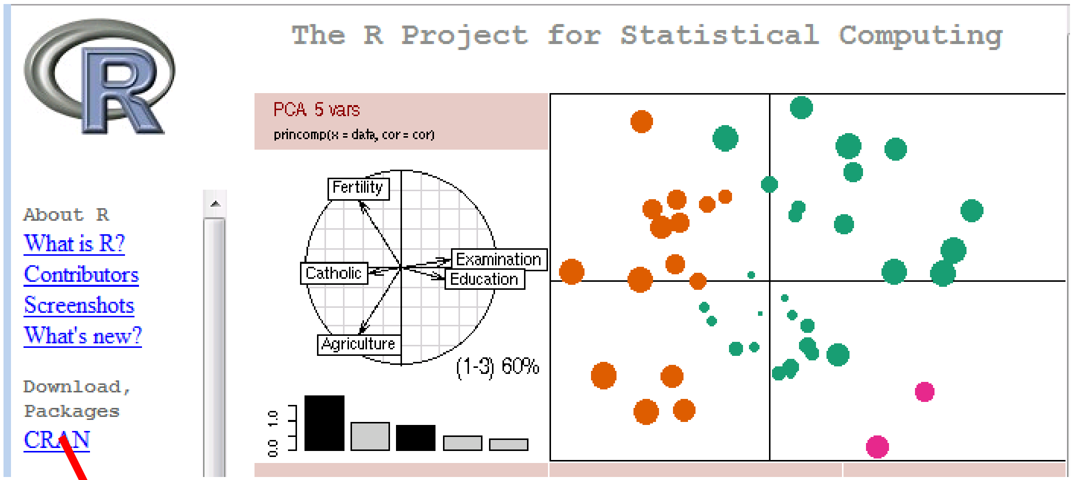


From the console:

install.packages("car",
dependencies = TRUE)

Packages

Install package car from a zip file



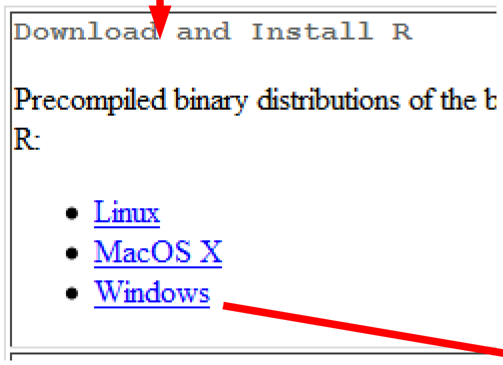
The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download,
Packages
CRAN

Sweden

http://ftp.sunet.se/pub/lang/CRAN/ Swedish University Computer Network, Uppsala



Download and Install R

Precompiled binary distributions of the b
R:

- Linux
- MacOS X
- Windows

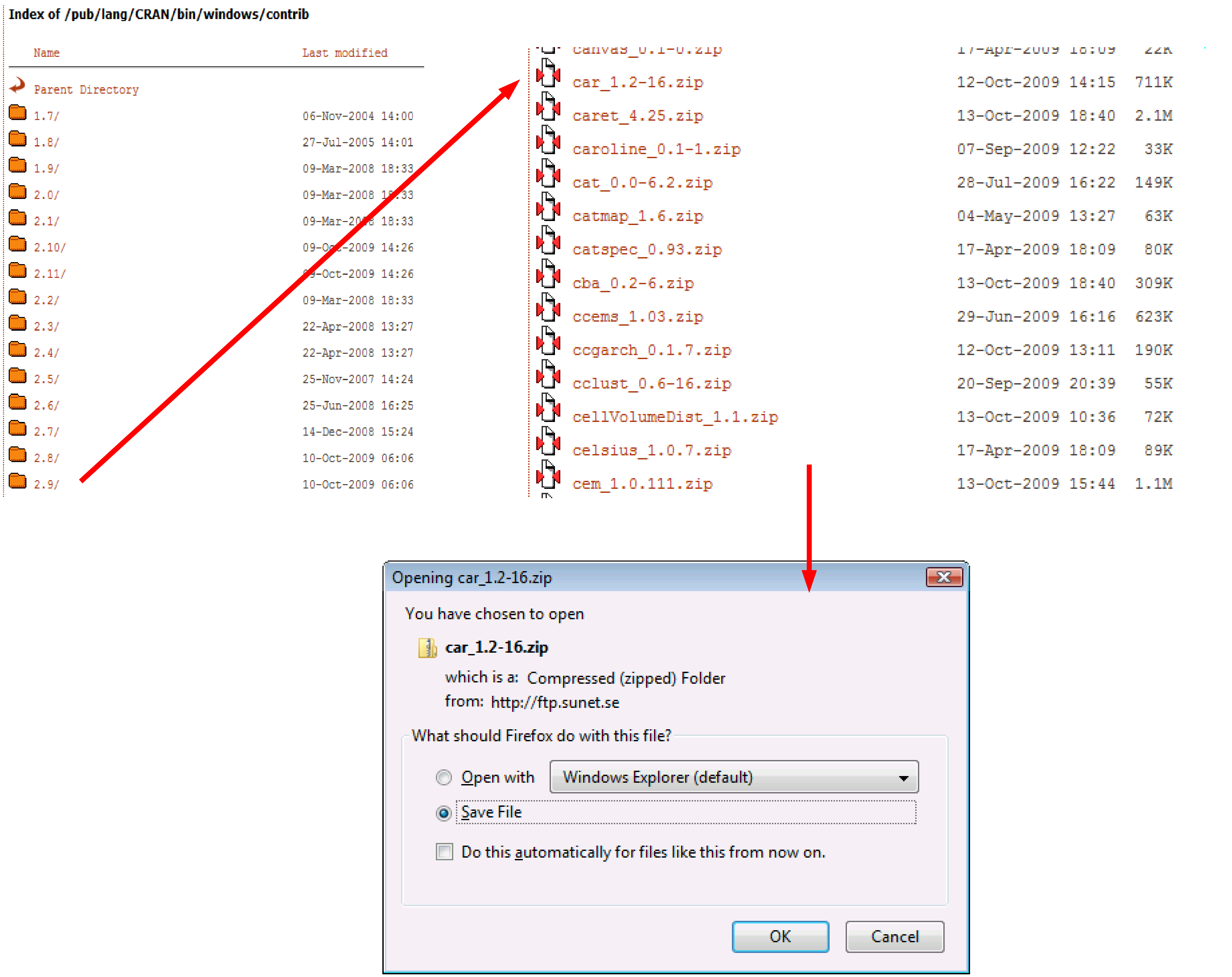
R for Windows

This directory contains binaries for a base distribution and packages to run on i386/x64 Windows.

Note: CRAN does not have Windows systems and cannot check these binaries for viruses. Use the normal pre

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch)
contrib	Binaries of contributed packages (managed by Uwe Ligges)



```
Index of /pub/lang/CRAN/bin/windows/contrib

   Name                              Last modified

   Parent Directory
   1.7/                              06-Nov-2004 14:00
   1.8/                              27-Jul-2005 14:01
   1.9/                              09-Mar-2008 18:33
   2.0/                              09-Mar-2008 18:33
   2.1/                              09-Mar-2008 18:33
   2.10/                             09-Oct-2009 14:26
   2.11/                             09-Oct-2009 14:26
   2.2/                              09-Mar-2008 18:33
   2.3/                              22-Apr-2008 13:27
   2.4/                              22-Apr-2008 13:27
   2.5/                              25-Nov-2007 14:24
   2.6/                              25-Jun-2008 16:25
   2.7/                              14-Dec-2008 15:24
   2.8/                              10-Oct-2009 06:06
   2.9/                              10-Oct-2009 06:06
```

```
   canvas_0.1-0.zip              17-Apr-2009 18:09   22K
   car_1.2-16.zip                12-Oct-2009 14:15  711K
   caret_4.25.zip                13-Oct-2009 18:40  2.1M
   caroline_0.1-1.zip            07-Sep-2009 12:22   33K
   cat_0.0-6.2.zip               28-Jul-2009 16:22  149K
   catmap_1.6.zip                04-May-2009 13:27   63K
   catspec_0.93.zip              17-Apr-2009 18:09   80K
   cba_0.2-6.zip                 13-Oct-2009 18:40  309K
   ccems_1.03.zip                29-Jun-2009 16:16  623K
   ccgarch_0.1.7.zip             12-Oct-2009 13:11  190K
   cclust_0.6-16.zip             20-Sep-2009 20:39   55K
   cellVolumeDist_1.1.zip        13-Oct-2009 10:36   72K
   celsius_1.0.7.zip             17-Apr-2009 18:09   89K
   cem_1.0.111.zip               13-Oct-2009 15:44  1.1M
```

Opening car_1.2-16.zip

You have chosen to open

car_1.2-16.zip

which is a: Compressed (zipped) Folder

from: http://ftp.sunet.se

What should Firefox do with this file?

○ Open with [Windows Explorer (default)]

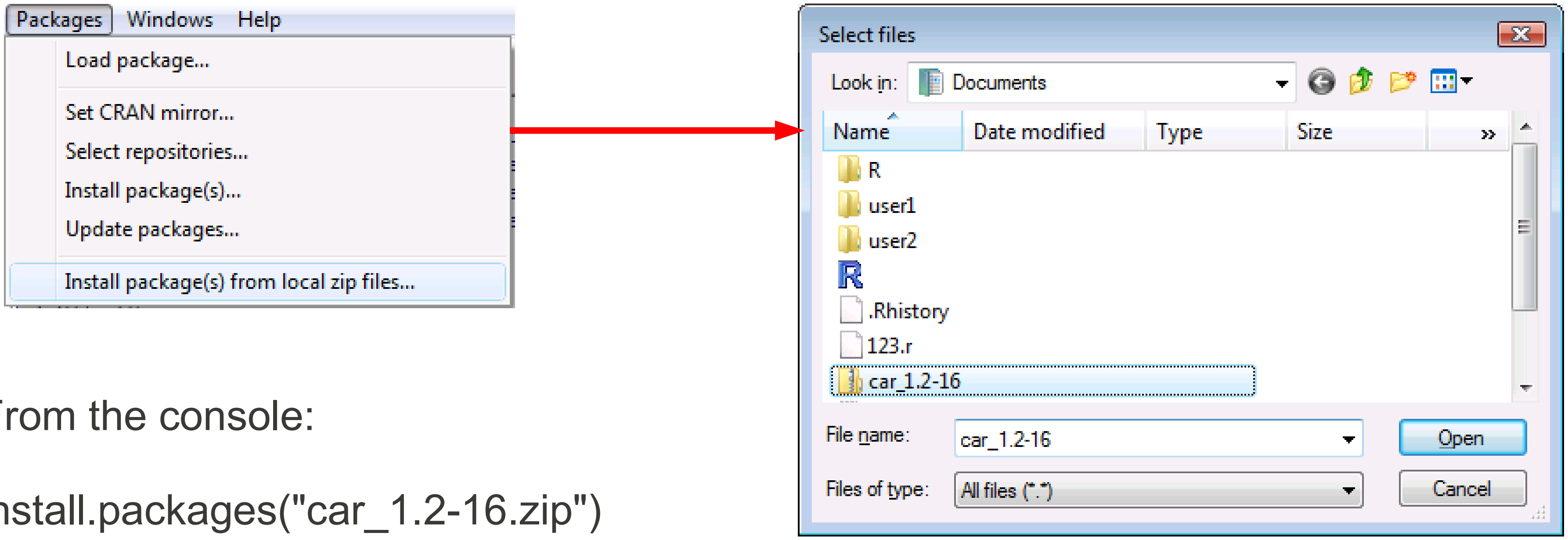
● Save File

☐ Do this automatically for files like this from now on.

Packages

Install package car from a zip file

Copy the zip file to the working directory



From the console:

install.packages("car_1.2-16.zip")

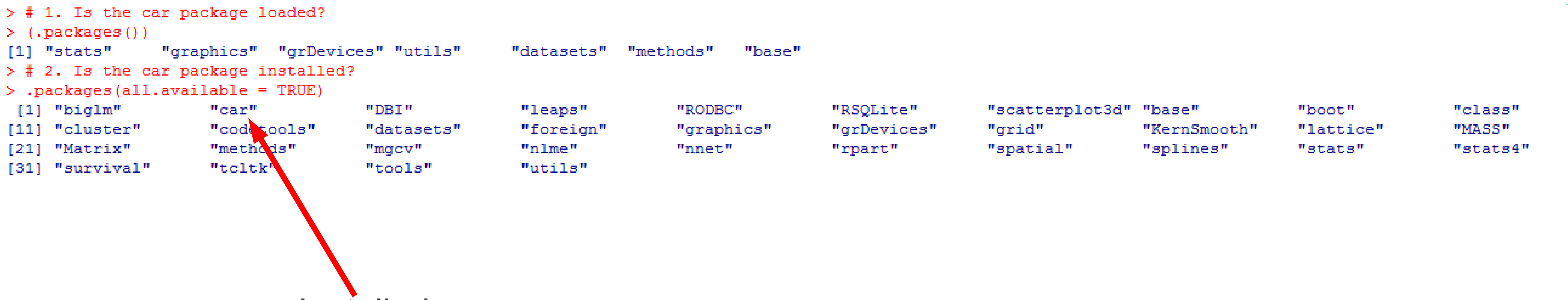
On linux or MacOsx:

R CMD INSTALL car.tar.gz

Packages

Double check:

Is the car package loaded?
Is the car package installed?

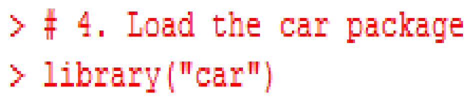


```
> # 1. Is the car package loaded?
> (.packages())
[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
> # 2. Is the car package installed?
> .packages(all.available = TRUE)
 [1] "biglm"        "car"          "DBI"          "leaps"        "RODBC"        "RSQLite"      "scatterplot3d" "base"         "boot"         "class"
[11] "cluster"      "codetools"    "datasets"     "foreign"      "graphics"     "grDevices"    "grid"         "KernSmooth"   "lattice"      "MASS"
[21] "Matrix"       "methods"      "mgcv"         "nlme"         "nnet"         "rpart"        "spatial"      "splines"      "stats"        "stats4"
[31] "survival"     "tcltk"        "tools"        "utils"
```

Installed

Packages

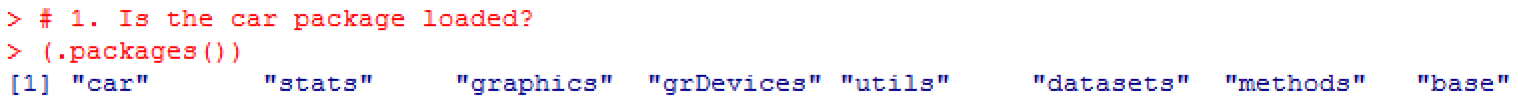
4. Load the car package



```
> # 4. Load the car package
> library("car")
```

Double check:

Is the car package loaded?



```
> # 1. Is the car package loaded?
> (.packages())
[1] "car"       "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
```

Loaded

Packages

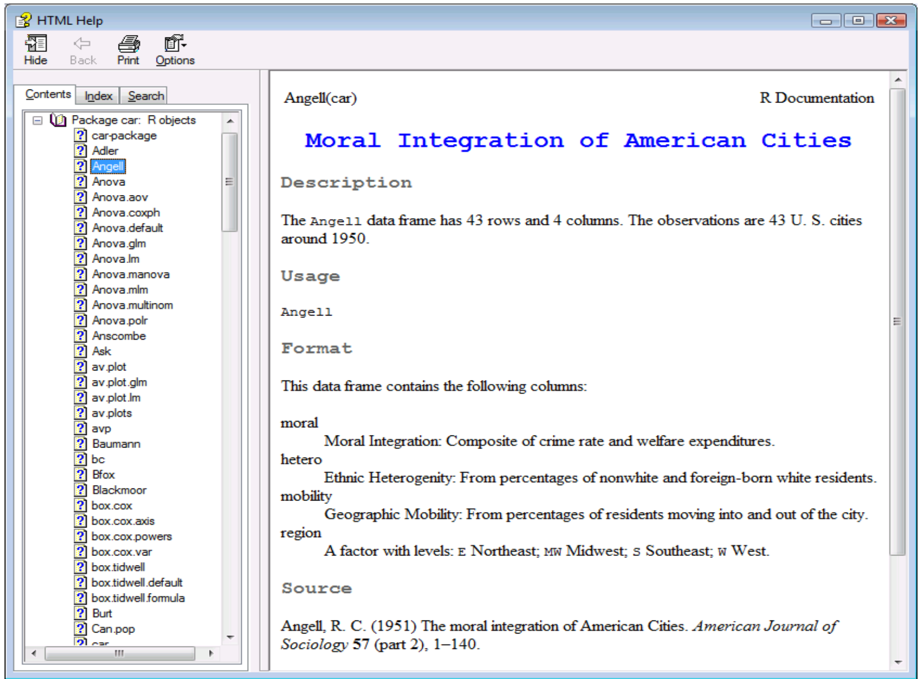
# 5. Is there help for the car package?

library(help = car)

```
                Information on package 'car'

Description:

Package:            car
Version:            1.2-16
Date:               2009/10/10
Title:              Companion to Applied Regression
Author:             John Fox <jfox@mcmaster.ca>. I am grateful to Douglas Bates, David Firth, Michael Friendly, Gregor Gorjanc, Spencer Graves, Richard
                    Heiberger, Georges Monette, Henric Nilsson, Derek Ogle, Brian Ripley, Sanford Weisberg, and Achim Zeileis for various suggestions and
                    contributions.
Maintainer:         John Fox <jfox@mcmaster.ca>
Depends:            R (>= 2.1.1), stats, graphics
Suggests:           MASS, nnet, leaps, survival
```

# 6. Find out information about the data frame Angell

help(Angell)



Angell(car) R Documentation

Moral Integration of American Cities

Description

The `Angell` data frame has 43 rows and 4 columns. The observations are 43 U. S. cities around 1950.

Usage

`Angell`

Format

This data frame contains the following columns:

moral
    Moral Integration: Composite of crime rate and welfare expenditures.
hetero
    Ethnic Heterogenity: From percentages of nonwhite and foreign-born white residents.
mobility
    Geographic Mobility: From percentages of residents moving into and out of the city.
region
    A factor with levels: `E` Northeast; `MW` Midwest; `S` Southeast; `W` West.

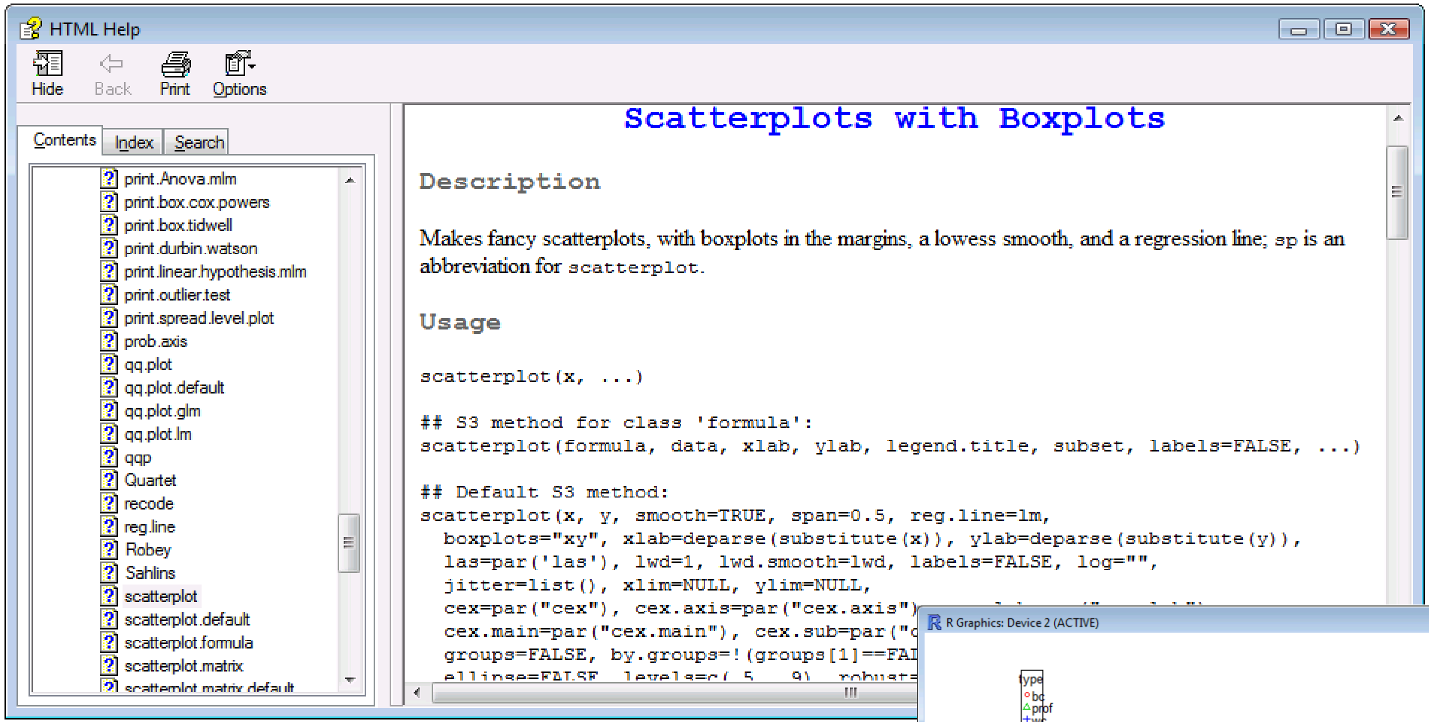
Source

Angell, R. C. (1951) The moral integration of American Cities. *American Journal of Sociology* 57 (part 2), 1–140.

Packages

7. Find out what the function scatterplot does

?scatterplot



print.Anova.mlm
print.box.cox.powers
print.box.tidwell
print.durbin.watson
print.linear.hypothesis.mlm
print.outlier.test
print.spread.level.plot
prob.axis
qq.plot
qq.plot.default
qq.plot.glm
qq.plot.lm
qqp
Quartet
recode
reg.line
Robey
Sahlins
scatterplot
scatterplot.default
scatterplot.formula
scatterplot.matrix
scatterplot.matrix.default

Scatterplots with Boxplots

Description

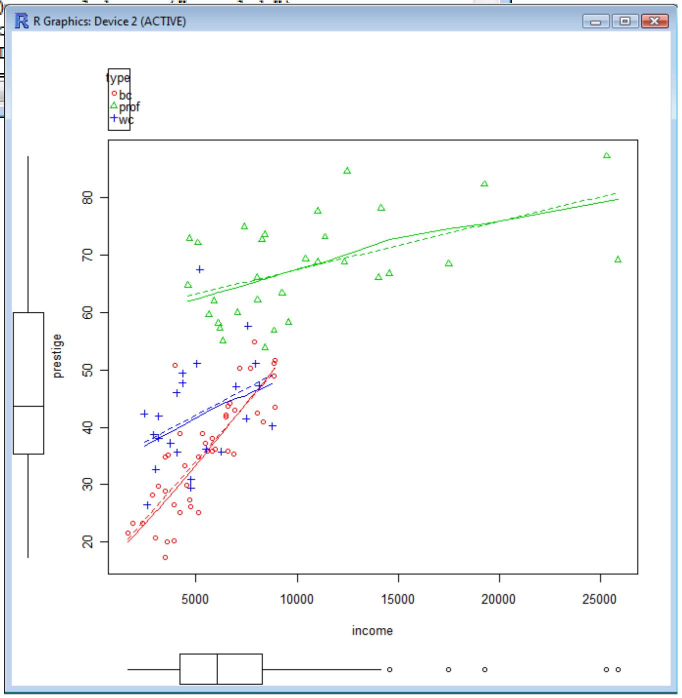
Makes fancy scatterplots, with boxplots in the margins, a lowess smooth, and a regression line; `sp` is an abbreviation for `scatterplot`.

Usage

```
scatterplot(x, ...)

## S3 method for class 'formula':
scatterplot(formula, data, xlab, ylab, legend.title, subset, labels=FALSE, ...)

## Default S3 method:
scatterplot(x, y, smooth=TRUE, span=0.5, reg.line=lm,
  boxplots="xy", xlab=deparse(substitute(x)), ylab=deparse(substitute(y)),
  las=par('las'), lwd=1, lwd.smooth=lwd, labels=FALSE, log="",
  jitter=list(), xlim=NULL, ylim=NULL,
  cex=par("cex"), cex.axis=par("cex.axis")
  cex.main=par("cex.main"), cex.sub=par("c
  groups=FALSE, by.groups=!(groups[1]==FAL
  ellipse=FALSE, levels=c(.5, .9), robust=
```

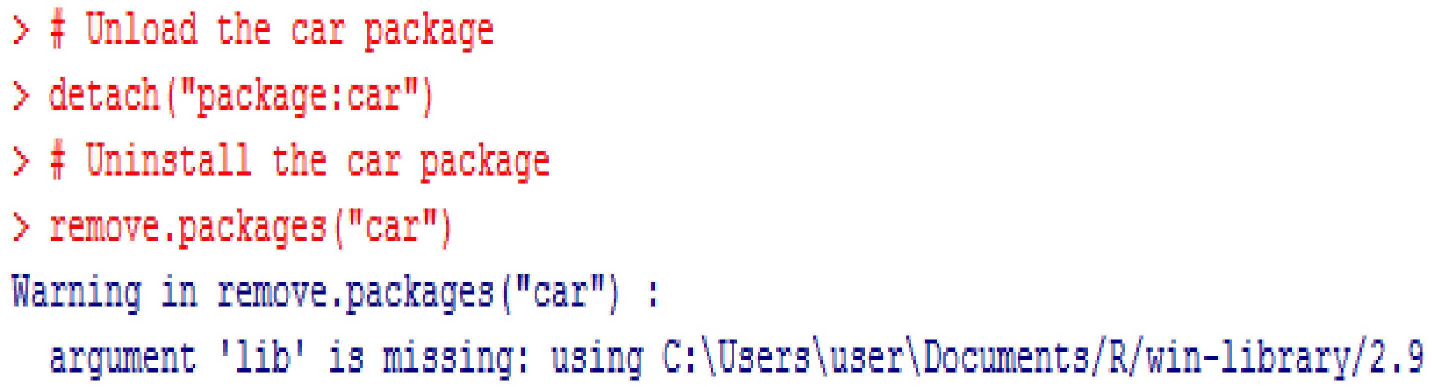
8. Run an example of scatterplot

scatterplot(prestige ~ income|type, data=Prestige, span=1)



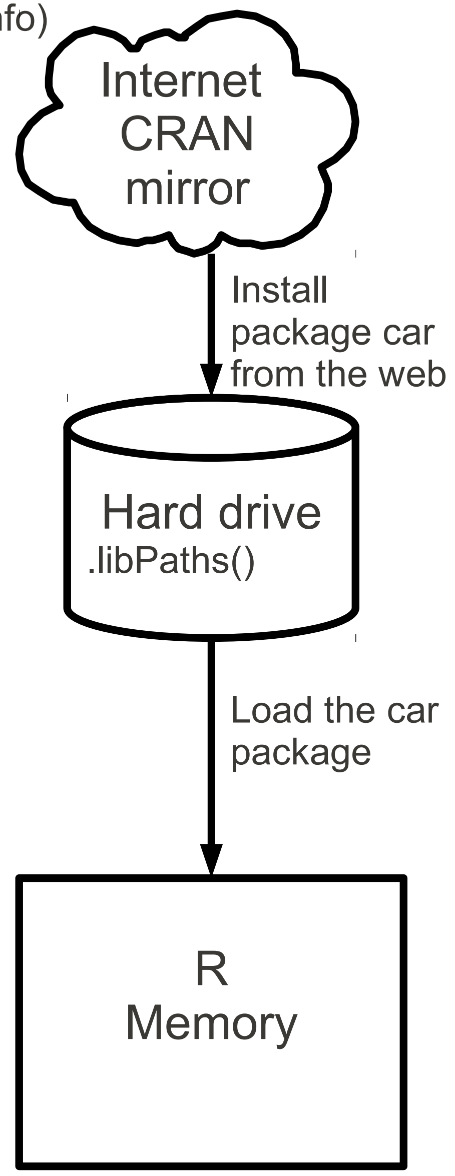
Packages

9. Unload the car package
10. Uninstall the car package

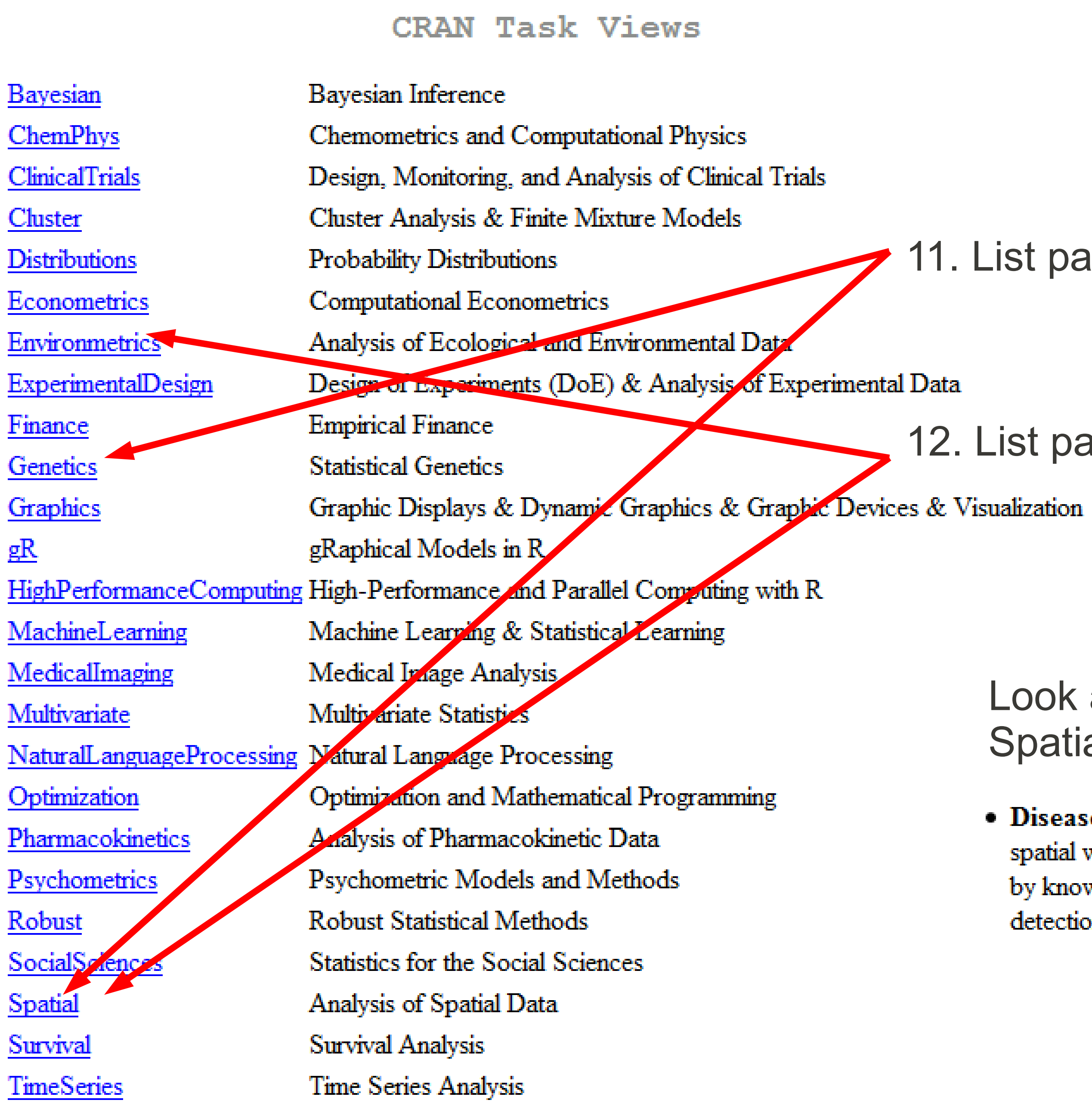


```
> # Unload the car package
> detach("package:car")
> # Uninstall the car package
> remove.packages("car")
Warning in remove.packages("car") :
  argument 'lib' is missing: using C:\Users\user\Documents/R/win-library/2.9
```

```
.libPaths() # get library location
dir(.libPaths()) # show files and directories on the library location

# 1. Is the car package loaded?
# search() is the "usual" command but it it also shows R objects (unnecessary info)
(.packages())
# 2. Is the car package installed?
# library() is the "usual" command but it it also shows the description (unnecessary info)
(.packages(all.available=TRUE))

# 3. Install package car from the web
install.packages("car", dependencies = TRUE)

# 2. Is the car package installed?
(.packages(all.available=TRUE))

dir(.libPaths()) # show files and directories on the library location

# 4. Load the car package
library("car")

# 1. Is the car package loaded?
(.packages())

# 5. Is there help for the car package?
library(help=car)

# 9. Unload the car package

# 1. Is the car package loaded?
(.packages())

# 10. Uninstall the car package

# 2. Is the car package installed?
(.packages(all.available=TRUE))
dir(.libPaths()) # show files and directories on the library location
```



Packages



CRAN Task Views

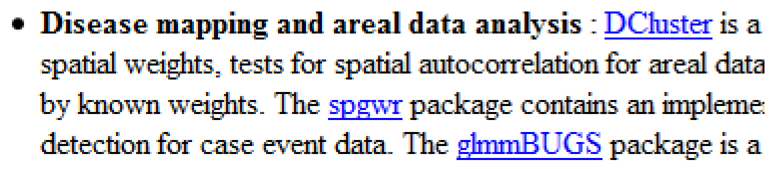
Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Design, Monitoring, and Analysis of Clinical Trials
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological 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 with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
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

11. List packages for epidemiology

Check out BioConductor!

12. List packages for environmental sciences

Look at the description of each view, Spatial has this:



- **Disease mapping and areal data analysis** : DCluster is a spatial weights, tests for spatial autocorrelation for areal data by known weights. The spgwr package contains an impleme detection for case event data. The glmmBUGS package is a

Packages

11. List packages for epidemiology

?? search the installed help files
For keywords “epidem”, “disease”, “illness”, etc...

R Site Search
http://search.r-project.org/

Rseek
http://www.rseek.org/

Read and maybe post a question on the Mailing List
R-help -- Main R Mailing List
https://stat.ethz.ch/mailman/listinfo/r-help

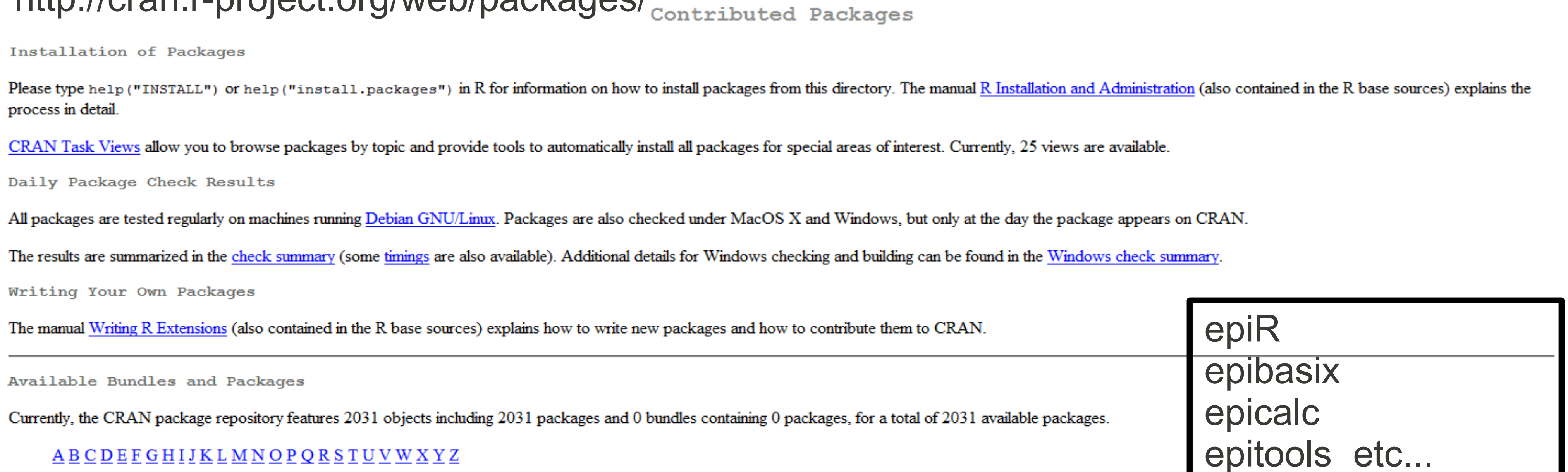
crantastic, a community site for R packages to search for, review and tag CRAN packages.
http://crantastic.org/

sos package
R related Search Engine
http://cran.r-project.org/web/packages/sos/

Stack Overflow a programming Q & A site
http://stackoverflow.com/

Packages

Contributed Packages
http://cran.r-project.org/web/packages/


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, 25 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.

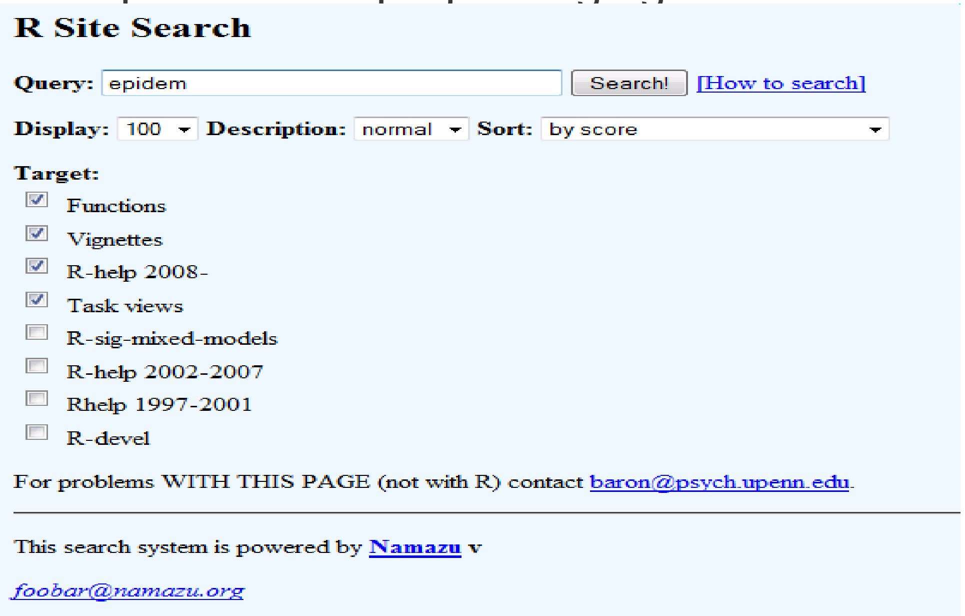
Available Bundles and Packages

Currently, the CRAN package repository features 2031 objects including 2031 packages and 0 bundles containing 0 packages, for a total of 2031 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

epiR
epibasix
epicalc
epitools etc...

R Site Search
http://search.r-project.org/cgi-bin/namazu.cgi


R Site Search

Query: epidem [Search!] [How to search]

Display: 100 **Description:** normal **Sort:** by score

Target:
- [✓] Functions
- [✓] Vignettes
- [✓] R-help 2008-
- [✓] Task views
- [ ] R-sig-mixed-models
- [ ] R-help 2002-2007
- [ ] Rhelp 1997-2001
- [ ] R-devel

For problems WITH THIS PAGE (not with R) contact baron@psych.upenn.edu.

This search system is powered by Namazu v

foobar@namazu.org

packages
References/to learn more:

The R book
Michael J. Crawley pp 4
2009 John Wiley & Sons Ltd

Basic statistics using R pp. 16
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics with R
Vincent Zoonekynd, pp 115
http://zoonek2.free.fr/UNIX/48_R/all.html

Introductory Statistics with R
Peter Dalgaard, pp 35
2010 Springer

Geographic Data Analysis
Pat Bartlein
http://geography.uoregon.edu/bartlein/courses/geog417/lectures/lec05.htm

Quick-R
Rob Kabacoff
http://www.statmethods.net/interface/packages.html

R console input

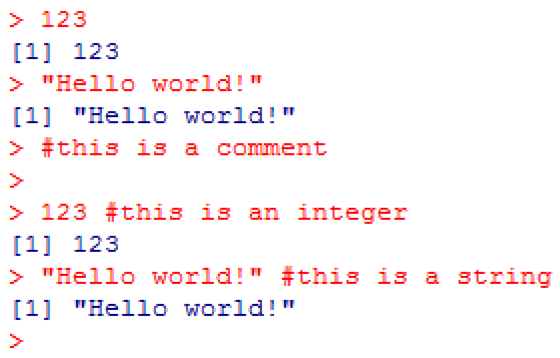
The console will accept R code, functions, expressions, variables and data.

Numbers can be positive or negative, and with a decimal part.

Strings are delimited by double quotes. Strings are text, character data.

Comments are marked with the # sign. Everything after a comment is ignored. Comments are useful for explaining the code, otherwise it would be hard trying to guess or remember what the code does.

