Introduction to R

Chao-Lung Yang, Ph.D.
Department of Industrial Management
National Taiwan University of Science and Technology

Notes

- The following materials are borrowed and modified based on slides and examples from
- 1. Dr. Hung Chen http://www.math.ntu.edu.tw/~hchen/Prediction/notes/R-programming.ppt
- 2. Dr. Henry Horng-Shing Lu
- http://www.stat.nctu.edu.tw/~misg/hslu/course/statistics/An_Introduction_of_R.pdf
- 3. Oscar Torres-Reyna *http://dss.princeton.edu/training/**RStudio**101.pdf*

R and S-Plus

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

R

- R is “GNU S” — A language and environment for data manipula-tion, calculation and graphical display.
  - R is similar to the award-winning S system, which was developed at Bell Laboratories by John Chambers et al.
  - a suite of operators for calculations on arrays, in particular matrices,
  - a large, coherent, integrated collection of intermediate tools for interactive data analysis,
  - graphical facilities for data analysis and display either directly at the computer or on hardcopy
  - a well developed programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.

What R Does and Does Not

- data handling and storage: numeric, textual
- matrix algebra
- hash tables and regular expressions
- high-level data analytic and statistical functions
- classes (“OO”)
- graphics
- programming language: loops, branching, subroutines

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

R and Statistics

- Packaging: a crucial infrastructure to efficiently produce, load and keep consistent software libraries from (many) different sources / authors
- Statistics: most packages deal with statistics and data analysis
- State of the art: many statistical researchers provide their methods as R packages