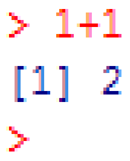
Examples:



```
> 123
[1] 123
> "Hello world!"
[1] "Hello world!"
> #this is a comment
>
> 123 #this is an integer
[1] 123
> "Hello world!" #this is a string
[1] "Hello world!"
>
```

Using R as a calculator

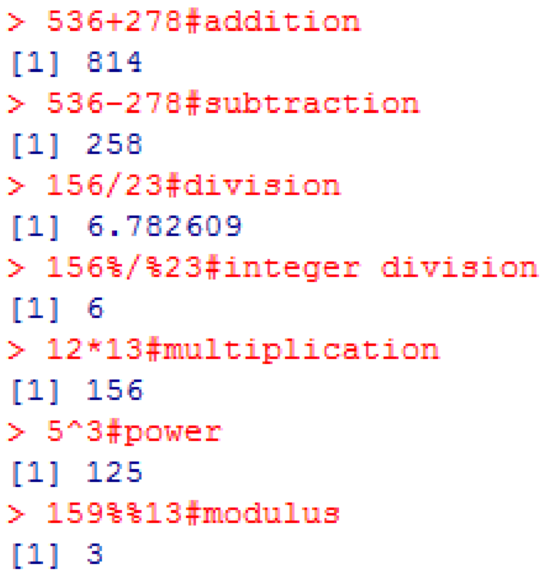
R can execute expressions directly from the console, like a calculator



```
> 1+1
[1] 2
>
```

Type 1+1 and enter

Mathematical operators

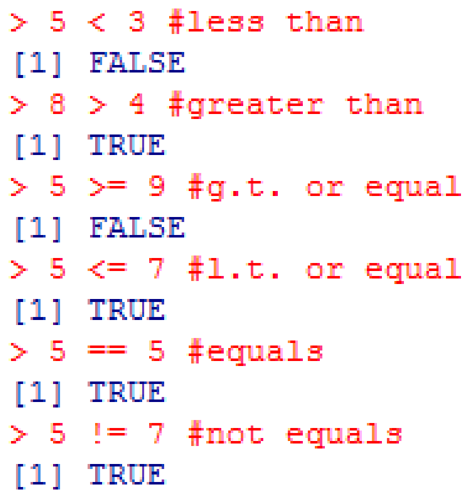


```
> 536+278#addition
[1] 814
> 536-278#subtraction
[1] 258
> 156/23#division
[1] 6.782609
> 156%/%23#integer division
[1] 6
> 12*13#multiplication
[1] 156
> 5^3#power
[1] 125
> 159%%13#modulus
[1] 3
```

Using R as a calculator

Comparison operators

The logical values are TRUE, FALSE and NA for missing values.



```
> 5 < 3 #less than
[1] FALSE
> 8 > 4 #greater than
[1] TRUE
> 5 >= 9 #g.t. or equal
[1] FALSE
> 5 <= 7 #l.t. or equal
[1] TRUE
> 5 == 5 #equals
[1] TRUE
> 5 != 7 #not equals
[1] TRUE
```

Using R as a calculator

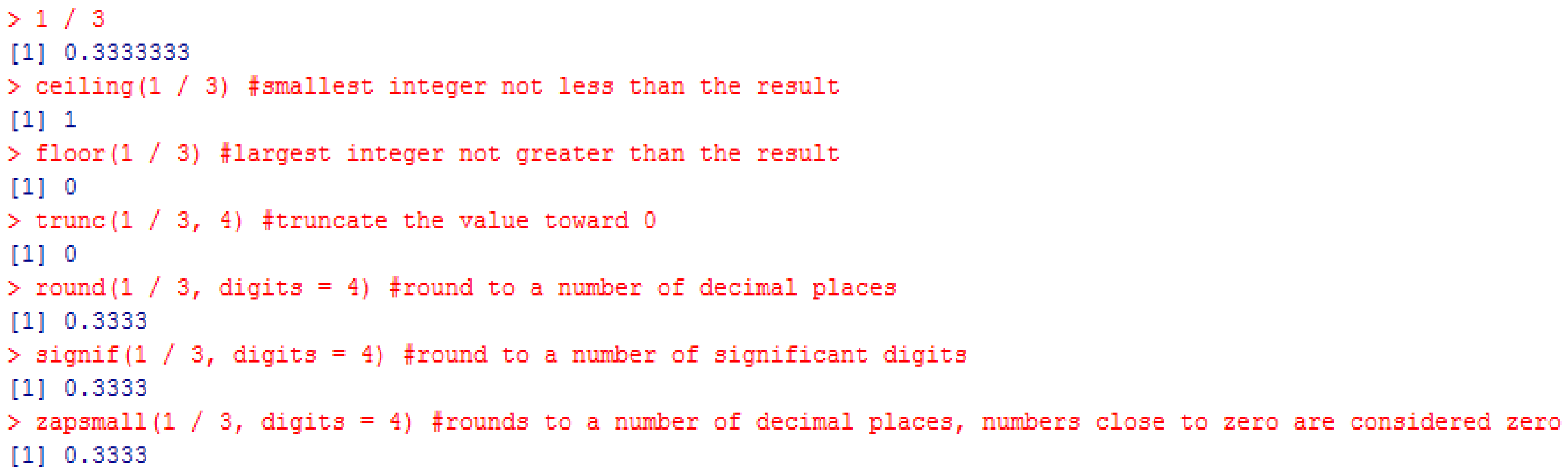
Logical operators

The logical values are TRUE, FALSE and NA for missing values.

```
> !FALSE # logical negation
[1] TRUE
> TRUE & FALSE # logical AND
[1] FALSE
> TRUE | FALSE # logical OR
[1] TRUE
> xor(TRUE, FALSE) # logical eXclusive OR
[1] TRUE
>
> TRUE && FALSE # logical AND
[1] FALSE
> TRUE || FALSE # logical OR
[1] TRUE
>
> c(T,F,F) & c(F,T,F) # logical AND
[1] FALSE FALSE FALSE
> c(T,F,F) | c(F,T,F) # logical OR
[1]  TRUE  TRUE FALSE
> c(T,F,F) && c(F,T,F) # logical AND
[1] FALSE
> c(T,F,F) || c(F,T,F) # logical OR
[1] TRUE
```

Using R as a calculator

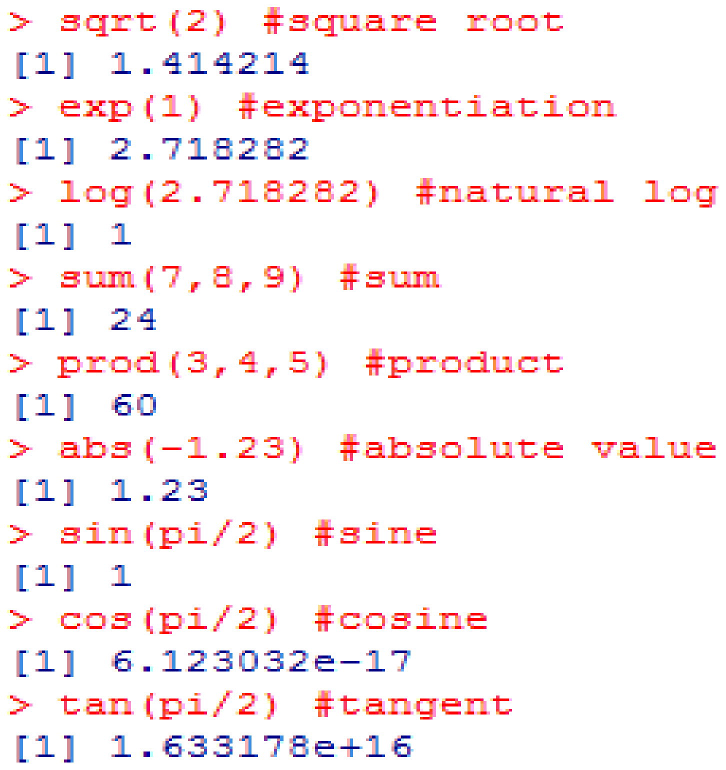
Rounding functions



```
> 1 / 3
[1] 0.3333333
> ceiling(1 / 3) #smallest integer not less than the result
[1] 1
> floor(1 / 3) #largest integer not greater than the result
[1] 0
> trunc(1 / 3, 4) #truncate the value toward 0
[1] 0
> round(1 / 3, digits = 4) #round to a number of decimal places
[1] 0.3333
> signif(1 / 3, digits = 4) #round to a number of significant digits
[1] 0.3333
> zapsmall(1 / 3, digits = 4) #rounds to a number of decimal places, numbers close to zero are considered zero
[1] 0.3333
```

Using R as a calculator

Mathematical functions

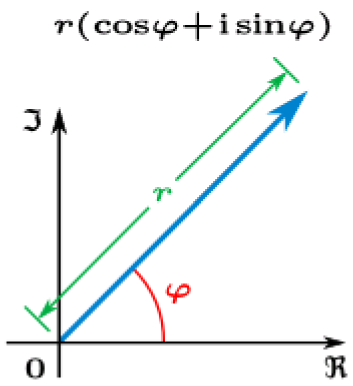


```
> sqrt(2) #square root
[1] 1.414214
> exp(1) #exponentiation
[1] 2.718282
> log(2.718282) #natural log
[1] 1
> sum(7,8,9) #sum
[1] 24
> prod(3,4,5) #product
[1] 60
> abs(-1.23) #absolute value
[1] 1.23
> sin(pi/2) #sine
[1] 1
> cos(pi/2) #cosine
[1] 6.123032e-17
> tan(pi/2) #tangent
[1] 1.633178e+16
```

Using R as a calculator

Complex functions

```
> mycomplexvar<-3+5i # a variable with a complex value
> mycomplexvar
[1] 3+5i
> Re(mycomplexvar) # real part
[1] 3
> Im(mycomplexvar) # imaginary part
[1] 5
> Conj(mycomplexvar) # complex conjugate
[1] 3-5i
> Mod(mycomplexvar) # complex modulus
[1] 5.830952
> Arg(mycomplexvar) # complex argument
[1] 1.030377
```



argument $\varphi = \pm\arctan\frac{y}{x}$

modulus r = $|x+iy| = \sqrt{x^2+y^2}.$

Using R as a calculator

R Built-in Constants

Constants that come with the R base package.

LETTERS: the 26 upper-case letters of the Roman alphabet;
letters: the 26 lower-case letters of the Roman alphabet;
month.abb: the three-letter abbreviations for the English month names;
month.name: the English names for the months of the year;
pi: the ratio of the circumference of a circle to its diameter.

```
pi * 10 # the perimeter of a circumference of diameter 10

# months in English
month.name
# months in your current locale
format(ISOdate(2009, 1:12, 1), "%B")
format(ISOdate(2009, 1:12, 1), "%b")
```

R as calculator
References/to learn more:

The R book
Michael J. Crawley pp 9
2010 John Wiley & Sons Ltd

Basic statistics using R pp. 35
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics: an introduction using R
Michael J. Crawley pp 281
2008 John Wiley & Sons Ltd

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/6bfdf37a91c966902de8395629e9fef6

Introductory Statistics with R
Peter Dalgaard, pp 3
2011 Springer

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/calculator.r

R Variables

Assigning values to objects = or <- or ->

```
> myvar <- 123 # to assign value 123 to variable "myvar"
> print(myvar) # display the variable
[1] 123
> #or
> myvar
[1] 123
> x = 5
> y <- 6
> 7 -> z
> x
[1] 5
> y
[1] 6
> z
[1] 7
> (myvar2 <- 456) # assign and display
[1] 456
```

Multiple assignments

```
> a <- b <- 55
> a
[1] 55
> b
[1] 55
>
> x <- (y <- c(5, 14,234))*2
> x;y
[1]  10  28 468
[1]   5  14 234
```

Multiple commands in one line

R Variables

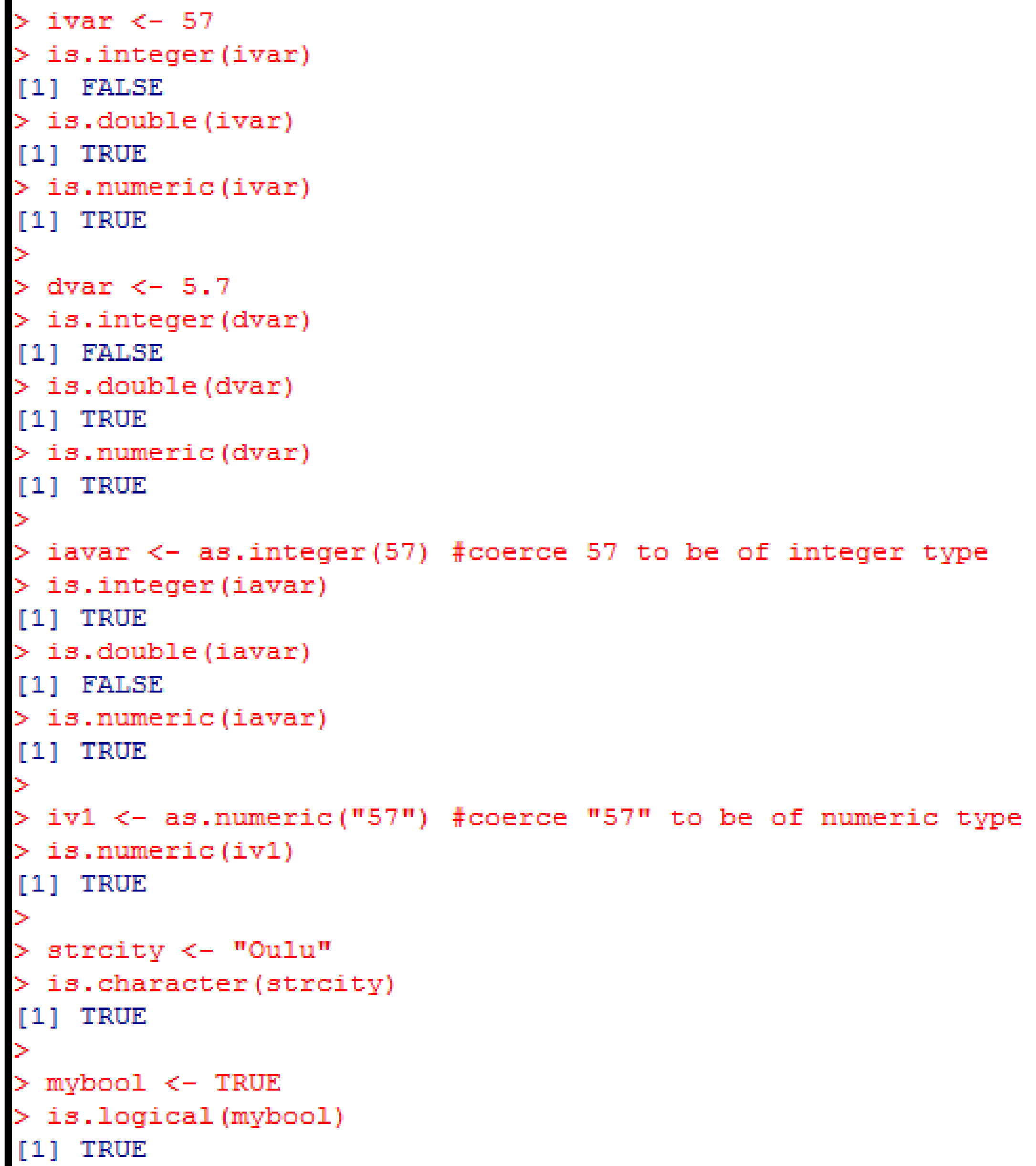
3 basic types of variables

- Numeric
- Character
- Boolean {true, false}

Functions to test an object's data type

is.integer, is.double, is.numeric, is.character and is.logical

as.integer is used to pass data to C or Fortran code

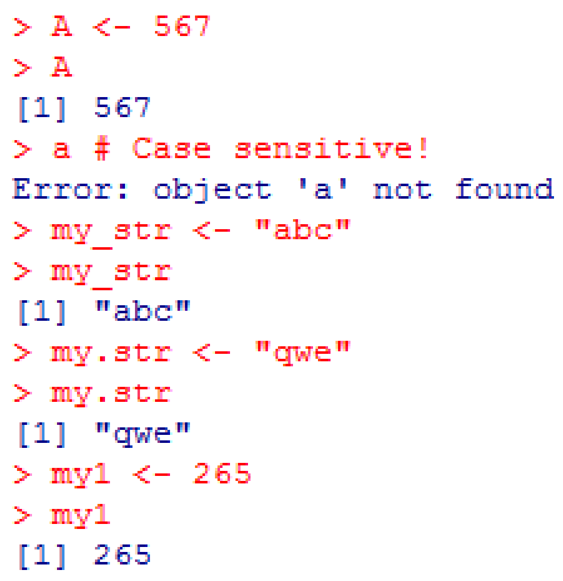


```
> ivar <- 57
> is.integer(ivar)
[1] FALSE
> is.double(ivar)
[1] TRUE
> is.numeric(ivar)
[1] TRUE
>
> dvar <- 5.7
> is.integer(dvar)
[1] FALSE
> is.double(dvar)
[1] TRUE
> is.numeric(dvar)
[1] TRUE
>
> iavar <- as.integer(57) #coerce 57 to be of integer type
> is.integer(iavar)
[1] TRUE
> is.double(iavar)
[1] FALSE
> is.numeric(iavar)
[1] TRUE
>
> iv1 <- as.numeric("57") #coerce "57" to be of numeric type
> is.numeric(iv1)
[1] TRUE
>
> strcity <- "Oulu"
> is.character(strcity)
[1] TRUE
>
> mybool <- TRUE
> is.logical(mybool)
[1] TRUE
```

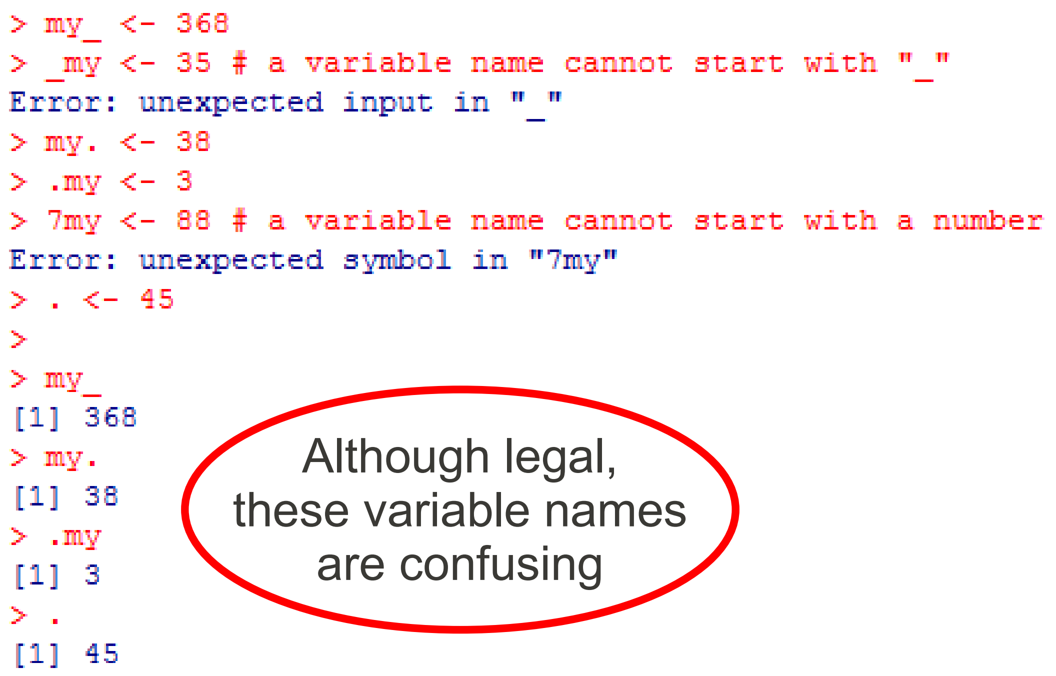
R Variables

Variable names

•Case sensitive
•R names depend on the operating system and country within which R is being run (technically on the locale settings)
•All alphanumeric symbols are allowed (and in some countries this includes accented letters) plus ‘.’ and ‘_’, with the restriction that a name must start with ‘.’ or a letter, and if it starts with ‘.’ the second character must not be a digit
•For portable R code (including that to be used in R packages) use only A–Za–z0–9



```
> A <- 567
> A
[1] 567
> a # Case sensitive!
Error: object 'a' not found
> my_str <- "abc"
> my_str
[1] "abc"
> my.str <- "qwe"
> my.str
[1] "qwe"
> my1 <- 265
> my1
[1] 265
```



```
> my_ <- 368
> _my <- 35 # a variable name cannot start with "_"
Error: unexpected input in "_"
> my. <- 38
> .my <- 3
> 7my <- 88 # a variable name cannot start with a number
Error: unexpected symbol in "7my"
> . <- 45
>
> my_
[1] 368
> my.
[1] 38
> .my
[1] 3
> .
[1] 45
```

Although legal,
these variable names
are confusing

R Variables

Reserved Words in R

These words should not be used as variable names or function names, to avoid parsing errors.

Reserved words:

if else repeat while function for in next break

TRUE   FALSE   NULL   Inf   NaN   NA

NA_integer_   NA_real_   NA_complex_   NA_character_

R Variables

Not Available / "Missing" Values

NA is a missing value indicator.

"Missing" Values are common in real world data because of no answers to surveys or missing data from sensors readings.

is.na() returns TRUE for missing elements
is.na() <- sets elements to NA

```
> x <- 5
> x
[1] 5
> is.na(x)
[1] FALSE
> y <- NA
> y
[1] NA
> is.na(y)
[1] TRUE
```

R Variables

Not Available / "Missing" Values

```
> z <- c(3,5,NA,6,7,8) # vector
> z
[1]  3  5 NA  6  7  8
> is.na(z) # which elements are NA
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
> is.na(z) <- c(1,5) # turn elements at position 1 and position 5 to NA
> z
[1] NA  5 NA  6 NA  8
```

```
> # math operators * + - / will return NA
> 5 * NA
[1] NA

> # comparison operators < <= > >= == != will return NA
> c(5, 5, NA) == c(5, NA, NA)
[1] TRUE   NA   NA
```

R Variables

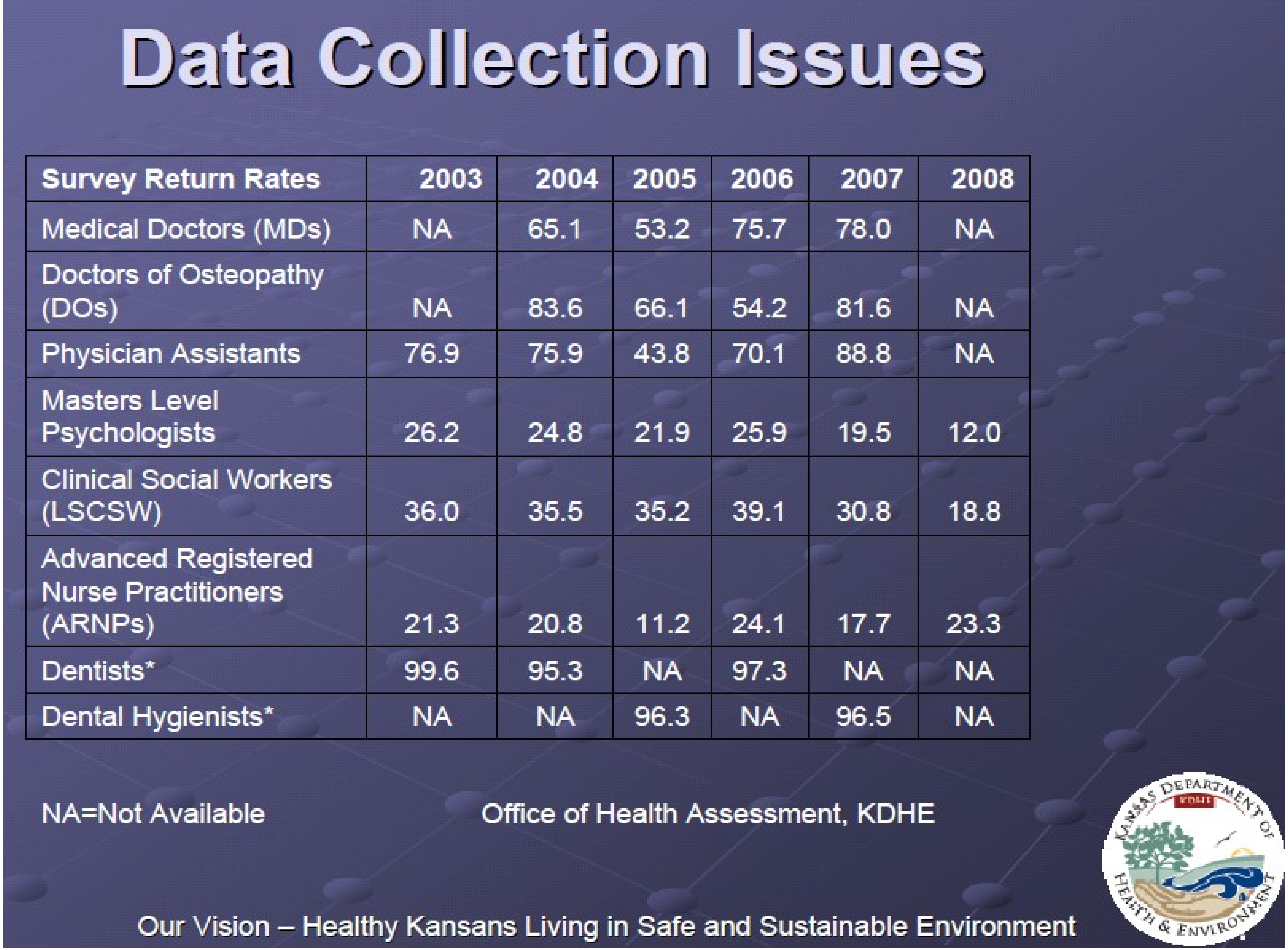
Not Available / "Missing" Values

```
> # NA is "undetermined" for logical expressions
> c(T, F) & c(NA, NA) # FALSE AND whatever is FALSE
[1]    NA FALSE
> c(T, F) | c(NA, NA) # TRUE OR whatever is TRUE
[1] TRUE   NA
> xor(NA,T)
[1] NA
```

```
> myvec <- c(7,4,NA,2,65)
> mean(myvec) # this will return NA
[1] NA
> mean(myvec, na.rm=T) # ignoring NA in a calculation
[1] 19.5
> na.omit(myvec) # omitting NA
[1]  7  4  2 65
attr(,"na.action")
[1] 3
attr(,"class")
[1] "omit"
```

R Variables

```
> x <- c(7, 6, NA, NA, 5)
> x[!is.na(x)] # get the data except the NAs
[1] 7 6 5
> na.omit(x) # get the data except the NAs, proper way
[1] 7 6 5
attr(,"na.action")
[1] 3 4
attr(,"class")
[1] "omit"
> mean(x)          # returns NA
[1] NA
> mean(x, na.rm=TRUE) # returns 6
[1] 6
> x[is.na(x)] <- 0 # replace NAs with 0
> x
[1] 7 6 0 0 5
```



Data Collection Issues

Survey Return Rates	2003	2004	2005	2006	2007	2008
Medical Doctors (MDs)	NA	65.1	53.2	75.7	78.0	NA
Doctors of Osteopathy (DOs)	NA	83.6	66.1	54.2	81.6	NA
Physician Assistants	76.9	75.9	43.8	70.1	88.8	NA
Masters Level Psychologists	26.2	24.8	21.9	25.9	19.5	12.0
Clinical Social Workers (LSCSW)	36.0	35.5	35.2	39.1	30.8	18.8
Advanced Registered Nurse Practitioners (ARNPs)	21.3	20.8	11.2	24.1	17.7	23.3
Dentists*	99.6	95.3	NA	97.3	NA	NA
Dental Hygienists*	NA	NA	96.3	NA	96.5	NA

NA=Not Available

Office of Health Assessment, KDHE

Our Vision – Healthy Kansans Living in Safe and Sustainable Environment

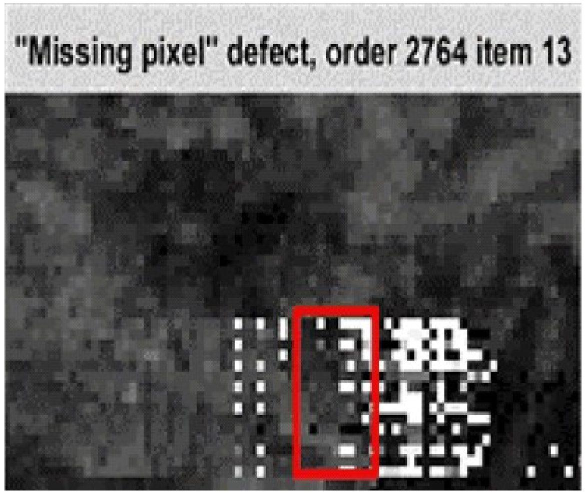
http://www.khpa.ks.gov/data_consortium/Docs/022009/WorkForceSurvey.pdf

Missing image data LANDSAT 5 - 7

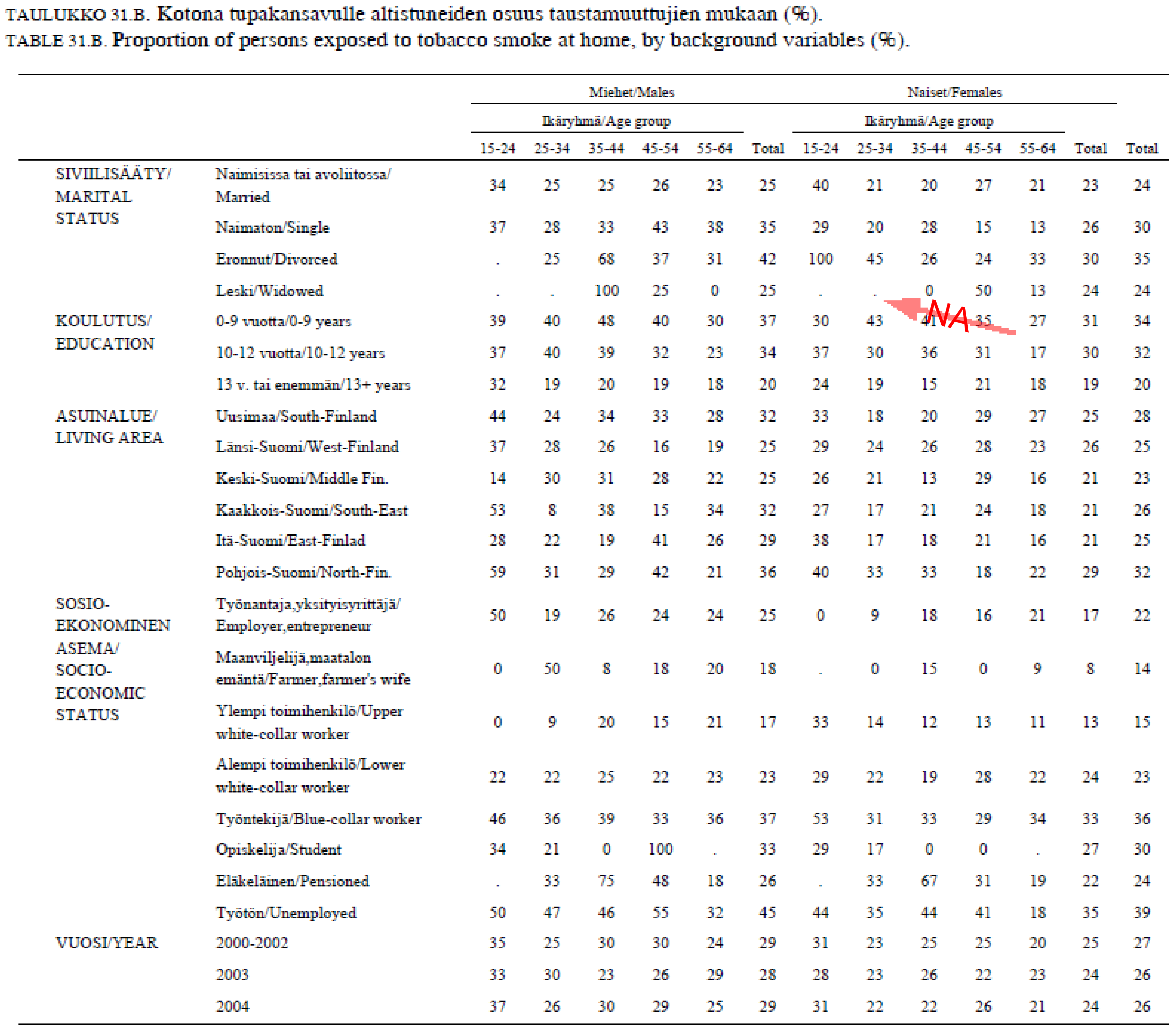
Anomalies description

Missing image data anomaly may be considered under different aspects. The most frequently case of
missing data may be called “missing pixels”. Usually, the “missing pixels” anomaly is correlated with
others anomalies (shifted swath – speckle - missing swath). Details are also provided about wrong or
missing auxiliary data that implies swath misalignment (See also *Anomaly slip 02*). This section describes
the following anomalies related to missing image data:

- Missing pixels.
- Missing pixels – shifted swath.
- Missing pixels – missing swath.
- Missing pixels – speckle.
- Corrupted Mirror Scan Correction Data (MSCD) – shifted swath.



http://earth.esa.int/pub/ESA_DOC/landsat_product_anomalies/GAEL-P157-SLP-001-03-01.pdf

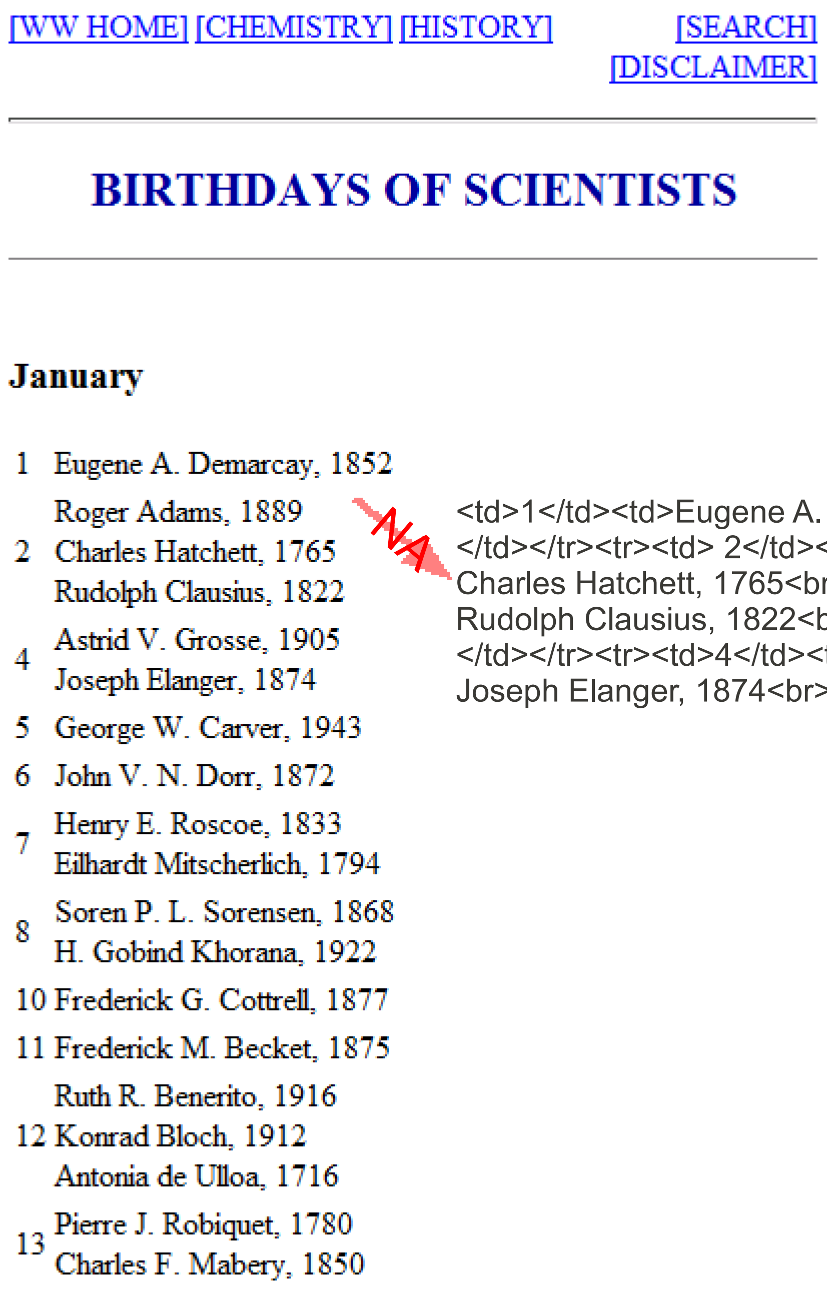


TAULUKKO 31.B. Kotona tupakansavulle altistuneiden osuus taustamuuttujien mukaan (%).
TABLE 31.B. Proportion of persons exposed to tobacco smoke at home, by background variables (%).

		Miehet/Males						Naiset/Females						
		Ikäryhmä/Age group						Ikäryhmä/Age group						
		15-24	25-34	35-44	45-54	55-64	Total	15-24	25-34	35-44	45-54	55-64	Total	Total
SIVIILISÄÄTY/ MARITAL STATUS	Naimisissa tai avoliitossa/ Married	34	25	25	26	23	25	40	21	20	27	21	23	24
	Naimaton/Single	37	28	33	43	38	35	29	20	28	15	13	26	30
	Eronnut/Divorced	.	25	68	37	31	42	100	45	26	24	33	30	35
	Leski/Widowed	.	.	100	25	0	25	.	.	0	50	13	24	24
KOULUTUS/ EDUCATION	0-9 vuotta/0-9 years	39	40	48	40	30	37	30	43	NA 41	35	27	31	34
	10-12 vuotta/10-12 years	37	40	39	32	23	34	37	30	36	31	17	30	32
	13 v. tai enemmän/13+ years	32	19	20	19	18	20	24	19	15	21	18	19	20
ASUINALUE/ LIVING AREA	Uusimaa/South-Finland	44	24	34	33	28	32	33	18	20	29	27	25	28
	Länsi-Suomi/West-Finland	37	28	26	16	19	25	29	24	26	28	23	26	25
	Keski-Suomi/Middle Fin.	14	30	31	28	22	25	26	21	13	29	16	21	23
	Kaakkois-Suomi/South-East	53	8	38	15	34	32	27	17	21	24	18	21	26
	Itä-Suomi/East-Finlad	28	22	19	41	26	29	38	17	18	21	16	21	25
	Pohjois-Suomi/North-Fin.	59	31	29	42	21	36	40	33	33	18	22	29	32
SOSIO-EKONOMINEN ASEMA/ SOCIO-ECONOMIC STATUS	Työnantaja,yksityisyrittäjä/ Employer,entrepreneur	50	19	26	24	24	25	0	9	18	16	21	17	22
	Maanviljelijä,maatalon emäntä/Farmer,farmer's wife	0	50	8	18	20	18	.	0	15	0	9	8	14
	Ylempi toimihenkilö/Upper white-collar worker	0	9	20	15	21	17	33	14	12	13	11	13	15
	Alempi toimihenkilö/Lower white-collar worker	22	22	25	22	23	23	29	22	19	28	22	24	23
	Työntekijä/Blue-collar worker	46	36	39	33	36	37	53	31	33	29	34	33	36
	Opiskelija/Student	34	21	0	100	.	33	29	17	0	0	.	27	30
	Eläkeläinen/Pensioned	.	33	75	48	18	26	.	33	67	31	19	22	24
	Työtön/Unemployed	50	47	46	55	32	45	44	35	44	41	18	35	39
VUOSI/YEAR	2000-2002	35	25	30	30	24	29	31	23	25	25	20	25	27
	2003	33	30	23	26	29	28	28	23	26	22	23	24	26
	2004	37	26	30	29	25	29	31	22	22	26	21	24	26

http://www.ktl.fi/attachments/suomi/julkaisut/julkaisusarja_b/2004b13.pdf

Using data from a website:



[WW HOME] [CHEMISTRY] [HISTORY] [SEARCH] [DISCLAIMER]

BIRTHDAYS OF SCIENTISTS

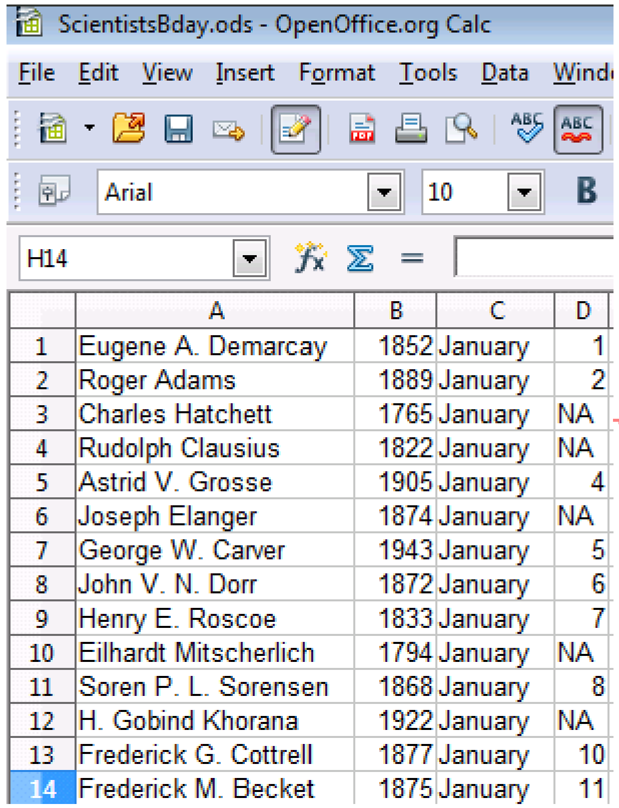
January

1	Eugene A. Demarcay, 1852
	Roger Adams, 1889
2	Charles Hatchett, 1765
	Rudolph Clausius, 1822
4	Astrid V. Grosse, 1905
	Joseph Elanger, 1874
5	George W. Carver, 1943
6	John V. N. Dorr, 1872
7	Henry E. Roscoe, 1833
	Eilhardt Mitscherlich, 1794
8	Soren P. L. Sorensen, 1868
	H. Gobind Khorana, 1922
10	Frederick G. Cottrell, 1877
11	Frederick M. Becket, 1875
	Ruth R. Benerito, 1916
12	Konrad Bloch, 1912
	Antonia de Ulloa, 1716
13	Pierre J. Robiquet, 1780
	Charles F. Mabery, 1850

NA

```
<td>1</td><td>Eugene A. Demarcay, 1852<br>
</td></tr><tr><td> 2</td><td>Roger Adams, 1889<br>
Charles Hatchett, 1765<br>
Rudolph Clausius, 1822<br>
</td></tr><tr><td>4</td><td>Astrid V. Grosse, 1905<br>
Joseph Elanger, 1874<br>
```

The data is clean and organized on a spreadsheet:



ScientistsBday.ods - OpenOffice.org Calc

File Edit View Insert Format Tools Data Wind

Arial 10

H14

	A	B	C	D
1	Eugene A. Demarcay	1852	January	1
2	Roger Adams	1889	January	2
3	Charles Hatchett	1765	January	NA
4	Rudolph Clausius	1822	January	NA
5	Astrid V. Grosse	1905	January	4
6	Joseph Elanger	1874	January	NA
7	George W. Carver	1943	January	5
8	John V. N. Dorr	1872	January	6
9	Henry E. Roscoe	1833	January	7
10	Eilhardt Mitscherlich	1794	January	NA
11	Soren P. L. Sorensen	1868	January	8
12	H. Gobind Khorana	1922	January	NA
13	Frederick G. Cottrell	1877	January	10
14	Frederick M. Becket	1875	January	11

NA

http://www.woodrow.org/teachers/ci/1992/activities/birthdays.html

R Variables

Finite Infinite and NaN Numbers

Infinite numbers are the result of finite numbers divided by zero
NaN (Not a Number) are the result of zero divided by zero

Inf ∞
-Inf -∞
NaN undetermined

is.finite() returns TRUE for a finite number
is.infinite() returns TRUE for an infinite number
is.nan() returns TRUE for a NaN

R Variables

```
a <- 1/2
a
is.finite(a)
is.infinite(a)
is.nan(a)

b <- 1/0
b
is.finite(b)
is.infinite(b)
is.nan(b)

c <- 0/0
c
is.finite(c)
is.infinite(c)
is.nan(c)
```

```
> a <- 1/2
> a
[1] 0.5
> is.finite(a)
[1] TRUE
> is.infinite(a)
[1] FALSE
> is.nan(a)
[1] FALSE
>
> b <- 1/0
> b
[1] Inf
> is.finite(b)
[1] FALSE
> is.infinite(b)
[1] TRUE
> is.nan(b)
[1] FALSE
>
> c <- 0/0
> c
[1] NaN
> is.finite(c)
[1] FALSE
> is.infinite(c)
[1] FALSE
> is.nan(c)
[1] TRUE
```

R Variables

Getting info from objects

class() returns the class attribute or the implicit class of this object

is() returns all the super-classes of this object's class

mode() to get or set the type or storage mode of an object

str() to compactly display the internal structure of an R object

length() to get or set the length of objects

dim() to retrieve or set the dimension of an object

nchar() to get or set the length of strings

object.size() to get an estimate of the memory used to store an R object

Common source of confusion:

class() vs is() vs mode()

length() vs dim() vs nchar()

R Variables

Type on R Editor:

```
Myint <- 567
is(myint)
Myreal <- 8.83
is(myreal)
mycomplex <- 34-7i
is(mycomplex)
mystring <- "quartz"
is(mystring)
myvector_i <- c(6,5,4)
is(myvector_i)
myvector_s <- c("a","b","c")
is(myvector_s)
mymatrix <- matrix(5,2,3)
is(mymatrix)
```

R Variables

is() returns all the super-classes of this object's class

```
> myint <- 567
> is(myint)
[1] "numeric" "vector"
> myreal <- 8.83
> is(myreal)
[1] "numeric" "vector"
> mycomplex <- 34-7i
> is(mycomplex)
[1] "complex" "vector"
>
> mystring <- "quartz"
> is(mystring)
[1] "character"          "vector"             "data.frameRowLabels"
> myvector_i <- c(6,5,4)
> is(myvector_i)
[1] "numeric" "vector"
>
> myvector_s <- c("a","b","c")
> is(myvector_s)
[1] "character"          "vector"             "data.frameRowLabels"
> mymatrix <- matrix(5,2,3)
> is(mymatrix)
[1] "matrix"    "array"     "structure" "vector"
```

All objects are vectors!

Scalars are vectors of length 1

R Variables

On R Editor, go to Edit/Replace and replace “is” with “class”

```
myint <- 567
class(myint)
myreal <- 8.83
class(myreal)
mycomplex <- 34-7i
class(mycomplex)
mystring <- "quartz"
class(mystring)
myvector_i <- c(6,5,4)
class(myvector_i)
myvector_s <- c("a","b","c")
class(myvector_s)
mymatrix <- matrix(5,2,3)
class(mymatrix)
```

Edit/Clear console to clear the previous calculations from the R Console

R Variables

```
> myint <- 567
> class(myint)
[1] "numeric"
> myreal <- 8.83
> class(myreal)
[1] "numeric"
> mycomplex <- 34-7i
> class(mycomplex)
[1] "complex"
> mystring <- "quartz"
> class(mystring)
[1] "character"
> myvector_i <- c(6,5,4)
> class(myvector_i)
[1] "numeric"
> myvector_s <- c("a","b","c")
> class(myvector_s)
[1] "character"
> mymatrix <- matrix(5,2,3)
> class(mymatrix)
[1] "matrix"
```

class() returns the class attribute or the implicit class of this object

This is the first class returned by is()

class(myvar) "class 1"

is(myvar) "class 1" "class 2" "class 3" ...

R Variables

On R Editor, go to Edit/Replace and replace “class” with “mode”

```
myint <- 567
mode(myint)
myreal <- 8.83
mode(myreal)
mycomplex <- 34-7i
mode(mycomplex)
mystring <- "quartz"
mode(mystring)
myvector_i <- c(6,5,4)
mode(myvector_i)
myvector_s <- c("a","b","c")
mode(myvector_s)
mymatrix <- matrix(5,2,3)
mode(mymatrix)
```

R Variables

mode() to get or set the type or storage mode of an object

```
> myint <- 567
> mode(myint)
[1] "numeric"
> myreal <- 8.83
> mode(myreal)
[1] "numeric"
> mycomplex <- 34-7i
> mode(mycomplex)
[1] "complex"
> mystring <- "quartz"
> mode(mystring)
[1] "character"
> myvector_i <- c(6,5,4)
> mode(myvector_i)
[1] "numeric"
> myvector_s <- c("a","b","c")
> mode(myvector_s)
[1] "character"
> mymatrix <- matrix(5,2,3)
> mode(mymatrix)
[1] "numeric"
```

The only difference is with matrix, let's try a data frame:

```
> mydataf <- data.frame(1,2,3)
> mode(mydataf)
[1] "list"
> class(mydataf)
[1] "data.frame"
> mydataf <- data.frame("a","b","c")
> mode(mydataf)
[1] "list"
> class(mydataf)
[1] "data.frame"
```

By default, a matrix is stored as numeric data in memory and a data frame as list data in memory. This can be changed, for achieving better performance or for compatibility.