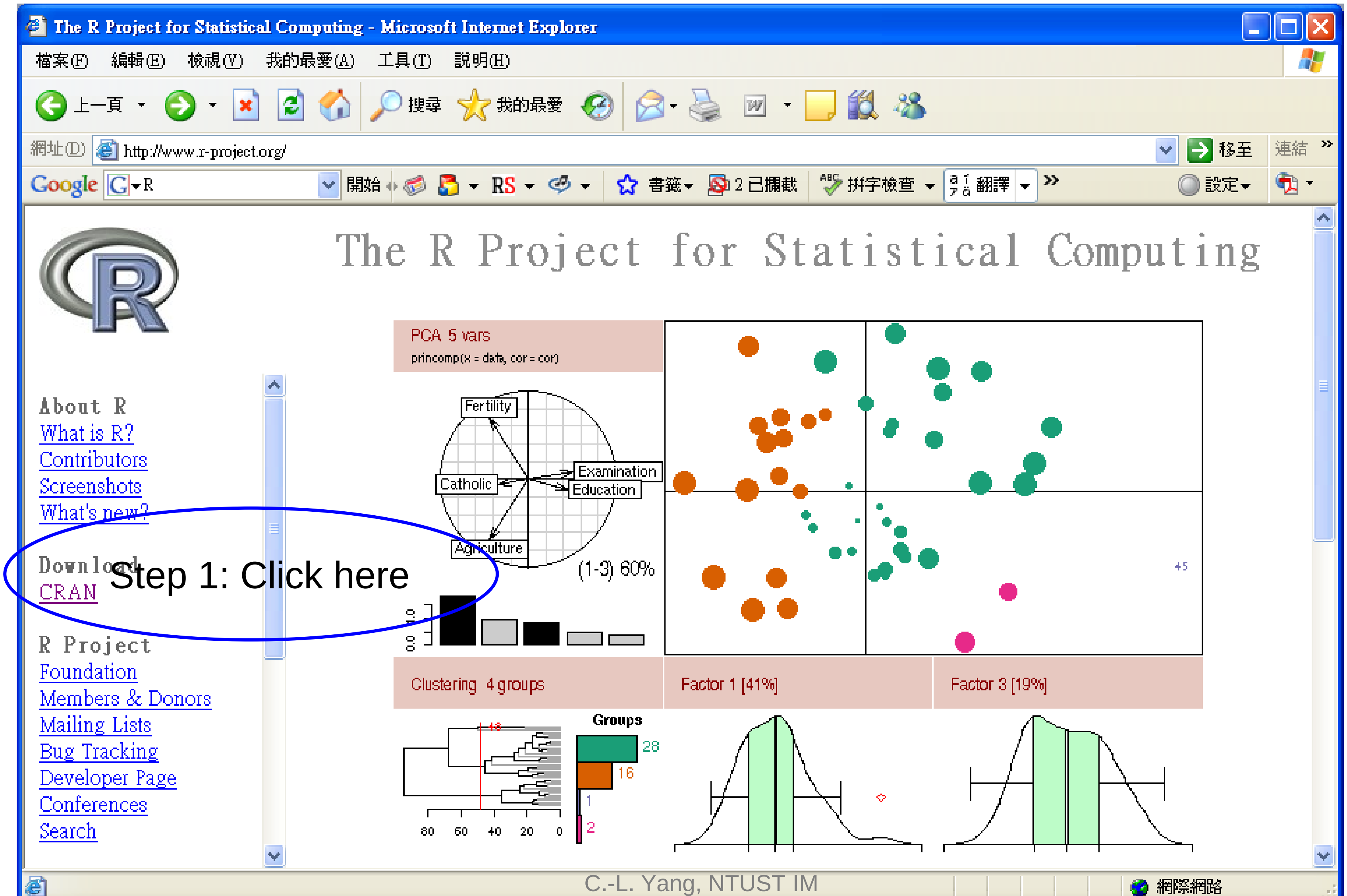
Data Analysis and Presentation

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

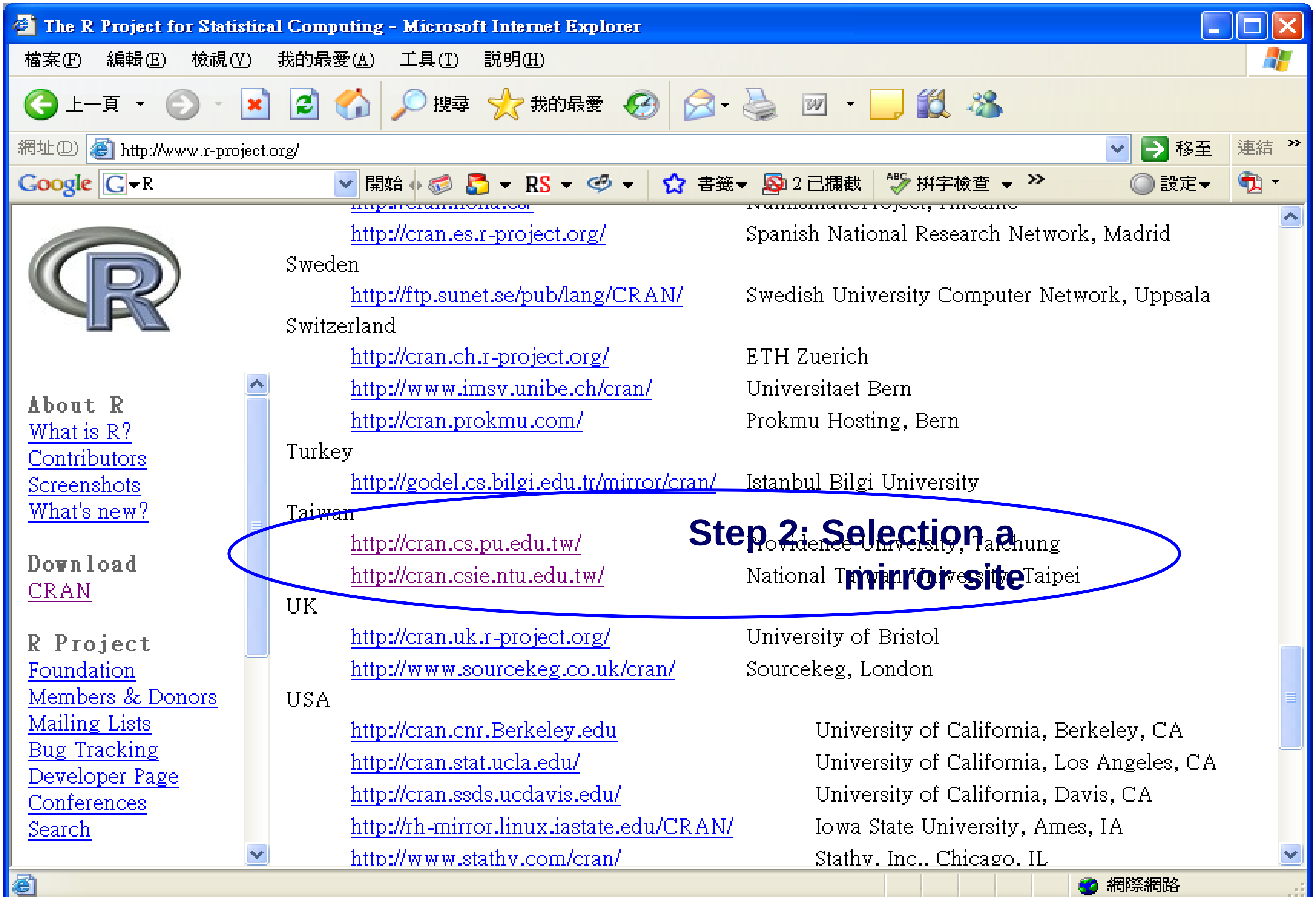
Installation of R

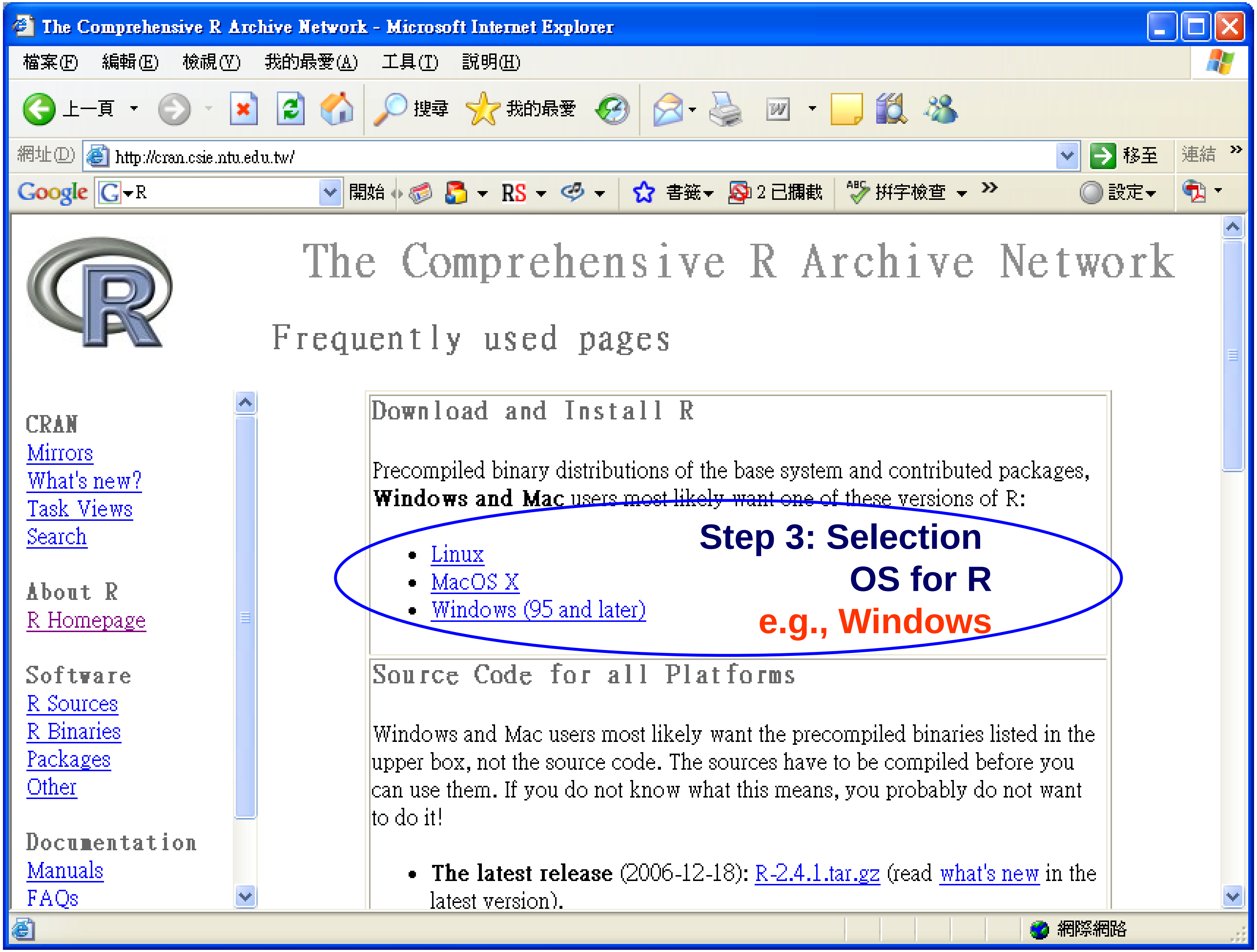
Installation of R

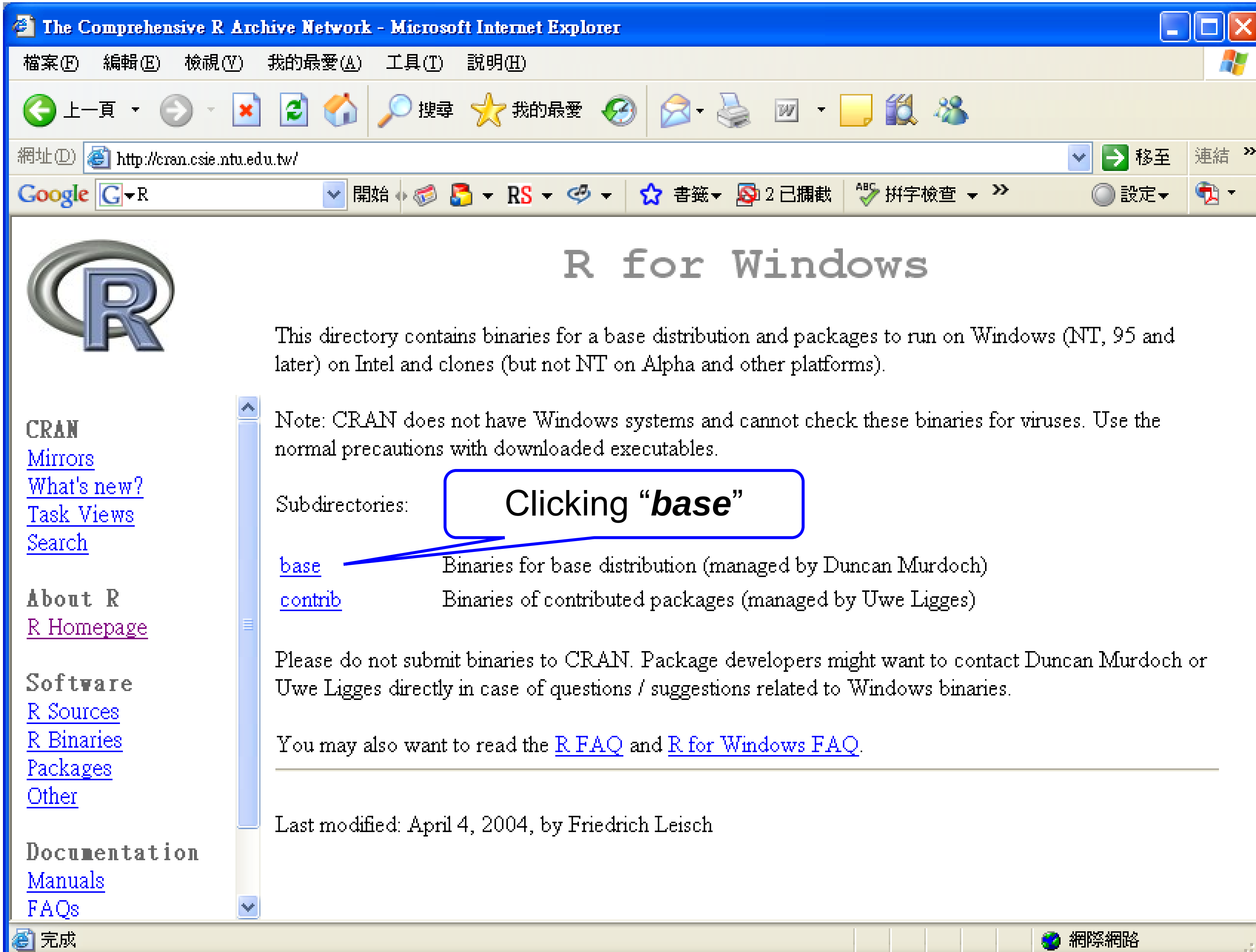
http://www.r-project.org/

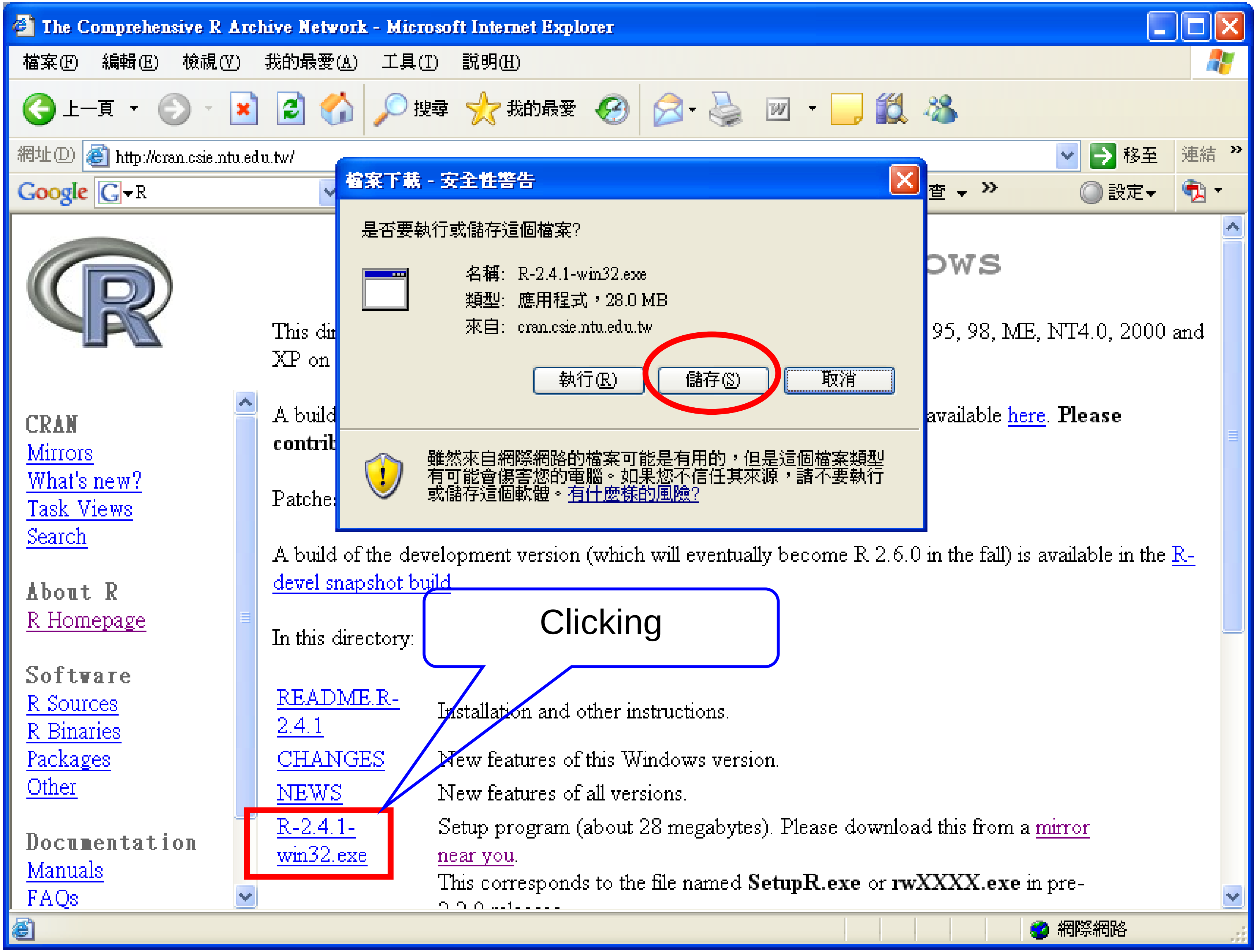


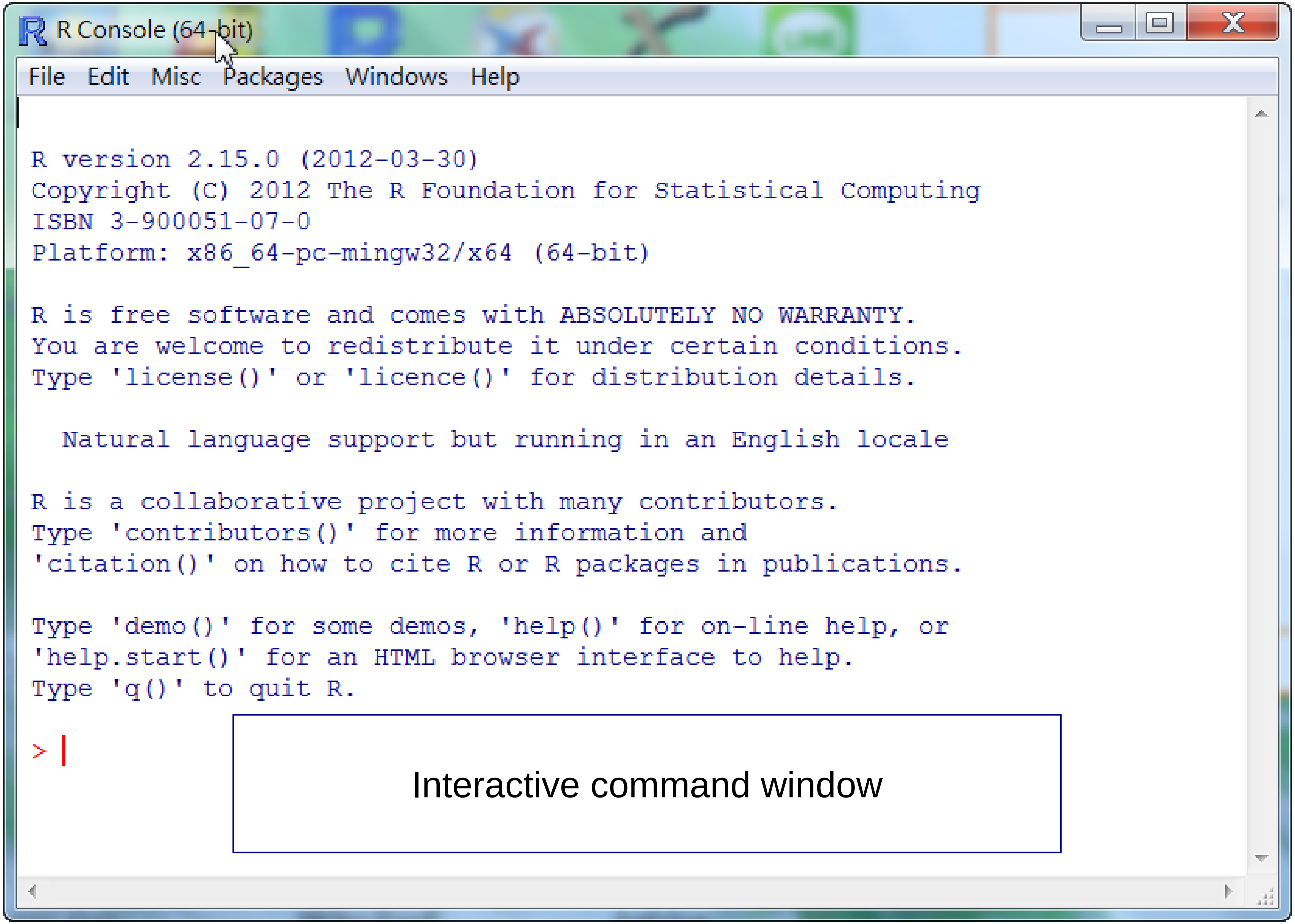
Selection a Mirror Site




The Comprehensive R Archive Network - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://cran.csie.ntu.edu.tw/
移至 連結
The Comprehensive R Archive Network
Frequently used pages
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
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 (95 and later)
Step 3: Selection
OS for R
e.g., 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 (2006-12-18): R-2.4.1.tar.gz (read what's new in the latest version).
網際網路


The Comprehensive R Archive Network - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D) http://cran.csie.ntu.edu.tw/
移至
連結
R for Windows
This directory contains binaries for a base distribution and packages to run on Windows (NT, 95 and later) on Intel and clones (but not NT on Alpha and other platforms).
Note: CRAN does not have Windows systems and cannot check these binaries for viruses. Use the normal precautions with downloaded executables.
Subdirectories:
Clicking "*base*"
base
Binaries for base distribution (managed by Duncan Murdoch)
contrib
Binaries of contributed packages (managed by Uwe Ligges)
Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.
You may also want to read the R FAQ and R for Windows FAQ.
Last modified: April 4, 2004, by Friedrich Leisch
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
完成
網際網路

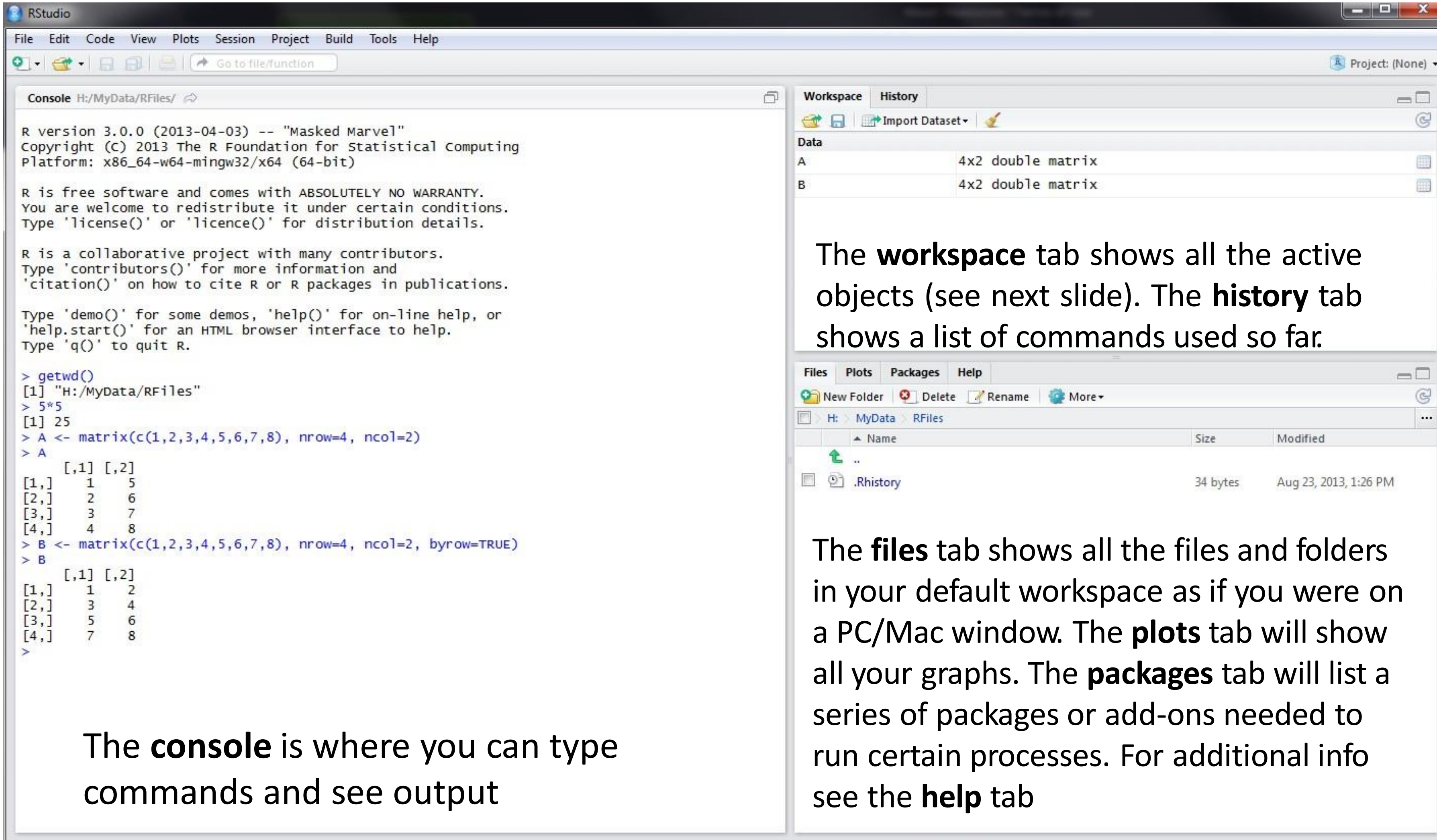

The Comprehensive R Archive Network - Microsoft Internet Explorer
檔案下載 - 安全性警告
是否要執行或儲存這個檔案?
名稱: R-2.4.1-win32.exe
類型: 應用程式，28.0 MB
來自: cran.csie.ntu.edu.tw
執行(R)
儲存(S)
取消
雖然來自網際網路的檔案可能是有用的，但是這個檔案類型有可能會傷害您的電腦。如果您不信任其來源，請不要執行或儲存這個軟體。有什麼樣的風險?
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
A build of the development version (which will eventually become R 2.6.0 in the fall) is available in the R-devel snapshot build
In this directory:
README.R-2.4.1
Installation and other instructions.
CHANGES
New features of this Windows version.
NEWS
New features of all versions.
R-2.4.1-win32.exe
Setup program (about 28 megabytes). Please download this from a mirror near you.
This corresponds to the file named SetupR.exe or rwXXXX.exe in pre-
Clicking


R Console (64-bit)
File Edit Misc Packages Windows Help
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)
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.
>
Interactive command window

RStudio

- RStudio allows the user to run R in a more user-friendly environment. It is open-source (i.e. free) and available at http://www.rstudio.com/

RStudio screen



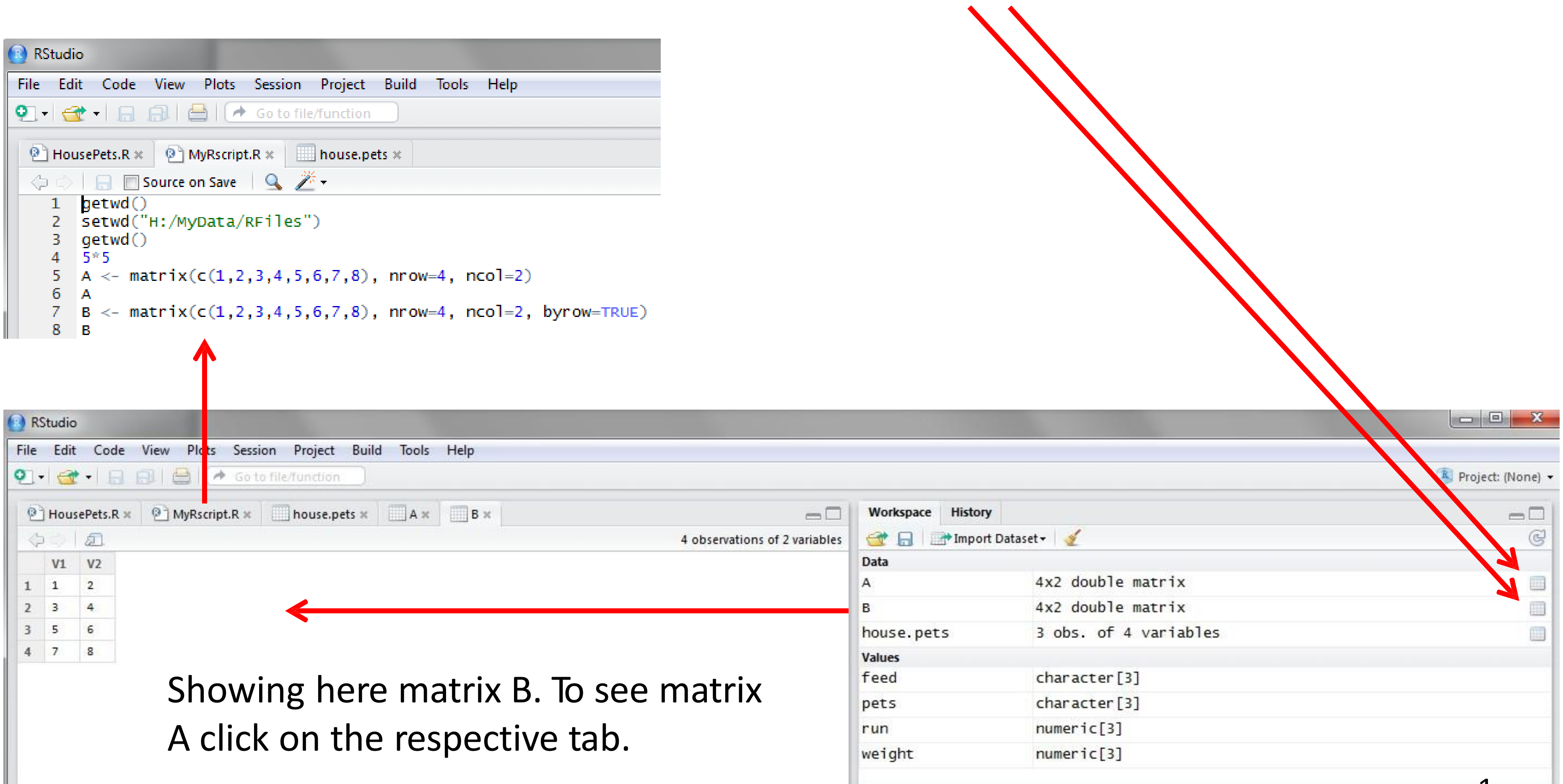
The **workspace** tab shows all the active objects (see next slide). The **history** tab shows a list of commands used so far.

The **files** tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your graphs. The **packages** tab will list a series of packages or add-ons needed to run certain processes. For additional info see the **help** tab

The **console** is where you can type commands and see output

Workspace tab (1)

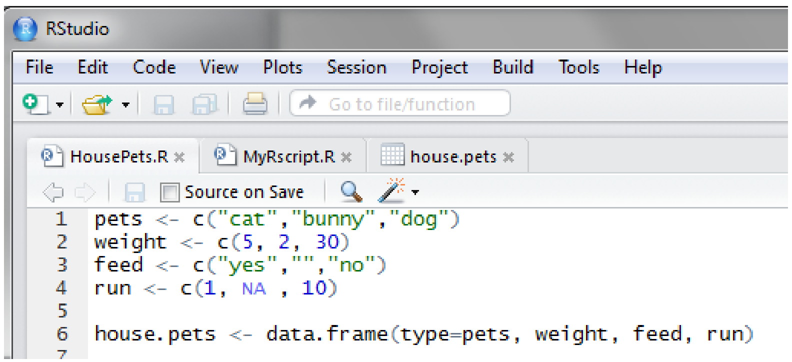
The workspace tab stores any object, value, function or anything you create during your R session. In the example below, if you click on the dotted squares you can see the data on a screen to the left.



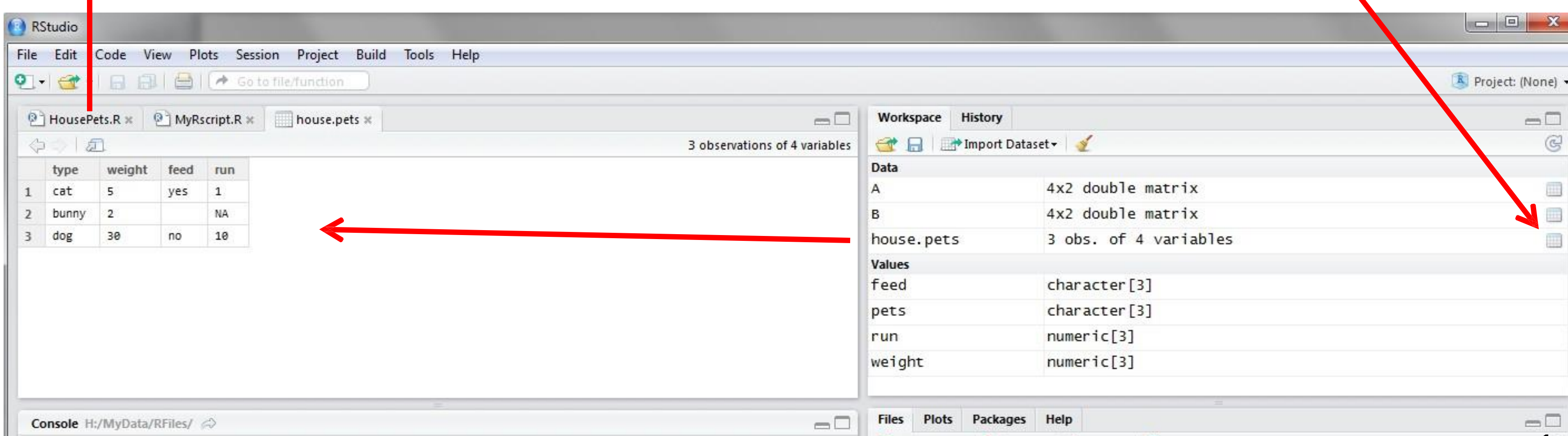
Showing here matrix B. To see matrix A click on the respective tab.

Workspace tab (2)

Here is another example on how the workspace looks like when more objects are added. Notice that the data frame `house.pets` is formed from different individual values or vectors.



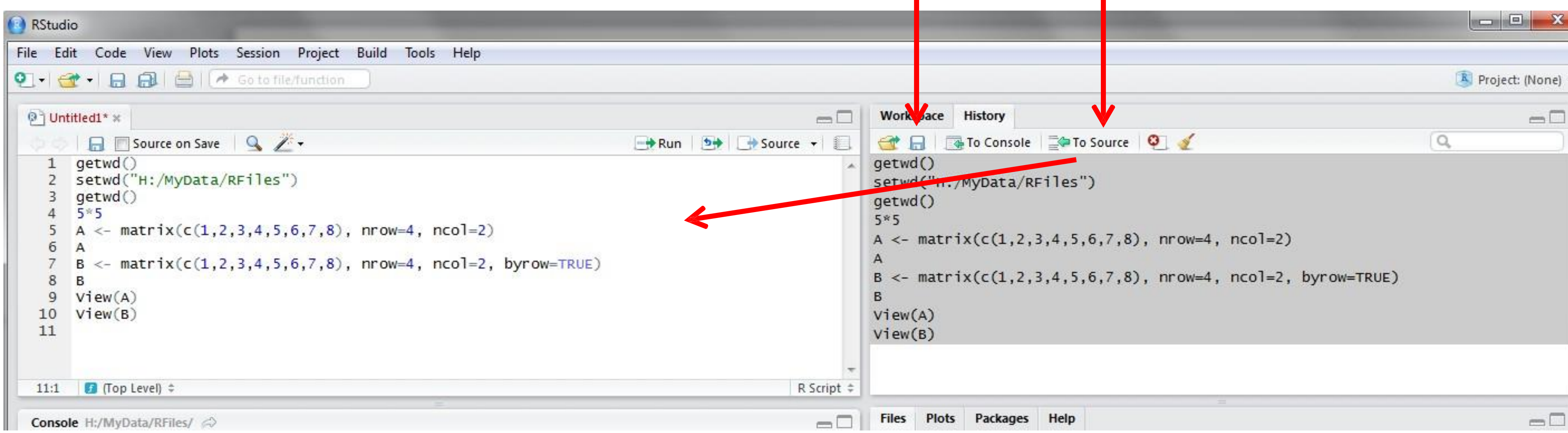
Click on the dotted square to look at the dataset in a spreadsheet form.



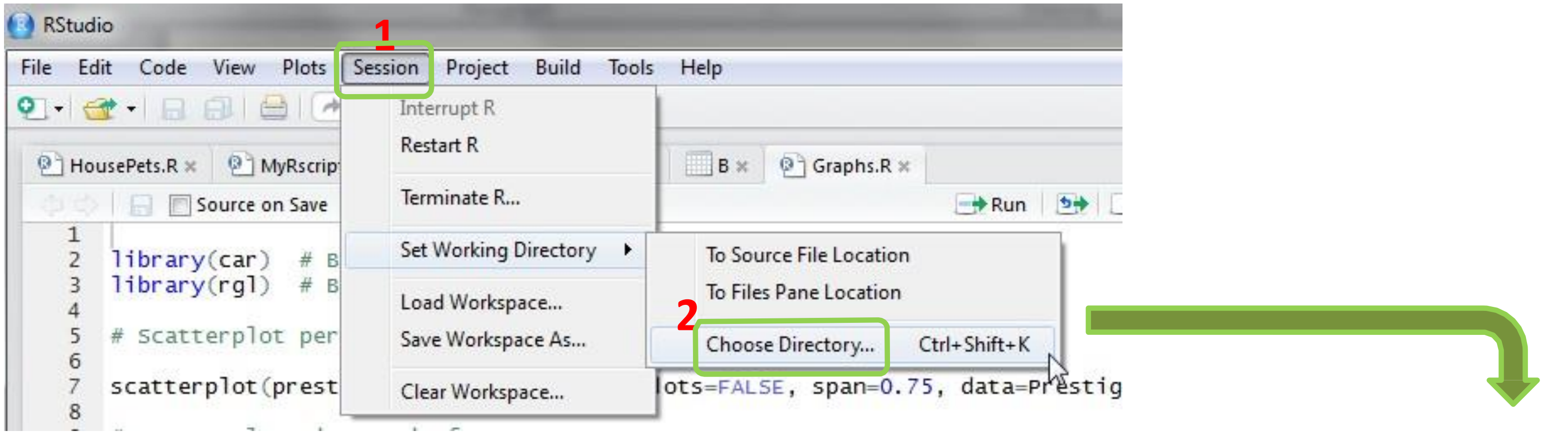
History tab

The history tab keeps a record of all previous commands. It helps when testing and running processes. Here you can either **save** the whole list or you can **select** the commands you want and send them to an R script to keep track of your work.

In this example, we select all and click on the “`To Source`” icon, a window on the left will open with the list of commands. Make sure to save the ‘untitled1’ file as an *.R script.

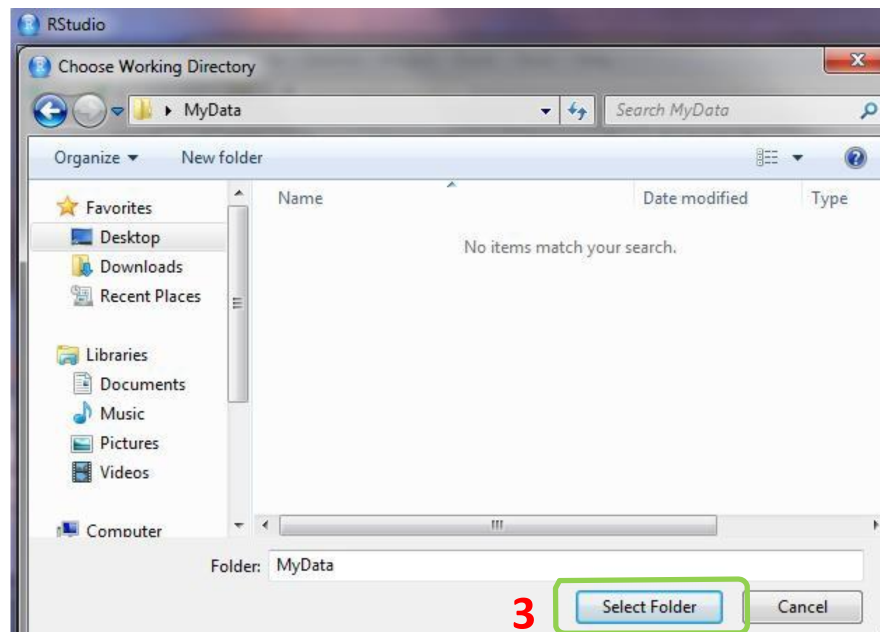


Changing the working directory



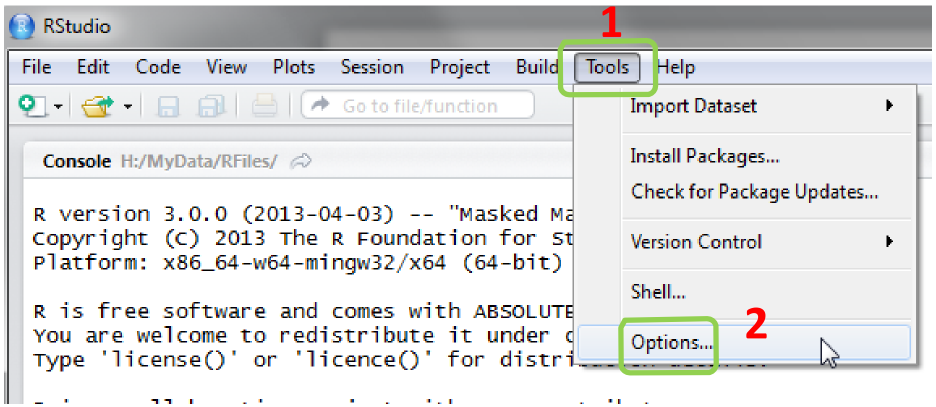
If you have different projects you can change the working directory for that session, see above. Or you can type:

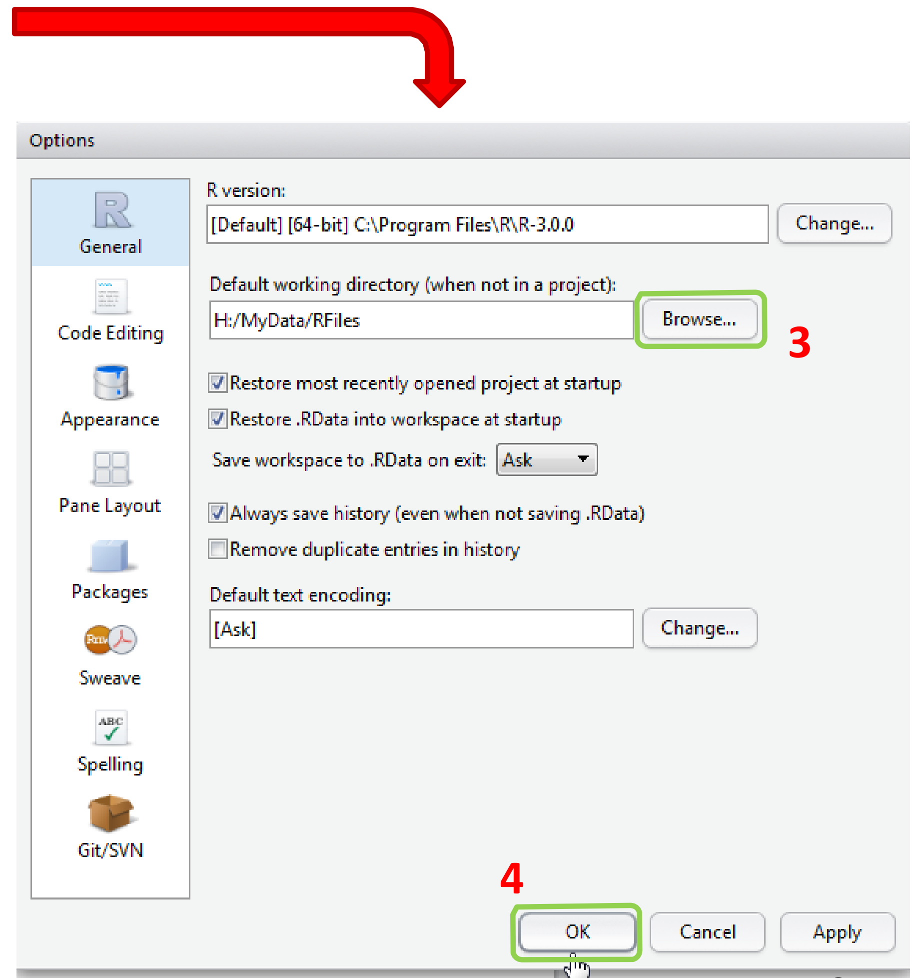
```
# Shows the working directory (wd)

getwd()

# Changes the wd

setwd("C:/myfolder/data")
```



More info see the following document: http://dss.princeton.edu/training/RStata.pdf