R Variables

On R Editor, go to Edit/Replace and replace “mode” with “length”, “dim” and “nchar”

myint <- 567	myint <- 567	myint <- 567
length(myint)	dim(myint)	nchar(myint)
myreal <- 8.83	myreal <- 8.83	myreal <- 8.83
length(myreal)	dim(myreal)	nchar(myreal)
mycomplex <- 34-7i	mycomplex <- 34-7i	mycomplex <- 34-7i
length(mycomplex)	dim(mycomplex)	nchar(mycomplex)
mystring <- "quartz"	mystring <- "quartz"	mystring <- "quartz"
length(mystring)	dim(mystring)	nchar(mystring)
myvector_i <- c(6,5,4)	myvector_i <- c(6,5,4)	myvector_i <- c(6,5,4)
length(myvector_i)	dim(myvector_i)	nchar(myvector_i)
myvector_s <- c("a","b","c")	myvector_s <- c("a","b","c")	myvector_s <- c("a","b","c")
length(myvector_s)	dim(myvector_s)	nchar(myvector_s)
mymatrix <- matrix(5,2,3)	mymatrix <- matrix(5,2,3)	mymatrix <- matrix(5,2,3)
length(mymatrix)	dim(mymatrix)	nchar(mymatrix)

R Variables

Length() is the number of elements, dim are the dimensions, nchar is the number of characters

```
> myint <- 567
> length(myint)
[1] 1
> myreal <- 8.83
> length(myreal)
[1] 1
> mycomplex <- 34-7i
> length(mycomplex)
[1] 1
> mystring <- "quartz"
> length(mystring)
[1] 1
> myvector_i <- c(6,5,4)
> length(myvector_i)
[1] 3
> myvector_s <- c("a","b","c")
> length(myvector_s)
[1] 3
> mymatrix <- matrix(5,2,3)
> length(mymatrix)
[1] 6
```

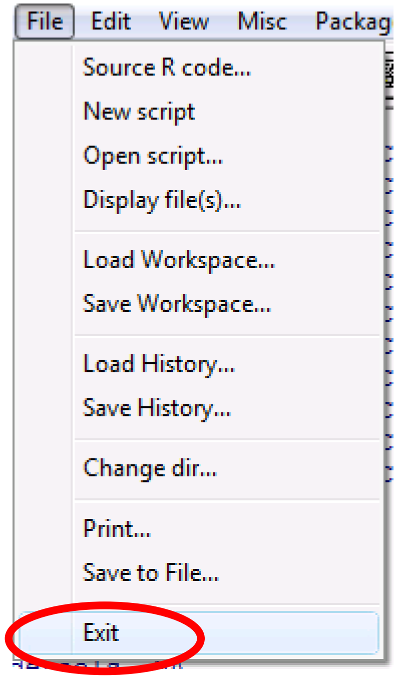
```
> myint <- 567
> dim(myint)
NULL
> myreal <- 8.83
> dim(myreal)
NULL
> mycomplex <- 34-7i
> dim(mycomplex)
NULL
> mystring <- "quartz"
> dim(mystring)
NULL
> myvector_i <- c(6,5,4)
> dim(myvector_i)
NULL
> myvector_s <- c("a","b","c")
> dim(myvector_s)
NULL
> mymatrix <- matrix(5,2,3)
> dim(mymatrix)
[1] 2 3
```

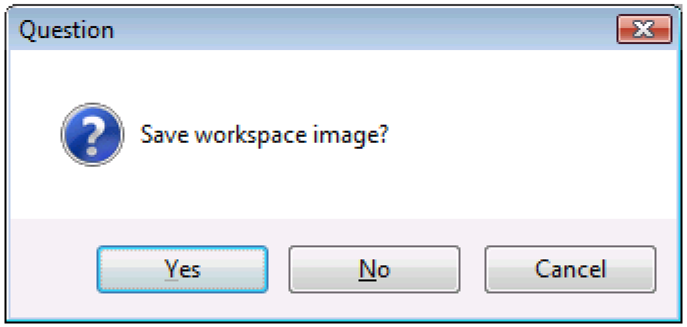
```
> myint <- 567
> nchar(myint)
[1] 3
> myreal <- 8.83
> nchar(myreal)
[1] 4
> mycomplex <- 34-7i
> nchar(mycomplex)
[1] 5
> mystring <- "quartz"
> nchar(mystring)
[1] 6
> myvector_i <- c(6,5,4)
> nchar(myvector_i)
[1] 1 1 1
> myvector_s <- c("a","b","c")
> nchar(myvector_s)
[1] 1 1 1
> mymatrix <- matrix(5,2,3)
> nchar(mymatrix)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

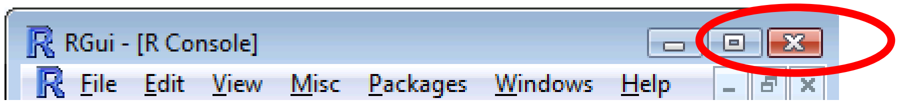
Quitting R

Command q()

Or File/Exit or close the editor window (on Windows)







save workspace image?

Yes will save all the objects from memory to a file .Rdata and it wil also save all the commands typed during the session to a file .Rhistory

Both files are saved on user\documents

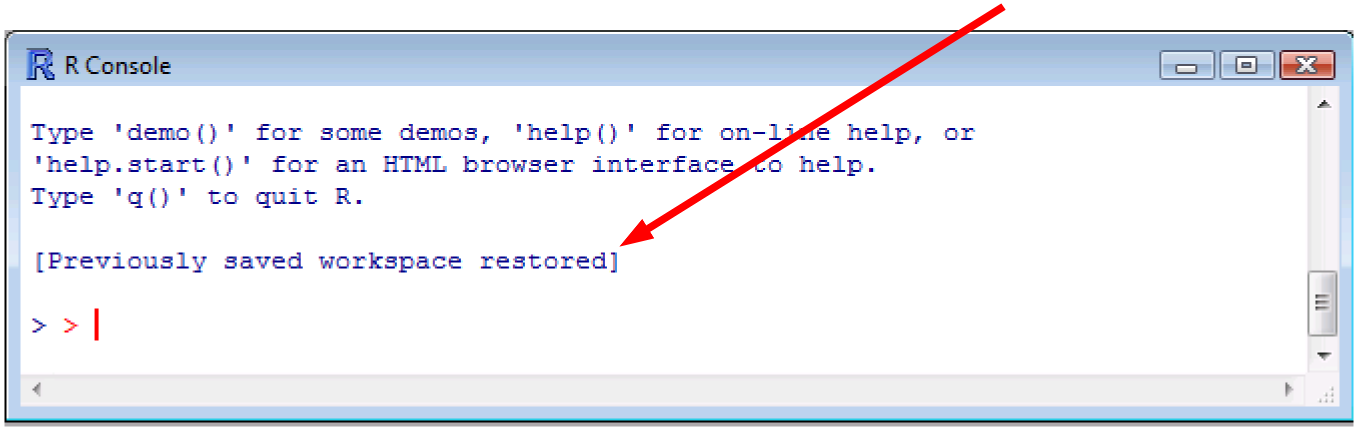
The file .Rhistory is plain text and it can be examined or edited.

To close R without the question:
q(save = "no")

R's workspace

R can save all the objects from memory to a file .Rdata and save all the commands typed during the session to a file .Rhistory, these are the default file names and they are saved on the working directory

Once a workspace is saved, it will be automatically loaded:



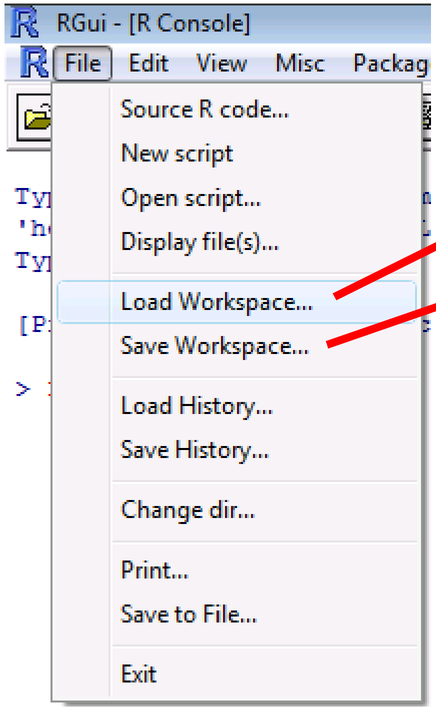
```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> >
```

By changing the working directory, many default workspace files can be used, on different directories.

But, the next session will open the default workspace, on the default working directory.

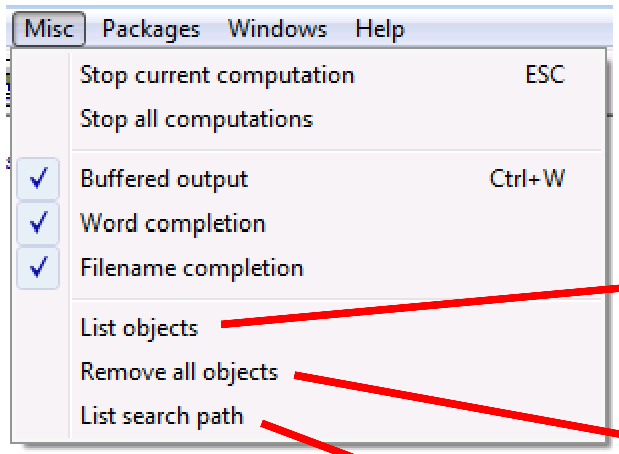
R's workspace

Workspace files can be saved and loaded from the File menu, with no need to change the working directory:



Or on the console:
load.image()
and
save.image()

R's workspace



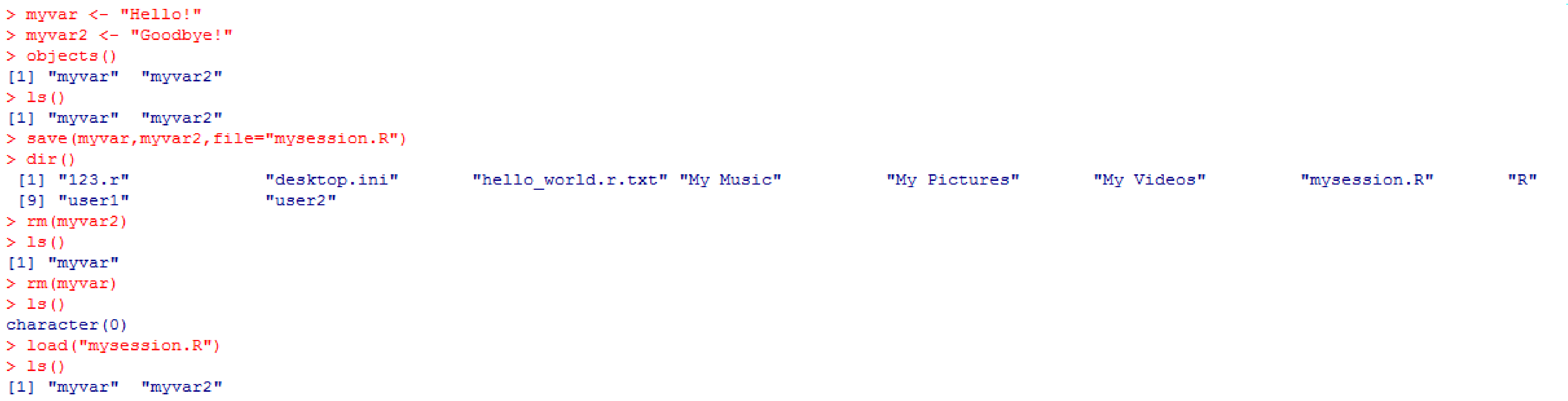
shows the contents of the workspace, sames as objects() or ls()

clears the workspace, sames as rm(list = ls(all = TRUE))

list of attached packages and R objects, sames as search()

R's workspace

objects() or ls() shows the contents of the workspace
save(var1, var2, varN, file="myfile.R") saves objects var1, var2 and varN to a file "myfile.R"
load("myfile.R") loads objects from file "myfile.R"
rm(var1) removes var1 from the workspace
rm(list = ls(all = TRUE)) clears the workspace
dir() shows the files on the working directory



```
> myvar <- "Hello!"
> myvar2 <- "Goodbye!"
> objects()
[1] "myvar"  "myvar2"
> ls()
[1] "myvar"  "myvar2"
> save(myvar,myvar2,file="mysession.R")
> dir()
 [1] "123.r"             "desktop.ini"       "hello_world.r.txt" "My Music"          "My Pictures"       "My Videos"         "mysession.R"       "R"
 [9] "user1"             "user2"
> rm(myvar2)
> ls()
[1] "myvar"
> rm(myvar)
> ls()
character(0)
> load("mysession.R")
> ls()
[1] "myvar"  "myvar2"
```

R's working directory

Working Directory
Default setting on Linux is $R_HOME\bin
Default setting on Windows is C:/Users/*MyUserName*/Documents

The command "system" executes OS commands

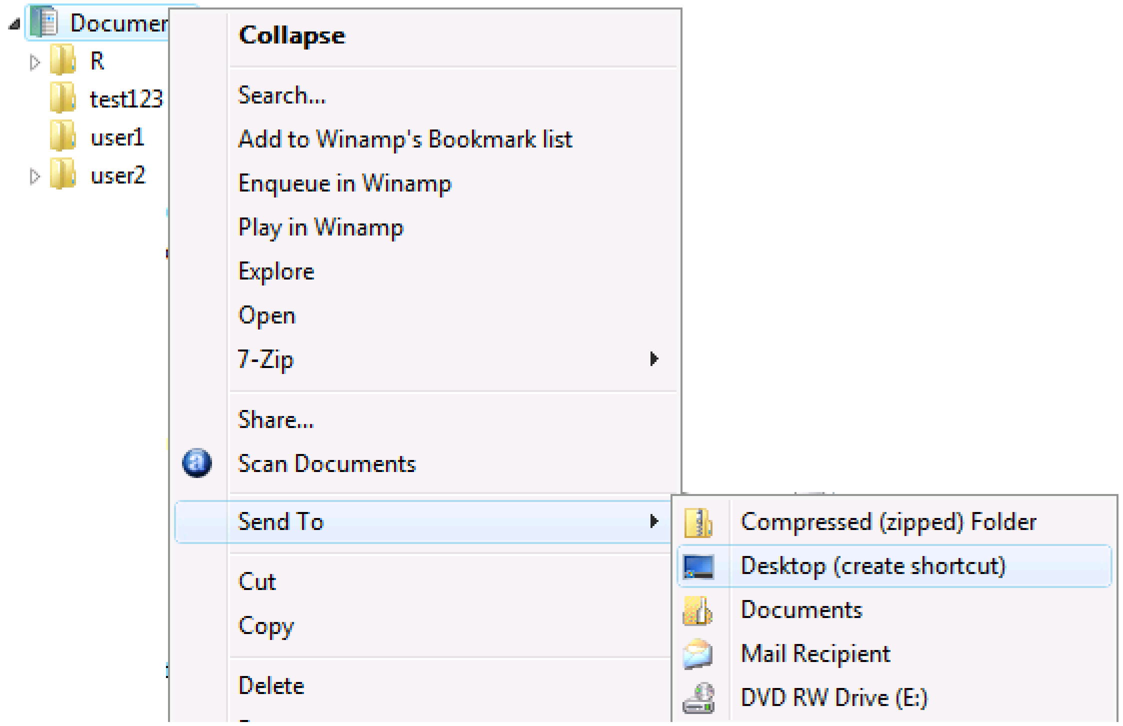
```
> getwd() # get the working directory
[1] "C:/Users/user/Documents"
> setwd("C:/Users/user/Documents/test123") # change the working directory
Error in setwd("C:/Users/user/Documents/test123") :
  cannot change working directory
> getwd() # it didn't change because the directory does not exist
[1] "C:/Users/user/Documents"
> system("md test123") # create a directory on Linux
Warning message:
In system("md test123") : md not found
> system(paste(Sys.getenv("COMSPEC"),"/c", "md test123")) # create a directory on
Windows
> setwd("C:/Users/user/Documents/test123")
> getwd()
[1] "C:/Users/user/Documents/test123"
```

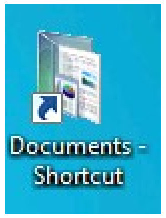
R's working directory and workspace

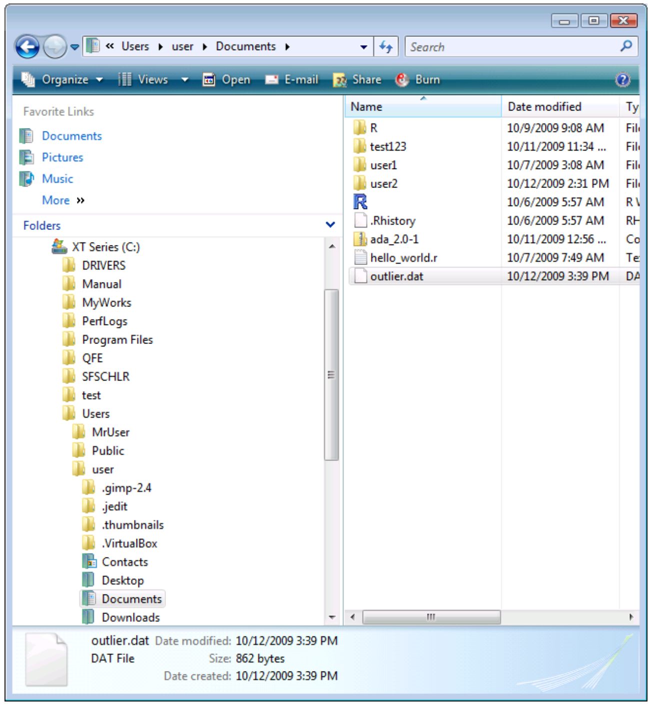
```
getwd()
myvar1 <- "variable 1 is a string"
myvar2 <- -2342.452
dir()
dir(all.files = T)
savehistory() # save the command history to the default file (.Rhistory)
save.image() # save the workspace to the default file (.RData)
dir() # it won't show .Rhistory and .RData
dir(all.files = T) # now it shows all the files!
file.show(".Rhistory") # display the history file, a text file is ok
file.show(".RData") # a binary data can't be displayed
```

R's working directory

Creating a shortcut on the desktop to the working directory







Name	Date modified	Ty
R	10/9/2009 9:08 AM	Fil
test123	10/11/2009 11:34 ...	Fil
user1	10/7/2009 3:08 AM	Fil
user2	10/12/2009 2:31 PM	Fil
	10/6/2009 5:57 AM	R \
.Rhistory	10/6/2009 5:57 AM	RH
ada_2.0-1	10/11/2009 12:56 ...	Co
hello_world.r	10/7/2009 7:49 AM	Te
outlier.dat	10/12/2009 3:39 PM	DA

outlier.dat Date modified: 10/12/2009 3:39 PM
DAT File Size: 862 bytes
Date created: 10/12/2009 3:39 PM

On Windows explorer, right click on the working directory and choose “Send To”, then choose “Desktop (create shortcut)”

programming R workspace
References/to learn more:

Basic statistics using R pp. 76
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/a70c8973cb8798b0bd0e6bdf7abd6ec7

Introductory Statistics with R
Peter Dalgaard, pp 31
2012 Springer

Quick-R
Rob Kabacoff
http://www.statmethods.net/interface/workspace.html

Data Structures in R

All objects are vectors

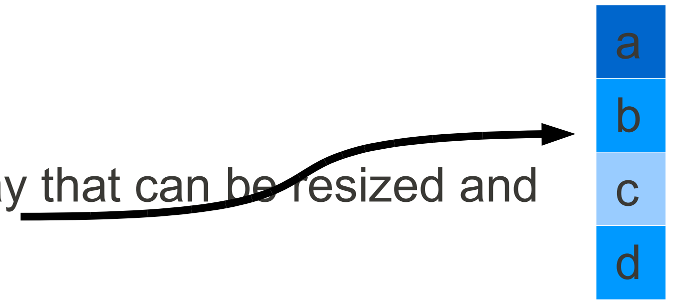
there are five other classes
for the basic data structures

- Factor
- Matrix
- Array
- Dataframe
- list

Data Structures in R

Vector

A vector is a dynamic array, that is, a unidimensional array that can be resized and allows elements to be added or removed.



Vector elements are numbered from 1 to n, n is the size of the vector. Elements can be accessed through their index with square brackets [ ], negative indeces = exclusion

3 types of vectors

- Numeric
- Character
- Boolean {true, false}

```
> c(734, 985, 43, 952)
[1] 734 985  43 952
> c("Helsinki","Tampere","Turku")
[1] "Helsinki" "Tampere"  "Turku"
> c(T,F,F,F,T,F,T,F,T,T)
 [1]  TRUE FALSE FALSE FALSE  TRUE
FALSE  TRUE FALSE  TRUE  TRUE
```

4 ways to create vectors

- : - colon operator
- c() - "concatenate" function
- seq() - "sequence" function
- rep() - repetition function

Data Structures in R

: - colon operator

Generates regular sequences from a starting value of the sequence to an end value of the sequence. The values are either a number (numeric or integer) or a factor.
The first element is *from* and the next ones' are *from* plus or minus one, up to or down to *to*.

Syntax:
from:*to*

The increment is always 1 or -1 for numeric arguments.
If *from* is integer then the result is integer, regardless of *to*.

from:*to* is equivalent to seq(*from*, *to*)

```
> 2:5 # sequence of numbers from 2 to 5
[1] 2 3 4 5
> 5:2 # sequence of numbers from 5 down to 2
[1] 5 4 3 2
> -3:4 # sequence of numbers from -3 to 4
[1] -3 -2 -1  0  1  2  3  4
> 0:pi # sequence of numbers from 0 to π
[1] 0 1 2 3
> pi:7 # sequence of numbers from π to 7
[1] 3.141593 4.141593 5.141593 6.141593
```

F(n+1) = F(n) + 1
or
F(n+1) = F(n) - 1

N integer implies F(n) integer
N real implies F(n) real

Data Structures in R

c() - "concatenate" function

Combine Values into a Vector or List.

c(myobj1, ..., myobjN, recursive=FALSE) combines all arguments from myobj1 to myobjN, with each element of the object as an element of the resulting vector, unless the object is a list, in which case the list is stored as one element of the resulting vector.

c(myobj1, ..., myobjN, recursive=TRUE) recursively combines all arguments from myobj1 to myobjN, with each element of the object as an element of the resulting vector, if the object can be listed, that is split into its elements.

```
> c(734, 985, 43, 952) # numeric vector
[1] 734 985  43 952
> c("Helsinki","Tampere","Turku") # string vector
[1] "Helsinki" "Tampere"  "Turku"
> c(T,F,F,F,T,F,T,F,T,T) # logical vector
 [1]  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
> c(23,10:16) # numeric vector
[1] 23 10 11 12 13 14 15 16
> c(T,F,F,5) # numeric vector
[1] 1 0 0 5
> c(1:5, 10.5, "next") # string vector
[1] "1"    "2"    "3"    "4"    "5"    "10.5" "next"
```

Data Structures in R

The elements of a vectors are of one data type only (Boolean, Numeric or Character) and mixing data types results in automatic data conversion.
Order of conversion: boolean → numeric → character

```
> c(T,F,F,55) # boolean becomes numeric
[1]  1  0  0 55
> c(TRUE, FALSE, F, "Turku") # boolean becomes character
[1] "TRUE"  "FALSE" "FALSE" "Turku"
> c(734, 985, "Turku") # numeric becomes character
[1] "734"   "985"   "Turku"
> c(TRUE, FALSE, F, T, -7.34, 72+9i, "Turku") # boolean and numeric become
character
[1] "TRUE"  "FALSE" "FALSE" "TRUE"  "-7.34" "72+9i" "Turku"
```

Data Structures in R

seq - "sequence" function

Generate regular sequences:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)), length.out = NULL, along.with = NULL, ...)

Arguments
... arguments passed to or from methods.
from, to the starting and (maximal) end value of the sequence.
by number: increment of the sequence.
length.out desired length of the sequence. A non-negative number, which for seq and seq.int will be rounded up if fractional.
along.with take the length from the length of this argument.

```
> seq(4, 9) # same as 4:9
[1] 4 5 6 7 8 9
> seq(1,10, by= 3) # numbers starting at 1, incrementing by 3, up to 10
[1]  1  4  7 10
> seq(1,15, length.out= 6) # 6 numbers evenly spaced between 1 and 15
[1]  1.0  3.8  6.6  9.4 12.2 15.0
> seq(along.with= 4:8) # the length of this argument will be the length of the output
[1] 1 2 3 4 5
> seq(7) # same as 1:7
[1] 1 2 3 4 5 6 7
> seq(length.out= 7) # same as 1:7
[1] 1 2 3 4 5 6 7
> seq(1,by=3, length.out= 9) # 9 numbers, starting in 1, incremented by 3
[1]  1  4  7 10 13 16 19 22 25
```

F(n+1) = F(n) + 1, F(n) [4, 9]

F(n+1) = F(n) + 3, F(n) [1, 10] the result is between 1 and 10

F(n+1) = F(n) + x, F(n) [1, 15] x = (15-1)/(6-1)

Data Structures in R

rep() - repetition function

Replicate elements of vectors and lists

rep(x, times, length.out, each)

Arguments
x is a scalar, a vector (including a list) or a pairlist or a factor
... further arguments:
times - a scalar or vector with the number of times repeat each element if times has the same length as the input, or to repeat the whole vector if times has length 1
length.out - an integer with the length of the result
each - an integer with the number of times each element of the input will be repeated

rep(x, times=1, length.out=NA, each=1) this is the default action

Data Structures in R

rep() - repetition function

```
> rep(14,3) # repeat number 14, 3 times
[1] 14 14 14
> rep(c(8,3,7),1:3) # repeat number 8, once, number 3, twice and number 7, thrice
[1] 8 3 3 7 7 7
> rep(c(8,3,7),1:3,4) # repeat number 8, 3 and 7 but limit the result to 4 elements
[1] 8 3 7 8
> rep(c(8,3,7),each=3) # repeat number 8,  number 3 and number 7, thrice
[1] 8 8 8 3 3 3 7 7 7
> rep(c(8,3,7), length.out=7,each=3) # repeat number 8,  number 3 and number 7, thrice - 
but limit the result to 7 elements
[1] 8 8 8 3 3 3 7
> rep(c(8,3,7), times=2,each=3) # repeat number 8,  number 3 and number 7, thrice - do this 
twice
 [1] 8 8 8 3 3 3 7 7 7 8 8 8 3 3 3 7 7 7
> rep(c(8,3,7), times=2,length.out=15,each=3) # repeat number 8,  number 3 and number 7, 
thrice - do this twice and limit the result to 15 elements
 [1] 8 8 8 3 3 3 7 7 7 8 8 8 3 3 3
```

Data Structures in R

```
rep(14,3) # repeat number 14, 3 times
rep(14,4)
rep(14,5)

rep(c(8,3,7),1:3) # repeat number 8, once, number 3, twice and number 7, thrice
rep(c(8,3,7),2:4)
rep(c(8,3,7),3:5)

rep(c(8,3,7),1:3,4) # repeat number 8, 3, and 7 but limit the result to 4 elements
rep(c(8,3,7),1:3,5)
rep(c(8,3,7),1:3,6)

rep(c(8,3,7),each=3) # repeat number 8, number 3 and number 7, thrice
rep(c(8,3,7),each=4)
rep(c(8,3,7),each=5)

rep(c(8,3,7), length.out=7,each=3) # repeat number 8,  number 3 and number 7, thrice -
but limit the result to 7 elements
rep(c(8,3,7), length.out=8,each=3)
rep(c(8,3,7), length.out=9,each=3)

rep(c(8,3,7), times=2,each=3) # repeat number 8,  number 3 and number 7, thrice - do
this twice
rep(c(8,3,7), times=3,each=3)
rep(c(8,3,7), times=4,each=3)
```

Data Structures in R

Extracting vector elements, or subsets

3 ways to extract vector elements $\left\{ \begin{array}{l} \text{By the element index(es)} \\ \text{By a logical expression} \\ \text{By keys} \end{array} \right.$

myvector

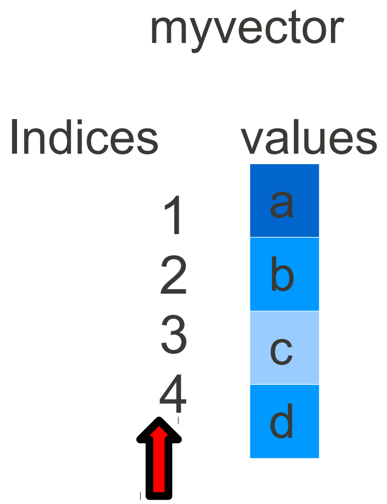
Indices	values
1	a
2	b
3	c
4	d

On vector "myvector"
Element 1 has value "a"

Data Structures in R

Extracting vector elements by the element index(es)

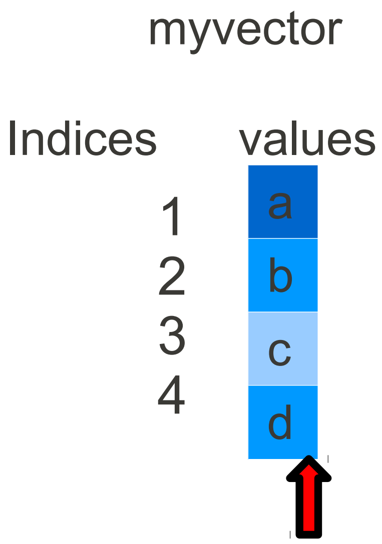
```
> myvec <- c(734, 985, 43, 952, 67, 28, 235, 885, 193)
> myvec
[1] 734 985  43 952  67  28 235 885 193
> myvec[5] # 5th element, starring Bruce Willis
[1] 67
> myvec[c(1,5,7)] # elements 1, 5 and 7
[1] 734  67 235
> myvec[-5] # all but the 5th element
[1] 734 985  43 952  28 235 885 193
> myvec[-c(1,5,7)] # all but elements 1, 5 and 7
[1] 985  43 952  28 885 193
> myvec[4:6] # elements 4 to 6
[1] 952  67  28
```



Data Structures in R

Extracting vector elements by a logical expression

The elements are selected by their value, regardless of their index



```
> myvec <- c(734, 985, 43, 952, 67, 28, 235, 885, 193)
> myvec
[1] 734 985  43 952  67  28 235 885 193
> myvec[myvec > 500] # only elements above 500
[1] 734 985 952 885
> myvec[(myvec %% 2)==0] # only even elements
[1] 734 952  28
> myvec[myvec %in% 100:500] # elements with values from 100 to 500
[1] 235 193
```

Data Structures in R

Extracting vector elements by keys

A key (name) can be used to access the vector's elements

The comand names() will add names to an existing vector, or they can be defined when creating the vector

```
> myvec <- c(734, 985, 43)
> myvec
[1] 734 985  43
> names(myvec) <- c("Helsinki","Tampere","Turku")
> myvec
Helsinki  Tampere    Turku
     734      985       43
> myvec["Helsinki"]
Helsinki
     734
> myvec[c("Turku","Tampere")]
  Turku Tampere
     43     985
> myvec2 <- c(Helsinki=734, Tampere=985, Turku=43)
> myvec2
Helsinki  Tampere    Turku
     734      985       43
```

Data Structures in R

subset

Subset returns subsets of vectors, matrices or data frames

subset(x, subset, ...)

for matrix or data frame:
subset(x, subset, select, drop = FALSE, ...)

x object to be subsetted.
subset logical expression indicating elements or rows to keep: missing values are taken as false.
select expression, indicating columns to select from a data frame.
drop passed on to [indexing operator.
... further arguments to be passed to or from other methods.

```
subset(airquality, Temp > 80, select = c(Ozone, Temp))
subset(airquality, Day == 1, select = -Temp)
subset(airquality, select = Ozone:Wind)
```

Data Structures in R

Operations on vectors

Most operations for scalars will work on vectors

```
> myvec1 <- c(3,6,7,8,12,23,94)
> 10 + myvec1 # adding a scalar
[1]  13  16  17  18  22  33 104
> 3 * myvec1 # multiplying by a scalar
[1]   9  18  21  24  36  69 282
> myvec1 ^ 2 # power by a scalar
[1]    9   36   49   64  144  529 8836
> log(myvec1) # natural logarithm
[1] 1.098612 1.791759 1.945910 2.079442 2.484907 3.135494 4.543295
> sin(myvec1) # sine
[1]  0.1411200 -0.2794155  0.6569866  0.9893582 -0.5365729 -0.8462204 -0.2452520
> myvec2 <- c(5,7,8,152,71,77,89)
> myvec1 + myvec2 # vector addition
[1]   8  13  15 160  83 100 183
> myvec1 * myvec2 # vector multiplicaton
[1]   15   42   56 1216  852 1771 8366
```

Data Structures in R

Vector set operations

set operations (union, intersection, asymmetric difference, equality and membership) on two vectors.
Union() is not the same as concatenation c() because c() will duplicate values that are common to both vectors.

```
> myvec1 <- c(3,6,7,8,12,23,94)
> myvec2 <- c(5,7,8,152,71,77)
> union(myvec1, myvec2) # set union
 [1]   3   6   7   8  12  23  94   5 152  71  77
> c(myvec1,myvec2) # notice the difference betwen union() and c()
 [1]   3   6   7   8  12  23  94   5   7   8 152  71  77
> intersect(myvec1, myvec2) # set intersection
[1] 7 8
> setdiff(myvec1, myvec2) # set difference
[1]  3  6 12 23 94
> setequal(myvec1, myvec2) # set equality
[1] FALSE
> is.element(4, myvec1) # set membership, is.element and %in% are synonims
[1] FALSE
> is.element(6, myvec1) # set membership
[1] TRUE
> 4 %in% myvec1 # set membership
[1] FALSE
> 6 %in% myvec1 # set membership
[1] TRUE
```

Data Structures in R

Sorting functions for vectors

```
> myvec <- c(734, NA, 985, 43, NA, 952, 67)
> myvec
[1] 734  NA 985  43  NA 952  67
> sort(myvec) # Sort a vector or factor
[1]  43  67 734 952 985
> sort(myvec, decreasing = TRUE) # Sort a vector or factor, decreasing
[1] 985 952 734  67  43
> rev(myvec) # Reverse elements
[1]  67 952  NA  43 985  NA 734
> unique(myvec) # Get non duplicate elements of a vector
[1] 734  NA 985  43 952  67
> order(myvec) # Sort an object, return the indeces
[1] 4 7 1 6 3 2 5
> order(myvec, na.last = FALSE) # Sort an object, return the indeces, NA at the begining
[1] 2 5 4 7 1 6 3
> order(myvec, na.last = TRUE) # Sort an object, return the indeces, NA at the end
[1] 4 7 1 6 3 2 5
> order(myvec, decreasing = FALSE) # Sort an object, return the indeces,increasing
[1] 4 7 1 6 3 2 5
> order(myvec, decreasing = TRUE) # Sort an object, return the indeces, decreasing
[1] 3 6 1 7 4 2 5
```

Data Structures in R

Difference and length functions for vectors

```
> myvec <- c(734, 985, 43, 952, 67, 28, 235, 885, 193)
> myvec
[1] 734 985  43 952  67  28 235 885 193
> diff(myvec) # difference between elements
[1]  251 -942  909 -885  -39  207  650 -692
> c(myvec[2]-myvec[1],myvec[3]-myvec[2],myvec[4]-myvec[3],myvec[5]-myvec[4])
[1]  251 -942  909 -885
> diff(myvec, lag = 2) # difference between elements, with a lag of 2
[1] -691  -33   24 -924  168  857  -42
> c(myvec[3]-myvec[1],myvec[4]-myvec[2],myvec[5]-myvec[3])
[1] -691  -33   24
> diff(myvec, differences = 2) # order of the difference of 2
[1] -1193  1851 -1794   846   246   443 -1342
> length(myvec) # Get the length of the vector
[1] 9
> length(myvec) <- 12 # Set the length of the vector
> myvec
 [1] 734 985  43 952  67  28 235 885 193  NA  NA  NA
> length(myvec) # Get the length of the vector
[1] 12
```

Data Structures in R

Statistical functions for vectors

```
> myvec1 <- c(3,6,7,8,12,23,94)
> summary(myvec1) # Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.00    6.50    8.00   21.86   17.50   94.00
> min(myvec1) # Min
[1] 3
> quantile(myvec1, probs=0.25) # 1st Qu.
25%
6.5
> median(myvec1) # median
[1] 8
> quantile(myvec1, probs=0.5) # median = 2nd Qu.
50%
  8
> mean(myvec1) # mean
[1] 21.85714
> quantile(myvec1, probs=0.75) # 3rd Qu.
 75%
17.5
> max(myvec1) # max
[1] 94
```

Data Structures in R

Statistical functions for vectors

```
> quantile(myvec1, probs=c(0.25, 0.75)) # 1st Qu. and 3rd Qu.
 25%  75%
 6.5 17.5
> IQR(myvec1) # inter-quartile range
[1] 11
> mad(myvec1) # robust alternative to IQR
[1] 5.9304
> sd(myvec1) # standard deviation
[1] 32.46243
> var(myvec1) # variance
[1] 1053.810
```

Data Structures in R

any(..., na.rm = FALSE) returns TRUE if at least one value is TRUE
all(..., na.rm = FALSE) returns TRUE if all the values are TRUE

na.rm = TRUE will ignore all the NAs

```
> #compare vectors, all elements are equal
> x <- c(7, 5, 6)
> y <- c(7, 5, 6)
> x==y
[1] TRUE TRUE TRUE
> all(x==y)
[1] TRUE
> any(x==y)
[1] TRUE
>
> #compare vectors, one element is equal
> x <- c(7, 5, 6)
> y <- c(7, 8, 9)
> x==y
[1]  TRUE FALSE FALSE
> all(x==y)
[1] FALSE
> any(x==y)
[1] TRUE
```

```
> #compare vectors, regardless
of element position
> x <- c(7, 5, 6)
> y <- c(5, 7, 6)
> x==y
[1] FALSE FALSE  TRUE
> sort(x)==sort(y)
[1] TRUE TRUE TRUE
```

Data Structures in R

```
> # comparing 2 vectors, by position and with NAs
> x <- y <- c(7, 6, NA, NA, 5)
> all(x==y)
[1] NA
> all(x==y , na.rm = TRUE)
[1] TRUE
> identical(x, y)
[1] TRUE
> all.equal(x, y)
[1] TRUE
> x[!is.na(x)]==y[!is.na(y)]
[1] TRUE TRUE TRUE
> all( x[!is.na(x)]==y[!is.na(y)] )
[1] TRUE
>
> # NA OR TRUE is TRUE
> # this will return TRUE despite the NAs
> any(x==y)
[1] TRUE
> # this will return NA, not FALSE
> y <- c(1, NA, 2, 3, 4)
> any(x==y)
[1] NA
```

Data Structures in R

a	t	i
b	g	k
c	b	m

Matrix

A matrix is a two-dimensional (m X n) object, like 2 or more vectors of the same size, side by side.
A matrix ha sonly one data type, automatic data conversion like a vector and the functions that apply to vectors also apply to matrices, excluding a few specific ones'.

3 types of matrices {
- Numeric
- Character
- Boolean {true, false}

3 ways to create matrices {
- matrix() - matrix function
- rbind() - row bind function
- cbind() - column bind function

Data Structures in R

matrix()

matrix creates a matrix from a set of values

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Arguments
data an optional data vector
nrow the desired number of rows
ncol the desired number of columns
byrow if TRUE, the matrix is filled by rows
dimnames list of names for rows or rows and columns

as.matrix tries to convert an object to a matrix.

is.matrix returns TRUE if an object is a matrix

Data Structures in R

```
> matrix(10,3,2) # matrix 3 x 2 with 5's
     [,1] [,2]
[1,]   10   10
[2,]   10   10
[3,]   10   10
> matrix(c(1,2,3),3,2)# matrix 3 x 2 with 2 columns with values [1,2,3]
     [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    3    3
> matrix(c(1,2),3,2,byrow = T)# matrix 3 x 2 with 3 rows with values [1,2]
     [,1] [,2]
[1,]    1    2
[2,]    1    2
[3,]    1    2
> matrix(1:6,3,2)# matrix 3 x 2 with ascending values from each column
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> matrix(1:6,3,2,byrow = T)# matrix 3 x 2 with ascending values from each row
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Data Structures in R

Setting row and column names

```
> mymatrix <- matrix(1:6,2,3,dimnames = list(c("row1", "row2"),c("col1", "col2", "col3")))
> mymatrix # row and column names
     col1 col2 col3
row1    1    3    5
row2    2    4    6
> mymatrix1 <- matrix(1:6,2,3,dimnames = list(c("row1", "row2")))
> mymatrix1 # row names
     [,1] [,2] [,3]
row1    1    3    5
row2    2    4    6
> mymatrix2 <- matrix(1:6,2,3,dimnames = list(NULL,c("col1", "col2", "col3")))
> mymatrix2 # column names
     col1 col2 col3
[1,]    1    3    5
[2,]    2    4    6
```

Data Structures in R

Setting row and column names, or changing them, on an existing matrix

```
> #using colnames, rownames
> mymatrix3 <- matrix(1:6,2,3)
> colnames(mymatrix3) = c("col1", "col2", "col3") # adding column names
> rownames(mymatrix3) = c("row1", "row2") # adding row names
> mymatrix3
     col1 col2 col3
row1    1    3    5
row2    2    4    6
> #using dimnames
> mymatrix4 <- matrix(1:6,2,3)
> dimnames(mymatrix4) = list(c("row1", "row2"),c("col1", "col2", "col3"))
> mymatrix4
     col1 col2 col3
row1    1    3    5
row2    2    4    6
```

Data Structures in R

cbind(), rbind()

Combine vector, matrix or data frames by columns or rows

```
> myvec <- seq(0,by=2, length.out= 8)
> rbind(myvec, 1:8)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
myvec    0    2    4    6    8   10   12   14
         1    2    3    4    5    6    7    8
> cbind(myvec, 1:8)
     myvec
[1,]     0 1
[2,]     2 2
[3,]     4 3
[4,]     6 4
[5,]     8 5
[6,]    10 6
[7,]    12 7
[8,]    14 8
```

Data Structures in R

Extracting matrix elements

```
> mymatrix <- matrix(1:6*10,3,2)
> mymatrix
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> mymatrix[1,1] # row 1, column 1
[1] 10
> mymatrix[3,2] # row 3, column 2
[1] 60
> mymatrix[1] # row 1, column 1
[1] 10
> mymatrix[2] # row 2, column 1
[1] 20
> mymatrix[2,1:2] # row 2, column 1 and 2
[1] 20 50
> mymatrix[1,] # row 1
[1] 10 40
> mymatrix[2,] # row 2
[1] 20 50
> mymatrix[,1] # column 1
[1] 10 20 30
> mymatrix[,2] # column 2
[1] 40 50 60
```

```
mymatrix <- matrix(1:6*10,3,2)

# if the row or column index is not
specified, the whole row or column
is taken
mymatrix[1,] # row 1
mymatrix[1,1:2] # row 1, all columns
explicitly selected
mymatrix[,1] # column 1
mymatrix[1:3,1] # column 1, all rows
explicitly selected

mymatrix[,] # if the row and column
index are  not specified, it's the
same
mymatrix # as the whole matrix

# a single index will show the matrix
elements by the order of insertion
# which is columns from top to
botton, rows from left to right
mymatrix[1]
mymatrix[2]
mymatrix[3]
mymatrix[4]
mymatrix[1:6]
```

Data Structures in R

Negative indices remove rows or columns

```
> mymatrix <- matrix(1:6*10,3,2)
> mymatrix
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> mymatrix[-1,-1] # remove row 1 and column 1
[1] 50 60
> mymatrix[-1,] # remove row 1
     [,1] [,2]
[1,]   20   50
[2,]   30   60
> mymatrix[-2,] # remove row 2
     [,1] [,2]
[1,]   10   40
[2,]   30   60
> mymatrix[,-1] # remove column 1
[1] 40 50 60
> mymatrix[,-2] # remove column 2
[1] 10 20 30
```

Data Structures in R

Extracting matrix elements by row or column names

```
> mymatrix <- matrix(1:6*10,2,3,dimnames = list(c("row1", "row2"),c("col1", "col2", "col3")))
> mymatrix
     col1 col2 col3
row1   10   30   50
row2   20   40   60
> mymatrix["row1","col1"]# row 1, column 1
[1] 10
> mymatrix["row2",]# row 2
col1 col2 col3
  20   40   60
> mymatrix[,c("col1","col3")]# column 1 and column 3
     col1 col3
row1   10   50
row2   20   60
```

Data Structures in R

Matrix info

```
> mymatrix <- matrix(1:6*10,2,3,dimnames = list(c("row1", "row2"),c("col1", "col2", "col3")))
> mymatrix
     col1 col2 col3
row1   10   30   50
row2   20   40   60
> dim(mymatrix) # dimensions of the matrix, 2 x 3
[1] 2 3
> length(mymatrix) # number of elements
[1] 6
> dimnames(mymatrix) # dimension names (rows and columns names')
[[1]]
[1] "row1" "row2"

[[2]]
[1] "col1" "col2" "col3"

> colnames(mymatrix) # rows names
[1] "col1" "col2" "col3"
> rownames(mymatrix) # columns names
[1] "row1" "row2"
> mode(mymatrix) # Storage Mode of this  Object
[1] "numeric"
> is(mymatrix) # all the super-classes of this object's class
[1] "matrix"    "array"     "structure" "vector"
> class(mymatrix) #  class attribute or the implicit class of this object
[1] "matrix"
```

Data Structures in R

Matrix calculations

```
> myvec <- seq(1,by=3, length.out= 9)
> mymatrix1 <- matrix(myvec,3,3)
> mymatrix2 <- matrix(9:1,3,3)
> # component-wise multiplication
> mymatrix1 * mymatrix2
     [,1] [,2] [,3]
[1,]    9   60   57
[2,]   32   65   44
[3,]   49   64   25
> # matrix multiplication
> mymatrix1 %*% mymatrix2
     [,1] [,2] [,3]
[1,]  222  132   42
[2,]  294  177   60
[3,]  366  222   78
> # matrix transpose
> t(mymatrix1)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]   10   13   16
[3,]   19   22   25
```

```
> myvec
[1]  1  4  7 10 13 16 19 22 25
> mymatrix1
     [,1] [,2] [,3]
[1,]    1   10   19
[2,]    4   13   22
[3,]    7   16   25
> mymatrix2
     [,1] [,2] [,3]
[1,]    9    6    3
[2,]    8    5    2
[3,]    7    4    1
```