Setting a default working directory





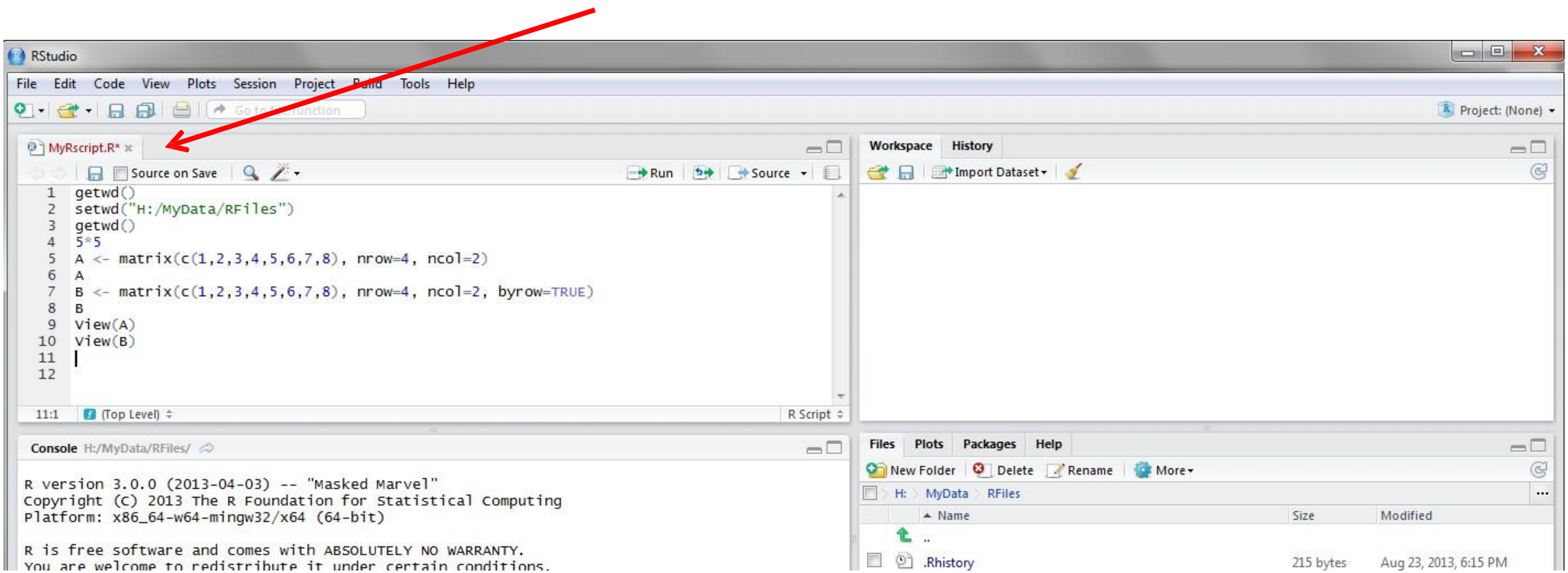
Every time you open RStudio, it goes to a default directory. You can change the default to a folder where you have your datafiles so you do not have to do it every time. In the menu go to Tools->Options

R script (1)

The usual Rstudio screen has four windows:

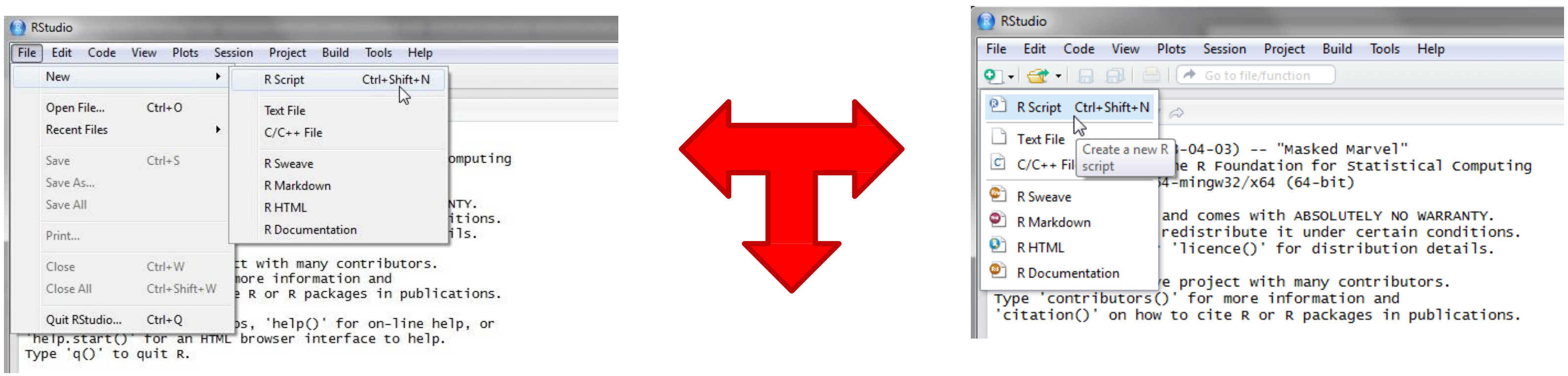
1. Console.
2. Workspace and history.
3. Files, plots, packages and help.
4. The R script(s) and data view.

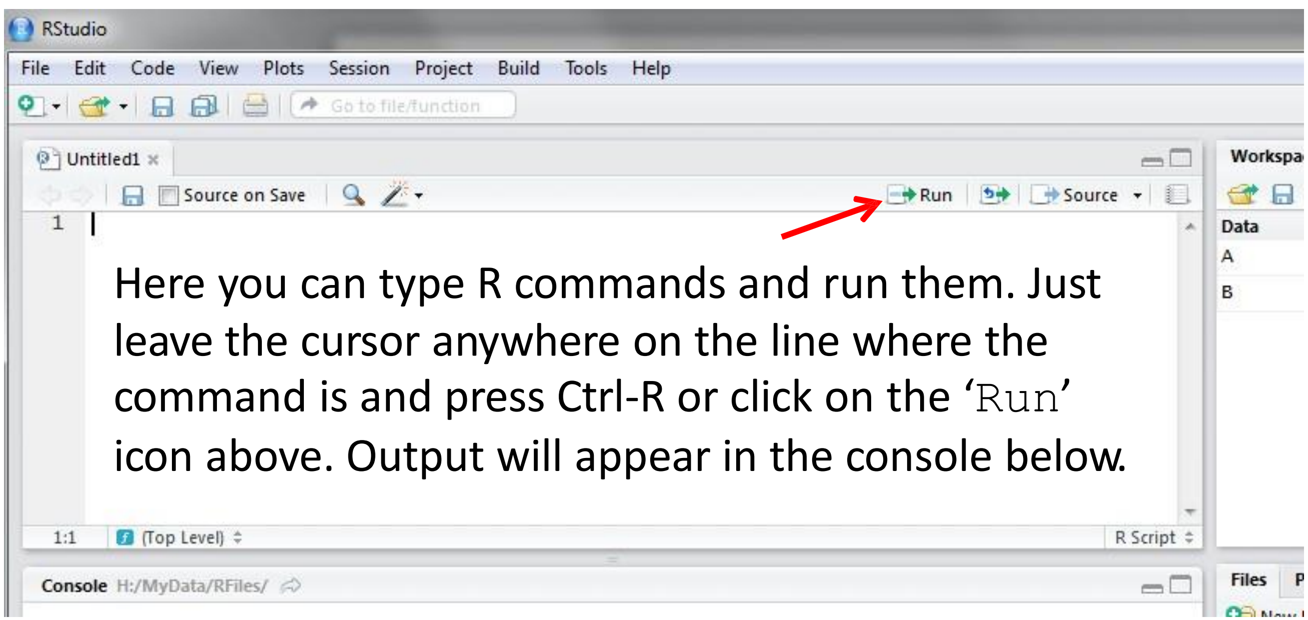
The R script is where you keep a record of your work. For Stata users this would be like the do-file, for SPSS users is like the syntax and for SAS users the SAS program.



R script (2)

To create a new R script you can either go to `File -> New -> R Script`, or click on the icon with the "+" sign and select "R Script", or simply press Ctrl+Shift+N. Make sure to save the script.

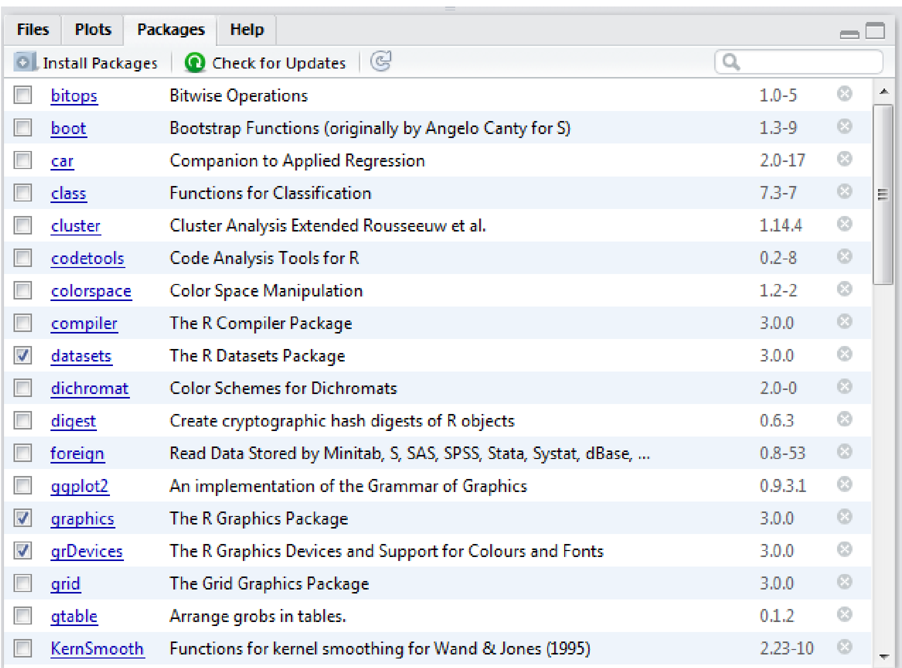


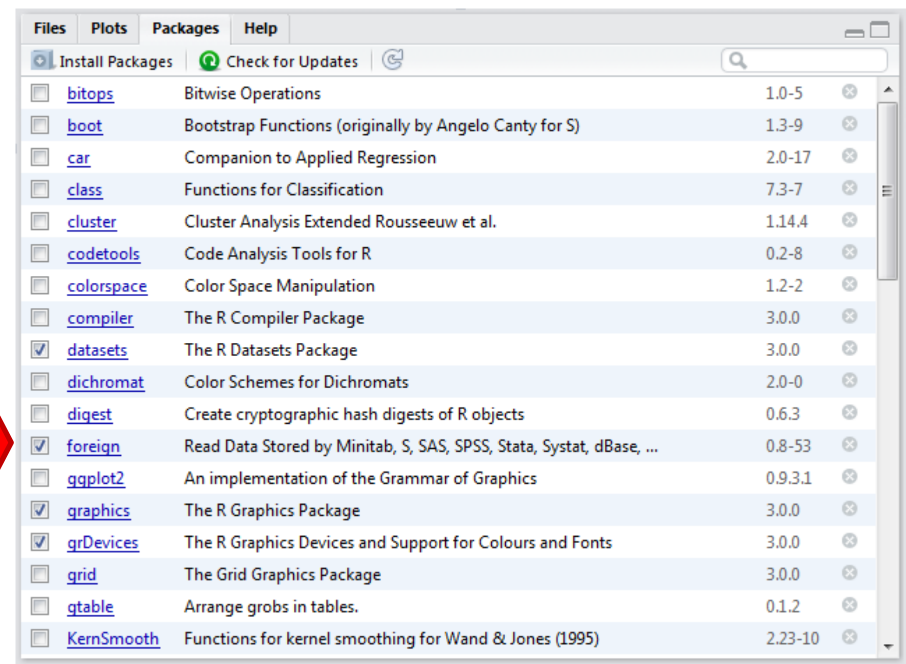


Here you can type R commands and run them. Just leave the cursor anywhere on the line where the command is and press Ctrl-R or click on the 'Run' icon above. Output will appear in the console below.

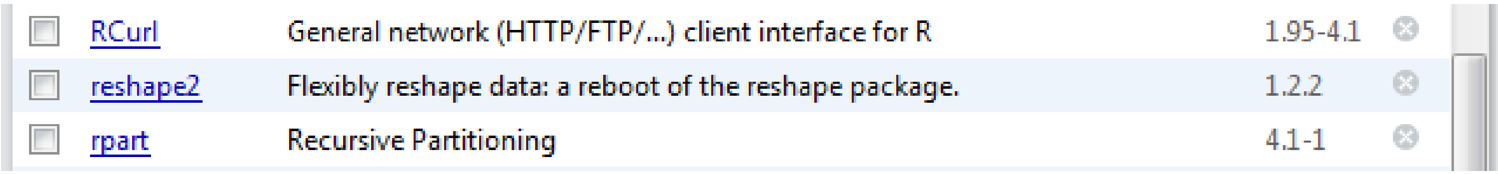
Packages tab

The package tab shows the list of add-ons included in the installation of RStudio. If checked, the package is loaded into R, if not, any command related to that package won't work, you will need select it. You can also install other add-ons by clicking on the 'Install Packages' icon. Another way to activate a package is by typing, for example, `library(foreign)`. This will automatically check the `--foreign` package (it helps bring data from proprietary formats like Stata, SAS or SPSS).

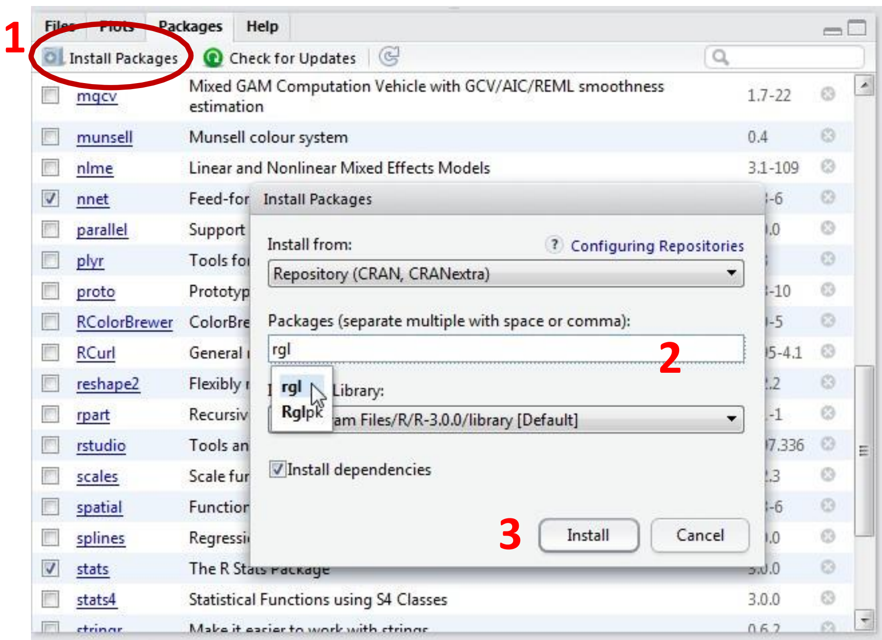




Installing a package



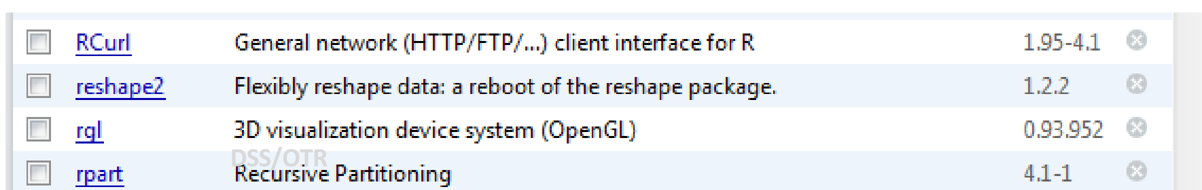
Before



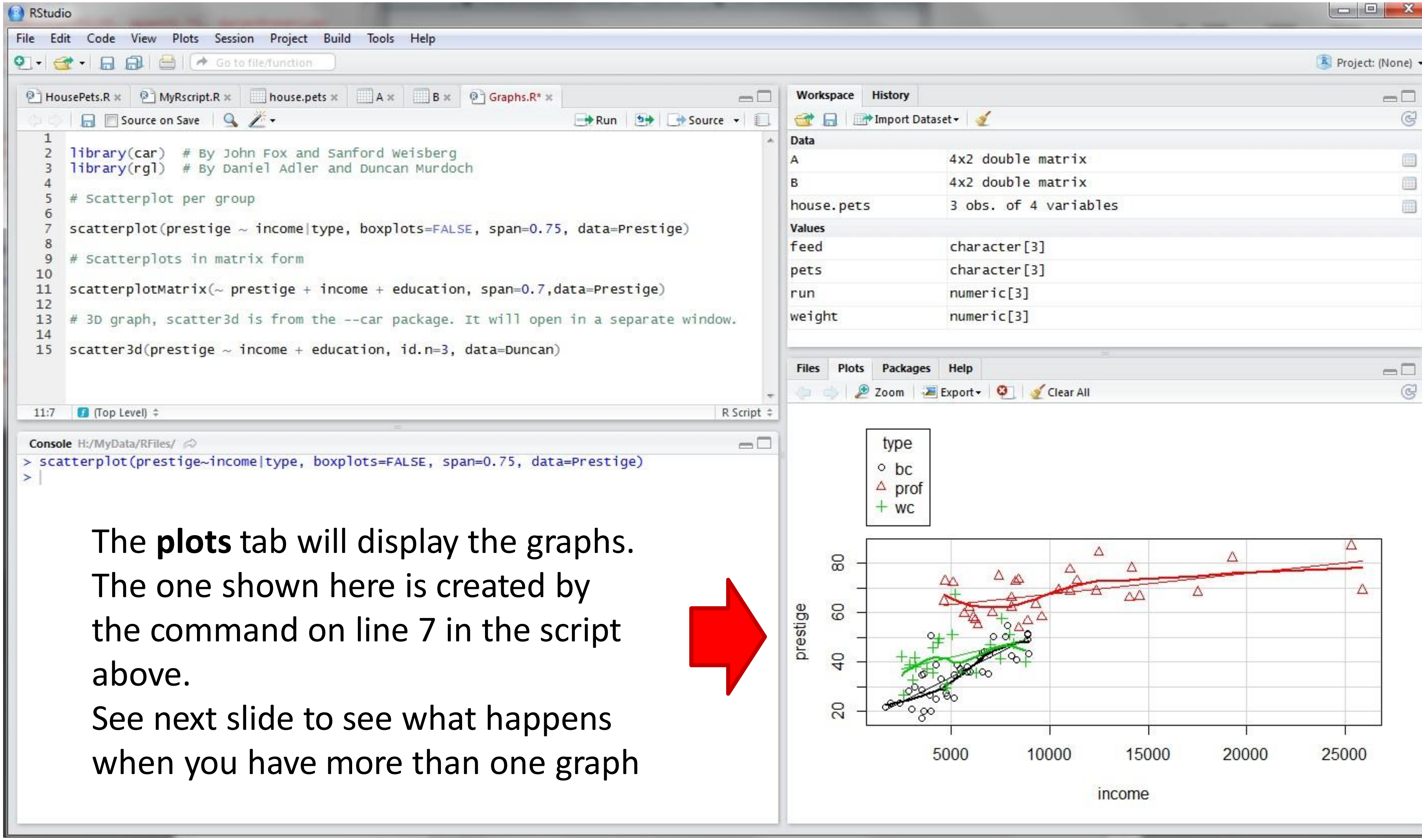
We are going to install the package – `rgl` (useful to plot 3D images). It does not come with the original R install.