Data Structures in R

Matrix calculations

```
> diag(1:4) # diagonal matrix 4 X 4
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
> diag(1,2) # Identity matrix 2 X 2
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> mymatrix <- matrix(1:9*10,3,3)
> det(mymatrix) # Determinant
[1] -5.32907e-13
> sum(diag(mymatrix)) # trace of a matrix
[1] 150
> eigen(mymatrix)$values  # Eigenvalues
[1]  1.611684e+02 -1.116844e+01 -5.019627e-15
> eigen(mymatrix)$vectors  # Eigenvectors
           [,1]       [,2]       [,3]
[1,] -0.4645473 -0.8829060  0.4082483
[2,] -0.5707955 -0.2395204 -0.8164966
[3,] -0.6770438  0.4038651  0.4082483
```

Data Structures in R

Matrix calculations

chol() Choleski factorization of a real symmetric positive-definite square matrix
qr() QR decomposition of a matrix
svd() singular-value decomposition of a rectangular matrix
crossprod() matrix cross-product
outer() outer product of arrays
scale() Scaling and centering of matrix
solve() Solve a system of equations
svd() singular-value decomposition of a rectangular matrix

Data Structures in R

Changing the matrix's elements

```
> #adding one row
> mymatrix <- matrix(1:6,2,3,byrow=T)
> mymatrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> mymatrix  <- rbind(mymatrix, 7:9)
> mymatrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> #adding one column
> mymatrix  <- cbind(mymatrix, seq(3.5,by=3,length.out = 3))
> mymatrix
     [,1] [,2] [,3] [,4]
[1,]    1    2    3  3.5
[2,]    4    5    6  6.5
[3,]    7    8    9  9.5
```

Data Structures in R

Changing the matrix's elements

```
> #changing an entire row
> mymatrix[3,] <- 1:4
> mymatrix
     [,1] [,2] [,3] [,4]
[1,]    1    2    3  3.5
[2,]    4    5    6  6.5
[3,]    1    2    3  4.0
> #changing an entire column
> mymatrix[,4] <- 7:9
> mymatrix
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    7
[2,]    4    5    6    8
[3,]    1    2    3    9
> #deleting one row
> mymatrix <- mymatrix[-2,]
> mymatrix
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    7
[2,]    1    2    3    9
> #deleting one column
> mymatrix <- mymatrix[,-4]
> mymatrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
```

Data Structures in R

Applying functions on matrix/array elements

apply() returns a vector or array or list, after applying a function to each of its members

apply(*object, margin, function*, ...)

object is the input array
margin are the subscripts where to apply the function, 1 indicates rows, 2 indicates columns, c(1,2) indicates rows and columns
function
... optional arguments for the function

```
> mymatrix <- matrix(1:6*10,2,3)
> mymatrix
     [,1] [,2] [,3]
[1,]   10   30   50
[2,]   20   40   60
> apply(mymatrix,1,max) # rows
[1] 50 60
> apply(mymatrix,2,max) # columns
[1] 20 40 60
> apply(mymatrix,c(1,2),max) # rows and columns, useless
```

```
#try:
apply(mymatrix,1,mean) # rows
apply(mymatrix,2,mean) # columns
```

```
#try:
apply(mymatrix,1,sort) # rows
apply(mymatrix,2,sort) # columns
```

Data Structures in R

Array

An array is a three-dimensional (m X n X p) object, like 2 or more matrices of the same dimensions, side by side.
An array has only one data type, automatic data conversion like a vector or matrix and the functions that apply to vectors and matrices also apply to arraya, excluding a few specific ones'.

array(data = NA, dim = length(data), dimnames = NULL) creates an array from data, dim are the dimensions and dimnames are optional names for the dimensions

as.array() tries to convert an object to an array

is.array() returns TRUE if the object is an array

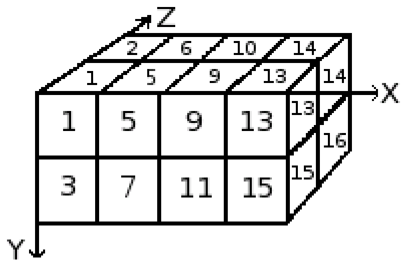
Dimension z = 2

2	6	10	14
8	19	17	16

Dimension z = 1

1	5	9	13
3	7	11	15

=

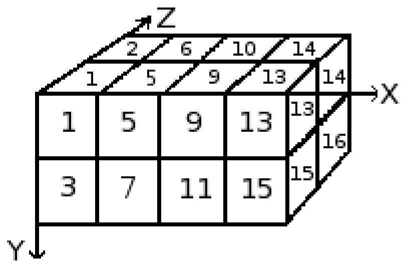


Data Structures in R

2	6	10	14
8	19	17	16

1	5	9	13
3	7	11	15

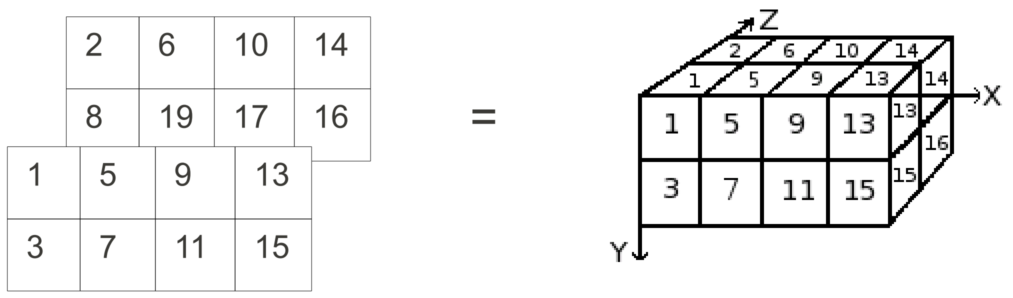
=



```
> array(c(1,3,5,7,9,11,13,15,2,8,6,19,10,17,14,16),c(2,4,2))
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    3    7   11   15

, , 2

     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    8   19   17   16
```

Notice how the element values are inserted by column

Data Structures in R



```
> # turning matrices into arrays
> # passing data by rows
> m1 <- matrix(c(1,5,9,13,3,7,11,15),2,4, byrow=T)
> m2 <- matrix(c(2,6,10,14,8,19,17,16),2,4, byrow=T)
> array(c(m1,m2),c(2,4,2))
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    3    7   11   15

, , 2

     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    8   19   17   16
```

Data Structures in R

Adding dimension names

City

	Age	Hgt	Wgt	BPM
Men	56	174	75	77
Women	67	166	55	70

Countryside

	Age	Hgt	Wgt	BPM
Men	64	178	78	63
Women	77	170	59	61

This data is fake, can anyone get real data?

```
> myarray<-array(c(56,67,174,166,75,55,77,70,64,77,178,170,78,59,63,61),c(2,4,2))
> dimnames(myarray) = list(c("men","women"),c("age","height","weight","pulse"),
+ c("city","countryside"))
> myarray
, , city

      age height weight pulse
men    56    174     75    77
women  67    166     55    70

, , countryside

      age height weight pulse
men    64    178     78    63
women  77    170     59    61
```

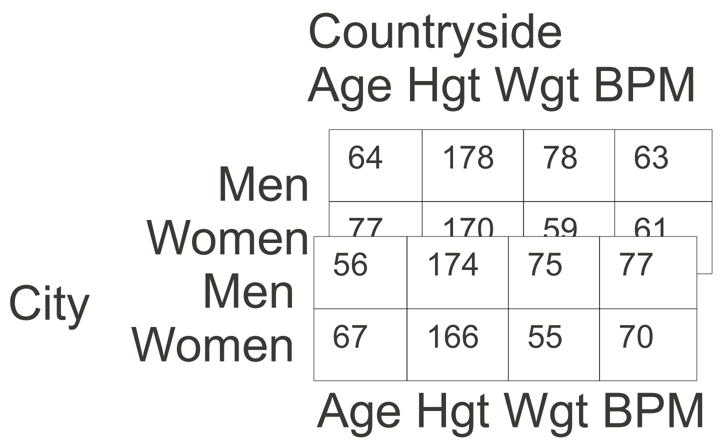
or

```
# dimension names defined directly:
myarray2<-
array(c(56,67,174,166,75,55,77,70,64,77,178,170,
78,59,63,61),c(2,4,2), dimnames =
list(c("men","women"),c("age","height","weight","pul
se"),c("city","countryside")))
```

Data Structures in R

```
> dimnames(myarray)
[[1]]
[1] "men"   "women"

[[2]]
[1] "age"    "height" "weight" "pulse"

[[3]]
[1] "city"       "countryside"

> myarray["women",,] # women's all info, all cities
       city countryside
age      67          77
height  166         170
weight   55          59
pulse    70          61
> myarray["women",,"countryside"] # women's all info, countryside
   age height weight  pulse
    77    170     59     61
> myarray[,,"countryside"] # all info, countryside
      age height weight pulse
men    64    178     78    63
women  77    170     59    61
> myarray[, "height",] # height
      city countryside
men    174         178
women  166         170
```

Accesing the array's elements

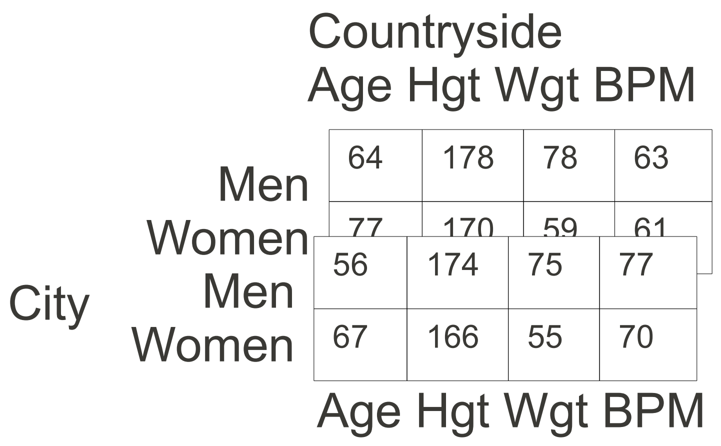


Same, with indices

```
myarray[2,,] # women's all info, all cities
myarray[2,, 2] # women's all info,
countryside
myarray[,,2] # all info, countryside
myarray[, 2,] # height
```

Data Structures in R

Operations on the array's elements



```
> apply(myarray,1,max) # rows
  men women
  178   170
```

Meaningless, age vs htg...

```
> apply(myarray,2,max) # columns
   age height weight  pulse
    77    178     78     77
```

Oldest, tallest...

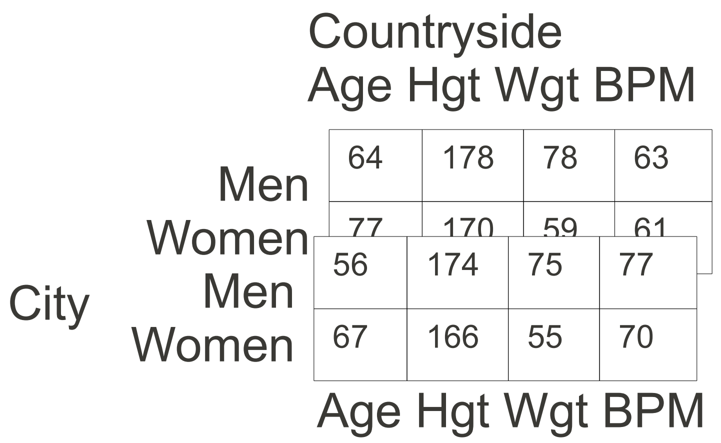
```
> apply(myarray,c(1,2),max) # rows and columns
      age height weight pulse
men    64    178     78    77
women  77    170     59    70
```

Oldest, tallest... by gender

```
apply(myarray,2,mean) # columns mean
apply(myarray,c(1,2),mean) # rows and columns mean
apply(myarray,2,quantile) # columns quartiles
apply(myarray,c(1,2),quantile) # rows and columns quartiles

apply(myarray,2,quantile,.5) # columns median
```

Data Structures in R

Changing the array's elements



```
myarray <- array(c(56,67,174,166,75,55,77,70,64,77,178,170,78,59,63,61),c(2,4,2))
dimnames(myarray) = list(c("men","women"),c("age","height","weight","pulse"),
c("city","countryside"))

myarray <- myarray[, -4,] # remove pulse, by index
myarray <- myarray[, colnames(myarray) != "age",] # remove age, by column name

myarray <- array(c(myarray,c(167,162,75,60,179,168,77,65)),c(2,2,4)) # adding 2 "places"
dimnames(myarray) = list(c("men","women"),c("height","weight"), c("city","countryside",
"p1","p2"))
```

Data Structures in R

List

A list is a vector containing elements of different types

- vector
- matrix
- array
- dataframe
- list

The elements are accessible by indices, like on a vector, there is just an extra square bracket [ ] for the list index and there might be other indices from contained elements.

```
> myvec <- 3:8
> mymatrix <- matrix(6:1*10,3,2)
> mydataframe <- as.data.frame(mymatrix)
> mylist <- list(myvec, mymatrix, mydataframe, 56,"test")
> mylist[[1]][1]
[1] 3
> mylist[[1]][2]
[1] 4
> mylist[[2]][1,1]
[1] 60
> mylist[[3]]$V1[2]
[1] 50
> mylist[[4]]
[1] 56
> mylist[[5]]
[1] "test"
```

```
> mylist
[[1]]
[1] 3 4 5 6 7 8

[[2]]
     [,1] [,2]
[1,]   60   30
[2,]   50   20
[3,]   40   10

[[3]]
  V1 V2
1 60 30
2 50 20
3 40 10

[[4]]
[1] 56

[[5]]
[1] "test"
```

Data Structures in R

Naming the elements of the list is recommended

Accessing the elements of the list

```
> myvec <- 3:8
> mymatrix <- matrix(6:1*10,3,2)
> mydataframe <- as.data.frame(mymatrix)
> mylist <- list(mv=myvec, mm=mymatrix,
mdf=mydataframe, mn=56,ms="test")
> mylist$mv[1]
[1] 3
> mylist$mv[2]
[1] 4
> mylist$mm[1,1]
[1] 60
> mylist$mdf$V1[2]
[1] 50
> mylist$mn
[1] 56
> mylist$ms
[1] "test"
```

```
> mylist
$mv
[1] 3 4 5 6 7 8

$mm
     [,1] [,2]
[1,]   60   30
[2,]   50   20
[3,]   40   10

$mdf
  V1 V2
1 60 30
2 50 20
3 40 10

$mn
[1] 56

$ms
[1] "test"
```

Data Structures in R

```
myvec <- 3:8
mymatrix <- matrix(6:1*10,3,2)
mydataframe <- as.data.frame(mymatrix)
mylist <- list(myvec, mymatrix, mydataframe, 56,"test")
is(mylist) # list, of course
length(mylist)
dim(mylist) # the dimensions of the elements don't count
mylist[1] # [1] <=> [[1]]
mylist[2]
mylist[3]
mylist[4]
mylist[5]
is(mylist[1]) # each element is a list
is(mylist[2])
is(mylist[3])
is(mylist[4])
is(mylist[5])
mylist[1:3]
```

Data Structures in R

Changing the elements of the list

```
myvec <- 3:8
mymatrix <- matrix(6:1*10,3,2)
mydataframe <- as.data.frame(mymatrix)
mylist <- list(mv=myvec, mm=mymatrix,
mdf=mydataframe, mn=56,ms="test")
# updating one element
mylist$ms <- "new test"
mylist$ms
mylist[[5]] <- "newer test"
mylist$ms
# inserting two elements
mylist <- c(mylist,wname="Friday", mday=13)
mylist
# deleting one element at a time
mylist$ms<- NULL
mylist[["mn"]]<- NULL
mylist[[1]]<- NULL
mylist
```

```
> mylist
$mv
[1] 3 4 5 6 7 8

$mm
     [,1] [,2]
[1,]   60   30
[2,]   50   20
[3,]   40   10

$mdf
  V1 V2
1 60 30
2 50 20
3 40 10

$mn
[1] 56

$ms
[1] "test"
```

Data Structures in R

Using the $ notation

```
myvec <- 3:8
mymatrix <- matrix(6:1*10,3,2)
mydataframe <- as.data.frame(mymatrix)
mylist <- list(mv=myvec, mm=mymatrix, mdf=mydataframe, mn=56,ms="test")
# inserting one element
mylist <- c(mylist,tree_info=list(family="Fagaceae", genus ="Fagus"))
mylist
mylist$tree_info # this is NULL, must specify the sub-elements
mylist$tree_info.family
mylist$tree_info.genus
mylist[["tree_info.family"]]
```

Data Structures in R

Factors

A factor is a vector that specifies a discrete clasification of other vectors. Factors store categorical data, qualitative values, non numeric such as gender, job, color, species, model, brand, etc... Or numeric but meaningless like model numbers or site numbers or zip codes.

```
> student.residence <-
c("Helsinki","Tampere","Turku","Helsinki","Helsinki","Turku","Oulu","Tampere","Helsinki","Tu
rku","Tampere","Helsinki")
> student.residence
 [1] "Helsinki" "Tampere"  "Turku"    "Helsinki" "Helsinki" "Turku"
 [7] "Oulu"     "Tampere"  "Helsinki" "Turku"    "Tampere"  "Helsinki"
> fstudent=as.factor(student.residence)
> fstudent
 [1] Helsinki Tampere  Turku    Helsinki Helsinki Turku    Oulu     Tampere
 [9] Helsinki Turku    Tampere  Helsinki
Levels: Helsinki Oulu Tampere Turku
> levels(fstudent)
[1] "Helsinki" "Oulu"     "Tampere"  "Turku"
> summary(fstudent)
Helsinki     Oulu  Tampere    Turku
       5        1        3        3
> student.height=c(175,162,170,170,192,170,115,155,150,130,220,160)
> student.height
 [1] 175 162 170 170 192 170 115 155 150 130 220 160
> tapply(student.height,fstudent,mean)
Helsinki     Oulu  Tampere    Turku
169.4000 115.0000 179.0000 156.6667
```

Data Structures in R

Sorted factors

Factor with levels of hierarchy

function ordered() turns a factor into a sorted factor

```
> # sort the cities by increasing longitude
> flevel.residence <- ordered(student.residence,
levels=c("Helsinki","Turku","Tampere","Oulu"))
> flevel.residence
 [1] Helsinki Tampere  Turku    Helsinki Helsinki Turku    Oulu     Tampere
 [9] Helsinki Turku    Tampere  Helsinki
Levels: Helsinki < Turku < Tampere < Oulu
> # check each student whether he/she lives south of Tampere
> flevel.residence < "Tampere"
 [1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE
TRUE
```

Data Structures in R

Data Frames

Data Frames are matrices with columns of different data types.

```
data.frame(..., row.names = NULL, check.rows = FALSE,
        check.names = TRUE,
        stringsAsFactors = default.stringsAsFactors())
```

... value or tag = value
row.names a column to be used as row names, or a vector with the row names
check.rows if TRUE then the rows are checked for consistency of length and names
check.names If TRUE then the names of the variables in the data frame are checked for syntax and uniqueness
stringsAsFactors true if character vectors should be converted to factors

```
> mydataf <- data.frame(age=c(25,22),height=c(174,166),weight=c(75,55),
city=c("Turku","Espoo"),row.names =c("Pekka","Anna"))
> mydataf
      age height weight  city
Pekka  25    174     75 Turku
Anna   22    166     55 Espoo
> is(mydataf)
[1] "data.frame" "list"       "oldClass"   "vector"
```

	Age	Hgt	Wgt	Name
Pekka	25	174	75	Pekka
Anna	22	166	55	Anna

Data Structures in R

Accessing the data frame's elements

```
> #getting the info for Anna
> mydataf[2,] # by index, row 2
     age height weight  city
Anna  22    166     55 Espoo
> mydataf["Anna",] # by key
     age height weight  city
Anna  22    166     55 Espoo
> #getting the weight for everybody
> mydataf[,3] # by index, column 3
[1] 75 55
> mydataf[,"weight"] # by key
[1] 75 55
> mydataf$weight # by list key
[1] 75 55
```

Data Structures in R

Sorting the data frame's elements

```
mydataf <- data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),
city=c("Turku","Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))
mydataf

#order by height
order(mydataf$height)
#order by name
order(row.names(mydataf))
#order by height and weight
order(mydataf$height, mydataf$weight)

#order by height
mydataf[order(mydataf$height),]
#order by name
mydataf[order(row.names(mydataf)),]
#order by height and weight
mydataf[order(mydataf$height, mydataf$weight),]
#order by height and weight, both decreasing
mydataf[order(mydataf$height, mydataf$weight, decreasing = T),]
#order by decreasing height and increasing weight
mydataf[order(-mydataf$height, mydataf$weight),]
```

Data Structures in R

Changing the data frame's elements

```
> mydataf <- data.frame(age=c(25,22),height=c(174,166),weight=c(75,55),
city=c("Turku","Espoo"),row.names =c("Pekka","Anna"))
> mydataf
      age height weight  city
Pekka  25    174     75 Turku
Anna   22    166     55 Espoo
> mydataf <- cbind(mydataf,course=c("Math","Art")) # adding a column
> mydataf
      age height weight  city course
Pekka  25    174     75 Turku   Math
Anna   22    166     55 Espoo    Art
> mydataf <- data.frame(mydataf, hobby=c("walking","reading")) # adding a column
> mydataf
      age height weight  city course   hobby
Pekka  25    174     75 Turku   Math walking
Anna   22    166     55 Espoo    Art reading
> mydataf <- rbind(mydataf,
zed=data.frame(age=28,height=199,weight=115,city="Oulu",course="Sports",hobby="sleep")) # adding
a row
> mydataf
      age height weight  city course   hobby
Pekka  25    174     75 Turku   Math walking
Anna   22    166     55 Espoo    Art reading
zed    28    199    115  Oulu Sports   sleep
```

Data Structures in R

Operations on the data frame's elements

```
> mean(mydataf[,1])  # the mean of all ages
[1] 23.5
> mean(mydataf[,c("height","weight")])  # the mean of height, weight
height weight
   170     65
> apply(mydataf,2,mean)  # ERROR!
   age height weight   city
    NA     NA     NA     NA
Warning messages:
1: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
2: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
3: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
4: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
> apply(mydataf[,1:3],2,mean)  # the mean of age, height, weight
   age height weight
  23.5  170.0   65.0
```

Data Structures in R

with() evaluate an expression in a data environment
with(data, expr, ...)

data data to use for constructing an environment, a list, a data frame, or an integer
expr expression to evaluate
... arguments to be passed to future methods

```
library(MASS)

    anorex.1 <- glm(anorexia$Postwt ~ anorexia$Prewt + anorexia$Treat +
offset(anorexia$Prewt), family = gaussian)
    summary(anorex.1)

with(anorexia, {
    anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt), family = gaussian)
    summary(anorex.1)
})
```

Data Structures in R

lapply, sapply applies a function over a list or vector

lapply returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X

sapply is a user-friendly version of lapply by default returning a vector or matrix if appropriate

lapply(X, FUN, ...)

sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)

X a vector (atomic or list) or an expressions vector
FUN the function to be applied to each element of X
... optional arguments to FUN
simplify if TRUE the result is simplified to a vector or matrix if possible
USE.NAMES if TRUE and if X is character, use X as names for the result unless it had names already
n number of replications
expr expression to evaluate repeatedly

```
at1 <- list(athlete="Johnson",coach="Earp",swimming=c(154,171,165), cycling=c(598,632,621),
running=c(1046,1102,1095),wetsuit=c(T,F,T))
# compute the list mean for each list element
mean(at1)
mean(at1$swimming) # one at at time...
lapply(at1,mean)
sapply(at1,mean)
```

Data Structures in R

Statistical functions

```
mydataf <- data.frame(age=c(25,22),height=c(174,166),weight=c(75,55),
city=c("Turku","Espoo"),row.names =c("Pekka","Anna"))
mydataf
sapply(mydataf, mean, na.rm=TRUE)
```

Other functions useful for sapply:

sd, var, min, max, med, range, and quantile

summary will return the Min. 1st Qu. Median Mean 3rd Qu. Max.

```
summary(mydataf)
```

fivenum will return Tukey's five number summary (minimum, lower-hinge, median, upper-hinge, maximum)

```
fivenum(mydataf[1:3])
```

Data Structures in R

string functions

```
# Concatenate strings
paste("a","b","c")
paste("a","b","c",sep="")

# Concatenate a vector of strings
myvec <- c("a","b","c")
paste(myvec)
paste(myvec,sep="",collapse="")

# extract part of the string
mystr <- "Hello world!"
substring(mystr, 7, 11)

# split a string into each character
mystr <- "Hello world!"
strsplit(mystr, "")

# split a string into pieces, using regex
mystr <- "Hello world!"
strsplit(mystr, " ")
strsplit(mystr, "\\s")
strsplit(mystr, "[\\seo]")
```

Data Structures in R

string functions

```
# find characters within the string, position + length
regexpr("e", mystr)

# replace one substring within the string, once only
sub("l","+",mystr)

# replace one substring within the string, for all matches
gsub("l","+",mystr)
```

format formats an R object for pretty printing

```
format(x, trim = FALSE, digits = NULL, nsmall = 0L,
       justify = c("left", "right", "centre", "none"),
       width = NULL, na.encode = TRUE, scientific = NA,
       big.mark = "",   big.interval = 3L,
     small.mark = "", small.interval = 5L,
   decimal.mark = ".", zero.print = NULL, drop0trailing = FALSE, ...)
```

Data Structures in R

date functions

```
Sys.Date() # current date

date() # current date and time
```

Use theformat() function to print dates

%d	day of the month (0-31)
%a	short week day
%A	long weekday
%m	month (00-12)
%b	short month
%B	long month
%y	2-digit year
%Y	4-digit year

```
format(Sys.Date(), format="%d %B %Y")
```

Data Structures in R

Data Type Conversion

Checking the data type

is.numeric(), is.character(), is.vector(), is.matrix(), is.data.frame()

Explicit conversion

as.numeric(), as.character(), as.vector(), as.matrix(), as.data.frame)

to / from	vector	Factor	Matrix	Array	Dataframe	list
vector	c(x,y)	as.factor(myvec, labels=c("L1", "L2", "L3")) as.factor(myvec,ordered=T, labels=c("L1", "L2", "L3"))	cbind(x,y) rbind(x,y)	array(x)	data.frame(x,y)	list(x)
Factor		ordered(f)				list
Matrix	as.vector(mymatrix)			array(x)	as.data.frame(mymatrix)	list
Array						list
Dataframe	mydataf[n,]		as.matrix(myframe)			list
list	unlist(mylist)					

Data Structures in R

Data Structures in R

Frequencies and Crosstabs

margin.table() compute the sum of table entries for a given index

margin.table(x, margin=NULL)

x an array
margin index number (1 for rows, etc...)

```
m <- matrix(1:9,3)
m
# row sum
margin.table(m,1)
sum(m[1,]);sum(m[2,]);sum(m[3,])
apply(m, 1, sum)
# column sum
margin.table(m,2)
sum(m[,1]);sum(m[,2]);sum(m[,3])
apply(m, 2, sum)

# note: there are functions for row sum and column sum:
colSums(m)
rowSums(m)
rowMeans(m)
colMeans(m)
```

Data Structures in R

Frequencies and Crosstabs

prop.table() Express table entries as a fraction of the marginal table

prop.table(x, margin=NULL)

x table
margin index, or vector of indices

```
m <- matrix(1:9,3)
m
prop.table(m) # cell percentages
prop.table(m, 1) # row percentages
prop.table(m, 2) # column percentages

prop.table(m) # cell percentages
m / sum(m)
sweep(m,1, margin.table(m),"/")

prop.table(m, 1) # row percentages
m[1,]/sum(m[1,])
m[2,]/sum(m[2,])
m[3,]/sum(m[3,])
sweep(m,1, margin.table(m,1),"/")

prop.table(m, 2) # column percentages
m[,1]/sum(m[,1])
m[,2]/sum(m[,2])
m[,3]/sum(m[,3])
sweep(m,2, margin.table(m,2),"/")
```

Data Structures in R

table() Cross tabulation and table creation

table(..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no", "ifany", "always"), dnn = list.names(...), deparse.level = 1)

... one of more objects which can be interpreted as factors
exclude levels to remove from all factors in .... If set to NULL, it implies useNA="always"
useNA whether to include extra NA levels in the table
dnn the names to be given to the dimensions in the result (the dimnames names)
deparse.level controls how the default dnn is constructed. See details
x an arbitrary R object, or an object inheriting from class "table" for the as.data.frame method
row.names a character vector giving the row names for the data frame
responseName The name to be used for the column of table entries, usually counts

Data Structures in R

```
x <- sample(c("heads","tails"),5, replace=T)
x
fx <- factor(x)
fx
table(fx)


mtcars
?mtcars
is(mtcars)
names(mtcars)
dim(mtcars)
rownames(mtcars);colnames(mtcars)
dimnames(mtcars)
# how many cars for each Number of cylinders
table(mtcars$cyl,dnn = list("Number of forward cylinders"))
# how many cars for each Number of cylinders / Number of forward gears
table(mtcars$cyl,mtcars$gear,dnn = list("Number of cylinders","Number of forward
gears"))
# how many cars for each Number of cylinders / Number of forward gears / Transmission
table(mtcars$cyl,mtcars$gear,mtcars$am,dnn = list("Number of cylinders","Number of
forward gears","Transmission"))
```

Data Structures in R

xtabs() reate a contingency table from cross-classifying factors

```
xtabs(formula = ~., data = parent.frame(), subset, na.action, exclude = c(NA, NaN),
drop.unused.levels = FALSE)
```

formula a formula object with the cross-classifying variables (separated by +) on the right hand side
data an optional matrix or data frame containing the variables in the formula formula
subset an optional vector specifying a subset of observations to be used
na.action a function which indicates what should happen when the data contain NAs
exclude a vector of values to be excluded when forming the set of levels of the classifying factors
drop.unused.levels a logical indicating whether to drop unused levels in the classifying factors

```
# convert from table to dataframe
hair.df=as.data.frame(HairEyeColor)
# convert from dataframe to table
xtabs(Freq~Hair+Eye+Sex,data=hair.df)

# crosstabulation of Hair and Eye
xtabs(Freq~Hair+Eye,data=hair.df)
# crosstabulation of Hair and Sex
xtabs(Freq~Hair+Sex,data=hair.df)
# crosstabulation of Eye and Sex
xtabs(Freq~Eye+Sex,data=hair.df)
```

Data Structures in R

ftable() create "flat" contingency tables

```
ftable(x, ...)
```

x, ... R objects which can be interpreted as factors
exclude values to use in the exclude argument of factor when interpreting non-factor objects
row.vars a vector of integers giving the numbers of the variables, or a character vector giving the names of the variables
col.vars a vector of integers giving the numbers of the variables, or a character vector giving the names of the variables

```
HairEyeColor
dim(HairEyeColor)
dimnames(HairEyeColor)
rownames(HairEyeColor)
colnames(HairEyeColor)

# the first variable ($Hair) on the rows
ftable(HairEyeColor, row.vars = 1)
# the 2nd  variable ($Eye) on the rows
ftable(HairEyeColor, row.vars = 2)
# the 3rd  variable ($Sex) on the rows
ftable(HairEyeColor, row.vars = "Sex") # by name

# the first variable ($Hair) on the rows
# on the columns the 2nd and 3rd ($Eye, $Sex)
ftable(HairEyeColor, row.vars = 1, col.vars=c(2,3))
ftable(HairEyeColor, row.vars = "Hair", col.vars=c("Eye","Sex"))
```

Data Structures in R

```
# the first variable ($Hair) on the rows
# on the columns the 3rd and 2nd ($Sex,$Eye )
ftable(HairEyeColor, row.vars = 1, col.vars=c(3,2))
ftable(HairEyeColor, row.vars = "Hair", col.vars=c("Sex","Eye"))

# the first and 2nd variables ($Hair,$Eye) on the rows
ftable(HairEyeColor, row.vars = 1:2)

# the first, 2nd and 3rd variables ($Hair,$Eye,$Sex) on the rows
ftable(HairEyeColor, row.vars = 1:3)
```

Data Structures in R

sweep return an array obtained from an input array by sweeping out a summary statistic

sweep(x, MARGIN, STATS, FUN="-", check.margin=TRUE, ...)

x an array
MARGIN a vector of indices giving the extents of x which correspond to STATS
STATS the summary statistic which is to be swept out
FUN the function to be used to carry out the sweep
check.margin If TRUE (the default), warn if the length or dimensions of STATS do not match the specified dimensions of x
... optional arguments to FUN

```
attitude
med.att <- apply(attitude, 2, median)
med.att
sweep(data.matrix(attitude), 2, med.att)# subtract the column medians
```

Data Structures in R

Attach a set of R objects to the search path

attach(*what*, pos = 2, name = deparse(substitute(what)), warn.conflicts = TRUE)

Arguments
what a data.frame, list, R data file or an environment
pos position in search() where to attach
name name to use for the attached database
warn.conflicts if true then it shows conflicts from attaching the database

Objects on *what* will be accessible directly through their names

Data Structures in R

```
mydataf <-
data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),city=c("Turk
u","Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))
mydataf
ls()
search()
attach(mydataf)
ls()
search()
detach(mydataf)
ls()
search()
rm(mydataf)
ls()
search()
```

Data Structures in R

```
> mydataf <-
data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),city=c("Turku","Espoo",
"Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))
> ls()
[1] "mydataf"
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
> attach(mydataf)
> ls()
[1] "mydataf"
> search()
 [1] ".GlobalEnv"        "mydataf"           "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
> detach(mydataf)
> ls()
[1] "mydataf"
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
> rm(mydataf)
> ls()
character(0)
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Data Structures in R

Accessing attached elements from a dataframe

```
mydataf <-
data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),city=c("Turk
u","Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))

> mydataf$weight
[1] 75 55
> weight
Error: object 'weight' not found


> attach(mydataf) # attach mydataf to the search path
> weight
[1] 75 55
> city
[1] Turku Espoo
Levels: Espoo Turku
> detach(mydataf) #  detach mydataf from the search path
> weight
Error: object 'weight' not found
```

Data Structures in R

Accessing attached elements from a dataset

```
data()
ToothGrowth
names(ToothGrowth)  #     len supp dose
supp # error
ToothGrowth$supp
attach(ToothGrowth)
supp
detach(ToothGrowth)
supp
```

Data Structures in R
References/to learn more:

The R book
Michael J. Crawley pp 15
2012 John Wiley & Sons Ltd

Basic statistics using R pp. 40
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics: an introduction using R
Michael J. Crawley pp 288
2010 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 34
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/a30d9f0c6a5ca66fdee17e6088a070ad

Introductory Statistics with R
Peter Dalgaard, pp 11
2013 Springer

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/vectors.r

Quick-R
Rob Kabacoff
http://www.statmethods.net/input/datatypes.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

Displaying data on R

dir() and **list.files()** lists the files in a directory

list.files(path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE, recursive = FALSE, ignore.case = FALSE)

dir(path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE, recursive = FALSE, ignore.case = FALSE)

path vector of full path names; the default is the working directory getwd()
pattern match an optional regular expression. Not wildcards!
all.files If TRUE, all file names will be returned, even hidden and system files or not visible for other reason
full.names If TRUE, the directory path is prepended to the file names
recursive logical If TRUE, the listing will recurse into sub-directories
ignore.case If TRUE, the search will be case-insensitive. It's always case-insensitive on Windows

Note:

R.home() # the path of R's home directory

Displaying data on R

```
dir()
list.files()
dir(pattern="^a") # list all files that start with "a"
dir(pattern="\\.R$") # list all files that end with ".R"
list.files(path = "c:/temp") # list all files from c:\temp
list.files(path = "c:/temp", all.files =TRUE) # list all files from c:\temp, even not
visible files
```

Displaying data on R

file.show() display one or more files, usually text files

file.show(..., header = rep("", nfiles), title = "R Information", delete.file = FALSE, pager = getOption("pager"), encoding = "")

... one or more character vectors containing the names of the files
header character vector (of the same length as the number of files specified in ...) giving a header for each file
title an overall title for the display
delete.file should the files be deleted after display? Used for temporary files
pager the pager to be used
encoding character string giving the encoding to be assumed for the file(s)

```
dir()
file.show(".Rhistory")
```

Displaying data on R

print displays values, expressions or variables

```
print(123) # displaying a number
print("abc") # displaying a string
print(123+567) # displaying an expression
```

cat concatenates and outputs objects
By default it will output to the Console (screen)

```
cat(... , sep = " ", fill = FALSE, labels = NULL)
```

... R objects
sep a character vector of strings to append after each element
fill a logical or (positive) numeric controlling how the output is broken into successive lines. If FALSE (default), only newlines created explicitly by "\n" are printed. Otherwise, the output is broken into lines with print width equal to the option width if fill is TRUE, or the value of fill if this is numeric. Non-positive fill values are ignored, with a warning.
labels character vector of labels for the lines printed. Ignored if fill is FALSE.

```
cat(5,"*",12,"=",5*12,"\n")
cat(5,"*",12,"=",5*12,"\n", sep = "_")
cat(5,"*",12,"=",5*12,"\n", sep = "   ")
cat(rep("0123456789",20), fill = T, width=3, labels = c("line 1","line 2","line 3","line 4"))
```

Displaying data on R

paste concatenate vectors to strings

```
paste(..., sep = " ", collapse = NULL)
```

... one or more R objects, to be converted to character vectors
sep a character string to separate the terms. Not NA_character_
collapse an optional character string to separate the results. Not NA_character_

```
paste(1:3) # same as as.character(1:3)
paste(1:3,sep = "") # separate terms - only 1 term, nothing to do
paste(c("one","two","three"))
paste(c("one","two","three"), sep = "")# separate terms - only 1 term, nothing to do
paste(c("one","two","three"), collapse="***") # separate results OK

paste(1,2,3)
paste(1,2,3,sep = "") # separate terms OK
paste("one","two","three")
paste("one","two","three", sep = "") # separate terms OK
paste("one","two","three", collapse="***") # separate results - only 1 result, nothing to do
```

Displaying data on R

Write write data to a connection or file

```
write(x, file = "data", ncolumns = if(is.character(x)) 1 else 5, sep = " ")
```

Arguments
x the data to be written out
file If "", print to the standard output connection
ncolumns the number of columns to write the data in
sep a string used to separate columns. Using sep = "\t" gives tab delimited output; default is " "

```
write("hello", file="")
write(1:10, file="")
write(c("one","two","three"), file="")

write(1:10, file="", sep = "")
write(c("one","two","three"), file="", sep = "")

write(1:10, file="", ncolumns = 3)
write(c("one","two","three"), file="", ncolumns = 2)
```

Redirecting data on R

sink redirects R output to a connection

sink.number() displays the number of current redirections

```
sink(file = NULL, append = FALSE, type = c("output", "message"),
     split = FALSE)
```

```
sink.number(type = c("output", "message"))
```

file a connection or a file name or NULL to stop
append If TRUE, output will be appended, otherwise, it will be overwritten
type either output stream or the messages stream
split if TRUE, output will be sent to both new and old streams

```
sink("output.txt") # creates a file to store the output
# the output will now be sent to file "output.txt"
print("Hello world!")
print(123*pi)
sink() # stop sending the output to the file
```

Editing data on R

edit invokes a text editor on an R object

edit(name = NULL, file = "", title = NULL, editor = getOption("editor"), ...)

```
vi(name = NULL, file = "")
emacs(name = NULL, file = "")
pico(name = NULL, file = "")
xemacs(name = NULL, file = "")
xedit(name = NULL, file = "")
```

Invoking a specific editor

name R object or file name to edit
file file name
title a title for the object
editor text editor to use
... further arguments to be passed to or from methods

Editing data on R

```
# open a file
dir()
edit(file="output.txt")

v1 <- c(734, 985, 43, 952)
v2 <- c("Helsinki","Tampere","Turku")
v3 <- c(T,F,F,F,T,F,T,F,T,T)
myarray<-array(c(56,67,174,166,75,55,77,70,64,77,178,170,78,59,63,61),c(2,4,2))
dimnames(myarray) = list(c("men","women"),c("age","height","weight","pulse"),
c("city","countryside"))
mydataf <-
data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),city=c("Turku","
Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))
mymatrix <- matrix(1:6*10,3,2)

# open the R Editor with the code to define the variables
edit(v1)
edit(v2)
edit(v3)
edit(myarray)

# open the R Data Editor
edit(mydataf)
edit(mymatrix)
```

Editing data on R

Fix an R object

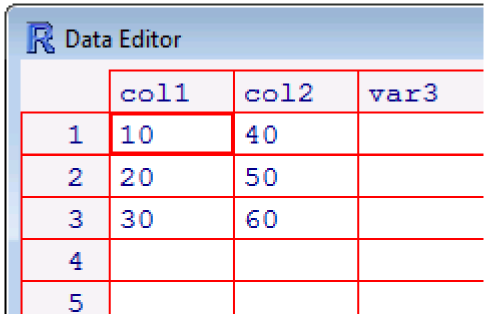
fix invokes edit on x and then updates x in the user's workspace

fix(x, ...)

x the name of an R object
... arguments to pass to edit

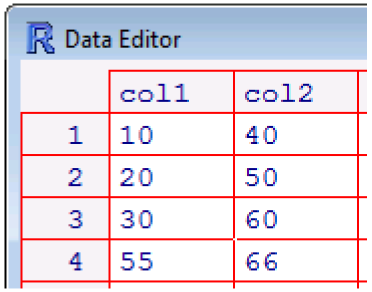
Editing data on R

```
> mymatrix <- matrix(1:6*10,3,2)
> mymatrix
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> edit(mymatrix)
     col1 col2
[1,]   10   40
[2,]   20   50
[3,]   30   60
[4,]   55   66
> mymatrix
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> fix(mymatrix)
> mymatrix
     col1 col2
[1,]   10   40
[2,]   20   50
[3,]   30   60
[4,]   55   77
```



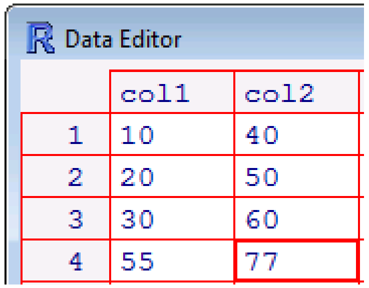
Data Editor

	col1	col2	var3
1	10	40	
2	20	50	
3	30	60	
4			
5			



Data Editor

	col1	col2
1	10	40
2	20	50
3	30	60
4	55	66



Data Editor

	col1	col2
1	10	40
2	20	50
3	30	60
4	55	77

Data input on R

Reading input from the console

```
> mydata1 <- scan()
1: 1
2: 2
3: 3
4:
Read 3 items
> mydata1
[1] 1 2 3
> is(mydata1)
[1] "numeric" "vector"
> mydata1 <- scan()
1: 1 2 3
4:
Read 3 items
> mydata1
[1] 1 2 3
> is(mydata1)
[1] "numeric" "vector"
```

Type
1 Enter 2 Enter 3 Enter Enter

Type
1 Space 2 Space 3 Enter Enter

Same input

Data input on R

Trying to enter text as input:

```
> mydata1 <- scan()
1: a
1: b
Error in scan(file, what, nmax, sep, dec, quote, skip,
nlines, na.strings,  :
  scan() expected 'a real', got 'a'
```

The default input data type is numeric, solution: use the *what* argument

Useful arguments for console input

what input data type logical, integer, numeric, complex, character

nmax maximum number of input values

nlines maximum number of lines of data

na.strings vector of elements interpreted as missing (NA) values

Data input on R

what input data type logical, integer, numeric, complex, character

```
> mydata1 <- scan(what=character())
1: a b c
4:
Read 3 items
> mydata1
[1] "a" "b" "c"
> mydata1 <- scan(what=character())
1: "one two" "three four"
3:
Read 2 items
> mydata1
[1] "one two"    "three four"
```

Quotes allow space on strings

```
> mydata1 <- scan(what=logical())
1: T
2: F
3: TRUE
4: FALSE
5:
Read 4 items
> mydata1
[1]  TRUE FALSE  TRUE FALSE
```

Logical will not accept 1 or 0

```
> mydata1 <- scan(what=complex())
1: -6576.9898
2: 3.54i
3: -5.543-.68767i
4:
Read 3 items
> mydata1
[1] -6576.990+0.00000i     0.000+3.54000i
-5.543-0.68767i
```

5i and 5*i are not the same

```
> mydata1 <- scan()
1: 25*pi
1: 15
Error in scan(file, what, nmax, sep, dec, quote,
skip, nlines, na.strings,  :
  scan() expected 'a real', got '25*pi'
> mydata1 <- scan()
1: sqrt(2)
1: 5
Error in scan(file, what, nmax, sep, dec, quote,
skip, nlines, na.strings,  :
  scan() expected 'a real', got 'sqrt(2)'
```

Constants and functions are not allowed

Data input on R

nmax maximum number of input values

nlines maximum number of lines of data

```
> mydata1 <- scan(nmax=4)
1: 9
2: 8
3: 7
4: 6
Read 4 items
> mydata1
[1] 9 8 7 6
> mydata1 <- scan(nmax=4)
1: 22 33 44 55 66 77 88 99
Read 4 items
> mydata1
[1] 22 33 44 55
```

```
> mydata1 <- scan(nlines=4)
1: 11
2: 22
3: 33
4: 44
Read 4 items
> mydata1
[1] 11 22 33 44
> mydata1 <- scan(nlines=4)
1: 1 2 3 4 5 6 7 8
9: 99 88 77
12: 66 55 44
15: 33 22 11
Read 17 items
> mydata1
 [1]  1  2  3  4  5  6  7  8 99 88 77 66 55 44 33 22 11
```

Data input on R

na.strings vector of elements interpreted as missing (NA) values

```
> mydata1 <- scan()
1: 1 2 NA 3 NA NA 4 5
9:
Read 8 items
> mydata1
[1]  1  2 NA  3 NA NA  4  5
> mydata1 <- scan(na.strings="*")
1: 9 8 * * 7 *
7:
Read 6 items
> mydata1
[1]  9  8 NA NA  7 NA
```

Data input on R

Reading input from the the web

```
# read a text file from the web to a string
con <- url("http://www.rni.helsinki.fi/~pek/r-koulutus/e2.dat") # open a connection
mytxt <- readLines(con) # read the file
close(con) # close the connection
mytxt

# execute code from the web
source("http://www.rni.helsinki.fi/~pek/r-koulutus/hello.R")

# download a file from the web
download.file("http://www.rni.helsinki.fi/~pek/r-koulutus/hello.R",destfile="hello.R")
download.file("http://www.rni.helsinki.fi/~pek/r-koulutus/e2.dat",destfile="e2.dat")
dir() # show the files on the working directory

# reading a data frame from the web
mydf <- read.table(url('http://www.rni.helsinki.fi/~pek/s-tools/e1.dat'))
mydf
class(mydf)
```

Data input on R

Reading input from a file

Comma-separated values (CSV) files
Text files containing data, each line of text is a row (record) of data and within each line, each column (field) of data is separated by a comma. Usually the first line has the column names.

Rules	Exceptions
•Records are separated by end-of-line characters •Fields are separated by commas •Leading or trailing spaces are part of the field data •Commas within fields are enclosed with double-quotes •double-quotes within fields are replaced by a pair of double-quotes •The first line might have the column names	•Line breaks can be placed inside double quotes •If the comma is used as a decimal sign then semicolons will separate the columns •Some implementations remove leading or trailing spaces •Some implementations enclose fields with leading or trailing spaces, within double-quotes •Some implementations enclose all fields within double-quotes

Data input on R

Reading input from a file

Tab delimited values (TAB) files
Text files containing data, each line of text is a row (record) of data and within each line, each column (field) of data is separated by a tab (ASCII 10). Usually the first line has the column names.

Rules

- Records are separated by end-of-line characters
- Fields are separated by tab
- Leading or trailing spaces are part of the field data
- There are no tabs within fields
- Line breaks can be placed within fields
- The first line might have the column names

Data input on R

Reading input from a file

Fixed Width Text Files
Text files containing data, each line of text is a row (record) of data and within each line, each column (field) of data has a constant, pre-defined number of characters. Usually the first line has the column names.

Rules

- Records are separated by end-of-line characters
- Fields have a fixed size
- Leading or trailing spaces are used as padding, unless anothe character is chosen for that purpose
- Line breaks can be placed within fields
- The first line might have the column names

Data input on R

Reading input from a file

File "hello.R" from the previous example should be on the working directory, "e2.dat" too.