Click on "`Install Packages`", write the name in the pop-up window and click on "`Install`".

After



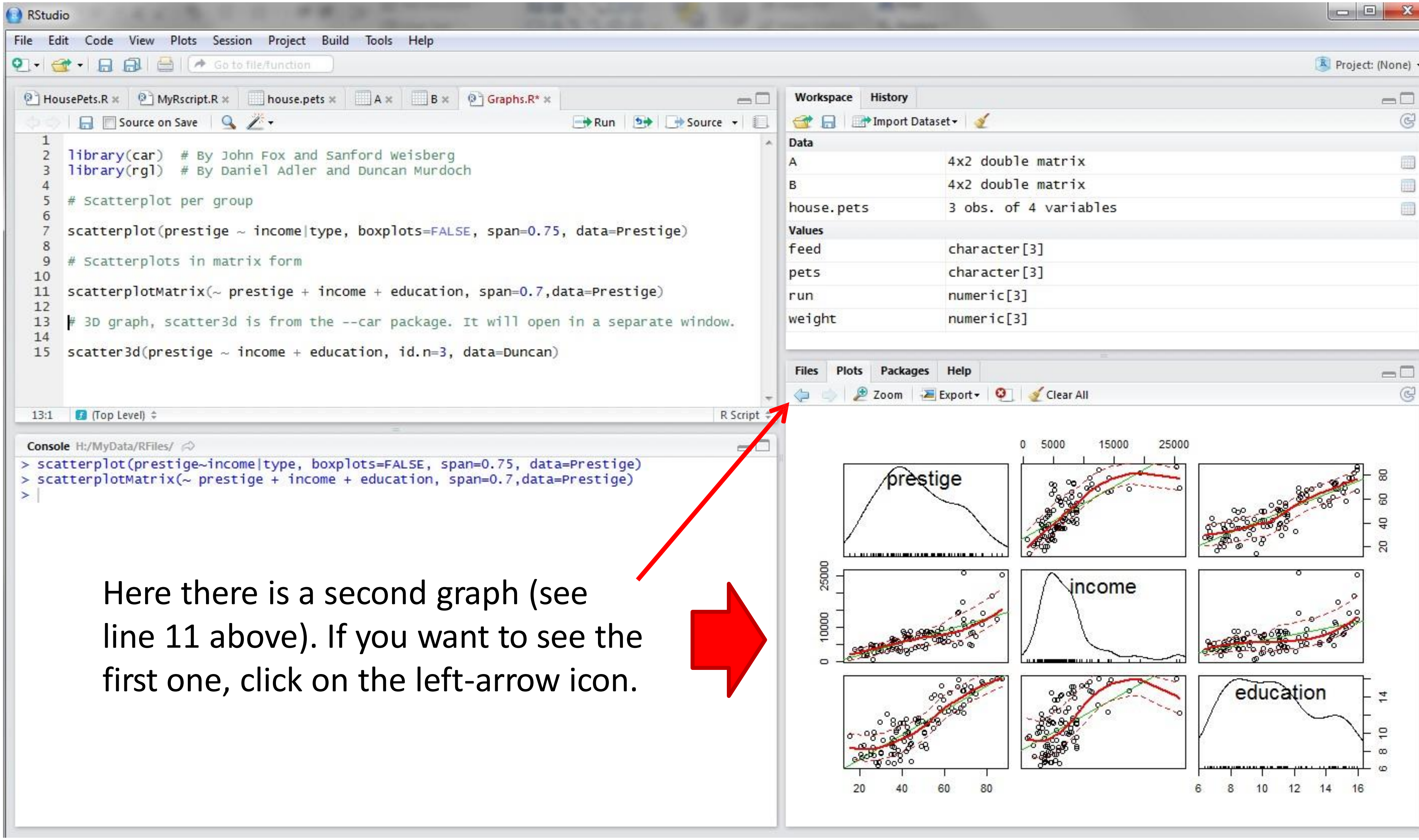
Plots tab (1)



The **plots** tab will display the graphs. The one shown here is created by the command on line 7 in the script above.

See next slide to see what happens when you have more than one graph

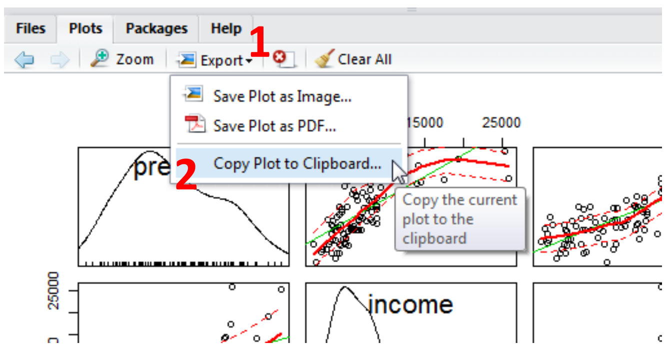
Plots tab (2)

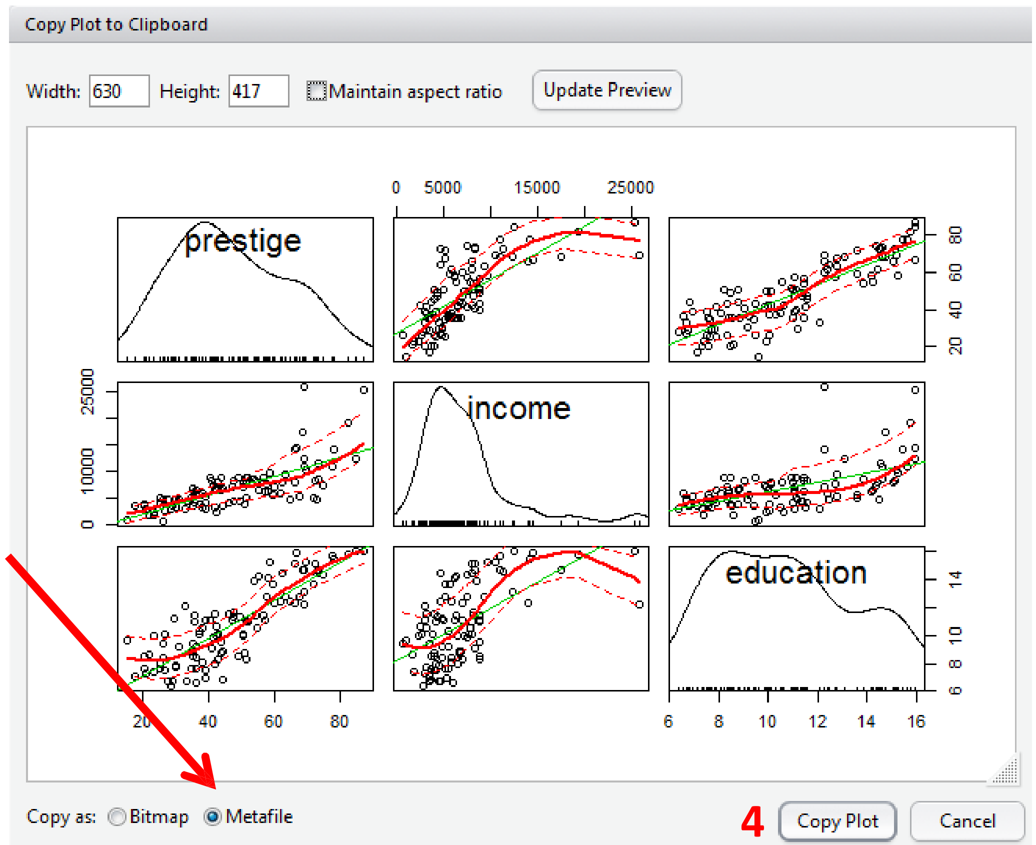


Here there is a second graph (see line 11 above). If you want to see the first one, click on the left-arrow icon.

Plots tab (3) – Graphs export

To extract the graph, click on "`Export`" where you can save the file as an image (PNG, JPG, etc.) or as PDF, these options are useful when you only want to share the graph or use it in a LaTeX document. Probably, the easiest way to export a graph is by copying it to the clipboard and then paste it directly into your Word document.

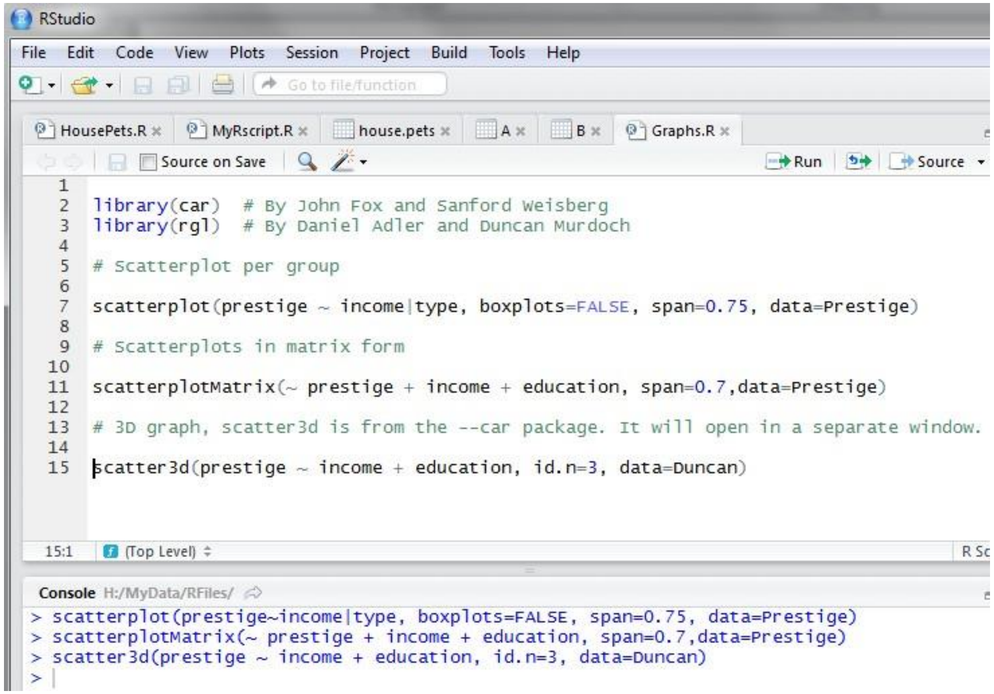




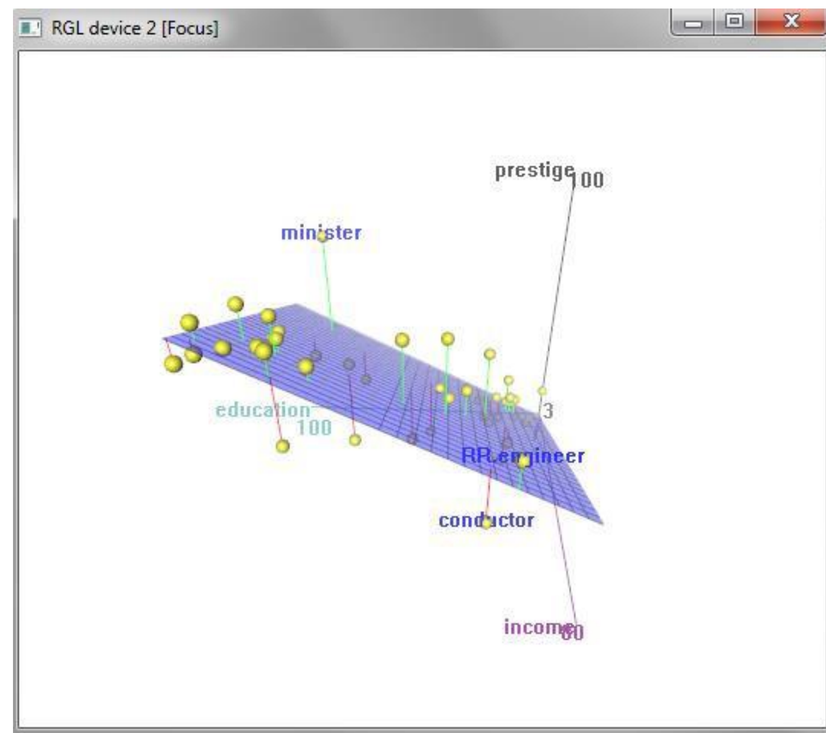
3 Make sure to select '`Metafile`'

5 Paste it into your Word document

3D graphs



3D graphs will display on a separate screen (see line 15 above). You won't be able to save it, but after moving it around, once you find the angle you want, you can screenshot it and paste it to you Word document.



Basic of R Programming

Environment Commands

```
> search() # loaded packages
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"

> ls() # Used objects
 [1] "bb"              "col"             "colorlut"        "EdgeList"
 [5] "Edges"           "EXP"             "f“

> rm(bb) # remove object “bb”

> ?lm  # ? Function name = look function

> args(lm) # look arguments in “lm”

> help(lm) #See detail of function (lm)
```

Operators

- Mathematic operators: + - * / ^
  - Mod: %%
  - sqrt, exp, log, log10, sin, cos, tan, …..
- Other operators:
  - $ component selection HIGH
  - [, [[subscripts, elements
  - : sequence operator
  - %*% matrix algebra
  - <, >, <=, >= inequality
  - ==, != comparison
  - ! not
  - &, |, &&, || and, or
  - ~ formulas
  - <- assignment (or = 1.9.1 later)

Demo Algebra, Operators and Functions

```
> 1+2
[1] 3
> 1 > 2
[1] FALSE
> 1 > 2 | 2 > 1
[1] TRUE
> 1:3
[1] 1 2 3
> A = 1:3
> A
[1] 1 2 3
> A*6
[1]  6 12 18
> A/10
[1] 0.1 0.2 0.3
> A %% 2
[1] 1 0 1
```

```
> B=4:6
> A*B
[1]  4 10 18
> A%*%B
     [,1]
[1,]   32
> A %*% t(B)
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]    8   10   12
[3,]   12   15   18
> A / B
[1] 0.25 0.40 0.50
> sqrt(A)
[1] 1.000000 1.414214 1.732051
> log(A)
[1] 0.0000000 0.6931472 1.0986123
```

```
> round(sqrt(A),2)
[1] 1.00 1.41 1.73
> ceiling(sqrt(A))
[1] 1 2 2
> floor(sqrt(A))
[1] 1 1 1
> eigen( A%*% t(B))
$values
[1] 3.200000e+01 5.835176e-16 2.480655e-16

$vectors
          [,1]       [,2]       [,3]
[1,] 0.2672612  0.3273463 -0.8890009
[2,] 0.5345225 -0.8217055  0.2540003
[3,] 0.8017837  0.4665237  0.3810004
> eigen( A%*% t(B))$values
[1] 3.200000e+01 5.835176e-16 2.480655e-16
```

Variables

```
> a = 49
> sqrt(a)
[1] 7
```

numeric

```
> a = "The dog ate my homework"
> sub("dog","cat",a)
[1] "The cat ate my homework“
```

character string

```
> a = (1+1==3)
> a
[1] FALSE
```

logical

Vectors, Matrices, and Arrays

- vector: an ordered collection of data of the same type
- > a = c(1,2,3)
- > a*2
- [1] 2 4 6

- In R, a single number is the special case of a vector with 1 element.
- Other vector types: character strings, logical

Vectors, Matrices, and Arrays

- matrix: a rectangular table of data of the same type

```
> A = matrix(1:12,3,4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
```

- array: 3-,4-,..dimensional matrix

```
> A[1,]
[1]  1  4  7 10
>
```

Lists

- vector: an ordered collection of data of the same type.

```
> a = c(7,5,1)
> a[2]
[1] 5
```

- list: an ordered collection of data of arbitrary types.

```
> doe = list(name="john",age=28,married=F)
> doe$name
[1] "john"
> doe$age
[1] 28
```

- Typically, vector elements are accessed by their index (an integer), list elements by their name (a character string). But both types support both access methods.

Data Frame

- data frame: is supposed to represent the typical data table that researchers come up with – like a spreadsheet.

- It is a rectangular table with rows and columns; data within each column has the same type (e.g. number, text, logical), but different columns may have different types.

Branching

if (logical expression) {
  statements
} **else** {
  alternative statements
}

else branch is optional

Loops

- When the same or similar tasks need to be performed multiple times; for all elements of a list; for all columns of an array; etc.
  - Monte Carlo Simulation
  - Cross-Validation (delete one and etc)

```
> for(i in 1:10) {
+     print(i*i)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
```

```
> i=1
> while(i<=10) {
+     print(i*i)
+     i=i+sqrt(i)
+ }
[1] 1
[1] 4
[1] 11.65685
[1] 27.68836
[1] 57.0912
```

lapply, sapply, apply

- When the same or similar tasks need to be performed multiple times for all elements of a list or for all columns of an array.
  - May be easier and faster than “for” loops
- lapply(li, *function*)
  - To each element of the list li, the function *function* is applied.
  - The result is a list whose elements are the individual *function* results.

```
> li = list("klaus","martin","georg")
> lapply(li, toupper)
[[1]]
[1] "KLAUS"

[[2]]
[1] "MARTIN"

[[3]]
[1] "GEORG"
```

lapply, sapply, apply

- sapply(li, fct)
- Like apply, but tries to simplify the result, by converting it into a vector or array of appropriate size

```
> li = list("klaus","martin","georg")
> sapply(li, toupper)
[1] "KLAUS"  "MARTIN" "GEORG"
```

```
> fct = function(x) { return(c(x, x*x, x*x*x)) }
> sapply(1:5, fct)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    1    4    9   16   25
[3,]    1    8   27   64  125
```

apply

apply(arr, margin, fct)
Apply the function fct along some dimensions of the array arr, according to margin, and return a vector or array of the appropriate size.

```
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> apply(A, 1, sum)
[1] 22 26 30
> apply(A, 2, sum)
[1]  6 15 24 33
```

Functions and Operators

Functions do things with data
“Input”: function arguments (0,1,2,…)
“Output”: function result (exactly one)

Example:

```
add = function(a,b)
{ result = a+b
  return(result) }
```

Operators:

Short-cut writing for frequently used functions of one or two arguments.
Examples: + - * / ! & | %%

Functions and Operators

- Functions do things with data
  - “Input”: function arguments (0,1,2,…)
  - “Output”: function result (exactly one)

Exceptions to the rule:

- Functions may also use data that sits around in other places, not just in their argument list: “scoping rules”*
- Functions may also do other things than returning a result. E.g., plot something on the screen: “side effects”

Statistical Functions of R

Distribution in R

- Notation:
  - Probability Density Function: d
  - Distribution Function: **p**
  - Quantile function: **q**
  - Random generation for distribution: **r**
- Example:
  - Normal distribution:
    - dnorm(x, mean=0, sd=1, log = FALSE)
    - pnorm(q, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)
    - qnorm(p, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)
    - rnorm(n, mean=0, sd=1)

- Weibull Distribution
  - dweibull(x, shape, scale = 1, log = FALSE)
  - pweibull(q, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
  - qweibull(p, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
  - rweibull(n, shape, scale = 1)
- Log Normal Distribution
  - dlnorm(x, meanlog = 0, sdlog = 1, log = FALSE)
  - plnorm(q, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
  - qlnorm(p, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
  - rlnorm(n, meanlog = 0, sdlog = 1)

Statistical Functions

Excel	**R**
NORMSDIST	pnorm(7.2,mean=5,sd=2)
NORMSINV	qnorm(0.9,mean=5,sd=2)
LOGNORMDIST	plnorm(7.2,meanlog=5,sdlog=2)
LOGINV	qlnorm(0.9,meanlog=5,sdlog=2)
GAMMADIST	pgamma(31, shape=3, scale =5)
GAMMAINV	qgamma(0.95, shape=3, scale =5)
GAMMALN	lgamma(4)
WEIBULL	pweibull(6, shape=3, scale =5)
BINOMDIST	pbinom(2,size=20,p=0.3)
POISSON	ppois(2, lambda =3)

Statistical models in R

- Regression analysis
  - a linear regression model with independent homoscedastic errors

$$y_i = \sum_{j=0}^{p} \beta_j x_{ij} + e_i, \qquad e_i \sim \mathrm{NID}(0, \sigma^2), \qquad i = 1, \ldots, n$$

- The analysis of variance (ANOVA)
  - Predictors are now all categorical/ qualitative.
  - The name Analysis of Variance is used because the original thinking was to try to partition the overall variance in the response to that due to each of the factors and the error.
  - Predictors are now typically called factors which have some number of levels.
  - The parameters are now often called *effects.*
  - The parameters are considered fixed but unknown —called *fixed-effects* models but *random-effects* models are also used where parameters are taken to be random variables.

One-Way ANOVA

- The model
  - Given a factor α occurring at i =1,…,I levels, with j = 1 ,…,J_i observations per level. We use the model
  - $y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$, i =1,…,I , j = 1 ,…,J_i
- Not all the parameters are identifiable and some restriction is necessary:
  - Set μ=0 and use I different dummy variables.
  - Set $\alpha_1 = 0$ — this corresponds to treatment contrasts
  - Set $\Sigma J_i \alpha_i = 0$ — ensure orthogonality
- Generalized linear models
- Nonlinear regression

Two-Way Anova

- The model $y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$.
  - We have two factors, α at I levels and β at J levels.
  - Let n_{ij} be the number of observations at level i of α and level j of β and let those observations be $y_{ij1}, y_{ij2}, \ldots$. A complete layout has $n_{ij} \geq 1$ for all i, j.
- The interaction effect $(\alpha\beta)_{ij}$ is interpreted as that part of the mean response not attributable to the additive effect of α_i and β_j.
  - For example, you may enjoy strawberries and cream individually, but the combination is superior.
  - In contrast, you may like fish and ice cream but not together.
- As of an investigation of toxic agents, 48 rats were allocated to 3 poisons (I,II,III) and 4 treatments (A,B,C,D).
  - The response was survival time in tens of hours. The Data:

Statistical Strategy and Model Uncertainty

- Strategy
  - *Diagnostics:* Checking of assumptions: constant variance, linearity, normality, outliers, influential points, serial correlation and collinearity.
  - *Transformation:* Transforming the response — Box-Cox, transforming the predictors — tests and polynomial regression.
  - *Variable selection:* Stepwise and criterion based methods
- Avoid doing too much analysis.
  - Remember that fitting the data well is no guarantee of good predictive performance or that the model is a good representation of the underlying population.
  - Avoid complex models for small datasets.
  - Try to obtain new data to validate your proposed model. Some people set aside some of their existing data for this purpose.
  - Use past experience with similar data to guide the choice of model.

Simulation and Regression

- What is the sampling distribution of least squares estimates when the noises are not normally distributed?
- Assume the noises are independent and identically distributed.
  1. Generate ε from the known error distribution.
  2. Form $y = X\beta + \varepsilon$.
  3. Compute the estimate of β.
- Repeat these three steps many times.
  - We can estimate the sampling distribution of using the empirical distribution of the generated , which we can estimate as accurately as we please by simply running the simulation for long enough.
  - This technique is useful for a theoretical investigation of the properties of a proposed new estimator. We can see how its performance compares to other estimators.
  - It is of no value for the actual data since we don't know the true error distribution and we don't know β.

File Input/Output of R

Write Data to a TXT File

- Usage:
- ***write***(x, file, …)

- x<-matrix(c(1.0, 2.0, 3.0, 4.0, 5.0, 6.0), 2, 3) ; x
- [,1] [,2] [,3]
- [1,] 1 3 5
- [2,] 2 4 6
- write(t(x), file="d:/out2.txt", ncolumns=3)
- write(x,   file="d:/out3.txt", ncolumns=3)

<u>d:/out2.txt</u>
1 3 5
2 4 6

<u>d:/out3.txt</u>
1 2 3
4 5 6

Write Data to a CSV File

- Usage:
- ***write.table***(x, file = "foo.csv", sep=", ",…)
- Example:
- x<-matrix(c(1.0, 2.0, 3.0, 4.0, 5.0, 6.0), 2, 3) ; x
- [,1] [,2] [,3]
- [1,] 1 3 5
- [2,] 2 4 6
- write.table(**t(x)**, file="d:/out4.txt", sep=",", col.names=FALSE, row.names=FALSE)
- write.table(x, file="d:/out5.txt",sep=",", col.names=FALSE, row.names=FALSE)

<u>d:/out4.txt</u>
1,2
3,4
5,6

<u>d:/out5.txt</u>
1,3,5
2,4,6

Read TXT and CSV File

- Usage:
- ***read.table*** (file,…)

- X=read.table(file="d:/out2.txt"); X
-   V1 V2 V3
- 1 1 3 5
- 2 2 4 6
- X=read.table(file="d:/out5.txt",sep=", ", header=FALSE); X

<u>d:/out2.txt</u>
1 3 5
2 4 6

<u>d:/out5.csv</u>
1,3,5
2,4,6

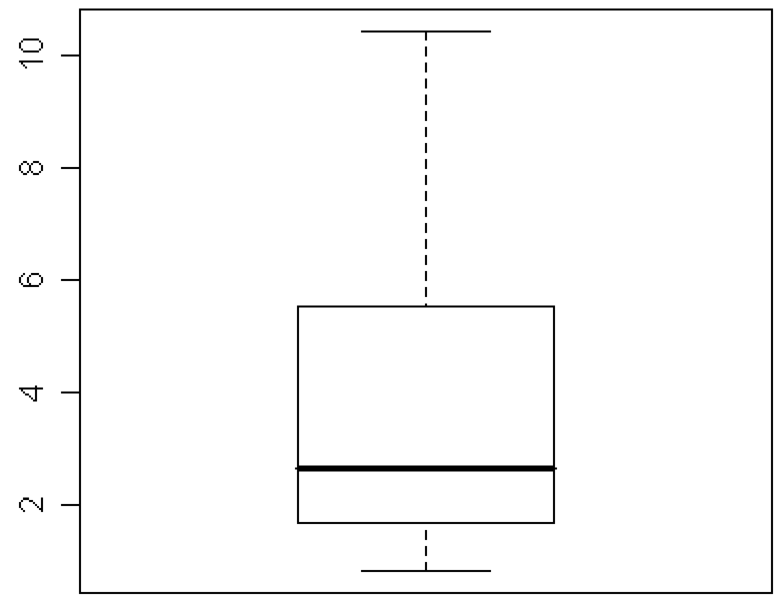
Demo Reading CSV File

```
> Data = read.table(file="d:/01.csv",header=TRUE, sep=",")
> Data
          Y        X1       X2
1  2.651680 13.808986 26.75896
2  1.875039 17.734523 37.89857
3  1.523964 19.891025 26.03624
4  2.984314 15.574261 30.21754
5 10.423087  9.293612 28.91459
6  0.840065  8.830160 30.38578
7  8.126936  9.615875 32.69579
> mean(Data$Y) # $ is used to identify which variable
[1] 4.060726
> mean(Data$X1)
[1] 13.53549
> mean(Data$X2)
[1] 30.41535
> sd(Data$Y)
[1] 3.691044
> summary(Data$Y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8401  1.7000  2.6520  4.0610  5.5560 10.4200
> boxplot(Data$Y)
```

01.csv

Book1

	A	B	C
1	Y	X1	X2
2	2.65168	13.80899	26.75896
3	1.875039	17.73452	37.89857
4	1.523964	19.89103	26.03624
5	2.984314	15.57426	30.21754
6	10.42309	9.293612	28.91459
7	0.840065	8.83016	30.38578
8	8.126936	9.615875	32.69579

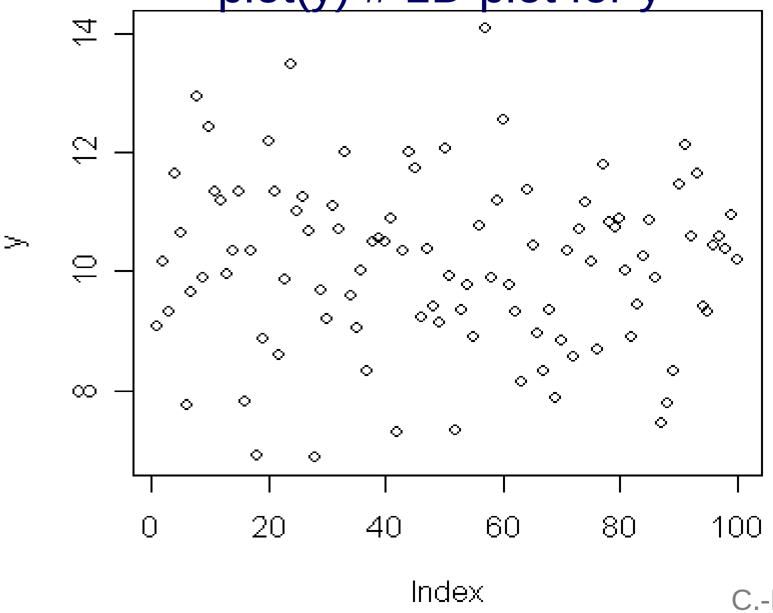


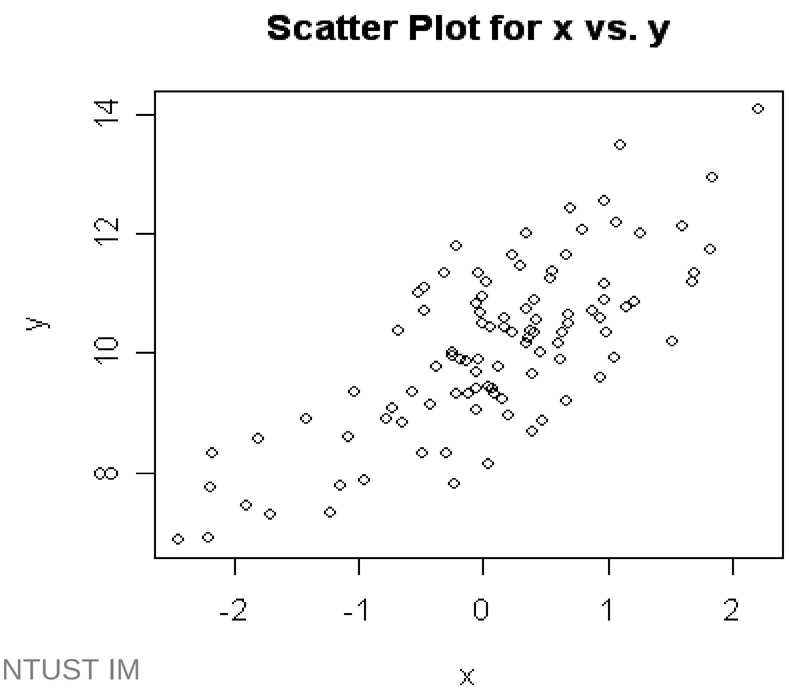
Plotting of R

Scatter Plot in R

- Scatter plot:
  - x = rnorm(100) # Generated 100 N(0,1)
  - y = 10 + 1.2*x + rnorm(100) # Simulated y
  - plot(y) # 1D plot for y
  - windows() # Create a new graph device
  - plot(x,y) # 2D plot for x vs y
  - title (main="Scatter Plot for x vs. y") # Add title in plot