```
# To read and execute it:
source("hello.R")
# To open a window for choosing a file to open:
source( file.choose() )

# read a text file from a file to a string
con <- file("e2.dat") # open a connection
mytxt <- readLines(con) # read the file
close(con) # close the connection
mytxt
```

Data input on R

Reading input from a file

scan()

Useful arguments for file input

what input data type logical, integer, numeric, complex, character, list

nmax maximum number of input values

nlines maximum number of lines of data

na.strings vector of elements interpreted as missing (NA) values

sep character that delimits fields, the default is white-space or end-of-line (unless within quotes)

dec decimal point character because of "." vs ","

skip the number of lines to skip from the beginning of the file

blank.lines.skip if true then blank lines are skipped

comment.char a character that marks comment lines, which are skipped

Data input on R

Reading input from a file

```
> mytxt <- scan("e2.dat") # read the file
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  scan() expected 'a real', got '#'

> mytxt <- scan("e2.dat", skip=2) # read the file, skip the 1st 2 lines
Read 14 items
> mytxt
 [1]  46 148  54 182  48 173  50 166  44 109  42 141  52 166

> mytxt <- scan("e2.dat", what = list("",""), skip=2)
Read 7 records
> mytxt
[[1]]
[1] "46" "54" "48" "50" "44" "42" "52"

[[2]]
[1] "148" "182" "173" "166" "109" "141" "166"
```

Data input on R

Reading input from a file

sep character that delimits fields, the default is white-space or end-of-line (unless within quotes)

blank.lines.skip if true then blank lines are skipped

comment.char a character that marks comment lines, which are skipped

```
> cat("12:34:56:78:90",file="numbers.txt") # create a text file with text "12:34:56:78:90"
> edit(file="numbers.txt")
> mytxt <- scan("numbers.txt", sep=":") # read the file
Read 5 items
> mytxt
[1] 12 34 56 78 90
>
> cat("12:34\n56:78:90",file="numbers.txt") # end-of-line also works
> edit(file="numbers.txt")
> mytxt <- scan("numbers.txt", sep=":") # read the file
Read 5 items
> mytxt
[1] 12 34 56 78 90
```

Data input on R

Reading input from a file

```
> # suppose that % is the symbol for lines with comments
> cat("12:ab\n% this is a comment between lines of data\n56:cd",file="numbers.txt")
> mytxt <- scan("numbers.txt", sep=":", what = list("","")) # read the file
Read 3 records
Warning message:
In scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  number of items read is not a multiple of the number of columns
> mytxt
[[1]]
[1] "12"
[2] "% this is a comment between lines of data"
[3] "cd"

[[2]]
[1] "ab" "56" ""

> # the comment was read as data, that is wrong
> #this is the correct way
> mytxt <- scan("numbers.txt", sep=":", what = list("",""), comment.char="%") # read the file
Read 2 records
> mytxt
[[1]]
[1] "12" "56"

[[2]]
[1] "ab" "cd"
```

Data input on R

Reading input from a file

read.table()

header if true, the first line of the file contains the names of the variables

sep character that delimits fields, the default is white-space or end-of-line (unless within quotes)

dec decimal point character because of "." vs ","

row.names a vector with the row names or the number of the column with the row names or the name of the column with the row names

col.names a vector of optional names for the variables ???

na.strings vector of elements interpreted as missing (NA) values

nrows maximum number of rows read

blank.lines.skip if true then blank lines are skipped

comment.char a character that marks comment lines, which are skipped

Data input on R

Reading input from a file

read.table() Reads a text file in table format and creates a data frame from it

read.csv(file, header = TRUE, sep = ",", quote="\"", dec=".", fill = TRUE, comment.char="", ...)	read comma separated value files (CSV)
read.csv2(file, header = TRUE, sep = ";", quote="\"", dec=",", fill = TRUE, comment.char="", ...)	CSV with comma as decimal point and a semicolon as field separator
read.delim(file, header = TRUE, sep = "\t", quote="\"", dec=".", fill = TRUE, comment.char="", ...)	read TAB delimited files (TAB)
read.delim2(file, header = TRUE, sep = "\t", quote="\"", dec=",", fill = TRUE, comment.char="", ...)	TAB with comma as decimal point
read.fwf(file, widths, header = FALSE, sep = "\t", skip = 0, row.names, col.names, n = -1, buffersize = 2000, ...)	read a table of fixed width formatted data

Data input on R

```
mydataf <- data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),
city=c("Turku","Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari Wan","Tove"))
mydataf

# saving with default values: separator = space, strings within "" and row names on the 1st line
write.table(mydataf, file = "z.txt")
edit(file="z.txt")
mydataf2 <- read.table("z.txt")
mydataf2

# saving as CSV
write.csv(mydataf, file = "z.csv")
edit(file="z.csv")
mydataf2 <- read.table("z.csv")
mydataf2
mydataf2 <- read.csv("z.csv")
mydataf2

# saving as TAB-delimited
write.table(mydataf, file = "z.tab", sep="\t")
edit(file="z.tab")
mydataf2 <- read.table("z.tab", sep="\t")
mydataf2

# saving as TAB-delimited, no "", no row names
write.table(mydataf, file = "z.tab", sep="\t",quote=F,row.names=F)
edit(file="z.tab")
mydataf2 <- read.table("z.tab", sep="\t",quote="")
mydataf2
```

Data input on R

read.ftable(), write.ftable() read, write "flat" contingency tables

Usage
read.ftable(file, sep = "", quote = "\"", row.var.names, col.vars, skip = 0)

write.ftable(x, file = "", quote = TRUE, append = FALSE, digits = getOption("digits"))

file either a character string naming a file or a connection which the data are to be read from or written to

sep character that delimits fields, the default is white-space or end-of-line (unless within quotes)

quote a character string giving the set of quoting characters for read.ftable

row.var.names a character vector with the names of the row variables

col.vars a list giving the names and levels of the column variables

skip the number of lines of the data file to skip before beginning to read data

x an object of class "ftable"

append If TRUE, the output from write.ftable is appended to the file

digits an integer giving the number of significant digits

Data input on R

Write an object to a file in ASCII format or read an object from a file

dget() and dput()

dget(filename) reads an R object from file "filename"
dput(obj, filename) writes an object "obj" to a file "filename", in ASCII format

```
> mydataf <- data.frame(age=c(25,22),height=c(174,166),weight=c(75,55),
city=c("Turku","Espoo"),row.names =c("Pekka","Anna"))
> dput(mydataf,"mydf.dat")
> mydataf2 <- dget("mydf.dat")
> mydataf2  == mydataf
      age height weight city
Pekka TRUE   TRUE   TRUE TRUE
Anna  TRUE   TRUE   TRUE TRUE
```

Data input on R

dump(list, file = "dumpdata.R", append = FALSE, control = "all", envir = parent.frame(), evaluate = TRUE)

list vector wi names of one or more R objects to be dumped.
file either a character string naming a file or a connection. "" indicates output to the console.
append if TRUE and file is a character string, output will be appended to file; otherwise, it will overwrite the contents of file.

source() reads R code from a file or a connection

source(file, local = FALSE, echo = verbose, print.eval = echo, verbose = getOption("verbose"), prompt.echo = getOption("prompt"), max.deparse.length = 150, chdir = FALSE, encoding = getOption("encoding"), continue.echo = getOption("continue"), skip.echo = 0, keep.source = getOption("keep.source"))

file a connection or a character string giving the pathname of the file or URL to read from
echo if TRUE, each expression is printed after parsing, before evaluation
print.eval if TRUE, the result of eval(i) is printed for each expression i; defaults to the value of echo
verbose if TRUE, more diagnostics (than just echo = TRUE) are printed during parsing and evaluation of input, including extra info for each expression
prompt.echo character; gives the prompt to be used if echo = TRUE
encoding The encoding(s) to be assumed when file is a character string: see file
skip.echo if echo = TRUE, how many lines to skip from the beginning

Data input on R

```
a <- 543.86
dump(a, "test_a.R") # error!
dump(ls(pattern ="a"), "test_a.R") # works...
dir(pattern="test")
ls()
rm(a)
a
ls()
source("test_a.R")
a
ls()
```

```
# to choose a file interactively:
source(file.choose())
```

Data input on R

```
mydataf <-
data.frame(age=c(25,22,26,28),height=c(174,166,174,170),weight=c(75,55,60,60),city=c("Turk
u","Espoo","Kuopio","Helsinki"),row.names =c("Pekka","Anna","Ari","Tove"))
dump(ls("mydataf"), file ="test_mydataf.R") # error!
ls()
search()
attach(mydataf)
ls()
search()
dump(ls("mydataf"), file ="test_mydataf.R")
dir(pattern="test")
detach(mydataf)
rm(mydataf)
ls()
search()
source("test_mydataf.R")
ls()
search() # "age"    "city"   "height" "weight"!
mydataf # got attached!
edit(file="test_mydataf.R") # this is why!
```

Data input on R

Write write data to a connection or file

```
write(x, file = "data", ncolumns = if(is.character(x)) 1 else 5, sep = " ")
```

Arguments
x the data to be written out
file If "", print to the standard output connection
ncolumns the number of columns to write the data in
sep a string used to separate columns. Using sep = "\t" gives tab delimited output; default is " "

```
write("hello", file="hello.txt")
write(1:10, file="1to10.txt")
write(c("one","two","three"), file="123.txt")

write(1:10, file="1to10b.txt", sep = "")
write(c("one","two","three"), file="123b.txt", sep = "")

write(1:10, file="1to10c.txt", ncolumns = 3)
write(c("one","two","three"), file="123c.txt", ncolumns = 2)

myvector<-c(1,2,3,4,5)
write(myvector,"myvector.txt")

mymatrix<-matrix(1:9,ncol=3,byrow=T)
write(t(mymatrix),"mymatrix.txt",ncol=ncol(mymatrix))
```

Data input on R

cat concatenates and outputs objects, also to a file
By default it will output to the Console (screen)

```
cat(... , file = "", sep = " ", fill = FALSE, labels = NULL, append = FALSE)
```

... R objects
file file name to get the output
sep a character vector of strings to append after each element
fill a logical or (positive) numeric controlling how the output is broken into successive lines. If FALSE (default), only newlines created explicitly by "\n" are printed. Otherwise, the output is broken into lines with print width equal to the option width if fill is TRUE, or the value of fill if this is numeric. Non-positive fill values are ignored, with a warning.
labels character vector of labels for the lines printed. Ignored if fill is FALSE.
append if TRUE, the outpur is appended at the end of the file

```
cat("Hello world!", file = "cattest.txt")
edit(file="cattest.txt")
cat("Hello aliens!", file = "cattest.txt", append = TRUE)
edit(file="cattest.txt")
```

Data input on R

save() saves R objects

```
save(..., list = character(0L),
     file = stop("'file' must be specified"),
     ascii = FALSE, version = NULL, envir = parent.frame(),
     compress = !ascii, eval.promises = TRUE, precheck = TRUE)
```

... the names of the objects to be saved
list A vector containing the names of objects to be saved
file a connection or the name of the file where the data will be saved
ascii if TRUE, an ASCII representation of the data is written
compress if TRUE, the filr is compressed
precheck if TRUE, the existence of the objects is checked before saving

load() loads datasets saved with save()

```
v1 <- c(734, 985, 43, 952)
v2 <- c("Helsinki","Tampere","Turku")
save(v1, v2, file = "v1v2.Rdata")
#remove all objects
rm(list=ls(all=TRUE))
v1;v2
load("v1v2.Rdata")
v1;v2
save(v1, v2, file = "v1v2.Rdata", ascii = TRUE)
edit(file="v1v2.Rdata")
```

Data input on R
References/to learn more:

The R book
Michael J. Crawley pp 97
2014 John Wiley & Sons Ltd

Basic statistics using R pp. 57
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics: an introduction using R
Michael J. Crawley pp 286
2012 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 91
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/81279218bad3be4326b943c4c3e62e4d

Introductory Statistics with R
Peter Dalgaard, pp 46
2014 Springer

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/data-input.r

Quick-R
Rob Kabacoff
http://www.statmethods.net/input/index.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

Control Flow

$$\text{Conditional statements} \left\{ \begin{array}{l} \text{if} \\ \text{ifelse} \\ \text{switch} \end{array} \right.$$

$$\text{Loop statements} \left\{ \begin{array}{l} \text{for} \\ \text{while} \\ \text{repeat} \end{array} \right.$$

Control Flow

if - Conditional statement

if(condition) expression # if the condition is true then the expression will execute
if(condition) expression else alternate.expression # if the condition is true then the expression will execute, otherwise the alternate.expression will execute

condition - a logical result, not NA. Only the first element of a vector is considered.
expression, alternate.expression - Either a simple expression, one command only, or a so called compound expression { expression1 ; expression2 } or:

```
{
expression1
expression2
}
```

To separate several expressions can be done with ; or newline but newline is more clear and understandable.
A newline before an else statement will cause an error.

```
> if (5 > 3) print ("OK") # this always returns "OK"
[1] "OK"
```

Control Flow

This code should be tested on the R editor, selecting a block if code + ctrl r

```
# get a random integer number from 1 to 10, say if it is greater than 5 or not
if (sample(1:10, 1) > 5) print ("random number > 5") else print ("random number < 5")

# same as above but in separate lines of code
if (sample(1:10, 1) > 5)
print ("random number > 5") else
print ("random number < 5")

# get a random integer number from 1 to 10, if greater than 5 then show its square value
# otherwise show it multiplied by 4
myrnd <- sample(1:10, 1)
if (myrnd > 5) {myrnd2 <- myrnd^2;print (myrnd2)} else {myrnd2 <- myrnd*4;print (myrnd2)}

# same as above but in separate lines of code
myrnd <- sample(1:10, 1)
if (myrnd > 5)
{
  myrnd2 <- myrnd^2
  print (myrnd2)
} else
{
  myrnd2 <- myrnd*4
  print (myrnd2)
}
```

Control Flow

"If" can be used as a function within expressions:

```
> x <- 5
> strwhartx <- if(is.complex(x)) "imaginary" else "real"
> strwhartx
[1] "real"
```

Control Flow

Conditional Element Selection

Ifelse - returns 1 out of 2 elements, depending on a logical condition.

ifelse(condition, condition.true.expression, condition.false.expression)

```
> x <- 5-7i
> ifelse(is.complex(x), "imaginary", "real") # if x is complex, return "imaginary", otherwise "real"
[1] "imaginary"
> x <- 16
> ifelse(is.complex(x), "imaginary", "real") # if x is complex, return "imaginary", otherwise "real"
[1] "real"
```

Control Flow

Ifelse on multiple elements

Ifelse can affect elements from vectors, matrices, etc... directly, with no need for loops or for "apply" functions

```
# get the sign (-1, 0, 1) from numbers
> ifelse(myvec >0, 1, ifelse(myvec <0, -1, 0))
[1] -1 -1 -1  0  1  1  1

# replace numbers with a word
# NA MISSING
# Inf  INFINITY
# >0 POSITIVE
# <0 NEGATIVE
# =0 ZERO
myvec <- c(-3:3, Inf, NA)
myvec
myvec <- sample(myvec) # random permutation
myvec
myvec.str <- ifelse(is.na(myvec),"MISSING",
        ifelse(is.infinite(myvec),"INFINITY",
                ifelse(myvec>0,"POSITIVE",
                        ifelse(myvec<0,"NEGATIVE",
                                "ZERO"
))))
myvec.str
```

Control Flow

switch - choose from several results depending upon an expression
It is not a statement like the C or C++ switch statement but a function, like the CASE WHEN THEN from SQL.

switch(expression, alternative1,alternative2,alternative3,alternative4,...)

```
> for(ch in c("c","k","a","B","A","b") ) print(switch(EXPR = ch,a=,A="ai",b="bee",c="see","????"))
[1] "see"
[1] "????"
[1] "ai"
[1] "????"
[1] "ai"
[1] "bee"
```

a=,A="ai" both "a" and "A" will return the same value
"????" is the default value (aka "otherwise") for values not in the alternatives' list

Numeric EXPR has no "otherwise"

```
> for(i in c(-1:3,9))  print(switch(i, 1,2,3,4))
NULL
NULL
[1] 1
[1] 2
[1] 3
NULL
```

Control Flow

for

for (var in seq) expr
break
next

for will cycle throught the elements of a vector sequentially until it reaches the last element or the *break* command is found within the loop. *Next* skips the current iteration.

```
> for(i in 4:7) print(i)
[1] 4
[1] 5
[1] 6
[1] 7
```

Loop through a sequence of numbers

```
> for(i in c(734, 985, 43, 952)) print(i)
[1] 734
[1] 985
[1] 43
[1] 952
```

Loop through a vector of numbers

```
> for(i in c("Helsinki","Tampere","Turku")) print(i)
[1] "Helsinki"
[1] "Tampere"
[1] "Turku"
```

Loop through a vector of strings

Control Flow

```
> for(i in 1:10)
+ {
+ print(i)
+ if (i==3) break
+ }
[1] 1
[1] 2
[1] 3
>
> for(i in 1:10)
+ {
+ if (i/2==i %/%2) next
+ print(i)
+ }
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
```

break exits the loop

next skips the current iteration

All the even numbers are skipped

Control Flow

while

while(cond) expr
break
next

while will test a condition and execute an expression if the condition is TRUE, then it will test the condition again and so forth.

break exits the loop
next skips the current iteration, it will cause an infinite loop unless the variable is updated before the *next* statement

Control Flow

```
# example of using while
n <- 1
while (n < 5)
{
print(n)
n <- n+1
}
# example of using while and next
n <- 0
while ((n <- n+1) < 5)
{
if (n==2) next
print(n)
}
```

The loop variable must be updated before reaching the next, otherwise the loop will be infinite

The code before *next* is executed
The code after *next* is skipped

```
# example of using while and next
n <- 0
while (n < 5)
{
n <- n+1
if (n==2) next
print(n)
}
```

The code before *next* is executed
The code after *next* is skipped

```
# example of using while and break
n <- 1
while (n < 5)
{
if (n==2) break
print(n)
n <- n+1
}
```

The code before *break* is executed
The code after *break* is skipped and the loop ends

Control Flow

```
print("Game: I will choose 3 numbers between 1 and 8, you have to guess them to win this
game")
x <- sample(1:8,3)
user.score <- c()
while (length(x)>0)
{
print("Give me a number between 1 and 8")
user.try <- scan(,what=numeric(),1)
if (user.try %in% x)
 {
 user.score <- c(user.score, user.try)
 x <- x[x != user.try]
 print("Correct!")
 }
else print("Wrong!")
if (length(x)==0)
 {
 print("You win! Now give 10 euros to the instructor and play again!")
 }
else
 {
 cat("You already guessed", ifelse((length(user.score)==0),"nothing!", paste(user.score,
collapse=", ")), " Try again!\n" )
 }
}
```

Control Flow

repeat

repeat expr
break

repeat will execute an expression and, from within that expression, test a condition, if the condition is TRUE, it will use *break* to stop, otherwise it will execute the expression again and so forth.

break exits the loop

Control Flow

```
# example of using repeat
n <- 1
repeat
{
print(n)
n <- n+1
if (n == 5) break
}

n <- 0
repeat
{
n <- n+1
print(n)
if (n == 4) break
}

# example of using repeat and next
n <- 0
repeat
{
n <- n+1
if (n == 2) next
print(n)
if (n == 4) break
}
```

The code before *break* is executed
The code after *break* is skipped and the loop ends

The code before *next* is executed
The code after *next* is skipped

The loop variable must be updated before reaching the next, otherwise the loop will be infinite

Control Flow

while vs *repeat*

while checks the conditional expression before entering the loop, it might not execute all at.

repeat executes the loop and then it checks the conditional expression anywhere from within the loop, usually, it will execute once, at least partially.

Control Flow
References/to learn more:

The R book
Michael J. Crawley pp 58
2015 John Wiley & Sons Ltd

Statistics: an introduction using R
Michael J. Crawley pp 283
2013 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 26
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/a70c8973cb8798b0bd0e6bdf7abd6ec7

Introductory Statistics with R
Peter Dalgaard, pp 44
2015 Springer

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/cond.r

Quick-R
Rob Kabacoff
http://www.statmethods.net/management/controlstructures.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

Function Syntax

R commands are R functions, syntax:

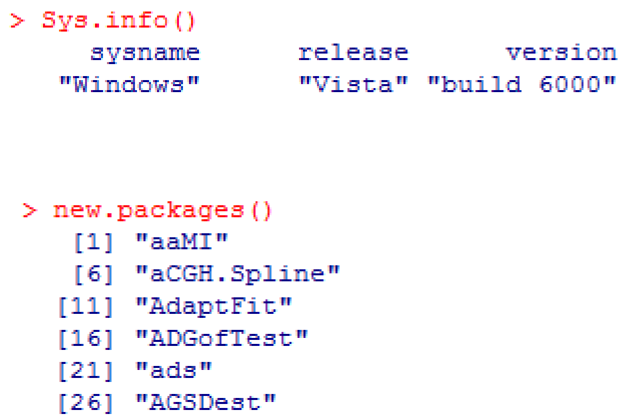
# calling a function, passing no parameters, the parenthesis are mandatory
result <- my_function()

# calling a function, passing parameters by position
result <- my_function(arg1,arg2,...,argN)

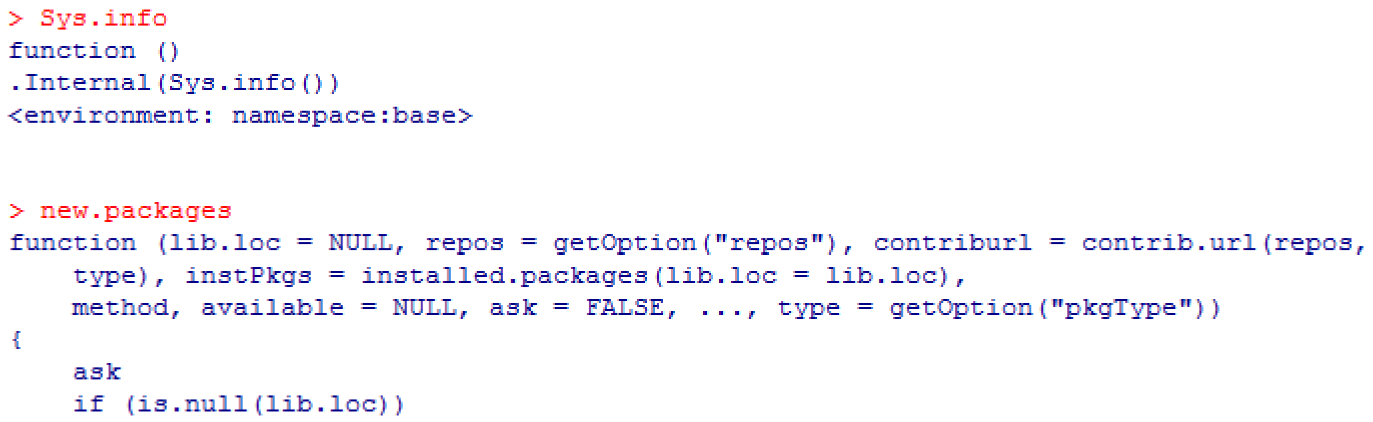
# calling a function, passing parameters by name
result <- my_function(arg_nameN=argN,arg_name1=arg1,...,arg_name2=arg2)

# calling a function, passing parameters by position and with optional parameters
result <-
my_function(arg1,arg2,...,argN,optional_arg1=value1,optional_arg2=value2,...optional_N=valueN)

Calling a function without parenthesis will return its code, unless it is an internal function.



```
> Sys.info()
    sysname       release       version
  "Windows"       "Vista" "build 6000"



 > new.packages()
    [1] "aaMI"
    [6] "aCGH.Spline"
   [11] "AdaptFit"
   [16] "ADGofTest"
   [21] "ads"
   [26] "AGSDest"
```



```
> Sys.info
function ()
.Internal(Sys.info())
<environment: namespace:base>


> new.packages
function (lib.loc = NULL, repos = getOption("repos"), contriburl = contrib.url(repos,
    type), instPkgs = installed.packages(lib.loc = lib.loc),
    method, available = NULL, ask = FALSE, ..., type = getOption("pkgType"))
{
    ask
    if (is.null(lib.loc))
```

Functions

To get the arguments of a function:

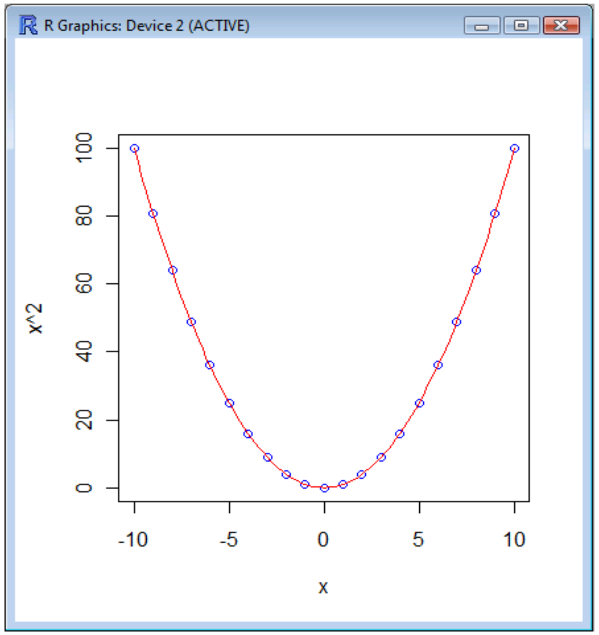
args() will return the arguments

```
args(plot) # get the arguments for function plot
args(graphics::plot) # specify package
```

Functions

y = f(x)
input(independent variable or argument)
output(dependent variable or value)

Example: quadratic function
y = f(x) = x^2
f: [-10,10] $\longrightarrow$ [0,100]
X <- -10:10
plot(x, x^2, col = "red",type="l")
points(x,x^2,col="blue")



Argument List of a Function

args() returns the argument names
and corresponding default values
of a function or primitive

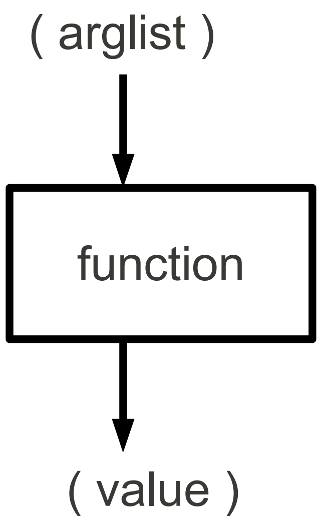
Functions

Function Definition

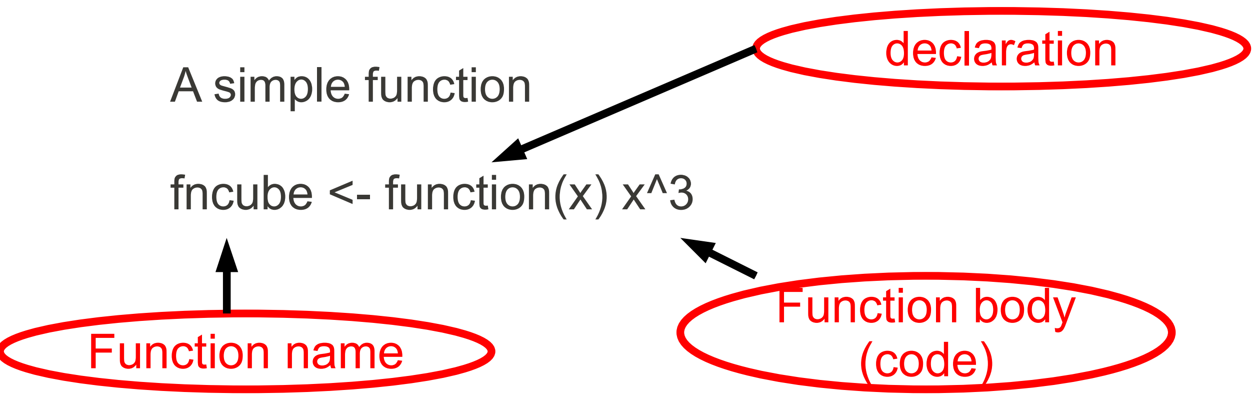
function(arglist) expr
return(value)

Arguments
arglist Empty or one or more terms
value An expression to be returned

The *return* command is unnecessary if the function end with the expression tobe returned



A simple function



```
fncube <- function(x) x^3
```

```
fncube(7)
fncube(1:5)
```

Functions

Example of using return

Improve the function fncube by returning 0 when the input is a character

```
fncube <- function(x) x^3

fncube2 <- function(x)
{
if (is.character(x)) return(0) else return(x^3)
}

fncube(123)
fncube("a")

fncube2(123)
fncube2("a")
```

Functions

A function with multiple parameters

```
fnpower <- function(x, n) x^n
fnpower(5, 3)
fnpower(1:5, 3)
fnpower(5, 1:3)
fnpower(1:8, 1:2)
c(1^1, 2^2, 3^1, 4^2, 5^1, 6^2, 7^1, 8^2)
fnpower(1:8, 1:4)
c(1^1, 2^2, 3^3, 4^4, 5^1, 6^2, 7^3, 8^4)
fnpower(1:8, 1:6) # error!
```

Functions

A recursive function

Fibonacci sequence, each element is the sum of the previous and the one before

Fibonacci F(n) = Fn-1 + Fn-2, F(0)=0, F(1)=1
n=0, 1, 2, ... F(n) = 0, 1, 1, 2, 3, 5, 8, 13, 21

```
# iterative implementation
Fibonacci <- function(v)
{
if (v<2) return(v)
t <- c(1, 1)
for (n in 3:v) t <- c(t, t[n-1]+t[n-2])
return(t[v])
}

Fibonacci2 <- function(n) ifelse(n==0, 0,ifelse(n==1 | n==2, 1, Fibonacci2(n-1)+Fibonacci2(n-
2) )  )


Fibonacci(8)
sapply(0:8,Fibonacci)
```

Functions

Default values

An argument can be optional and have a default value

```
my.foo <- function(x, y) {
return( x^3 + y*9 )
}

# calling the function
# passing arguments by position
my.foo(4, 3)
my.foo(4) # error!
# passing arguments by name
my.foo(y=3, x=4)

# default values
my.foo <- function(x, y=3) {
return( x^3 + y*9 )
}

# calling using the default value
my.foo(4)
```

Functions

Passing functions as arguments

```
# passing a function as an argument
my.foo <- function(x, y=3, foo2) {
return( foo2(x^3 + y*9) )
}

my.foo(2, 3, sin)
sin(2^3 + 3*9)


# passing a function as an argument, with parameters for that function
my.foo <- function(x,y=3, foo2, ...) {
return( foo2(x, x^3 + y*9, ...) )
}

my.foo(2:20,3, plot,col = "blue",type="l")
```

Functions

Passing an arbitrary number of arguments

```
my.foo <- function(x,y=3, ...) {
return( x^3 + y*9 +mean( ...) )
}

my.foo(2,3,76,45,43,976,34)
2^3 + 3*9 +mean(76,45,43,976,34)
```

Functions
References/to learn more:

The R book
Michael J. Crawley pp 47
2016 John Wiley & Sons Ltd

Statistics: an introduction using R
Michael J. Crawley pp 292
2014 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 27
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/a70c8973cb8798b0bd0e6bdf7abd6ec7

Introductory Statistics with R
Peter Dalgaard, pp 46
2016 Springer

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/f-own.r

Chem 351 Archives Page
David Harvey
http://fs6.depauw.edu:50080/~harvey/Chem%20351/PDF%20Files/Handouts/RDocs/Writing%20Functions%20Using%20R.pdf

Quick-R
Rob Kabacoff
http://www.statmethods.net/management/userfunctions.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

Graphics on R

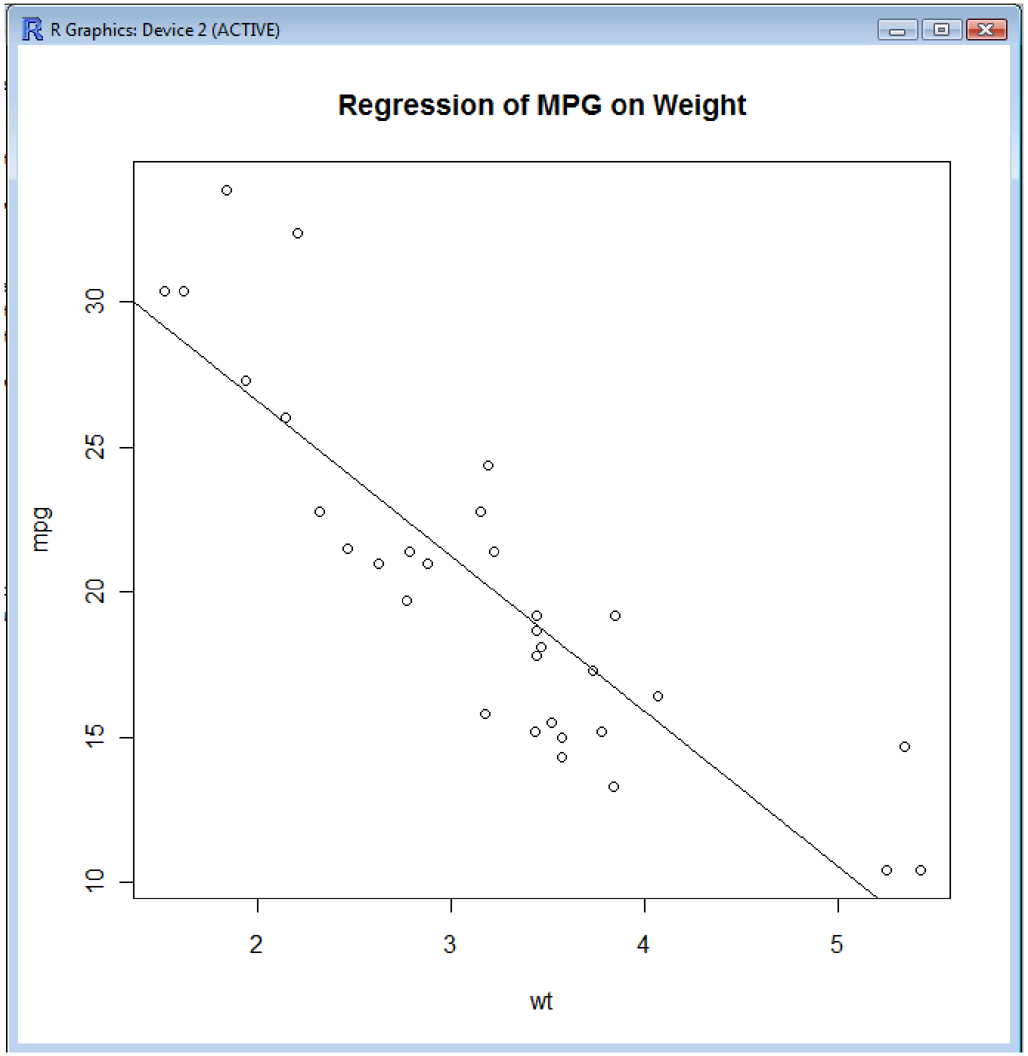
The "graphics" package contains many functions for drawing graphics

$$\text{high level functions} \left\{ \begin{array}{l} \text{plot} \\ \text{barplot} \\ \text{dotchart} \\ \text{stripchart} \\ \text{pie} \\ \text{hist} \\ \text{boxplot} \\ \text{pairs} \\ \text{stem} \\ \text{mosaicplot} \\ \text{qqnorm} \\ \text{contour} \\ \text{persp} \\ \text{image} \end{array} \right.$$

$$\begin{array}{l} \text{low level functions} \\ \text{(add details to the} \\ \text{graphic)} \end{array} \left\{ \begin{array}{l} \text{axis} \\ \text{title} \\ \text{text} \\ \text{legend} \\ \text{points} \\ \text{lines} \\ \text{abline} \\ \text{polygon} \\ \text{qqline} \end{array} \right.$$

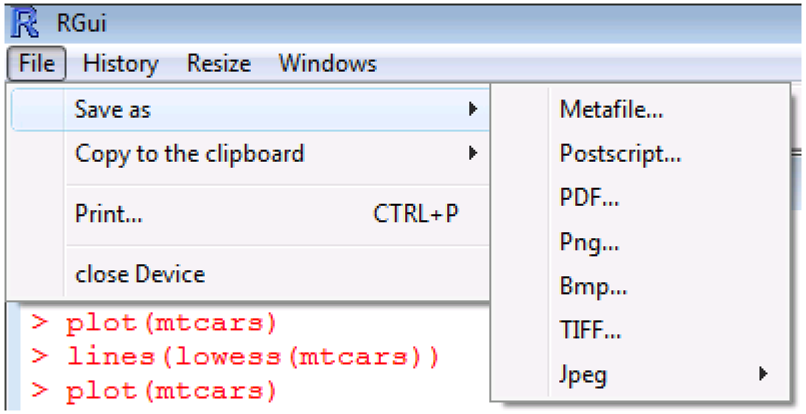
Graphics on R

```
mtcars
?mtcars

attach(mtcars)
plot(wt, mpg)
abline(lm(mpg~wt))
title("Regression of MPG on Weight")
detach(mtcars)

# instead of attach, "with" would work,
not adding to the search path
with(mtcars, {
plot(wt, mpg)
abline(lm(mpg~wt))
title("Regression of MPG on Weight")
})
```



Graphics on R

The graphic can be saved as an image file from the menu:



```
> plot(mtcars)
> lines(lowess(mtcars))
> plot(mtcars)
```

Or from code:

```
win.metafile("MPGonWeight.wmf")
postscript("MPGonWeight.ps")
pdf("MPGonWeight.pdf")
png("MPGonWeight.png")
bmp("MPGonWeight.bmp")
tiff("MPGonWeight.jpg")
jpeg("MPGonWeight.jpg")
```

Graphics on R

Plotting the sine and a parabola:

```
plot(sin, -pi, 2*pi)
plot(function(r) r^2, -pi, 2*pi)
```

The second plot will overwrite the first one

If that is not the efect wanted:

Multiple graphics

- Each graphic on a window
- All graphics on separate parts of one window
- All graphics merged together, on one window

Graphics on R

Several graphic windows (graphic devices)

To create a graphic device (different commands for different OSs), that will become the active graphic device:
windows() or win.graph()   Windows
X11()   Unix
macintosh()   Mac

The first device is device 2, then device 3, etc...

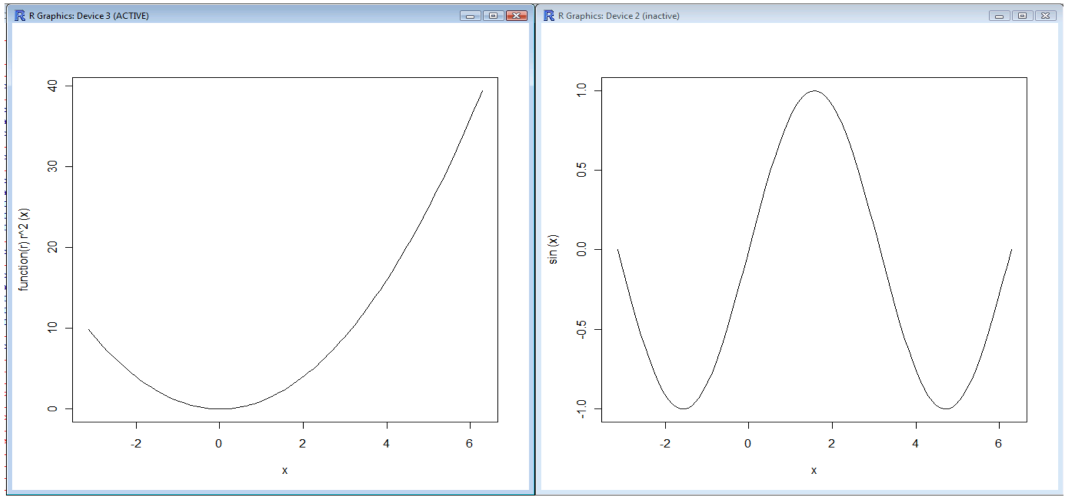
To make a graphic device the active one:
dev.set(2) # set active graphic device 2

To close the active graphic device
dev.off()

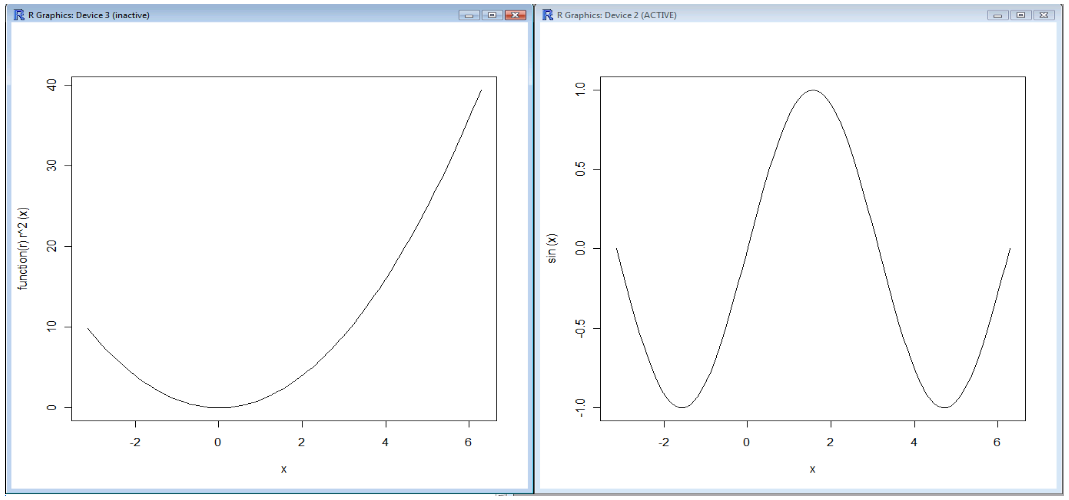
To close the graphic device 5
dev.off(5)

Graphics on R

```
plot(sin, -pi, 2*pi)
windows()
plot(function(r) r^2, -pi, 2*pi)
```

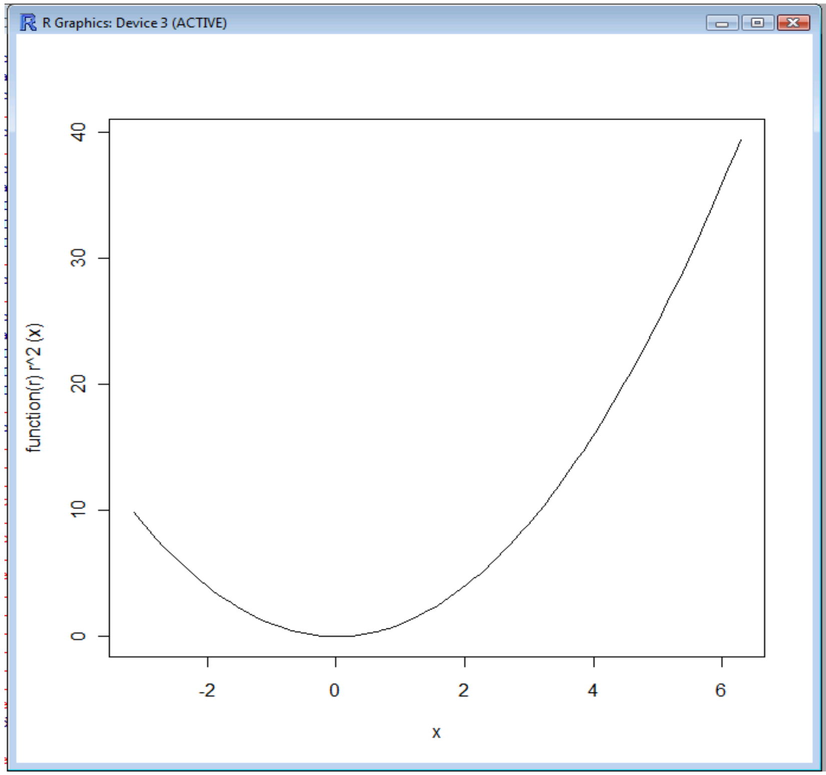

R Graphics: Device 3 (ACTIVE)
R Graphics: Device 2 (inactive)
function(r) r^2 (x)
sin (x)
x
x