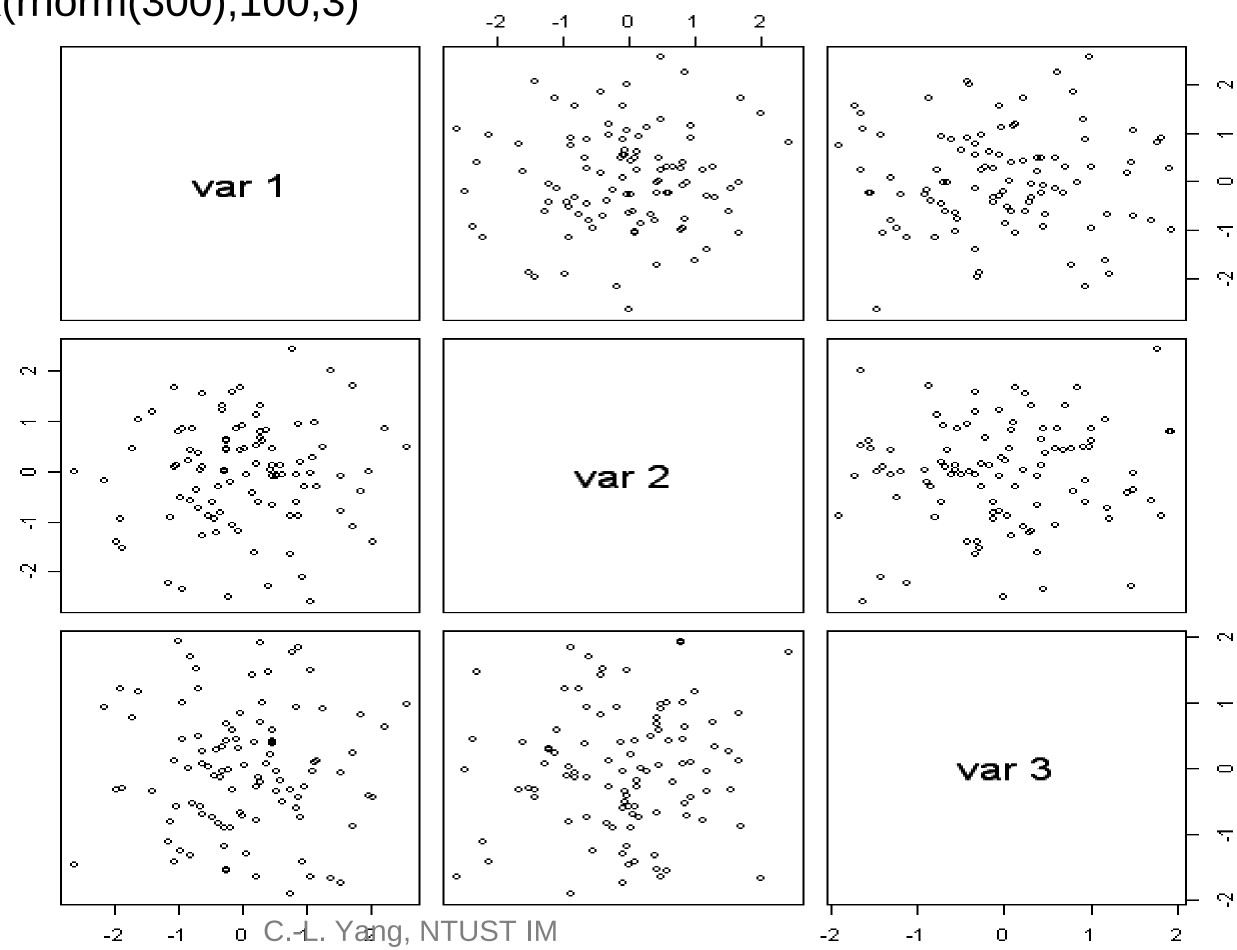
plot(y) # 1D plot for y





Scatterplot Matrices in R

- A matrix of scatterplots is produced.
  - **Usage:**
    - pairs(x, ...)
  - Example:
    - X = matrix(rnorm(300),100,3)
    - pairs(X)



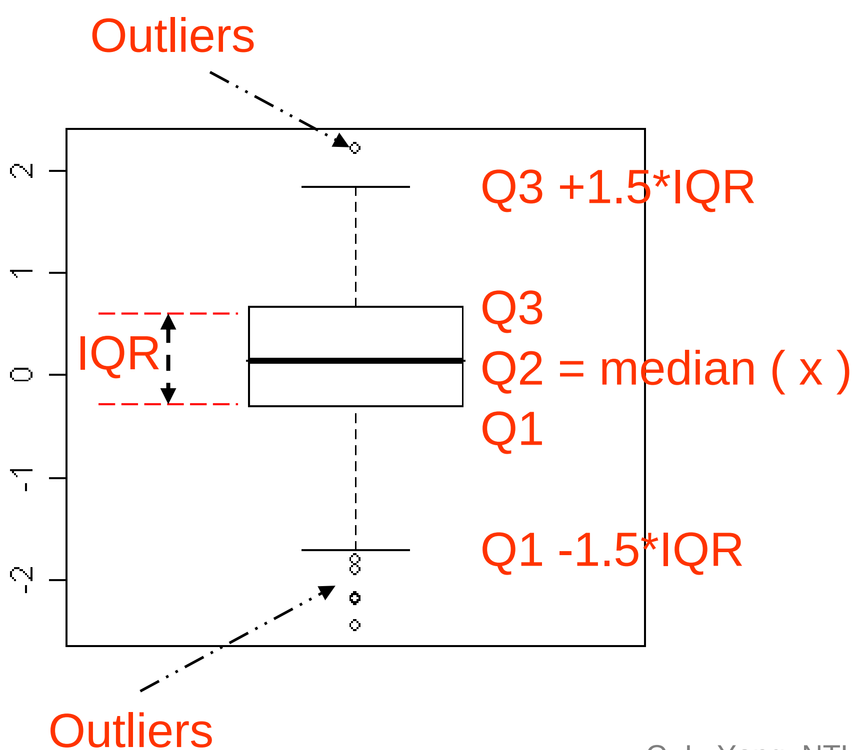
Box Plot in R

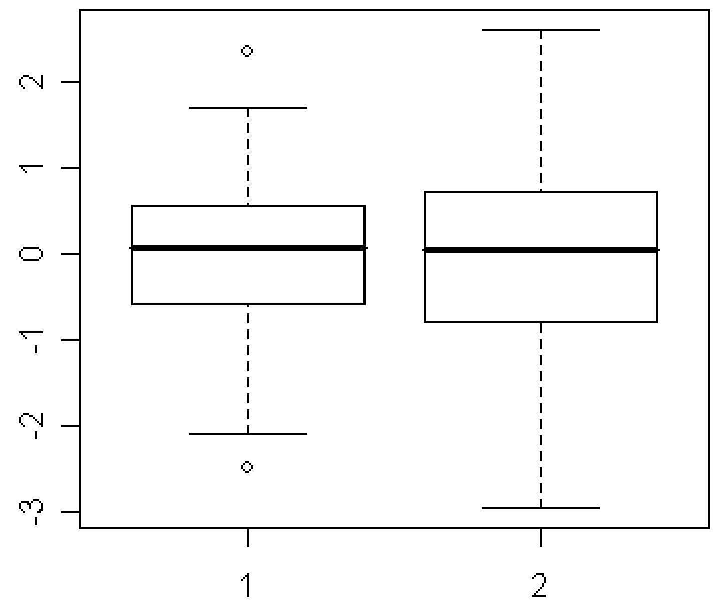
- Box plot:
  - Produce box-and-whisker plot of the given (grouped) values.
  - **Usage:** boxplot(x, ...)
  - Example1:
    - X=rnorm(100)
    - boxplot(x)

Example2:

x=rnorm(100); y=rnorm(100);

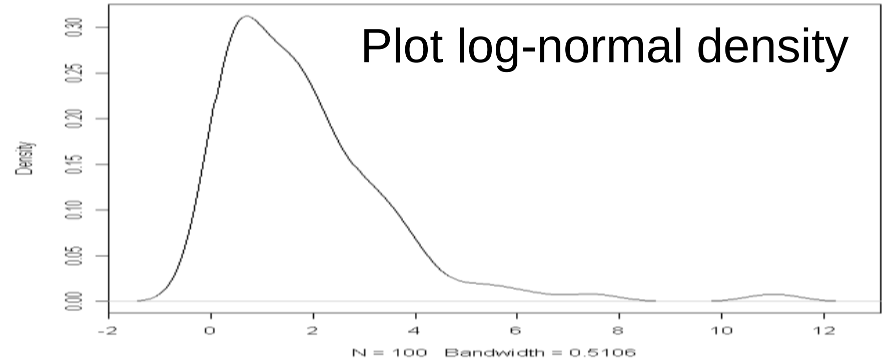
boxplot(x,y)

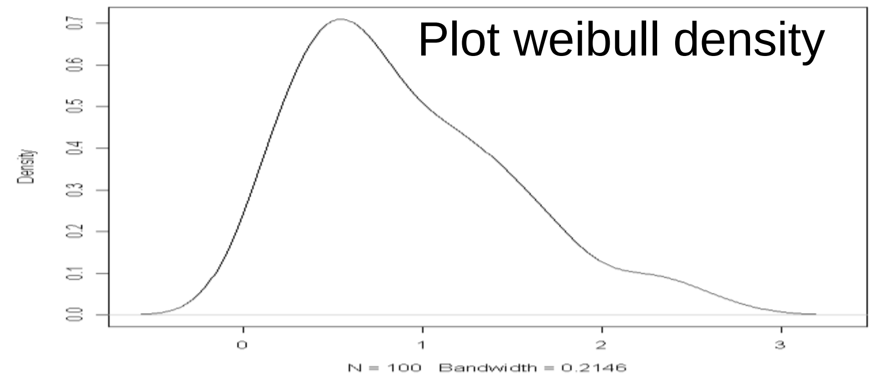


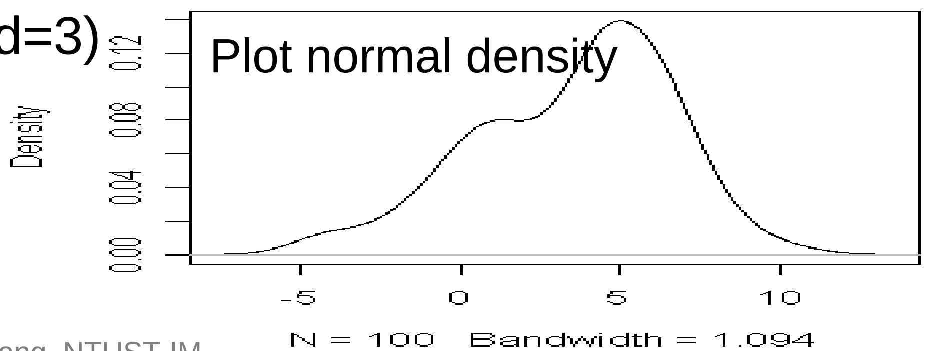


Kernel Density Plot

- Kernel Density:
  - density(x,…)
  - Kernel Density Plot:
    - plot(density(x,…))
  - Examples:
    - X= rlnorm(100,0,1)
    - plot(density(X))
    - Y= rweibull(100,1.5,1)
    - plot(density(Y))
    - x= rnorm(100, mean=3, sd=3)
    - plot(density(x))

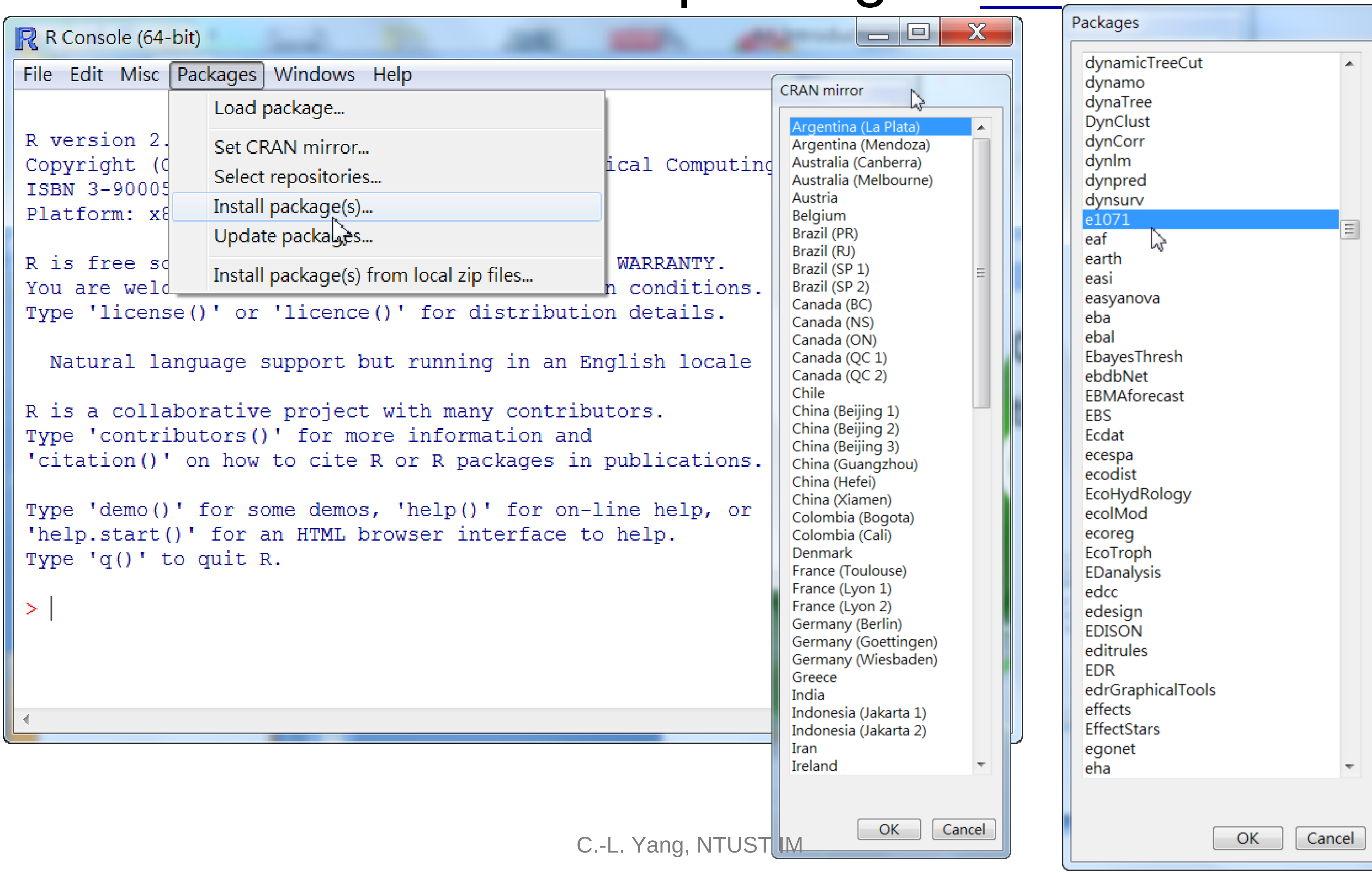






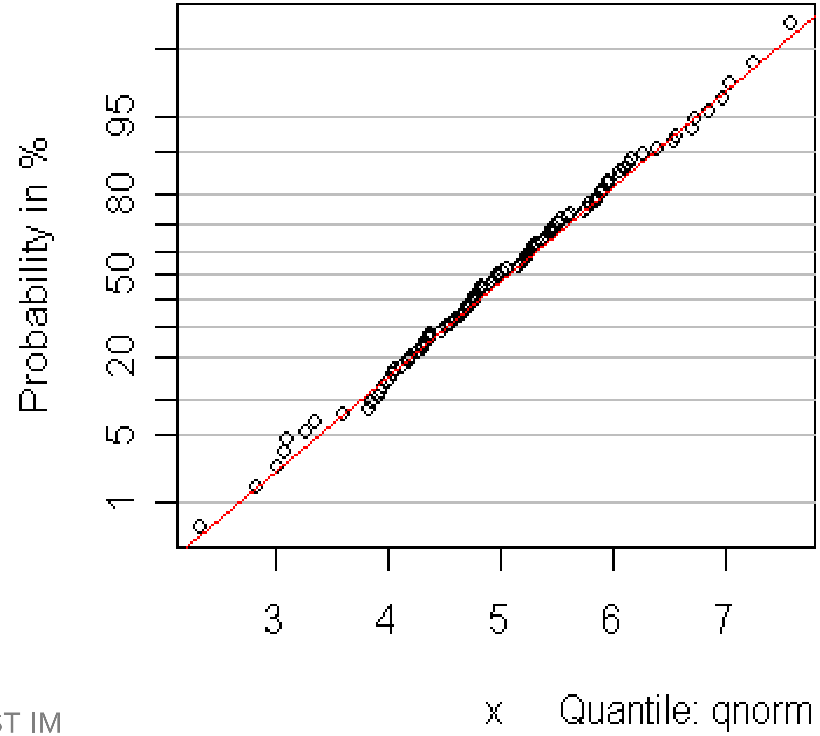
Probability Plots

- Need installation R package : **e1071**



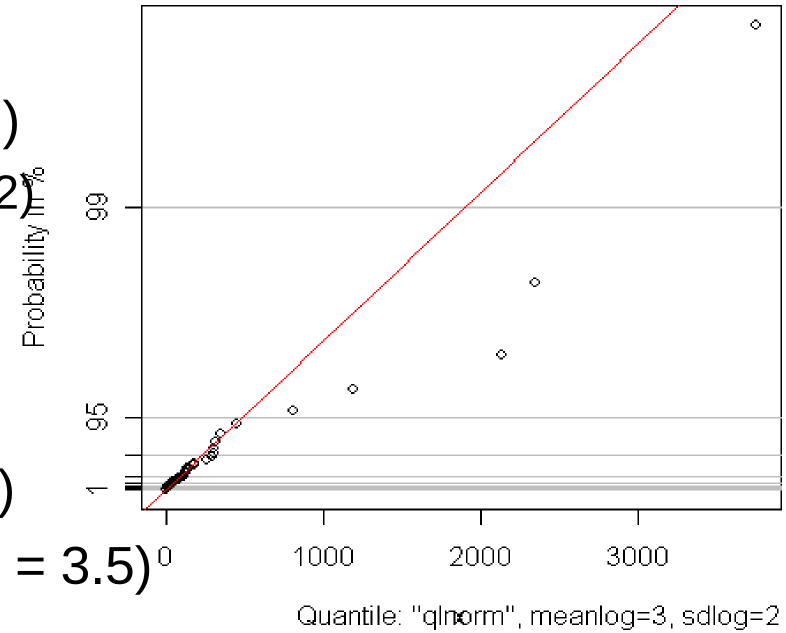
Using Probability Plot

- **Description**
  - Generates a probability plot for a specified theoretical distribution, i.e., basically a <u>qqplot</u> where the y-axis is labeled with probabilities instead of quantiles. The function is mainly intended for teaching the concept of quantile plots.
  - **Usage**
    - probplot(x,…)
    - Example1:
      - x <- rnorm(100, mean=5)
      - probplot(x)