```
dev.set(2) # set active graphic device 2
```


R Graphics: Device 3 (inactive)
R Graphics: Device 2 (ACTIVE)
function(r) r^2 (x)
sin (x)
x
x

Graphics on R

dev.off() # close the active graphic device



dev.off() # close the active graphic device

This graph is gone too

Graphics on R

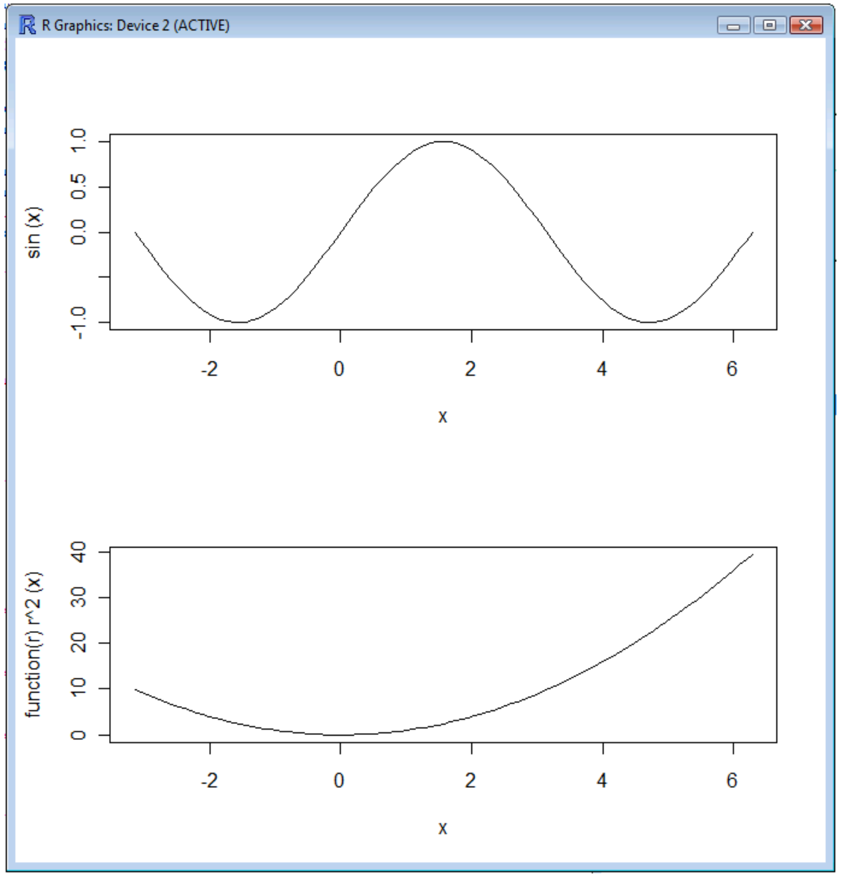
Multiple graphs in one window

par() set or query graphical parameters

Many parameters but the one needed:

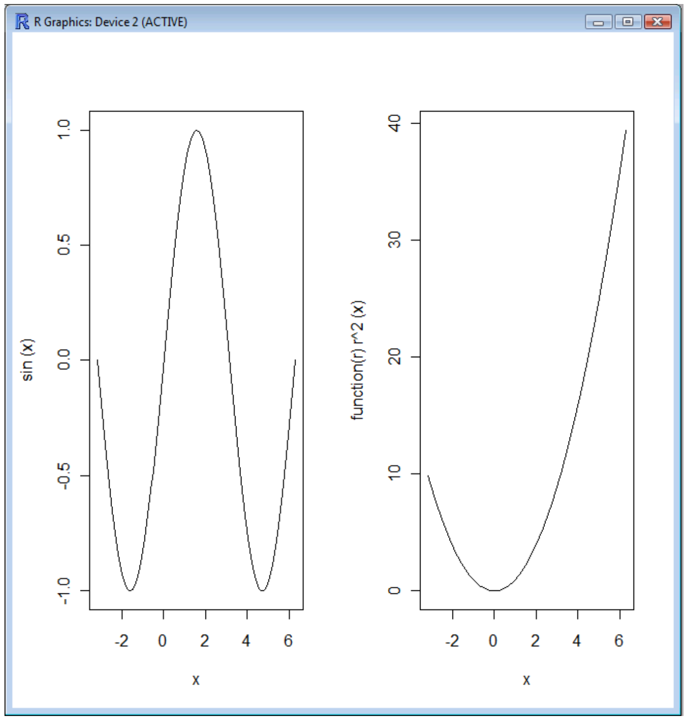
mfcol, mfrow A vector of the form c(nr, nc). Subsequent figures will be drawn in an nr-by-nc array on the device by columns (mfcol), or rows (mfrow)

```
par(mfrow=c(2,1))
plot(sin, -pi, 2*pi)
plot(function(r) r^2, -pi, 2*pi)
```

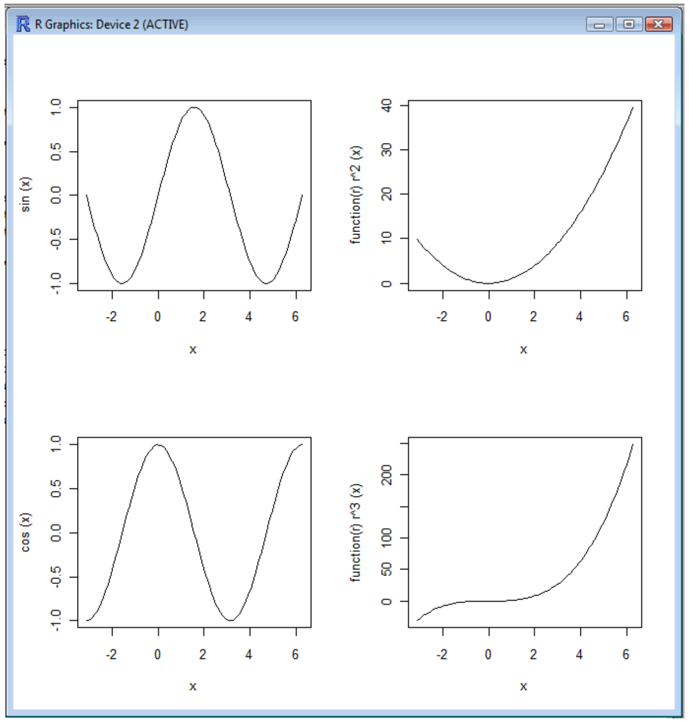


Graphics on R

```
par(mfrow=c(1,2))
plot(sin, -pi, 2*pi)
plot(function(r) r^2, -pi, 2*pi)
```



```
par(mfrow=c(2,2))
plot(sin, -pi, 2*pi)
plot(function(r) r^2, -pi, 2*pi)
plot(cos, -pi, 2*pi)
plot(function(r) r^3, -pi, 2*pi)
```



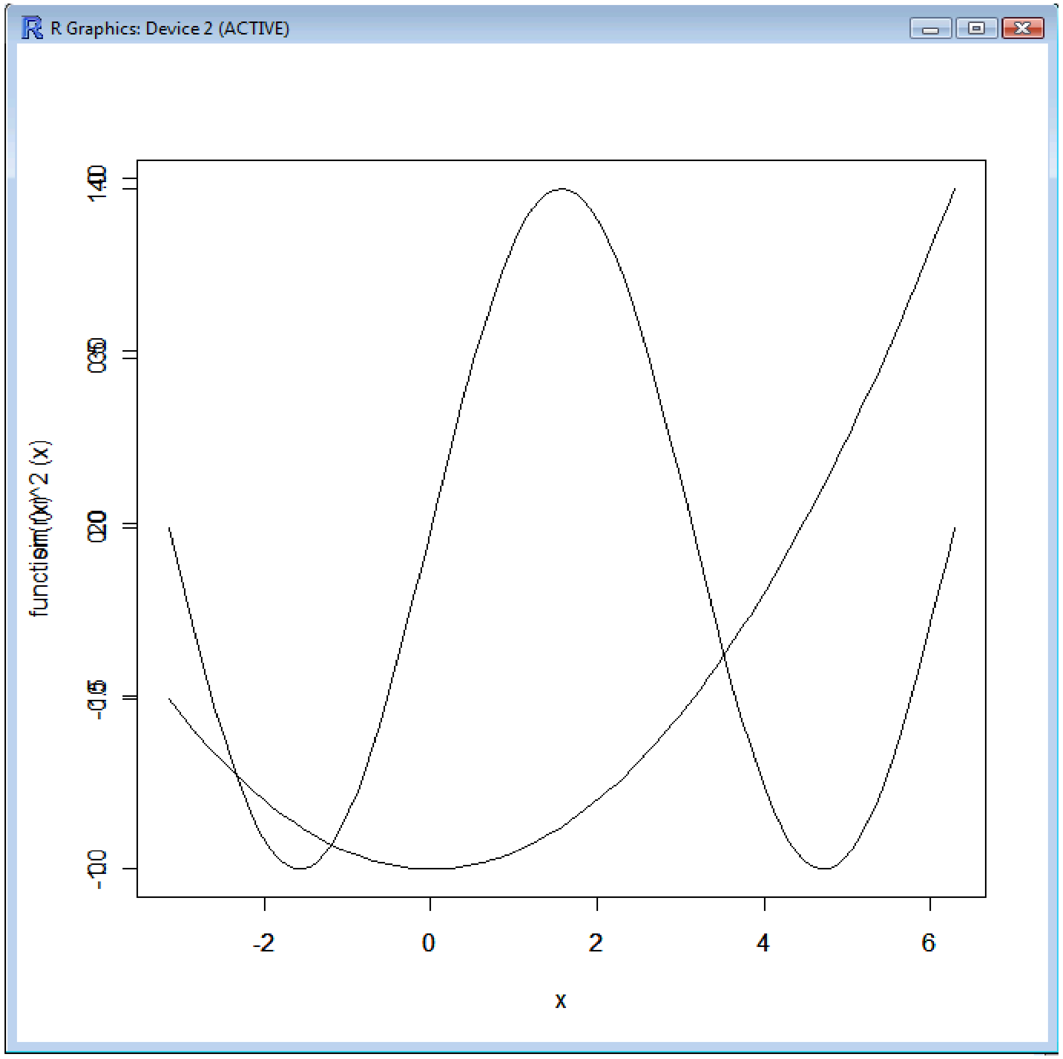
Graphics on R

graphics merged together, on one window

```
plot(sin, -pi, 2*pi)
par(new=T)
plot(function(r) r^2, -pi, 2*pi)
```

The y axis has a different title and scale, making the overlay look funny...

But the two graphs are there!



Graphics on R

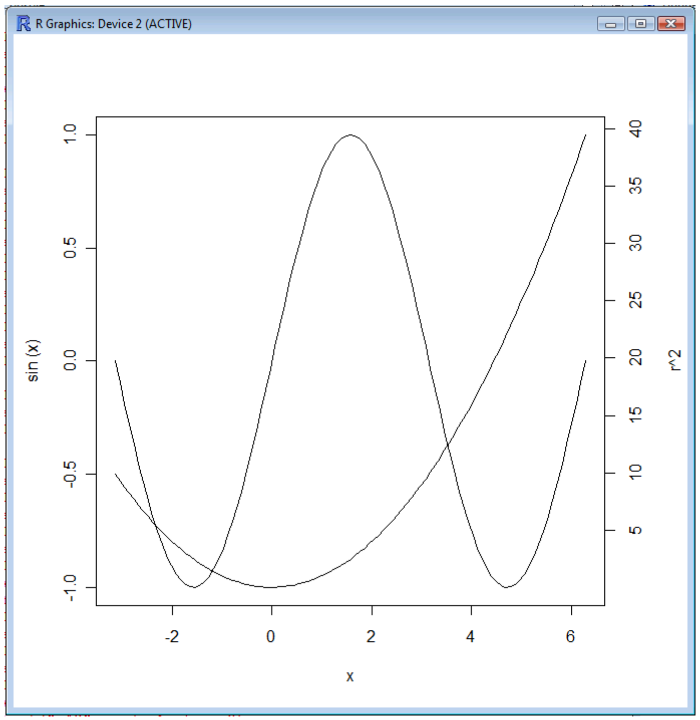
Overlaying graphs with different scales $\left\{\begin{array}{l}\text{Use different y axis}\\ \\ \text{Use the same scale}\end{array}\right.$

Graphics on R

Using different y axis on the same plot

Usually misleading, this seldom used but it is just an example of R's graphing capabilities

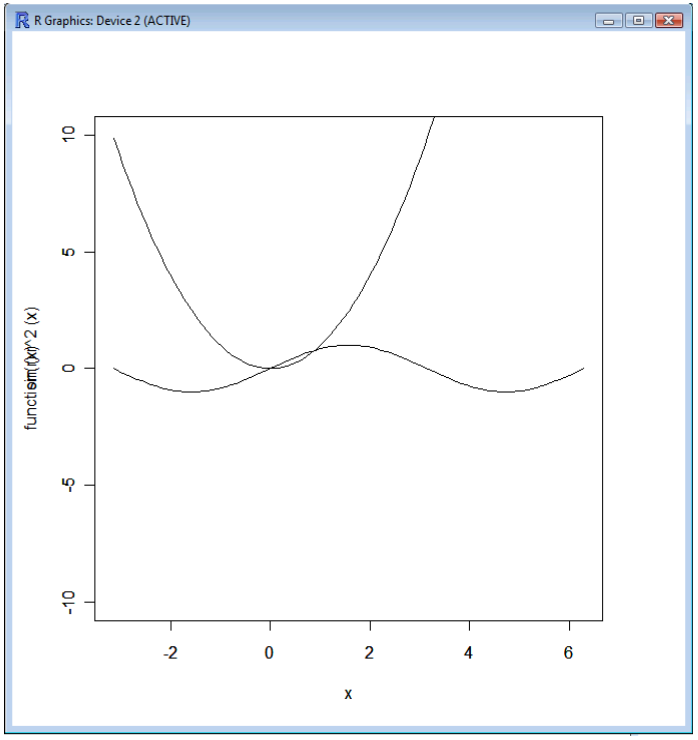
```
par(mar = c(5, 4, 4, 4) + 0.3)  # Leave space for z axis
plot(sin, -pi, 2*pi)
par(new = TRUE)
plot(function(r) r^2, -pi, 2*pi, axes = FALSE, bty = "n", xlab = "", ylab = "")
axis(side=4, at = pretty( c(pi^2, 4*pi^2) ))
mtext("r^2", side=4, line=3)
```

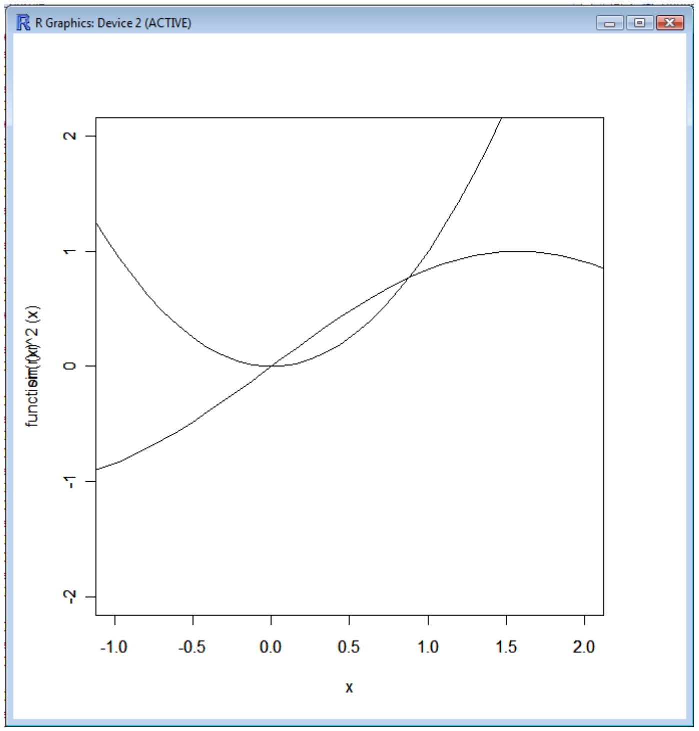


Graphics on R

Choosing a range for the x or y axis

```
plot(sin, -pi, 2*pi, ylim=c(-10,10))
par(new=T)
plot(function(r) r^2, -pi, 2*pi, ylim=c(-10,10))

plot(sin, -pi, 2*pi, xlim=c(-1,2), ylim=c(-2,2))
par(new=T)
plot(function(r) r^2, -pi, 2*pi, xlim=c(-1,2),
ylim=c(-2,2))
```





Graphics on R

plot()

The plot() function is very versatile and very useful

plot() can draw {

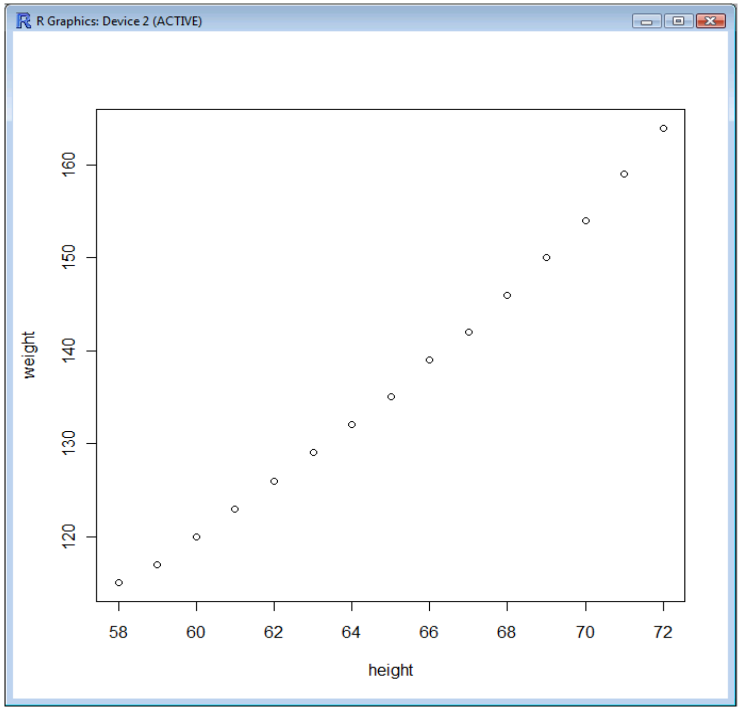
simple plot
function plot
line chart
scatterplot
density plot

Graphics on R

Simple plot

A simple plot plot(X) has each element of a discrete variable X ploted on the y-axis and the element's index on the x-axis

```
# simple plot
women
plot(women)
```



Function plot

A function plot is a simple plot for a continuous variable

```
# function plot
x = seq(-2,2)
y = x^2
# edgy graph!
plot(x,y,type="l",xlab="X axis",ylab="Y axis",main="Parabola", col = "red")
# better
sp <- spline(x, y) # spline interpolation of data points
lines(sp, col = "blue")
# much  better
sp <- spline(x, y,n=20) # interpolation at n points spanning [xmin, xmax]
lines(sp, col = "green")
```

Graphics on R

Line chart

A line chart is a simple plot with consecutive plots connected by lines

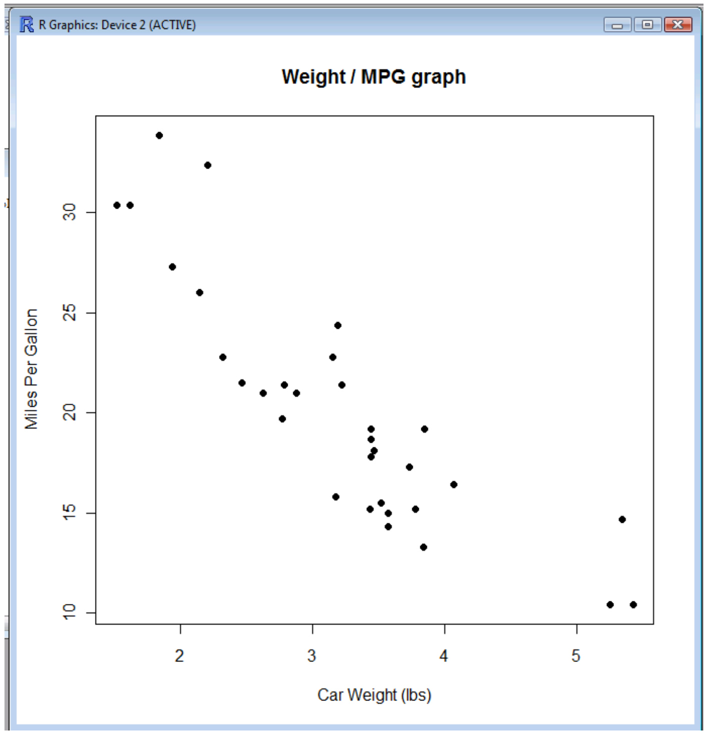
```
# line chart

x <- c(1:5); y <- x # create some data
par(pch=22, col="blue") # plotting symbol and color
par(mfrow=c(2,4)) # all plots on one page
opts = c("p","l","o","b","c","s","S","h")
for(i in 1:length(opts))
{
  heading = paste("type=",opts[i])
  plot(x, y, main=heading)
  lines(x, y, type=opts[i])
}


x <- c(1:5); y <- x^4 # create some data
par(pch=22, col="blue") # plotting symbol and color
par(mfrow=c(2,4)) # all plots on one page
opts = c("p","l","o","b","c","s","S","h")
for(i in 1:length(opts))
{
  heading = paste("type=",opts[i])
  plot(x, y, main=heading)
  lines(x, y, type=opts[i])
}
```

Graphics on R

Scatterplot

A scatterplot plot(X, Y) has each element of a variable Y ploted on the y-axis and the corresponding element for variable X on the x-axis

```
# scatterplot
attach(mtcars)
plot(wt, mpg, main="Weight / MPG graph", xlab="Car Weight (lbs)", ylab="Miles Per Gallon", pch=19)
```

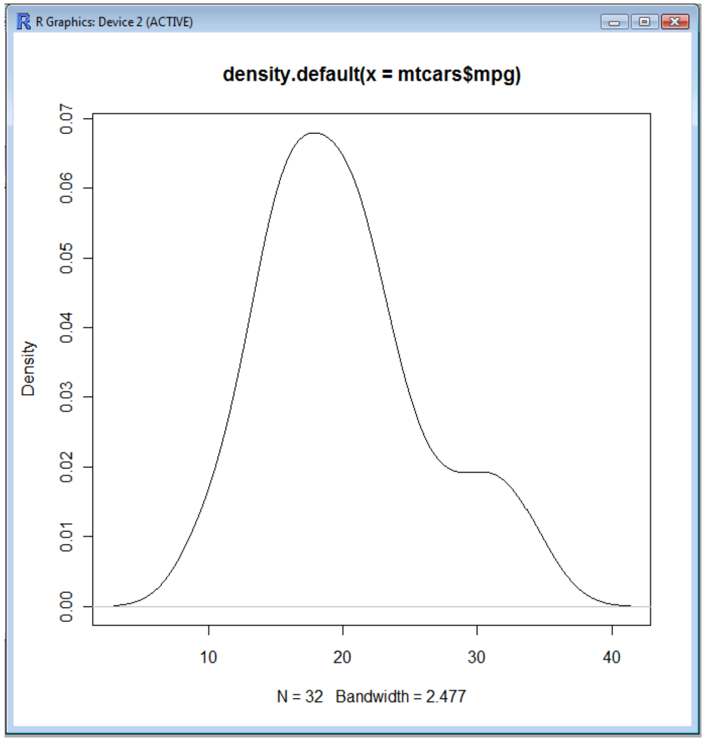


Graphics on R

Kernel density plots

Kernel density plots nicely visualize the shape of a distribution. They can be better than histograms, even with normal curves because histograms are strongly affected by the number of bins used and by outliers.

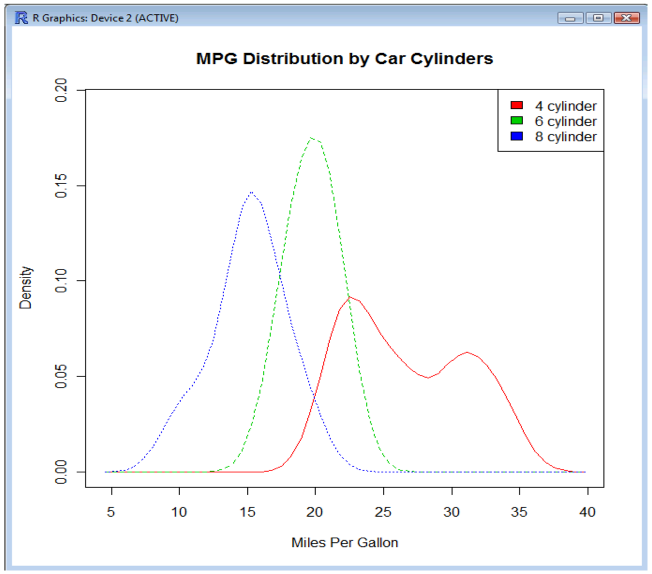
```
# Kernel density plot
d <- density(mtcars$mpg) # kernel density estimates
plot(d)

# Filled density plot
d <- density(mtcars$mpg)
plot(d, main="Kernel Density of Miles Per Gallon")
polygon(d, col="red", border="blue")
```



Graphics on R

Kernel density for comparing groups

To compare the kernal density plots of two or more groups, the sm package has the function sm.density.compare():
sm.density.compare(x, factor)
x numeric vector
factor grouping variable



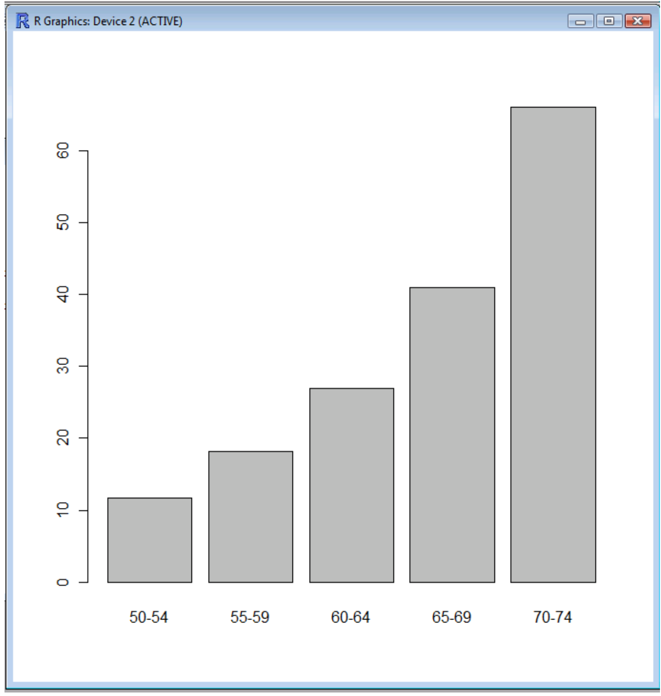
```
# Compare MPG distributions for cars with 4,6, or 8 cylinders
library(sm)
attach(mtcars)

# create value labels
cyl.f <- factor(cyl, levels= c(4,6,8),  labels = c("4 cylinder", "6 cylinder", "8 cylinder"))

# plot densities
sm.density.compare(mpg, cyl, xlab="Miles Per Gallon")
title(main="MPG Distribution by Car Cylinders")

# add legend
colfill<-c(2:(2+length(levels(cyl.f))))
legend("topright", levels(cyl.f), fill=colfill)
```

Graphics on R

barplot

boxplot(X) is a plot that, if X is a vector, the vector elements are the heights of the bars in the plot, if X is a matrix, the matrix columns are the heights of the bars in the plot, stacked after the first bar (column)
If the argument beside=TRUE, then the values in each column are juxtaposed, not stacked.
The argument horiz=TRUE creates an horizontal barplot.

```
VADeaths
class(VADeaths)
dimnames(VADeaths)
# simple barplot
barplot(VADeaths[,"Rural Male"])
# stacked barplots
barplot(VADeaths[,c("Rural Male", "Rural Female")])
# juxtaposed barplots
barplot(VADeaths[,c("Rural Male", "Rural Female")],beside=T)
# stacked barplots
barplot(VADeaths)
# juxtaposed barplots
barplot(VADeaths,beside=T)
```

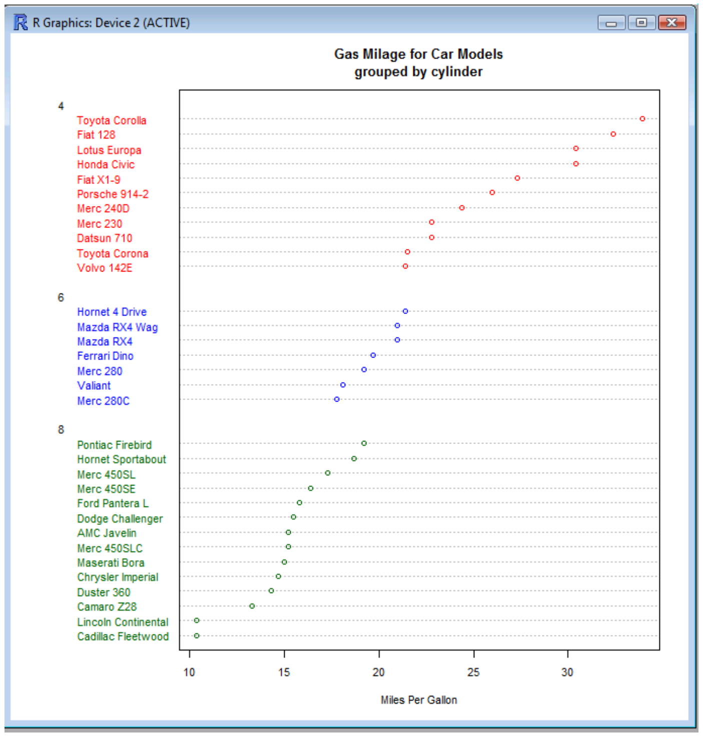


Graphics on R

dotchart

dotchart(X) plots a dot chart or dot plot which plots the values of variable X in groups

```
# Simple Dotplot
dotchart(mtcars$mpg,labels=row.names(mtcars),cex=.7,
main="Gas Milage for Car Models",xlab="Miles Per Gallon")

# Dotplot: Grouped Sorted and Colored
# Sort by mpg, group and color by cylinder
x <- mtcars[order(mtcars$mpg),] # sort by mpg
x$cyl <- factor(x$cyl) # it must be a factor
x$color[x$cyl==4] <- "red"
x$color[x$cyl==6] <- "blue"
x$color[x$cyl==8] <- "darkgreen"
dotchart(x$mpg,labels=row.names(x),cex=.7,groups= x$cyl,main="Gas Milage for Car
Models\ngrouped by cylinder",xlab="Miles Per Gallon",gcolor="black", color=x$color)
```

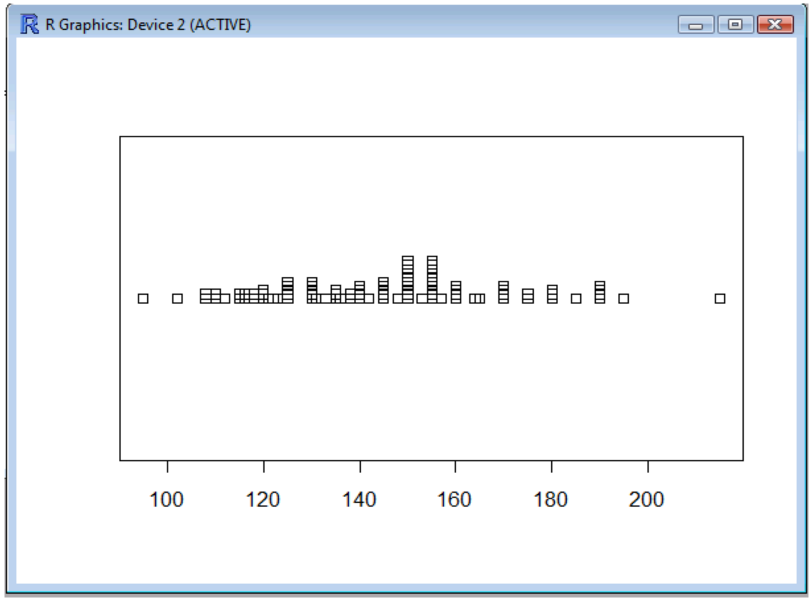


stripchart

A stripchart(X) plots a one dimensional or dot plot of the variable X, this is a good alternative to boxplots when sample sizes are small

Data from Cartoon Guide to Statistics, from Larry Gonick, Woollcott Smith, Collins Reference, 1993
The weights of some Penn State students, in 1992

```
mydataf2 <- read.csv("PennState92.csv", header=F,row.names=1)
mydataf2
# put all the data in one vector
v1 <- c(as.matrix(mydataf2[1,]),as.matrix(mydataf2[2,]))
v1 <- v1[!is.na(v1)]
# nice strip chart
stripchart(v1)
# nice strip chart with groups
stripchart(v1, method = "stack",xlim = c(min(v1),max(v1)))
```

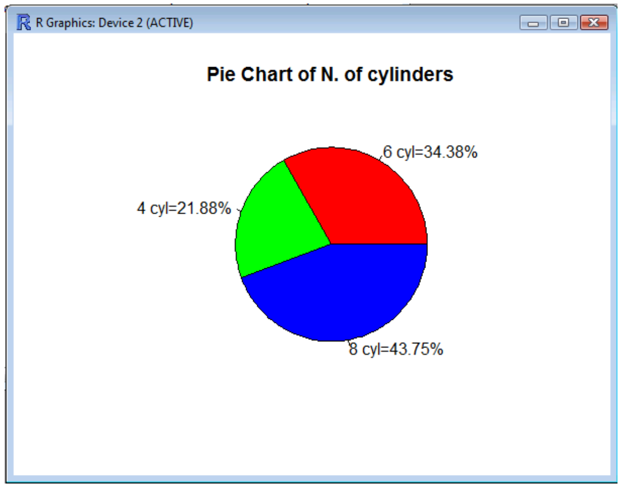


Graphics on R

pie

pie(x) draws a circle (pie) cut into segments (slices), each slice represents a unique value from the elements of x and the sixe of the slice and the relative frequency of each unique value is represented by the size of the slice.

```
# simple pie
pie(unique(mtcars$cyl), labels = unique(mtcars$cyl), main="Pie Chart of N. of cylinders")

# pie with percentages and colors
with(mtcars, {
n.cyl <- unique(cyl)
percent.cyl <-round(table(cyl)/dim(mtcars)[1]*100,2)
lbls <- paste(n.cyl," cyl=",percent.cyl,"%", sep="")
pie(n.cyl, labels = lbls , main="Pie Chart of N. of cylinders", col=rainbow(length(lbls)))
})
```

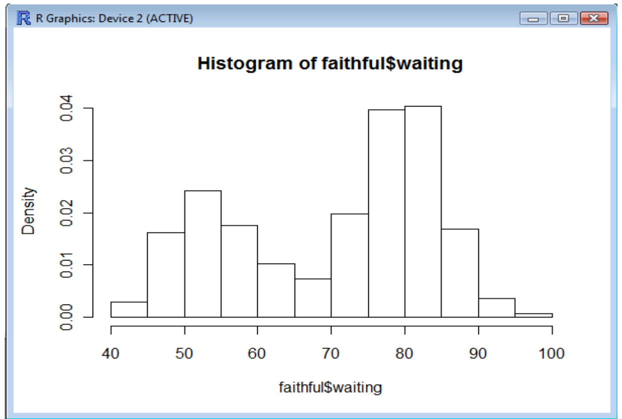


Graphics on R

hist

hist(X) is an histogram, a bar plot with the frequencies of the values in X on the y-axis and the ranges of values on the x-axis
A cumulative distribution curve is the proportion of X on the y-axis, up to the current position on the x-axis

```
# simple histogram
hist(faithful$waiting, prob=TRUE)

# Frequency polygon
# http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=101
h <- hist(faithful$waiting, prob=TRUE, plot=FALSE)
# compute the frequency polygon
diffBreaks <- h$mids[2] - h$mids[1]
xx <- c( h$mids[1]-diffBreaks, h$mids, tail(h$mids,1)+diffBreaks )
yy <- c(0, h$density, 0)
# draw the histogram
hist(faithful$waiting, prob = TRUE, xlim=range(xx),border="gray", col="gray90")
# adds the frequency polygon
lines(xx, yy, lwd=2, col = "royalblue")

# cumulative distribution
h <- hist(faithful$waiting)
h$counts <- cumsum(h$counts)
plot(h)
```

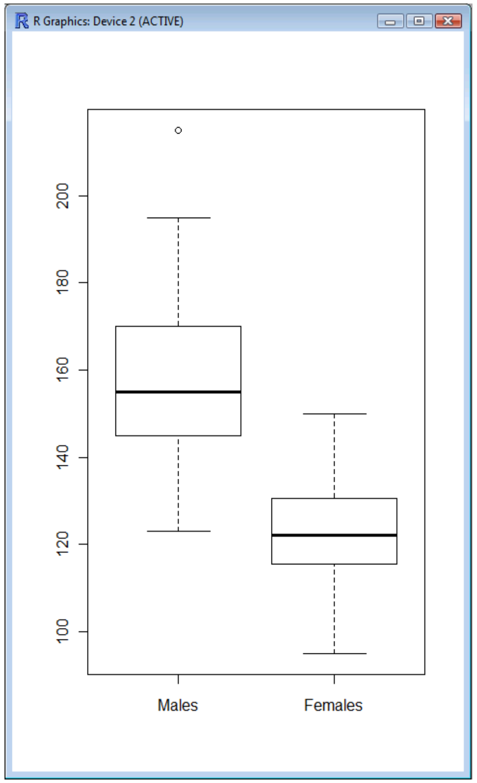


Graphics on R

boxplot

boxplot(X) is a box-and-whisker plot with the values of variable X, this is an effective way to summarize larger datasets

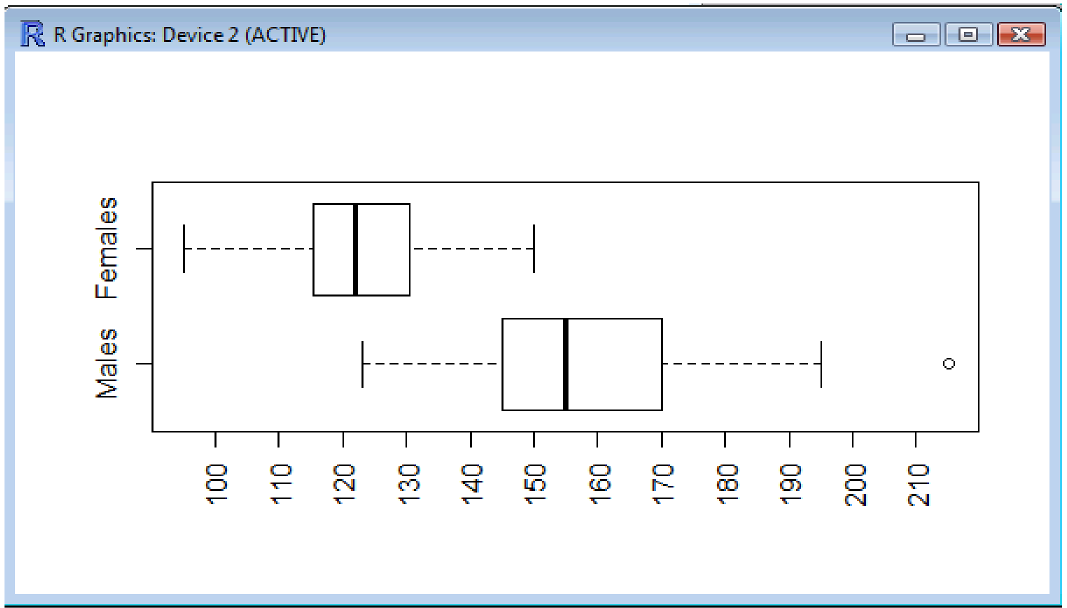
```
mydataf2 <- read.csv("PennState92.csv", header=F,row.names=1)
mydataf2
# plot the data for Males and Females
apply(mydataf2, 1, summary)
boxplot(as.numeric(mydataf2[1,]), as.numeric(mydataf2[2,]), names=c("Males","Females"))
```



Graphics on R

Changing the scale

```
mydataf2 <- read.csv("PennState92.csv", header=F,row.names=1)
mydataf2

# plot the data for Males and Females
apply(mydataf2, 1, summary)
# use a y-axis scale of 10
boxplot(as.numeric(mydataf2[1,]), as.numeric(mydataf2[2,]), names=c("Males","Females"),
horizontal=T, xaxt = "n")
axis(1, 10:21*10,  las = 2)
```

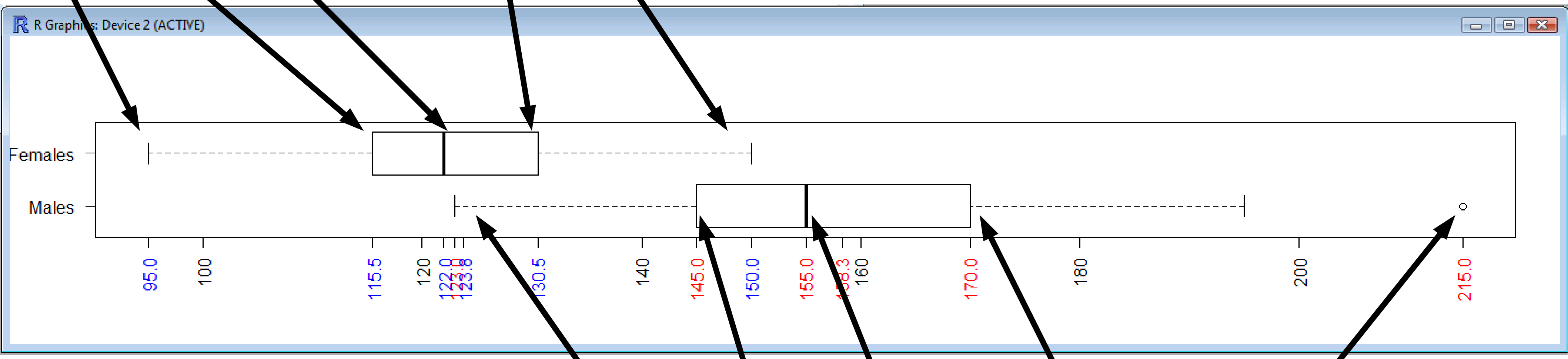


Graphics on R

Adding references to points

```
mydataf2 <- read.csv("PennState92.csv", header=F,row.names=1)
mydataf2

# plot the data for Males and Females
# summary points on the y-axis
boxplot(as.numeric(mydataf2[1,]), as.numeric(mydataf2[2,]), names=c("Males","Females"), horizontal=T,
las = 2)
summdtf <- apply(mydataf2, 1, summary)
axis(1, summdtf$Males,  las = 2, col.axis="red")
axis(1, summdtf$Females,  las = 2, col.axis="blue")
```

```
$Females
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   95.0   115.5   122.0   123.8   130.5   150.0    22.0
```

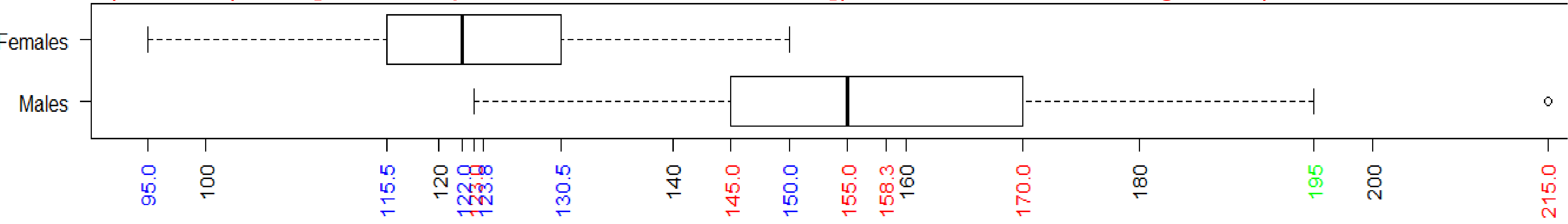


```
> apply(mydataf2, 1, summary)
$Males
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  123.0   145.0   155.0   158.3   170.0   215.0
```

Graphics on R

Calculating the whiskers

```
boxplot(as.numeric(mydataf2[1,]), as.numeric(mydataf2[2,]), names=c("Males","Females"),
horizontal=T,  las = 2)
summdtf <- apply(mydataf2, 1, summary)
axis(1, summdtf$Males,  las = 2, col.axis="red")
axis(1, summdtf$Females,  las = 2, col.axis="blue")

female <- as.numeric(mydataf2[2,])
female <- female[!is.na(female)]
male <- as.numeric(mydataf2[1,])
IQRmale <- IQR(male, na.rm =T) # interquartile range
IQRfemale <- IQR(female, na.rm =T) # interquartile range
q1male <- quantile(male,.25, na.rm =T)
q3male <- quantile(male,.75, na.rm =T)
q1female <- quantile(female,.25, na.rm =T)
q3female <- quantile(female,.75, na.rm =T)
# whiskers = Q1 - 1.5 * IQR and Q3 + 1.5 * IQR
min(female[female > q1female  - 1.5 * IQRfemale])
max(female[female < q3female  + 1.5 * IQRfemale])
min(male[male > q1male  - 1.5 * IQRmale])
max(male[male < q3male  + 1.5 * IQRmale])

axis(1, max(male[male < q3male  + 1.5 * IQRmale]),  las = 2, col.axis="green")
```

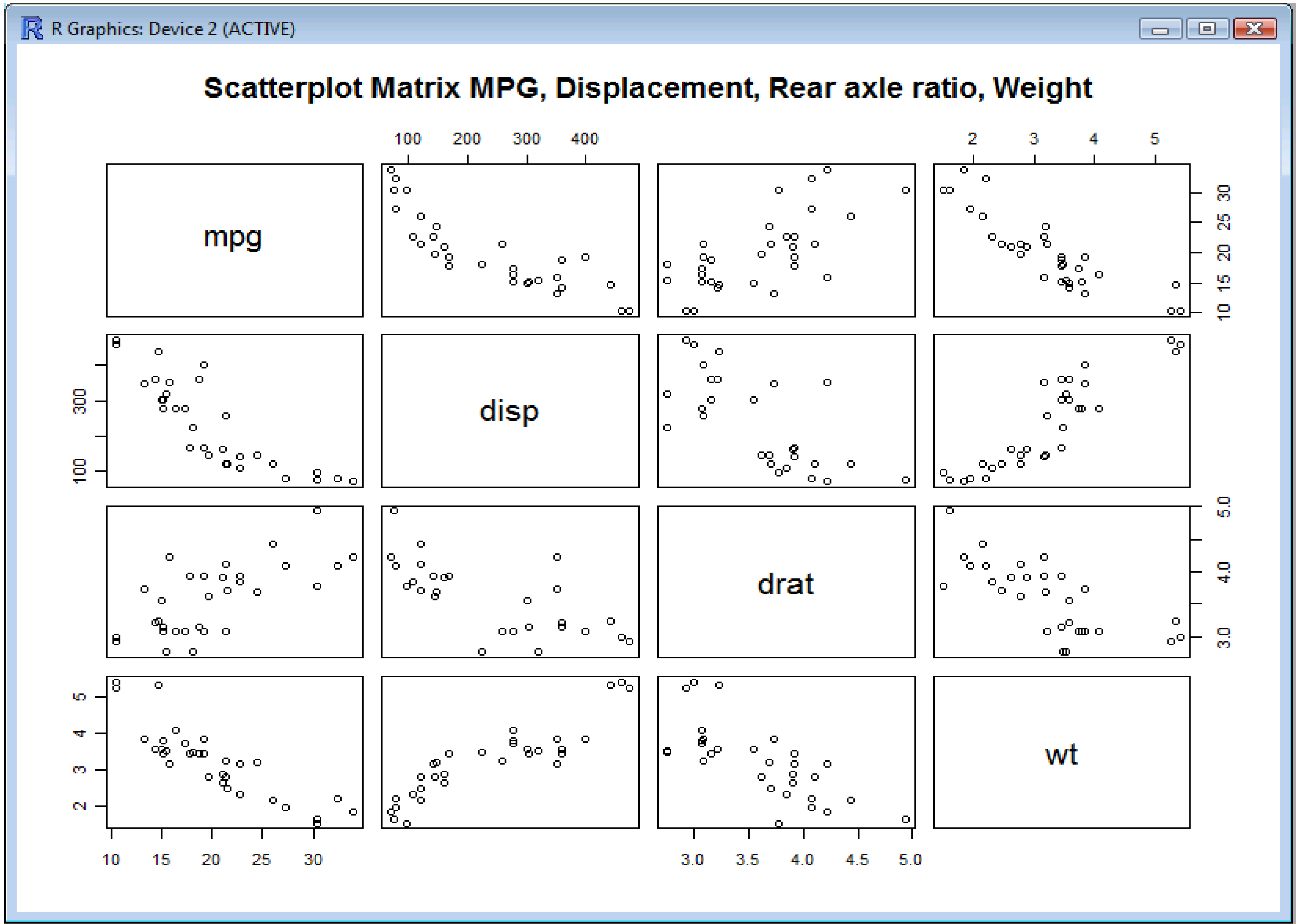


Graphics on R

pairs

pairs() shows a matrix with all the scatterplots for the columns of variable X

```
pairs(~mpg+disp+drat+wt,data=mtcars, main="Scatterplot Matrix MPG, Displacement, Rear axle ratio, Weight")
```



Graphics on R

stem

stem(X) creates a stem-and-leaf plot, which shows the shape of a distribution and displays each observation, useful for small datasets

```r
mydataf2 <- read.csv("PennState92.csv", header=F,row.names=1)
mydataf2
# put all the data in one vector
v1 <- c(as.matrix(mydataf2[1,]),as.matrix(mydataf2[2,]))
v1 <- v1[!is.na(v1)]
# stem-and-leaf plot
stem(v1)
```

```
  The decimal point is 1 digit(s) to the right of the |

   8 | 5
  10 | 288002556688
  12 | 000123555550000013555688
  14 | 000025555580000000000355555555557
  16 | 000045000055
  18 | 000500005
  20 | 5
```

Stem

Leaf

Graphics on R

Details about the R stem()

Data from Basic Biostatistics, by Burt Gertsman, chapter 3

stem(x, scale = 1, width = 80, atom = 1e-08)

x a numeric vector
scale This controls the plot length
width The desired width of plot
atom a tolerance

```
myvec <- c(14, 17, 18, 19, 22, 22, 23, 24, 24, 26, 26, 27, 28, 29, 30, 30, 30, 31, 32, 33, 34, 34, 35, 36, 37,
38)
stem(myvec) # this is wrong!
length(myvec) # n=26
stem(myvec,atom =26) # OK!
# Too squished to see shape
# Split stem
stem(myvec,atom =1) # OK!

myvec <- c(14, 17, 18, 19, 22, 22, 23, 24, 26, 26, 27, 28, 29, 30, 30, 30, 31, 32, 33, 34, 34, 35, 36, 37, 38)
stem(myvec) # this is wrong!
length(myvec) # n=25
stem(myvec,atom =25) # OK!
# Too squished to see shape
# Split stem
stem(myvec) # OK!
```

Graphics on R

mosaicplot

mosaicplot() draws a mosaic plot, a relationship betwen two or more categorical variables, the widht of the bars is horizontally and vertically proportional to the probabilities associated with the categorical variables

```
mosaicplot(Titanic, main = "Survival on the Titanic", color = TRUE)
## Formula interface for tabulated data:
mosaicplot(~ Sex + Age + Survived, data = Titanic, color = TRUE)

## Formula interface for raw data: visualize cross-tabulation of numbers
## of gears and carburettors in Motor Trend car data.
mosaicplot(~ gear + carb, data = mtcars, color = TRUE, las = 1)
# color recycling
mosaicplot(~ gear + carb, data = mtcars, color = 2:3, las = 1)
```

Graphics on R

Examples to explain mosaicplot()

```
Titanic
is(Titanic)
dim(Titanic)
dimnames(Titanic) # Class Sex Age Survived

# Overall gender proportion the Titanic
mosaicplot(~ Sex, main = "Overall gender proportion on the Titanic", data = Titanic, color =
TRUE)
# ladies first
mosaicplot(~ Sex, main = "Overall gender proportion on the Titanic", data = Titanic[,2:1,,],
color = TRUE)

# split vertically by survival rate
mosaicplot(~ Sex+ Survived, main = "Overall gender/survival proportion on the Titanic", data =
Titanic[,2:1,,], color = TRUE)

#Overall age/survival proportion on the Titanic
mosaicplot(~ Age+ Survived, main = "Overall age/survival proportion on the Titanic", data =
Titanic, color = TRUE)
```

Graphics on R

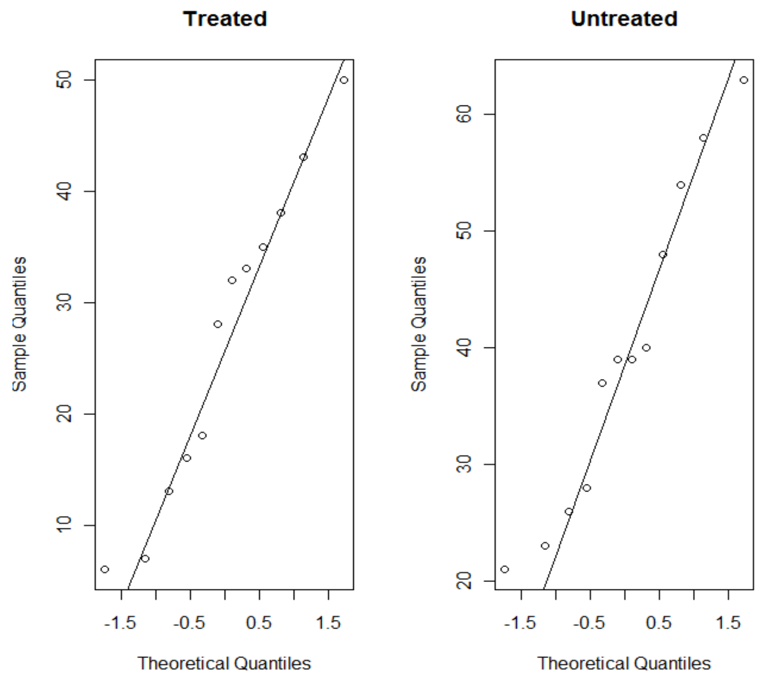
qqnorm and qqline

qqnorm(X) draws a normal probability chart for variable X, with the values of variable X on the y-axis and their associated probability based on a cummulative frequency on the x-axis, assuming a normal distribution

qqline(X) draws the expected linear relationship, assuming a normal distribution

Data from Transcriptomics Bioinformatics, by Attila Gyenesei
"An experiment was conducted to evaluate the effectiveness of a treatment for tapeworm in the stomachs of sheep. A random sample of 24 worm-infected lambs of the same age and health was randomly divided into two groups. 12 were injected with the drug and the remaining 12 were left untreated. After a 6-month period the worm counts were recorded"

```
sheep <- read.table("sheep.txt", sep="\t", header=T)
par(mfrow=c(1,2))
qqnorm(sheep$treated, main="Treated")
qqline(sheep$treated)
qqnorm(sheep$untreated, main="Untreated")
qqline(sheep$untreated)
```



Graphics on R

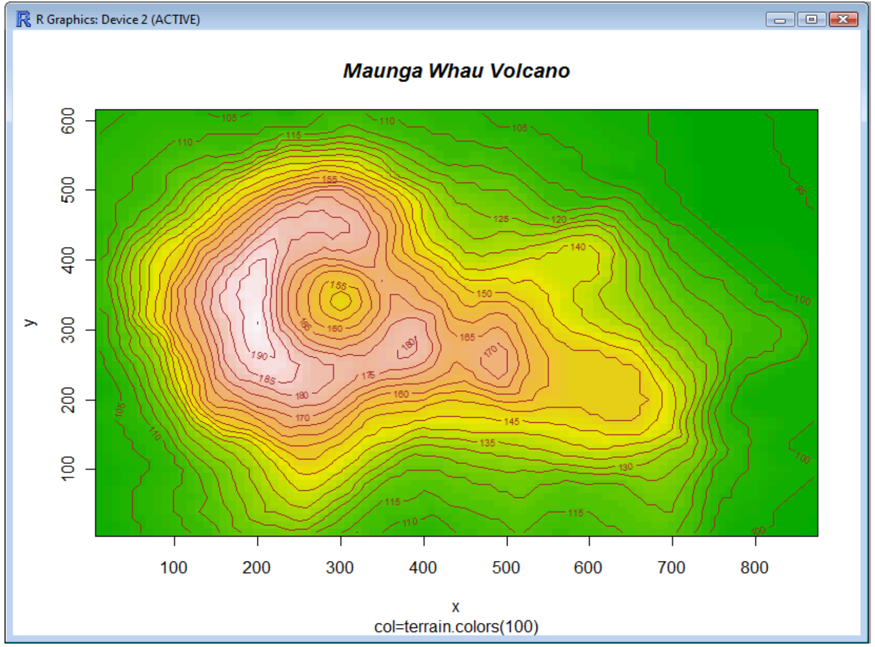
contour

contour(X,Y,Z) draws a contour plot, with vector X for the rows, vector Y for the columns and matrix X for the data

Example from R Graph Gallery by Romain François
http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=22

contour plot Maunga Whau Volcano

```
x <- 10*(1:nrow(volcano)); x.at <- seq(100, 800, by=100)
y <- 10*(1:ncol(volcano)); y.at <- seq(100, 600, by=100)
# Using Terrain Colors
image(x, y, volcano, col=terrain.colors(100),axes=FALSE)
contour(x, y, volcano, levels=seq(90, 200, by=5), add=TRUE, col="brown")
axis(1, at=x.at)
axis(2, at=y.at)
box()
title(main="Maunga Whau Volcano", sub = "col=terrain.colors(100)", font.main=4)
```

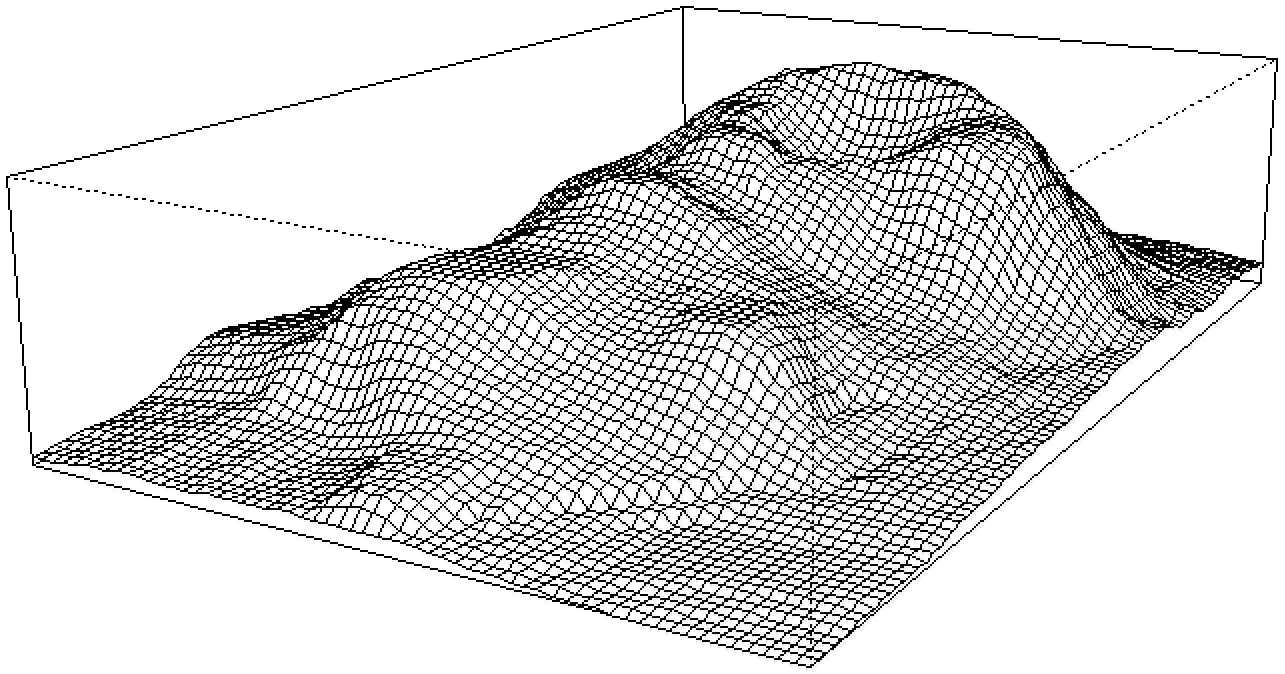


Graphics on R

persp

persp(X,Y,Z) draws a 3d graph, with vector X for the rows, vector Y for the columns and matrix X for the data

```
## (2) Visualizing a simple DEM model

z <- 2 * volcano        # Exaggerate the relief
x <- 10 * (1:nrow(z))   # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))   # 10 meter spacing (E to W)
persp(x, y, z, theta = 120, phi = 15, scale = FALSE, axes = FALSE)
```

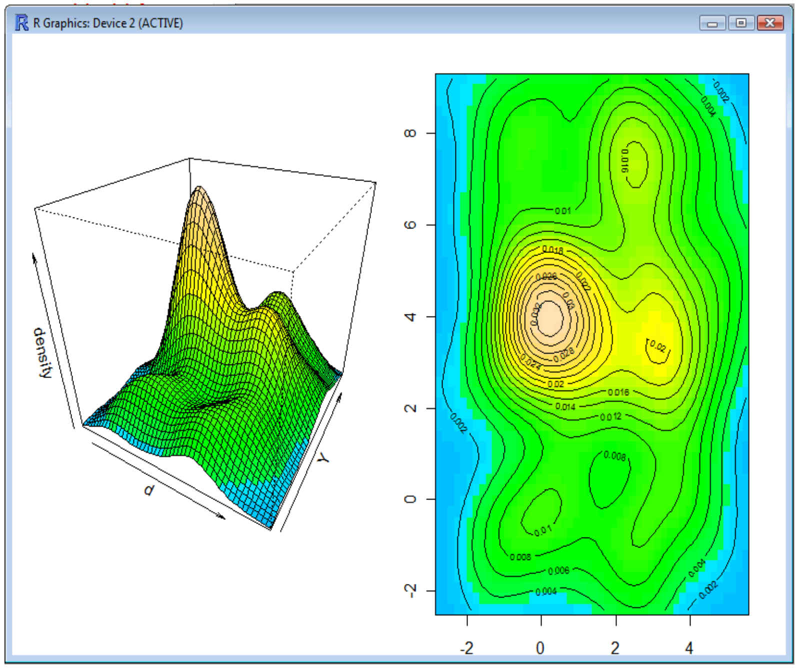


Graphics on R

Example from R Graph Gallery by Romain François
http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=1

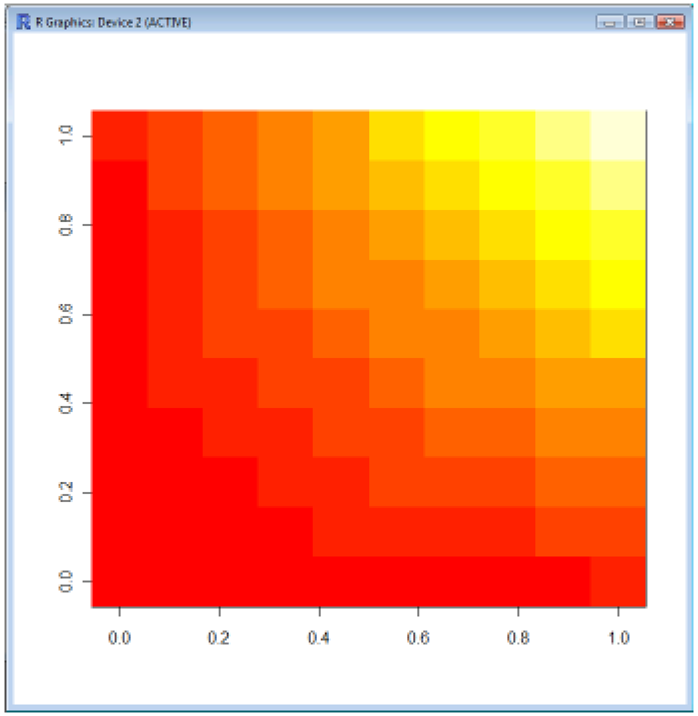
Kernel density estimator in R2 Perspective plot and contour plot

```
require(MASS)
set.seed(125)
x <- rnorm(150,mean=3*rbinom(150,prob=.5,size=1),sd=1)
y <- rnorm(150,mean=4*rbinom(150,prob=.5,size=2),sd=1)
d <- kde2d(x,y,n=50)
kde2dplot <- function(d,              # a 2d density computed by kde2D
                    ncol=50,         # the number of colors to use
                    zlim=c(0,max(z)), # limits in z coordinates
                    nlevels=20,       # see option nlevels in contour
                      theta=30,        # see option theta in persp
                      phi=30)         # see option phi in persp
                      {
z   <- d$z
nrz <- nrow(z)
ncz <- ncol(z)
couleurs  <- tail(topo.colors(trunc(1.4 * ncol)),ncol)
fcol      <- couleurs[trunc(z/zlim[2]*(ncol-1))+1]
dim(fcol) <- c(nrz,ncz)
fcol      <- fcol[-nrz,-ncz]
par(mfrow=c(1,2),mar=c(0.5,0.5,0.5,0.5))
persp(d,col=fcol,zlim=zlim,theta=theta,phi=phi,zlab="density")
par(mar=c(2,2,2,2))
image(d,col=couleurs)
contour(d,add=T,nlevels=nlevels)
box()
}
kde2dplot(d)
```

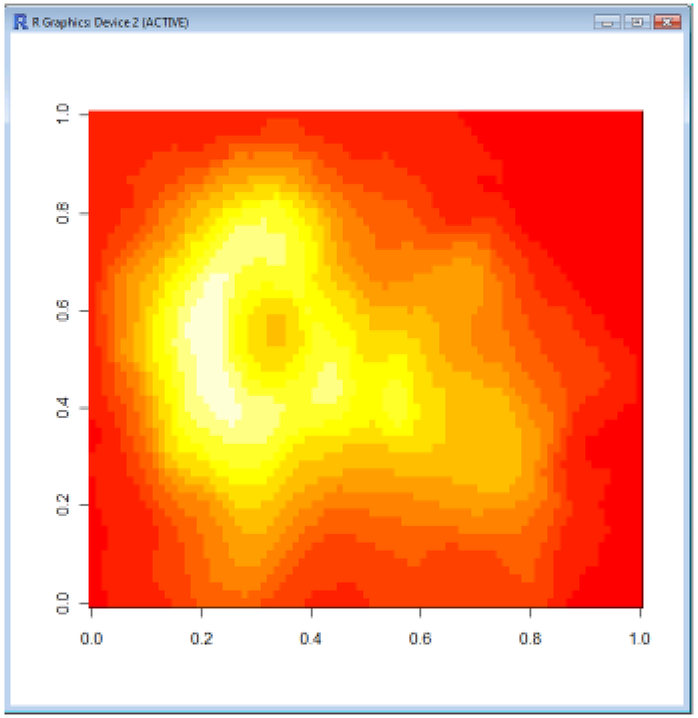


Graphics on R

image() Creates a grid of colored or gray-scale rectangles with colors corresponding to the values in z



```
x <- 1:10
y <- 1:10
m <- outer(x,y)
m
image(m)
```



```
volcano
image(volcano)
```

Graphics on R
References/to learn more:

The R book
Michael J. Crawley pp 135
2017 John Wiley & Sons Ltd

Basic statistics using R pp. 110
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics: an introduction using R
Michael J. Crawley pp 297
2015 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 147
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/a68d739b891b9a30368f756ba473b81d

Introductory Statistics with R
Peter Dalgaard, pp 71
2017 Springer

Geographic Data Analysis
Pat Bartlein
http://geography.uoregon.edu/bartlein/courses/geog417/lectures/lec02.htm

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/g-intro.r

Chem 351 Archives Page
David Harvey
http://fs6.depauw.edu:50080/~harvey/Chem%20351/PDF%20Files/Handouts/RDocs/Graphing%20Data%20in%20R%20-%20A%20Gallery%20of%20Plots.pdf

Thomas AP Statistics
thomasmathematics.com
http://www.thomasmathematics.com/Aims/Ch1Aim50001.pdf

Quick-R
Rob Kabacoff
http://www.statmethods.net/graphs/index.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

Sexual Discrimination at Berkeley

UCBAdmissions(datasets)

This data set is frequently used for illustrating Simpson's paradox, see Bickel et al. (1975). At issue is whether the data show evidence of sex bias in admission practices. There were 2691 male applicants, of whom 1198 (44.5%) were admitted, compared with 1835 female applicants of whom 557 (30.4%) were admitted. This gives a sample odds ratio of 1.83, indicating that males were almost twice as likely to be admitted. In fact, graphical methods (as in the example below) or log-linear modelling show that the apparent association between admission and sex stems from differences in the tendency of males and females to apply to the individual departments (females used to apply more to departments with higher rejection rates).

Simpson's paradox
http://en.wikipedia.org/wiki/Simpson%27s_paradox

Admissions by Department

		Male	Female
Department A	Admitted	512	89
	Rejected	313	19
Department B	Admitted	353	17
	Rejected	207	8
Department C	Admitted	120	202
	Rejected	205	391
Department D	Admitted	138	131
	Rejected	279	244
Department E	Admitted	53	94
	Rejected	138	299
Department F	Admitted	22	24
	Rejected	351	317

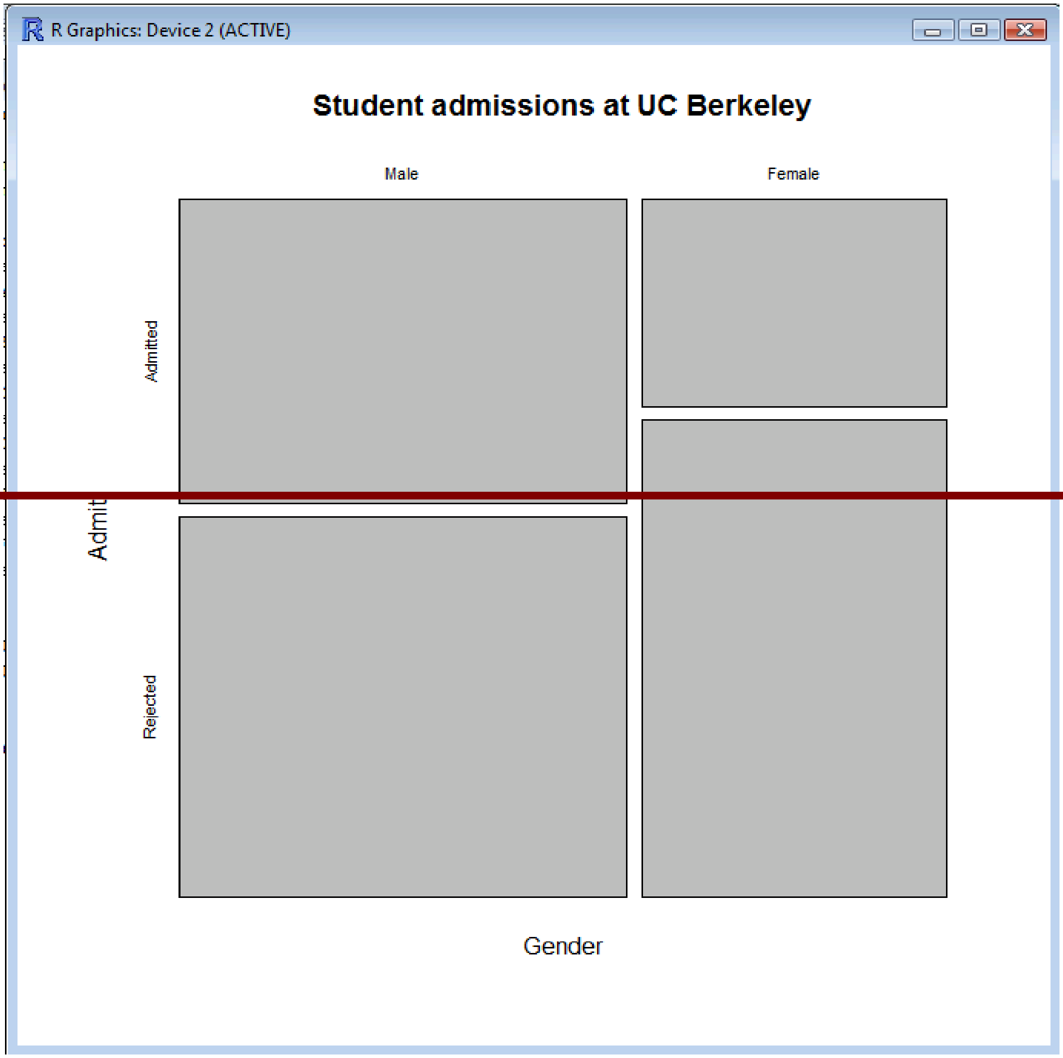
Gender	Admitted	Rejected	%Admitted
Male	1198	1493	44.5
Female	557	1278	30.4

More males are admitted than females is this discrimination?

```
UCBAdmissions
is(UCBAdmissions) # contingency table!
dim(UCBAdmissions)
rownames(UCBAdmissions)
colnames(UCBAdmissions)
dimnames(UCBAdmissions)

# creating the table Gender Admitted Rejected %Admitted
> apply(UCBAdmissions, 1:2, sum)
          Gender
Admit      Male Female
  Admitted 1198    557
  Rejected 1493   1278
> apply(UCBAdmissions, 1:2, sum)[1,]/apply(UCBAdmissions, 2, sum)
     Male    Female
0.4451877 0.3035422

# creating the table Admissions by Department

xtabs(Freq~Dept+Gender+Admit,data=UCBAdmissions)

apply(UCBAdmissions, c(3,2,1), sum)
```

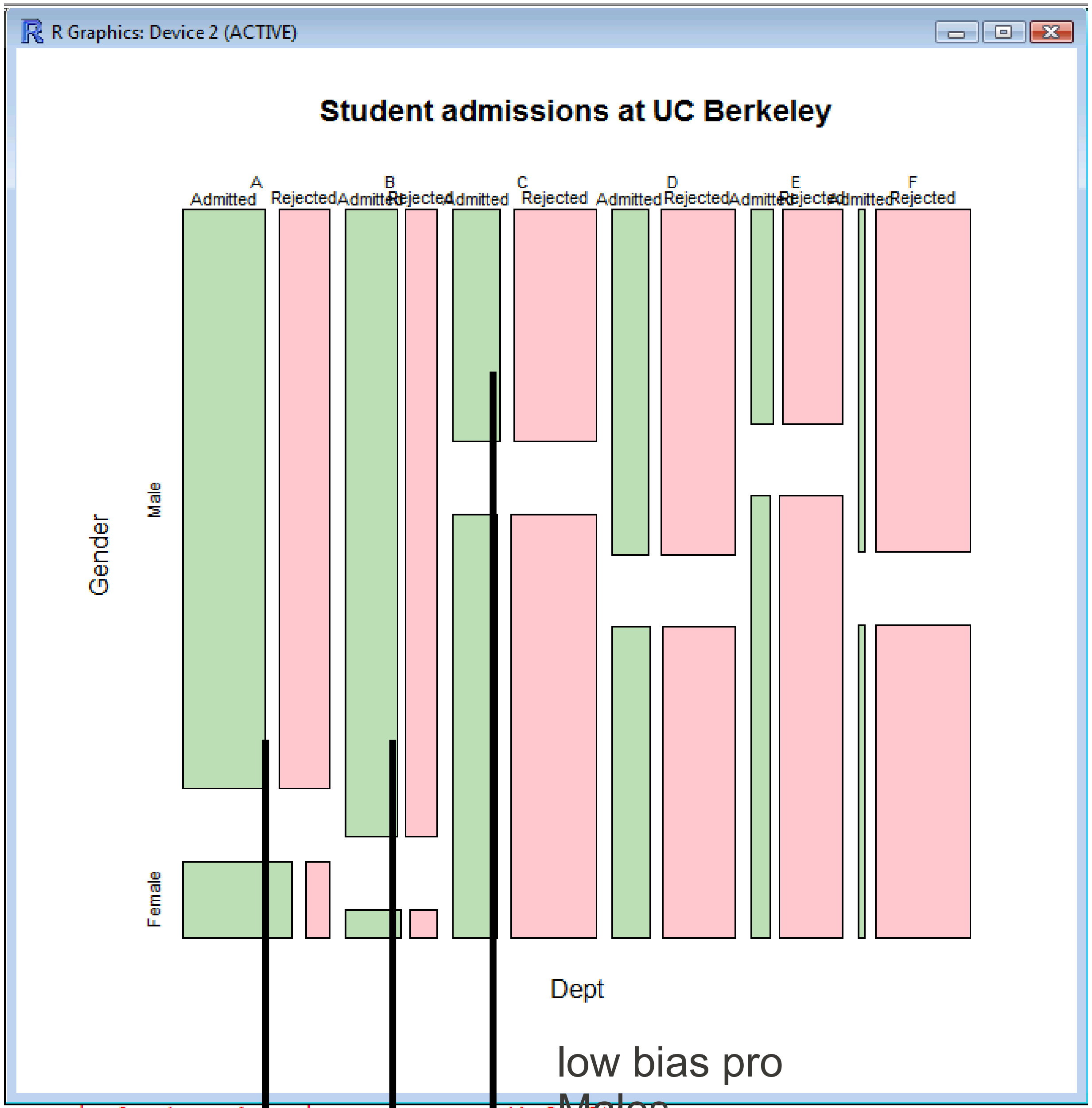
On a mosaicplot, 2 variables are independent when their proportions are the same, this is not the case

```
mosaicplot(apply(UCBAdmissions, c(2, 1), sum), main = "Student admissions at UC Berkeley")
```



More males are admitted!

```
mosaicplot(UCBAdmissions, sort = 3:1,col = hcl(c(120, 10)),main = "Student admissions at UC Berkeley")
```



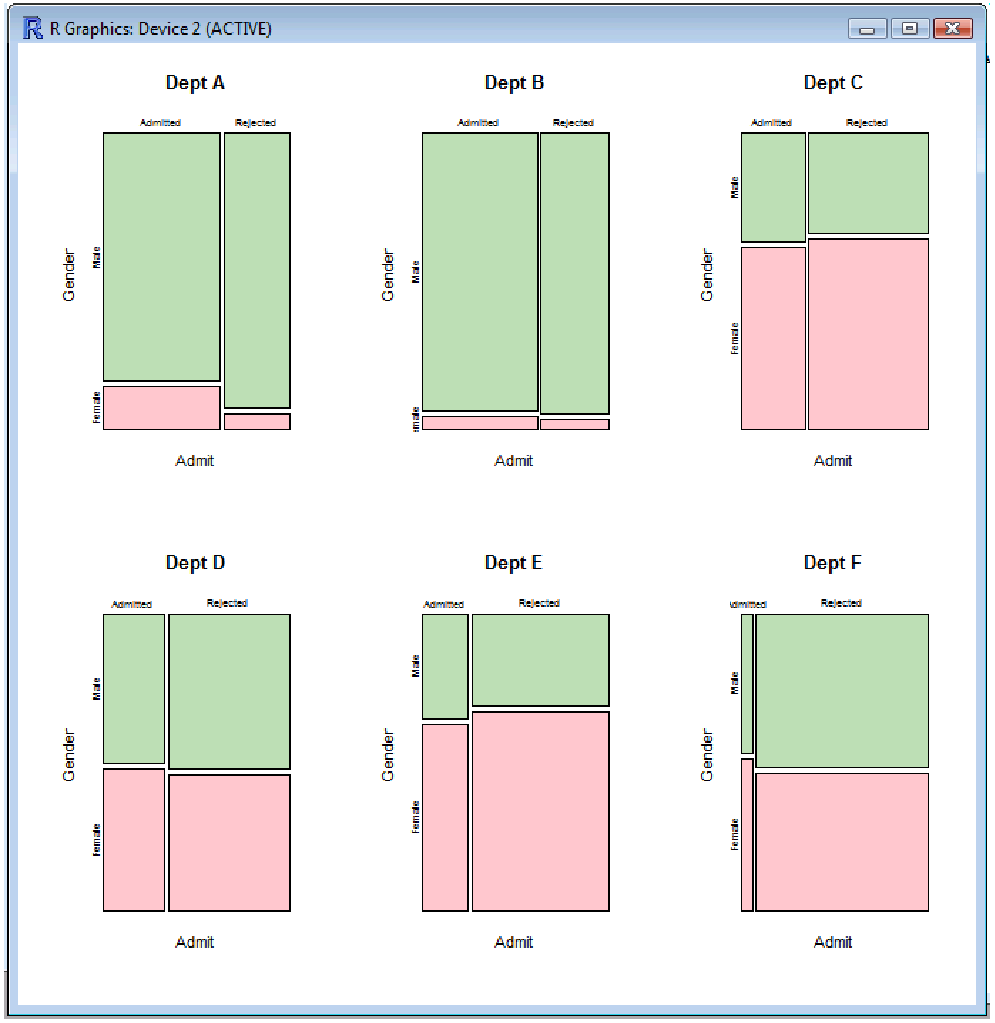
High bias pro Females

low bias pro Females

low bias pro Males

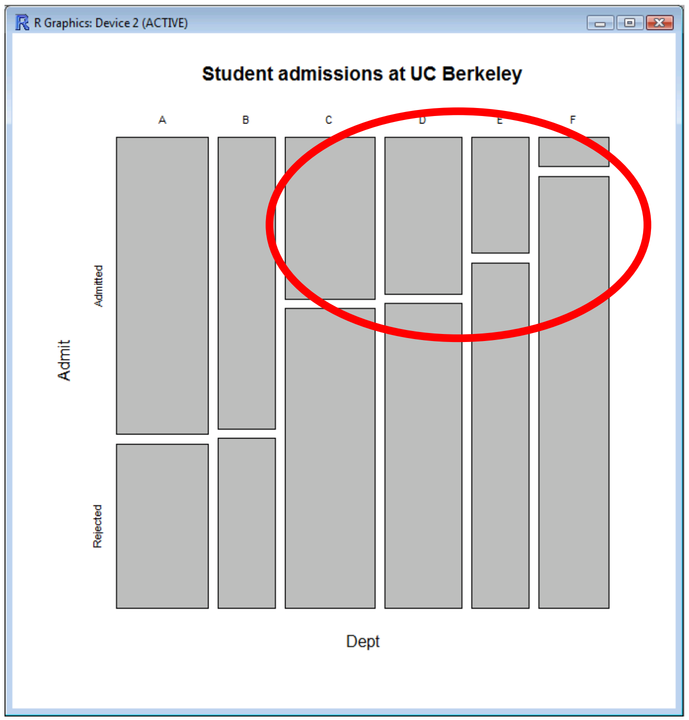
More clear picture:

```
par(mfrow = c(2,3))
for (dpt in LETTERS[LETTERS <= "F"]) mosaicplot(UCBAdmissions[,,dpt], sort = c(1,2), main
= paste("Dept",dpt),col = hcl(c(120, 10)))
```

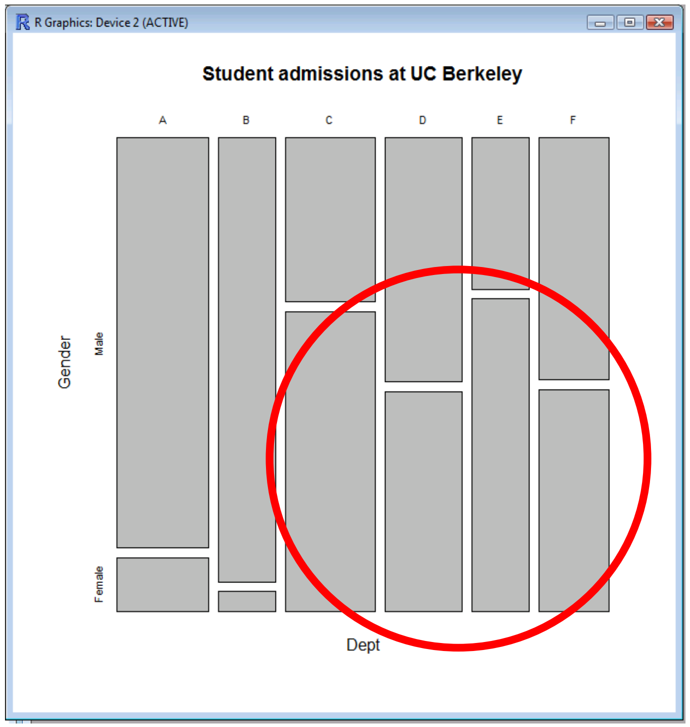


There is very low bias and it favors females, so, why the huge disparity in admissions?

```
# which departments admitted less people?
mosaicplot(apply(UCBAdmissions, c(3, 1), sum), main = "Student admissions at UC Berkeley")
```

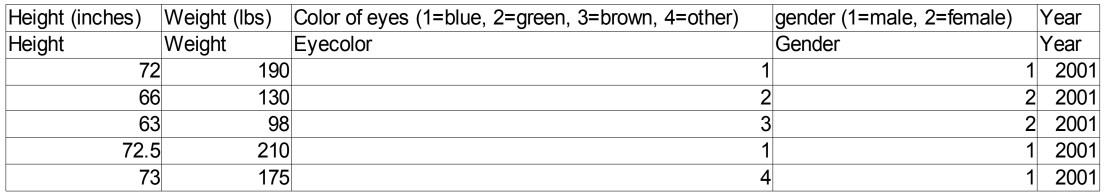


```
# which departments did females applied to mostly?
mosaicplot(apply(UCBAdmissions, c(3, 2), sum), main = "Student admissions at UC Berkeley")
```

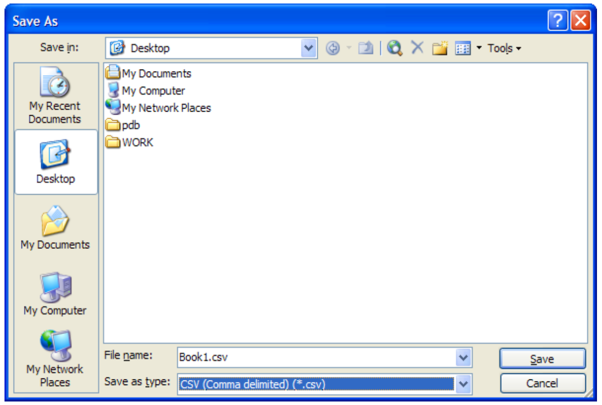


Females applied mostly to departments that admitted less people, basically competing against each other, while males took the departments more accessible

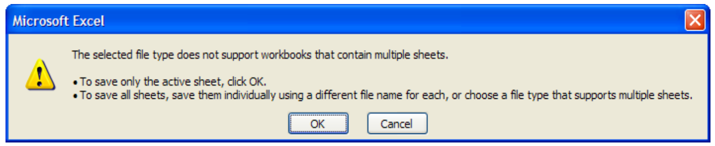
Saving CSV and TAB from Excel



Height (inches)	Weight (lbs)	Color of eyes (1=blue, 2=green, 3=brown, 4=other)	gender (1=male, 2=female)	Year
Height	Weight	Eyecolor	Gender	Year
72	190	1	1	2001
66	130	2	2	2001
63	98	3	2	2001
72.5	210	1	1	2001
73	175	4	1	2001

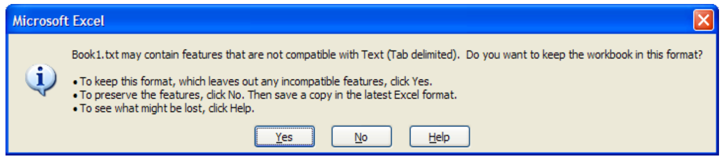


File name: Book1.csv
Save as type: CSV (Comma delimited) (*.csv)



The selected file type does not support workbooks that contain multiple sheets.

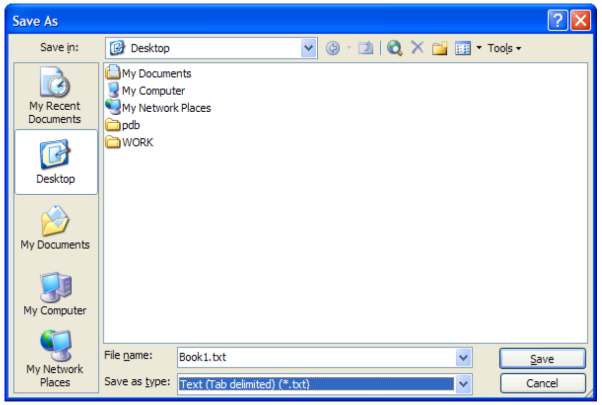
- To save only the active sheet, click OK.
- To save all sheets, save them individually using a different file name for each, or choose a file type that supports multiple sheets.



Book1.txt may contain features that are not compatible with Text (Tab delimited). Do you want to keep the workbook in this format?

- To keep this format, which leaves out any incompatible features, click Yes.
- To preserve the features, click No. Then save a copy in the latest Excel format.
- To see what might be lost, click Help.

Save as height_weight2.csv



File name: Book1.txt
Save as type: Text (Tab delimited) (*.txt)

data from:
ECO 231W Econometrics, Summer 07, Session A
Instructor: Tak Wai Chau
http://troi.cc.rochester.edu/~tchau/eco231/height_weight.xls

Let's read the table and check out its data:

```
DataStudents<-read.csv("height_weight2.csv",skip=1)
DataStudents # Height Weight Eyecolor Gender Year
```

(a) Calculate the sample means and standard deviations from each variable.

```
mean(DataStudents$Height)
mean(DataStudents$Weight)
sd(DataStudents$Height)
sd(DataStudents$Weight)
# or
mean(DataStudents[,c("Height","Weight")]) # the mean of height, weight
sd(DataStudents[,c("Height","Weight")]) # the sd of height, weight
```

(b) Calculate the sample means and standard deviations for height and weight, this time by gender.

```
mean(DataStudents[which(DataStudents$Gender==1),c("Height","Weight")]) # the mean of height, weight
mean(DataStudents[which(DataStudents$Gender==2),c("Height","Weight")]) # the mean of height, weight
sd(DataStudents[which(DataStudents$Gender==1),c("Height","Weight")]) # the sd of height, weight
sd(DataStudents[which(DataStudents$Gender==2),c("Height","Weight")]) # the sd of height, weight
# or
aggregate(DataStudents[,c(1,2,4)], list(DataStudents[,4]), mean)
aggregate(DataStudents[,c(1,2,4)], list(DataStudents[,4]), sd)
```

(c) Calculate the sample means and standard deviations for height and weight, this time by color of eyes.

```
aggregate(DataStudents[,1:3], list(DataStudents[,3]), mean)
aggregate(DataStudents[,1:3], list(DataStudents[,3]), sd)
```

(d) Suppose it is a random sample of students in the university, test the null hypothesis that the mean weight is 200lb for male students against a two-sided alternative.

Null hypothesis H0:$\mu = 200$ Alternative hypothesis H1:$\mu \neq 200$
5% significance level
m=177.9864
s=28.42943
n= length(which(DataStudents$Gender==1)) = 147
T=(177.9864-200)/(28.42943 / sqrt(147))=-9.388184

On R:

```
t.test(DataStudents$Weight[DataStudents$Gender==1], NULL,"two.sided", mu = 200, paired = FALSE,
var.equal = FALSE, conf.level = 0.95)
```

```
        One Sample t-test

data:  DataStudents$Weight[DataStudents$Gender == 1]
t = -9.3882, df = 146, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 200
95 percent confidence interval:
 173.3522 182.6206
sample estimates:
mean of x
 177.9864
```

200 is over the confidence interval, in the rejection zone, so it has to be rejected

(e) Suppose it is a random sample of students in the university, test the null hypothesis that mean weights are the same for male and female students.

null hypothesis: mean weights are the same for male and female students

Null hypothesis H0:μ1 = μ2   Alternative hypothesis H1:μ ≠ μ2
5% significance level
m1=177.9864
m2=133.5093
s1=28.42943
s2=20.10362
n1= length(which(DataStudents$Gender==1))   = 147
n2= length(which(DataStudents$Gender==2))   = 54
T=(177.9864 - 133.5093) / sqrt(28.42943 ^ 2 / 147 + 20.10362 ^ 2 / 54) = 12.34402

On R:

```
t.test(DataStudents$Weight[DataStudents$Gender==1],
DataStudents$Weight[DataStudents$Gender==2],"two.sided",paired = FALSE, var.equal = FALSE,
conf.level = 0.95)

        Welch Two Sample t-test

data:  DataStudents$Weight[DataStudents$Gender == 1] and
DataStudents$Weight[DataStudents$Gender == 2]
t = 12.344, df = 133.349, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 37.35046 51.60382
sample estimates:
mean of x mean of y
 177.9864  133.5093
```

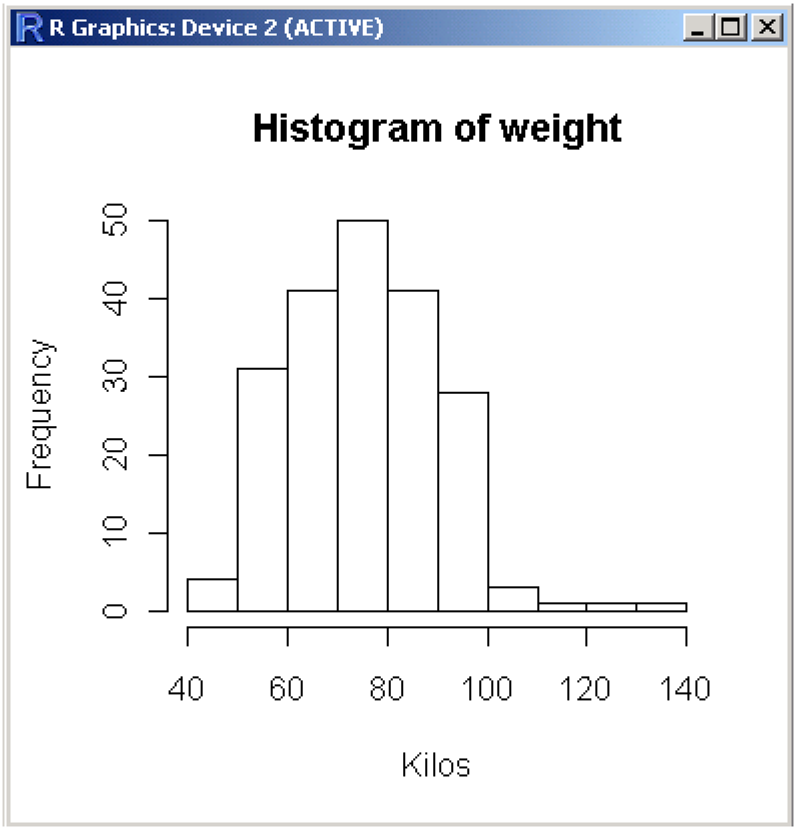
Null hypothesis rejected

To do a dot plot in R:

```
dotchart(DataStudents$Weight, main='Students Weight',xlab='Weight in pounds')

stripchart(DataStudents$Weight, method = "stack",xlim =
c(min(DataStudents$Weight),max(DataStudents$Weight)))
```

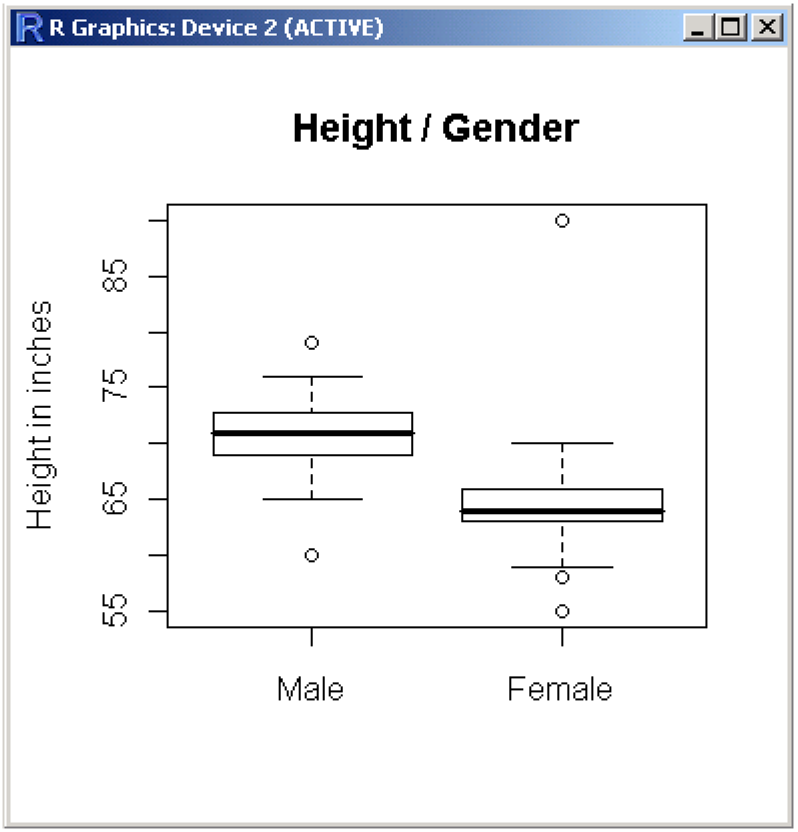
To examine a distribution of the weight, an histogram is quite useful:

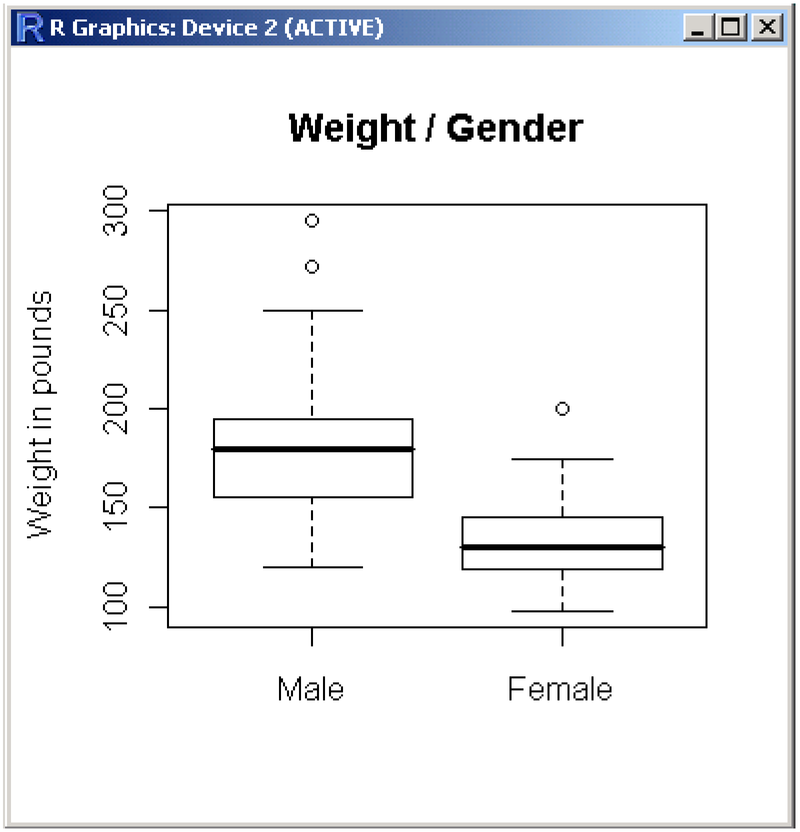
```
hist(DataStudents$Weight*0.45359237,main='Histogram of weight',xlab='Kilos')
```



Are men taller and heavier than women?

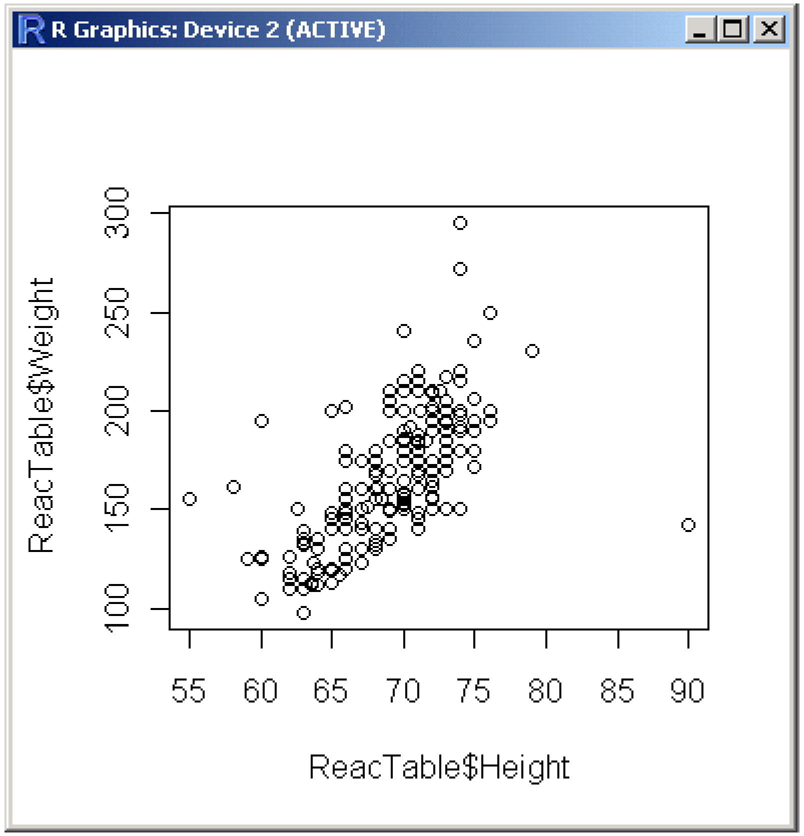
```
boxplot(DataStudents$Height ~ DataStudents$Gender, main='Height / Gender',
ylab='Height in inches',names=c('Male','Female'))
boxplot(DataStudents$Weight ~ DataStudents$Gender, main='Weight / Gender',
ylab='Weight in pounds',names=c('Male','Female'))
```





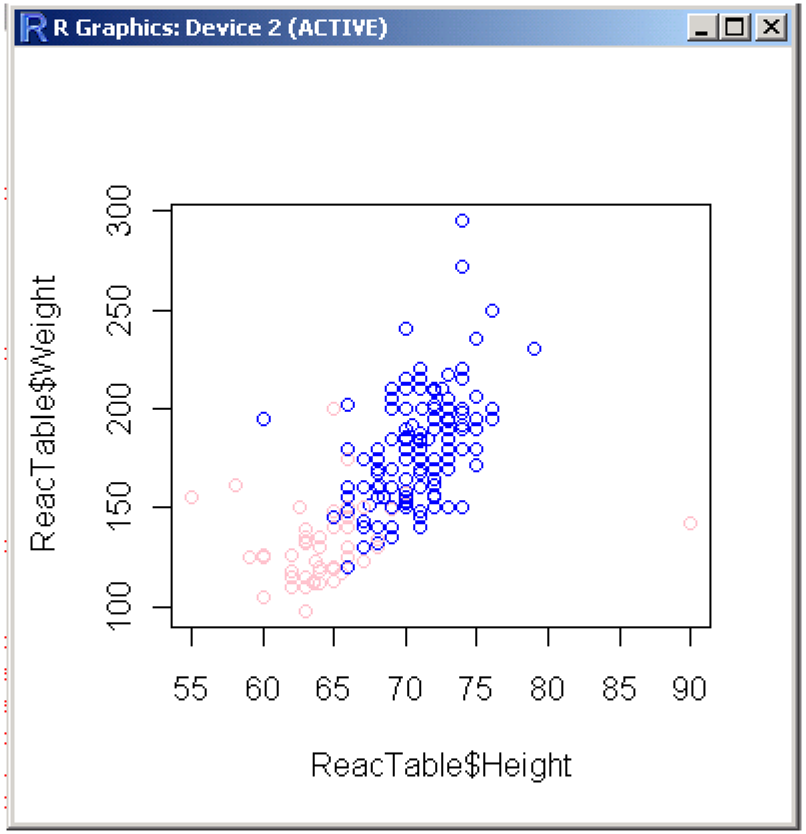
How are weight and height related? A scatter plot will show all the data.

```
plot(DataStudents$Height, DataStudents$Weight)
```



That could be more interesting if the gender was also involved:

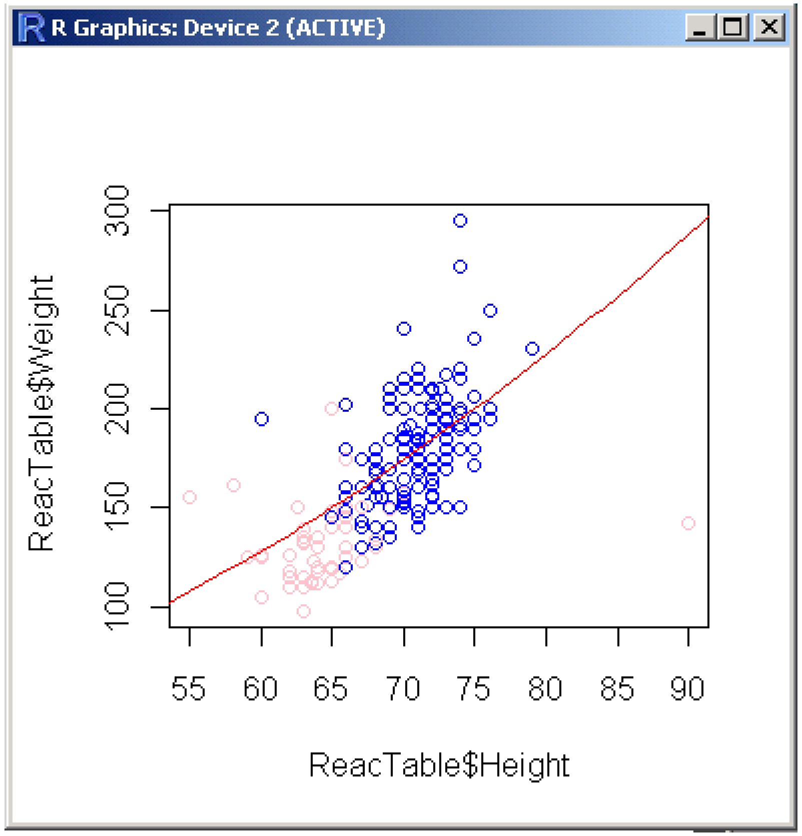
```
sex<-ifelse(DataStudents$Gender==1,'blue','pink')
plot(DataStudents$Height, DataStudents$Weight, col=sex)
```



Who's fat?
Using the BMI(Body Mass Index) formula, BMI Overweight >= 25
BMI=(weight in pounds * 703) / height in inches^2
So, the curve that separates Overweight people from the rest is:
weight = (25 * height in inches^2)/703

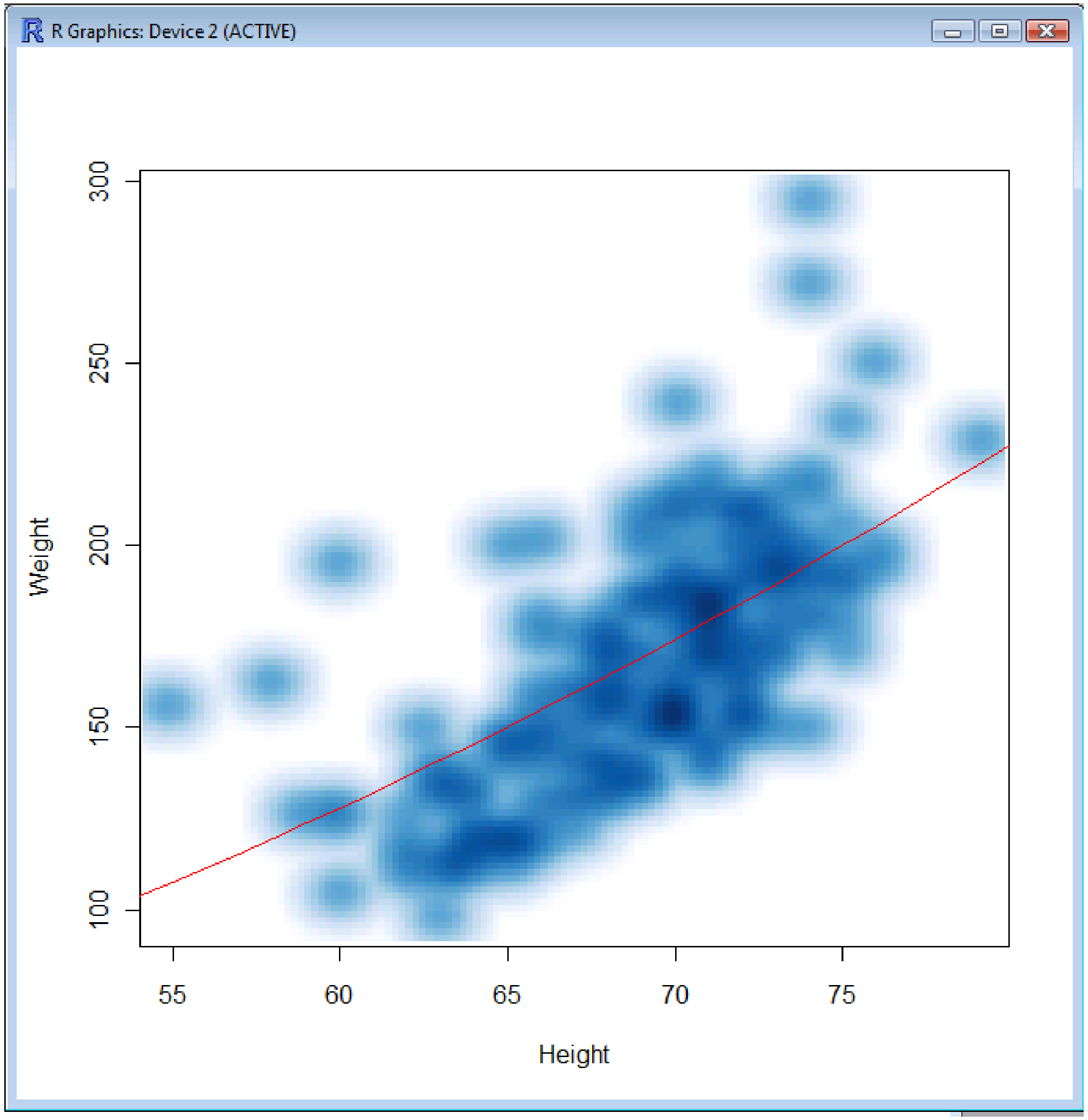
In R:

```
sex<-ifelse(DataStudents$Gender==1,'blue','pink')
plot(DataStudents$Height, DataStudents$Weight, col=sex)
x1<- 50:100 #height
y1<-  25 * x1 * x1 / 703  #weight
points(x1, y1,type='l',col='red')
```



It seems like more men than women are Overweight in this sample data. But there are almost 3 times more men than women and the scatter plot shows one plot for one or more coincident data values and draws the blue over the pink!

```
smoothScatter(DataStudents, nrpoints=0)
x1<- 50:100 #height
y1<-  25 * x1 * x1 / 703  #weight
points(x1, y1,type='l',col='red')
```



Connecting to Excel through ODBC

```
library(RODBC)
connection <- odbcConnectExcel("Forbes2000.xls", readOnly = TRUE)
#odbcConnectExcel2007
connection
sqlTables(connection)

odbcGetInfo(connection)

sqlFetch(connection,'Sheet1$')
sqlQuery(connection, "select * from [Sheet1$]")
dfForbes2000 <- sqlFetch(connection,'Sheet1$')

class(dfForbes2000)
# names of the columns
names(dfForbes2000)
colnames(dfForbes2000)
# names of the rows are the row numbers, usually plenty of them!
rownames(dfForbes2000)


close(connection)

layout(matrix(1:2, nrow = 2))
hist(dfForbes2000$marketvalue)
hist(log(dfForbes2000$marketvalue))
```

Problem: the data from a chemical analysis comes in several columns (table 1), each for a different concentration, 10, 20 and 30. AUC is area under the concentration-time curve. The concentration value is stored with the column name, but some statistical analysis would require it to be on a column of its own (table 2)

The data is on a TAB delimited file datawnoise.txt and it will have to be converted and saved onto file datawnoise2.txt

reactime	AUC10	AUC20	AUC30
10	361	729	1105
25	541	1089	1645
55	721	1449	2185
80	901	1809	2725
85	1081	2169	3265
105	1261	2529	3805
110	1441	2889	4345
135	1621	3249	4885
150	1801	3609	5425
155	1981	3969	5965

Table 1

ReacTime	Concentration	AUC
10	10	361
10	20	729
10	30	1105
25	10	541
25	20	1089
25	30	1645
55	10	721
55	20	1449
55	30	2185
80	10	901
80	20	1809
80	30	2725
85	10	1081
85	20	2169
85	30	3265
105	10	1261
105	20	2529
105	30	3805
110	10	1441
110	20	2889
110	30	4345
135	10	1621
135	20	3249
135	30	4885
150	10	1801
150	20	3609
150	30	5425
155	10	1981
155	20	3969
155	30	5965

Table 2

```r
# read table data, TAB separated
RTable<-read.table("datawnoise.txt", header = T, sep = "\t")
# examine the data
RTable
# store the number of rows and columns
iNrows<-dim(RTable)[1]
iNrows
iNcols<-dim(RTable)[2]
iNcols
# reactime values are needed for each AUC value
rep(RTable$reactime,iNcols-1)
# sort the repeated reactime values
ReacTime<-sort(rep(RTable$reactime,iNcols-1))
ReacTime
#get the AUC col names
sColName<-colnames(RTable)[-1]
sColName
# the concentration values are extracted from the AUC column names'
sub( "\\D+", "", sColName,perl = TRUE)
# concentration values are needed for each original row
Concentration<-rep(sub( "\\D+", "", sColName,perl = TRUE),iNrows)
Concentration
# convert to vector, by columns
AUC<-c(t(as.matrix(RTable[-1])))
#create a matrix with the new data
newdata<-cbind(ReacTime,Concentration,AUC)
# save the new data
write.table(newdata, file = "datawnoise2.txt", sep = "\t",row.names =FALSE, quote =FALSE)
```

Example from R Graph Gallery by Romain François
http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=139

Scatterplots with smoothed densities color representation

```
library("geneplotter")  ## from BioConductor
require("RColorBrewer") ## from CRAN

  x1  <- matrix(rnorm(1e4), ncol=2)
  x2  <- matrix(rnorm(1e4, mean=3, sd=1.5), ncol=2)
  x   <- rbind(x1,x2)

  layout(matrix(1:4, ncol=2, byrow=TRUE))
  op <- par(mar=rep(2,4))
  smoothScatter(x, nrpoints=0)
  smoothScatter(x)
  smoothScatter(x, nrpoints=Inf,
                colramp=colorRampPalette(brewer.pal(9,"YlOrRd")),
                bandwidth=40)
  colors  <- densCols(x)
  plot(x, col=colors, pch=20)

par(op)
```

Open PennState92.xls
Save as
PennState92.csv

```
PennStudents<-read.csv("PennState92.csv",row.names=1,header =F)
PennStudents
```

Statistical Inference

Distributions

standard univariate discrete distributions

binom	Binomial Distribution
nbinom	Binomial negative Distribution
pois	Poisson Distribution
geom	Geometric Distribution
hyper	Hipergeometric Distribution

Statistical Inference

Distributions

standard univariate continuous distributions $\left\{ \right.$

unif	Uniform Distribution
norm	Normal Distribution
lnorm	Log-normal Distribution
chisq	Chi Square Distribution
t	Student t Distribution
f	f Distribution Distribution
exp	Exponential Distribution
gamma	Gamma Distribution
weibull	Weibull Distribution
cauchy	Cauchy Distribution
beta	Beta Distribution
logis	Logistic
signrank	Wilcoxon Signed Rank Statistic
wilcox	Wilcoxon Rank Sum Statistic

Statistical Inference

Distributions

Multivariate continuous distributions

mvrnorm	multivariate normal (pkg MASS)
wish	Wishart (pkg MCMCpack)
iwish	inverse Wishart (pkg MCMCpack)
dirichlet	Dirichlet (pkg MCMCpack)
mvnorm	multivariate normal (pkg mvtnorm)
mvt	multivariate t (pkg mvtnorm)

Multivariate discrete distributions

multinom	multinomial

Statistical Inference

Distributions

$$\text{Functions for distribution "dist"} \left\{ \begin{array}{l} \text{ddist(x, ... params ..., log=FALSE) density function or probability density function, log=TRUE for log-likelihoods} \\ \text{pdist(q, ... params ..., lower.tail=TRUE, log.p=FALSE) distribution function (cumulative density function), lower.tail=FALSE for one-tailed upper p-values, log.p=TRUE for very small p-values} \\ \text{qdist(p, ... params ...,lower.tail=TRUE,log.p=FALSE) quantile function (inverse cumulative density function)} \\ \text{rdist(n, ... params ...) random deviate generator, n is the number of deviates} \end{array} \right.$$

Statistical Inference

Distributions

R has several algorithms for pseudo random number generators (RNG), these algorithms will generate the same sequence of pseudo random numbers by specifying the seed for the algorithm (to start the sequence) and the version number (the algorithms are updated for bugs and improvements)

RNG functions

RNGkind
RNGversion
set.seed

For simplicity, the examples will use the default RNG and change the seed to assure reproducibility of results

```
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
set.seed(2012) # setting a seed for the RNG
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
set.seed(2012) # setting a seed for the RNG
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
```

the RNG will return the same sequences for reproducibility of tests

Statistical Inference

Distributions

```
rnorm(1) # draw a sample of size 1 from a normal distribution
rnorm(5) # draw a sample of size 5 from a normal distribution
rnorm(5,mean=1,sd=3) # draw a sample of size 5 from a normal distribution with mean 1 and
standard deviation 3
rnorm(60, 4, 7) # draw a sample of size 60 from a normal distribution with mean 4 and standard
deviation 7

dnorm(0) # density for the normal distribution on point 0
dnorm(1) # density for the normal distribution on point 1
dnorm(3) # density for the normal distribution on point 3
pnorm(0)  # acumulated probability for the normal distribution below point 0
pnorm(3) # acumulated probability for the normal distribution below point 3
qnorm(0.5)   # quantile 50% of the normal distribution is 0
qnorm(0.9986501) # quantile for pnorm(3)
x<-seq(-4,4,length=200) # create a sequence of 200 values [-4, 4]
plot(x,dnorm(x),type="l") # plot a normal distribution

rpois(50, lambda=3) # draw a sample of size 50 from a Poisson distribution with lambda=3

rbinom(100, 40, .25) # draw a sample of size 100 from a Binomial distribution with size=40 and
prob=.25
```

Statistical Inference

Sampling

sample draws a random sample from a population, replacement=T for sampling with replacement

```
sample(5) # random permutation of sequence [1, 5]
sample(20) # random permutation of sequence [1, 20]

sample(seq(3:45), 10) # random sample of size 10 from [3, 45]

sample(4, 10, prob = c(0.3, 0.5, 0.1, 0.1), replace = T) # random sample of size 10 from sequence [1, 4] with different probabilities of being chosen P(1)=.3, P(2)=.5 etc...

sample(c(0,1), 20, replace = TRUE) # 20 Bernoulli trials

sample(c("heads","tails"),1) # flipping a coin once
sample(c("heads","tails"),5, replace=T) # flipping a coin 5 times

sample(6,1) # rolling a dice once
sample(6,20,replace=T) # rolling a dice 20 times

sample(c("rock","paper", "scisors"),1) # draw rock-paper-scisors once

sample(39, 7) # drawing lottery numbers
```

Statistical Inference

Tests

One-Sample and Paired Data	t.test	t-test
	wilcox.test	Wilcoxon signed rank
Two-Sample	t.test	t-test
	wilcox.test	Wilcoxon 2-sample rank-sum
k-Sample	kruskal.test	Kruskal-Wallis
	oneway.test	One-way ANOVA
Unified Unpaired Nonparametric Tests	spearman2	

Statistical Inference

Tests

```
# Student's sleep data
plot(extra ~ group, data = sleep)
# t test
t.test(extra ~ group, data = sleep)
```

Statistical Inference

Tests

Explanatory / response	continuous	categorical
categorical	Logistic regression	Cpntigency tables, 2x2, Chi2, Fisher
continuous	Regression, correlation	Anova, t-test

Statistical Inference

Correlations

cor( ) correlations

cov( ) covariances

cor.test( ) test a single correlation coefficient

corrgram( ) plot correlograms

```
# Correlations/covariances among numeric variables in
# dataframe mtcars. Use listwise deletion of missing data.
cor(mtcars, use="complete.obs", method="kendall")
cov(mtcars, use="complete.obs")

# Correlation matrix from mtcars
# with mpg, cyl, and disp as rows
# and hp, drat, and wt as columns
x <- mtcars[1:3}
y <- mtcars[4:6]
cor(x, y)
```

Statistical Inference

Correlations

```
# First Correlogram Example
library(corrgram)
corrgram(mtcars, order=TRUE, lower.panel=panel.shade,  upper.panel=panel.pie,
text.panel=panel.txt,  main="Car Milage Data in PC2/PC1 Order")

# Second Correlogram Example
library(corrgram)
corrgram(mtcars, order=TRUE, lower.panel=panel.ellipse,  upper.panel=panel.pts,
text.panel=panel.txt,  diag.panel=panel.minmax,  main="Car Milage Data in PC2/PC1 Order")

# Third Correlogram Example
library(corrgram)
corrgram(mtcars, order=NULL, lower.panel=panel.shade,  upper.panel=NULL,
text.panel=panel.txt,  main="Car Milage Data (unsorted)")
```

http://www.statmethods.net/stats/correlations.html

Statistical Inference

Linear Regression

Model Fitting

Multiple linear regression: lm, ols [Residuals: residuals.ols]
General linear model: glm, glmD (just change glm to glmD in call)
Binary logistic model: glm, lrm
Ordinal logistic model: lrm [Residuals: residuals.lrm]
Parametric survival models: survreg, psm
Cox proportional hazards model: coxph, cph [Residuals: residuals.coxph, residuals.cph]
Buckley-James censored least squares regression: bj

After-Fitting Analysis

Specifications for predictor transformations used by Design: specs
Predictions and confidence intervals: predict.Design
Overly influential observations: which.influence
Sensitivity to unmeasured confounder in lrm: sensuc
Create S function to evaluate fitted equation: Function
Compose LATEX code for typesetting algebraic expressions containing model fit: latex.Design
Odds and hazard ratios and effect differences: summary.Design
General contrasts and CLs: contrast.Design
ANOVA: anova.Design
Fast backward stepdown variable selection: fastbw
Huber-White-Efron robust covariance matrix estimator with optional cluster sampling adjustment: robcov
Bootstrap nonparametric covariance matrix estimator with optional cluster sampling adjusting: bootcov
Generate data frame with predictor combinations: gendata

Statistical Inference

Linear Regression

Years of experience and executive salaries in millions

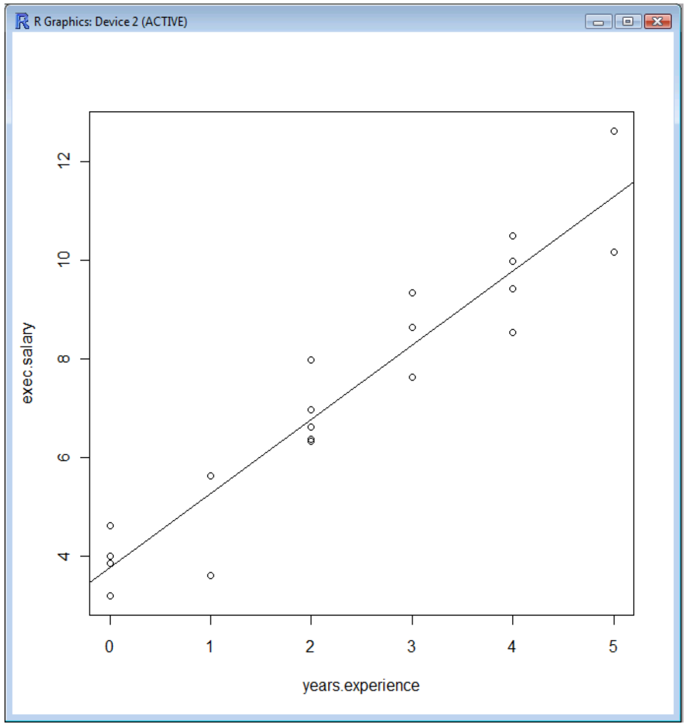
```
exec.df = read.delim("salary.txt",col.names=c("years.experience","exec.salary"), header = F)
class(exec.df)
plot(exec.df)
attach(exec.df)
exec.salary.lm  = lm(exec.salary~years.experience)    # regression
abline(exec.salary.lm) # regression line
summary(exec.salary.lm) # SUMMARY OF THE REGRESSION PROCESS
exec.salary.lm$residuals # check residuals
plot(exec.salary.lm$residuals) # plot residuals
abline(h=0)
# horizontal line on y=0 because residuals are centered around it
# data far from this line was not predicted well by the regression model, because the residual is high

exec.salary.lm$fitted.values # predicted salary for each executive, by the adjusted model
predict.lm(exec.salary.lm,data.frame(years.experience=0)) # to predict the salary for a new executives, with 0 years of experience
predict.lm(exec.salary.lm,data.frame(years.experience=c(1.5,2,3.5))) # to predict the salary for 3 new executives, with 1.5, 2 and 3.5 years of experience
```

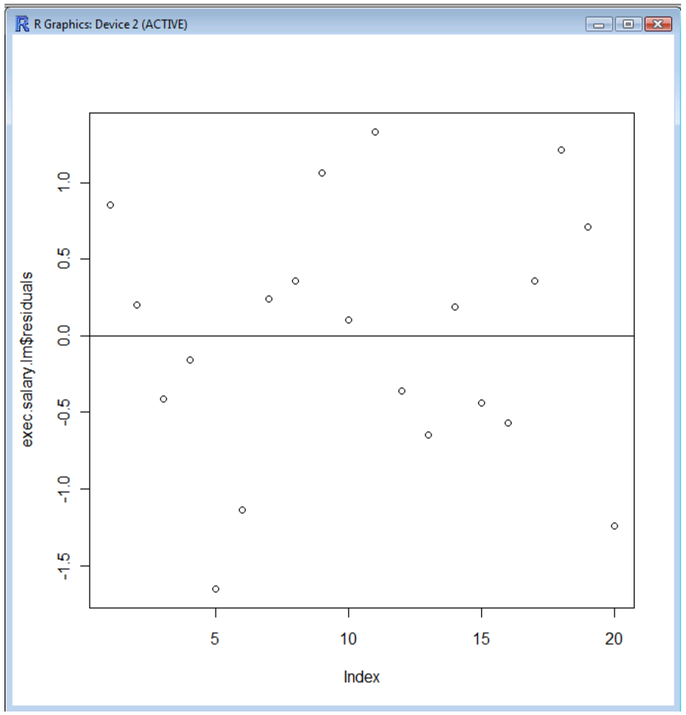
Statistical Inference

Linear Regression

Simple Linear Regression

0	4.61
2	6.97
2	6.36
2	6.61
1	3.61
5	10.15
0	4.00
3	8.63
3	9.34
0	3.86
5	12.62
4	9.42
3	7.63
4	9.97
2	6.33
0	3.19
1	5.62
2	7.98
4	10.49
4	8.54





Statistical Inference

Linear Regression

Multiple linear regression

Instead of a regression line there is a regression plane

```
data(mtcars) # load dataset
attach(mtcars)
cars.lm = lm(mpg~hp+wt) # explain gas milleage in function of power and weight
summary(cars.lm)
# the model is: gas milleage = 37.22 - 0.03 power - 3.87 weight
# the more powerful the car, the lower the MPG, less milles per gallon
# the heavier the car, the lower the MPG, less milles per gallon
# R-Squared is 82%, these 2 variables explain the gas milleage very well

# let's draw the residuals to check if any car behaves differently
plot(cars.lm$residuals)
abline(h=0)

# to predict how many milles per gallon a car with 150 horse power and weight 2.t tons:
predict.lm(cars.lm,data.frame(hp=150,wt=2.5))
```

Statistical Inference

ANOVA

When is Anova Used?
• All explanatory variables are categorical—unquantified and unordered
• The explanatory variables are called ‘factors’; each has two or more levels.
• If there is one factor with two levels, use Student’s t.
• If there is one factor with three+ levels, use one-way Anova.
• If there are two factors, use two-way Anova.
• For three factors, use three-way Anova, and so on…
• If every combination of factors is present, you have a factorial design, allowing you to study interactions between variables (and order no longer matters!).

Statistical Inference

ANOVA

Modelling the mileage (mpg) with variables weight (wt), transmission type (am), and/or the number of cylinders (cyl), 3 models:

```
data(mtcars) # load dataset
res.lm = lm(mpg ~ wt, data = mtcars)
res.lm2 = lm(mpg ~ wt + cyl, data = mtcars)
res.lm3 = lm(mpg ~ wt + cyl + am, data = mtcars)

#Applying anova() to a single model object produces an analysis of variance for computing the F-test of whether the constant mean model is appropriate
anova(res.lm)

# there is a relationship between mpg and wt

# Applying anova() to two model objects for test if nested models produces an analysis of variance for computing the F-test of whether the extra term is warranted.

anova(res.lm, res.lm2)

# the differences are significant, the number of cylinders seems to have a statistically significant effect.

# Applying anova() to three nested models produces sequential F-tests.
anova(res.lm, res.lm2, res.lm3)

# This shows that in the model mpg modeled by wt and cyl, the cyl variable is statistically significant.
# However, in the full model of mpg modeled by wt, cyl, and am, the variable am is not statistically significant.
```

# http://wiener.math.csi.cuny.edu/st/stRmanual/anova.pdf

Statistical Inference
References/to learn more:

The R book
Michael J. Crawley pp 370
2018 John Wiley & Sons Ltd

Basic statistics using R pp. 213
Jarno Tuimala (CSC) and Dario Greco (HY)
http://www.csc.fi/english/csc/courses/archive/R2008s

Statistics: an introduction using R
Michael J. Crawley pp 125
2016 John Wiley & Sons Ltd

Statistics with R
Vincent Zoonekynd, pp 620
http://zoonek2.free.fr/UNIX/48_R/all.html

Aprendizaje del software estadístico R: un entorno para simulación y computación estadística
Prof. Alberto muñoz garcía
Departamento de Estadística
Universidad Carlos III de Madrid
http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/resolveUid/4b28fd8154f6521f963aa058ec6baf31

Introductory Statistics with R
Peter Dalgaard, pp 109
2018 Springer

Geographic Data Analysis
Pat Bartlein
http://geography.uoregon.edu/bartlein/courses/geog417/lectures/lec10.htm

Software Tools, Part 1: introduction to R software
Petri Koistinen
http://www.rni.helsinki.fi/~pek/s-tools/test-ci.r

Chem 351 Archives Page
David Harvey
http://fs6.depauw.edu:50080/~harvey/Chem%20351/PDF%20Files/Handouts/RDocs/Using%20R%20for%20Linear%20Regression.pdf

Thomas AP Statistics
thomasmathematics.com
http://www.thomasmathematics.com/Aims/Ch3Aim30001.pdf

Quick-R
Rob Kabacoff
http://www.statmethods.net/stats/correlations.html

The Stem and Tendril simplified R manual
Professors Franzblau, Poje and Verzani of the College of Staten Island
http://wiener.math.csi.cuny.edu/st/stRmanual/

- The rgdal Package

October 17, 2008
Title for the Geospatial Data Abstraction Library
Version .5-27
Date -10-09
Depends (>= 2.3.0), methods, sp
LazyLoad
Description bindings to Frank Warmerdam's Geospatial Data Abstraction Library (GDAL) (>= 1.3.1) and access to projection/transformation operations from the PROJ.4 library. The
GDAL and PROJ.4 libraries are external to the package, and, when installing the package from source, must be correctly installed first. Both GDAL raster and OGR vector map data can be imported into R, and GDAL raster data and OGR vector data exported. Use is made of classes defined in the sp package.
Author H. Keitt <tkeitt@mail.utexas.edu>, Roger Bivand <Roger.Bivand@nhh.no>, Edzer Pebesma <e.pebesma@geo.uu.nl>, Barry Rowlingson
Maintainer Bivand <Roger.Bivand@nhh.no>
License (>= 2)
URL ://www.gdal.org, http://rgdal.sourceforge.net/, http://sourceforge.net/projects/rgdal/
System Requirements for building from source:GDAL >= 1.3.1 library from http://www.gdal.org/download.html and PROJ.4 (proj >= 4.4.9) from http://proj.maptools.org/

- OGR Simple Feature Library

The OGR Simple Features Library is a C++ open source library (and commandline tools) providing read (and sometimes write) access to a variety of vector file formats including ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, and Mapinfo mid/mif and TAB formats.

OGR is a part of the GDAL library.
http://www.gdal.org/ogr/

R and GIS

R has several packages that can work with GIS data, the most commonly used is the rgdal Package. The acronym rgdal stands for "R Geospatial Data Abstraction Library".
The rgdal Package provides bindings to Frank Warmerdam's Geospatial Data Abstraction Library (GDAL), this library can work with both raster and vector data in many of the available GIS formats in use. The vector library (OGR) is incorporated into GDAL (raster library) and it is fine to mention either one as separate libraries or GDAL as a whole. RGDAL can work with GDAL raster and OGR vector map files, and it can use both together.

Using rgdal

Loading the library:

```
> library(rgdal)
Loading required package: sp
Geospatial Data Abstraction Library extensions to R successfully loaded
Loaded GDAL runtime: GDAL 1.5.3, released 2008/09/09
GDAL_DATA: C:/PROGRA~1/R/R-28~1.0/library/rgdal/gdal
Loaded PROJ.4 runtime: Rel. 4.6.1, 21 August 2008
PROJ_LIB: C:/PROGRA~1/R/R-28~1.0/library/rgdal/proj
>
```

To get a list of the available drivers:

```
> getGDALDriverNames()
     name                          long_name create  copy
1 AAIGrid              Arc/Info ASCII Grid  FALSE  TRUE
2    ADRG     ARC Digitized Raster Graphics   TRUE FALSE
3     AIG             Arc/Info Binary Grid  FALSE FALSE
...
```

These are just the first 3, there are over 70, including geoTIFF, ESRI HDR, Erdas IMG, Idrisi RST, USGS DEM, etc...

```
# To get the gdal version:
getGDALVersionInfo()

# Loading dem30m_erdas.img (about 5 Megs):
dem30 <- readGDAL('dem30m_erdas.img')

# Getting the projection, datum, etc...
proj4string(dem30)

# What kind of variable is dem30?
class(dem30)

# Getting detailed information about dem30:
summary(dem30)

# dimensions and their names
dim(dem30)
names(dem30)

# Plotting a density map:
plot(density(dem30$band1))

# read as GDALReadOnlyDataset
dem30 <- GDAL.open('dem30m_erdas.img', read.only = TRUE)
# Displaying the raster
displayDataset(dem30,offset=c(0,0), region.dim=dim(dem30), reduction=1, band=1)

# Lattice (trellis) plot method for spatial data with attributes
spplot(as(dem30, "SpatialGridDataFrame"))
```

```
# storing the elevation on a matrix
rt <- as.matrix(getRasterTable(dem30)[,3])

# fixing the dimensions
rm<-matrix(rt,dim(dem30)[1],dim(dem30)[2],byrow=T)
dim(rt)
dim(rm)

# using function image to show the DEM
image(1:dim(dem30)[1],1:dim(dem30)[2],rm)

# using function persp to show the DEM
v1 <- seq(1,dim(dem30)[1],100)
v2 <- seq(1,dim(dem30)[2],100)
persp(1:length(v1), 1:length(v2),rm[v1,v2] )
```

Cholera mortalities, Soho

Load unit0_slides.R

and read unit0_slides-2x2.pdf

To learn more:

One-day introductory course on Spatial Data Analysis with R
www.bias-project.org.uk/ASDARcourse

Geographic Data Analysis
Geog 4/517, Pat Bartlein
http://geography.uoregon.edu/bartlein/courses/geog417/syll09.htm

R for Medicine and Biology
Paul D. Lewis pp 58
Jones and Bartlett Series in Biomedical Informatics

Applied Spatial Data Analysis with R
http://www.amazon.com/Applied-Spatial-Data-Analysis-Use/dp/0387781706