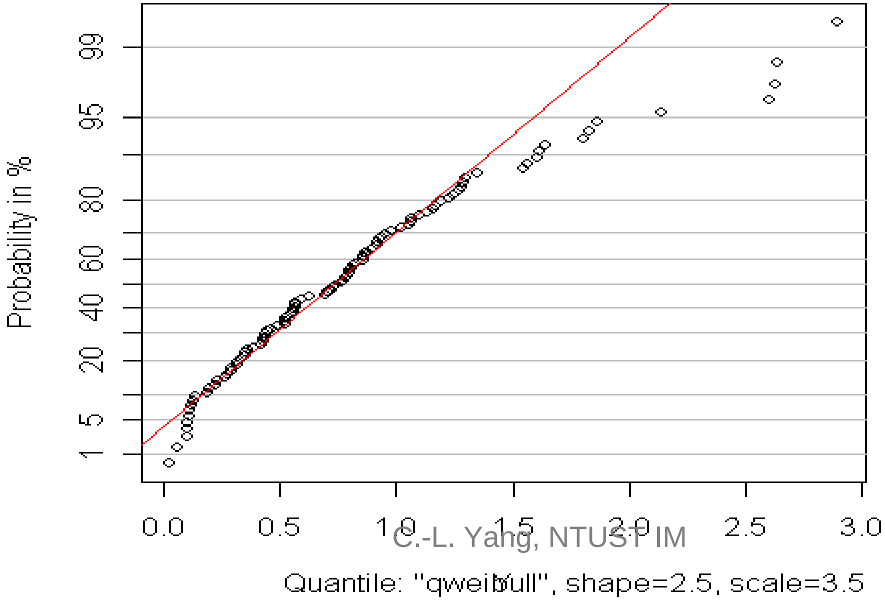


More Examples

- Example2:
  - x <- rlnorm(100,meanlog = 3, sdlog = 2)
  - probplot(x, "qlnorm", meanlog = 3, sdlog = 2)



- Example 3:
  - x=rweibull(100, shape=2.5, scale = 3.5)
  - probplot(x,"qweibull", shape=2.5, scale = 3.5)



Pareto Chart

- Load package: **library(qcc)**
  - Using command:

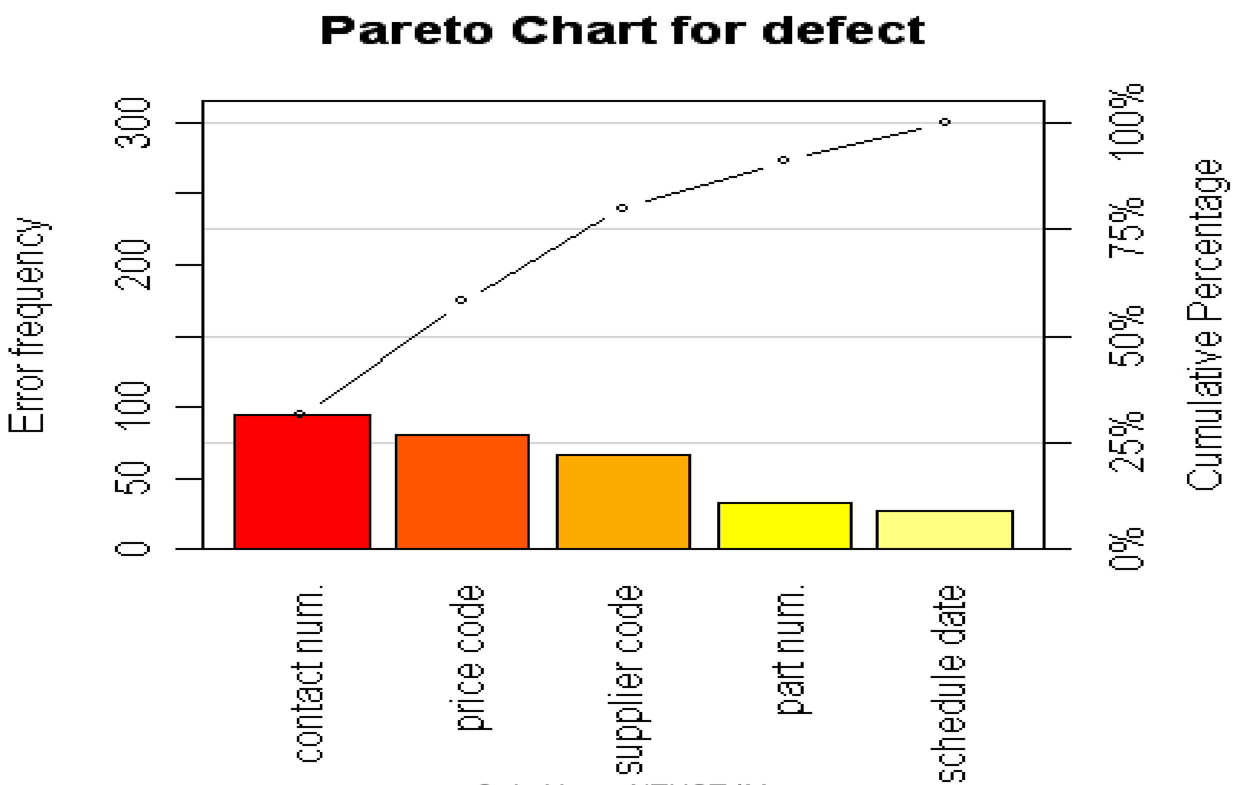
    pareto.chart(x, ylab = "Frequency", xlab, ylim, main, col = heat.colors(length(x)), ...)
- Example:

```
defect <- c(80, 27, 66, 94, 33) # Frequency
names(defect) <- c("price code", "schedule date",
   "supplier code", "contact num.", "part num.") # names
pareto.chart(defect, ylab = "Error frequency")#plot
```

Pareto Chart Analysis

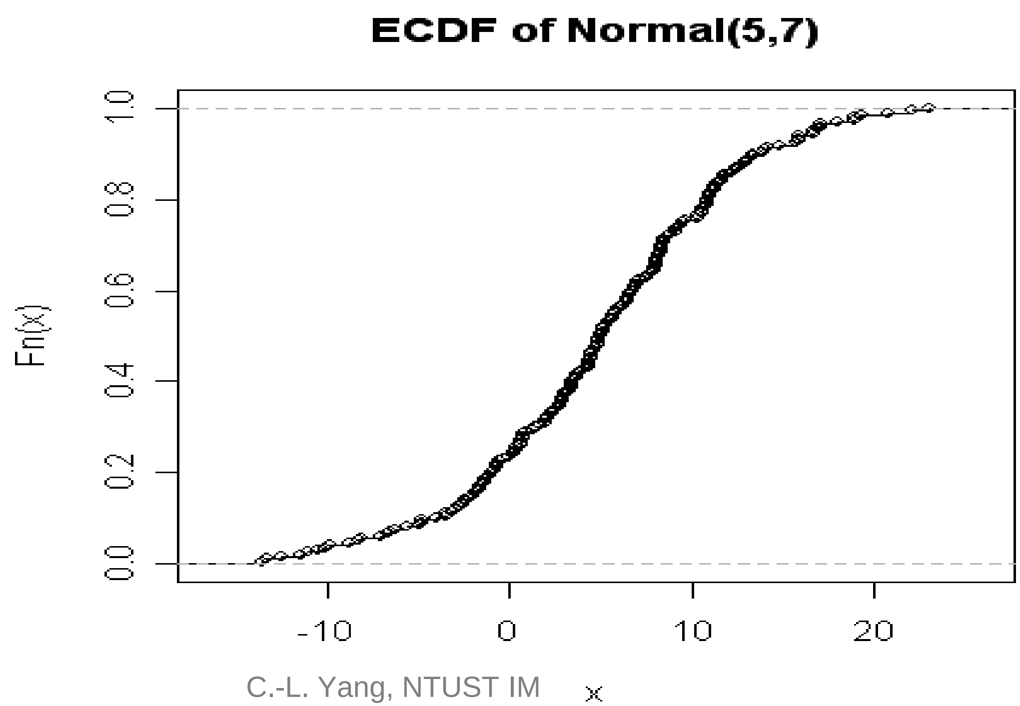
Pareto chart analysis for defect

	Frequency	Cum.Freq.	Percentage	Cum.Percent.
contact num.	94	94	31.33333	31.33333
price code	80	174	26.66667	58.00000
supplier code	66	240	22.00000	80.00000
part num.	33	273	11.00000	91.00000
schedule date	27	300	9.00000	100.00000



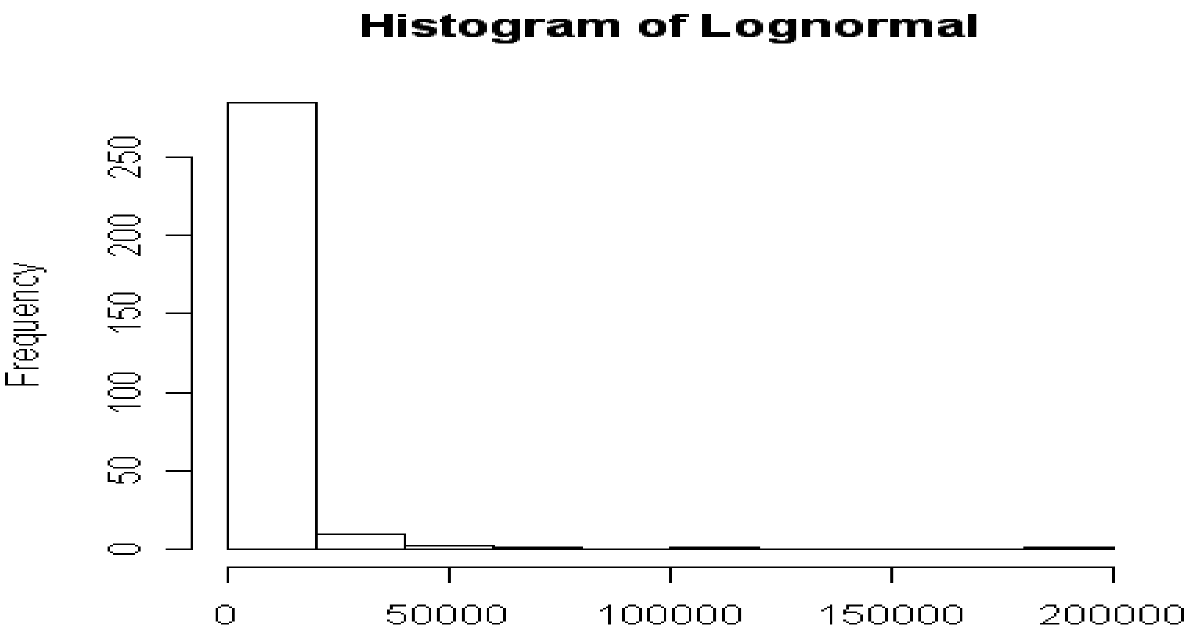
Empirical CDF

- Empirical Cumulative Distribution Function
- Usage: ecdf(x); plot(ecdf(x),…)
- Example:
  - X= rnorm(200,mean=5,sd=7)
  - plot(ecdf(X),main= “ECDF of Normal(5,7)")



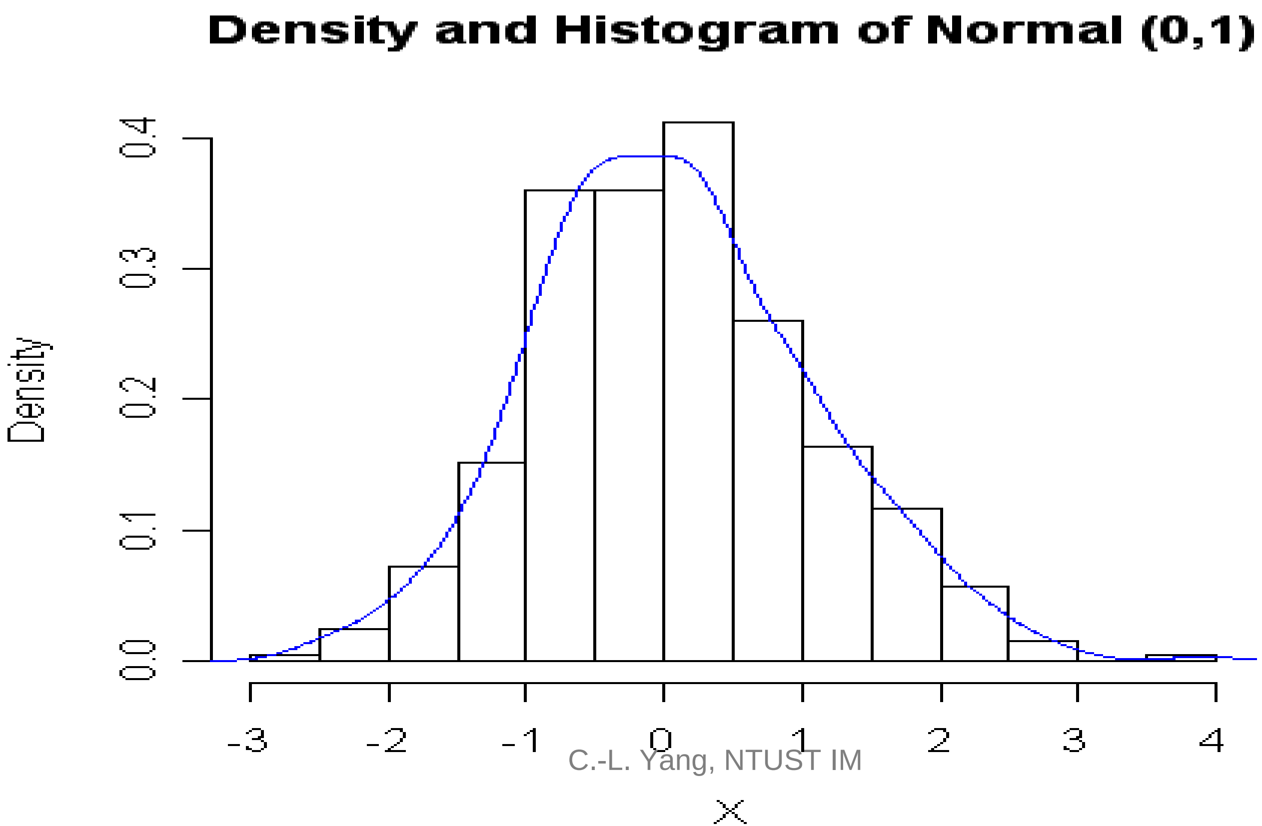
Histogram (1)

- The generic function ***hist*** computes a histogram of the given data values.
- Usage: **hist(x, probability = !freq, ...)**
- Example:
  - X=rlnorm(300,lmean=5, logsd=3)
  - hist(X) # Using default setting



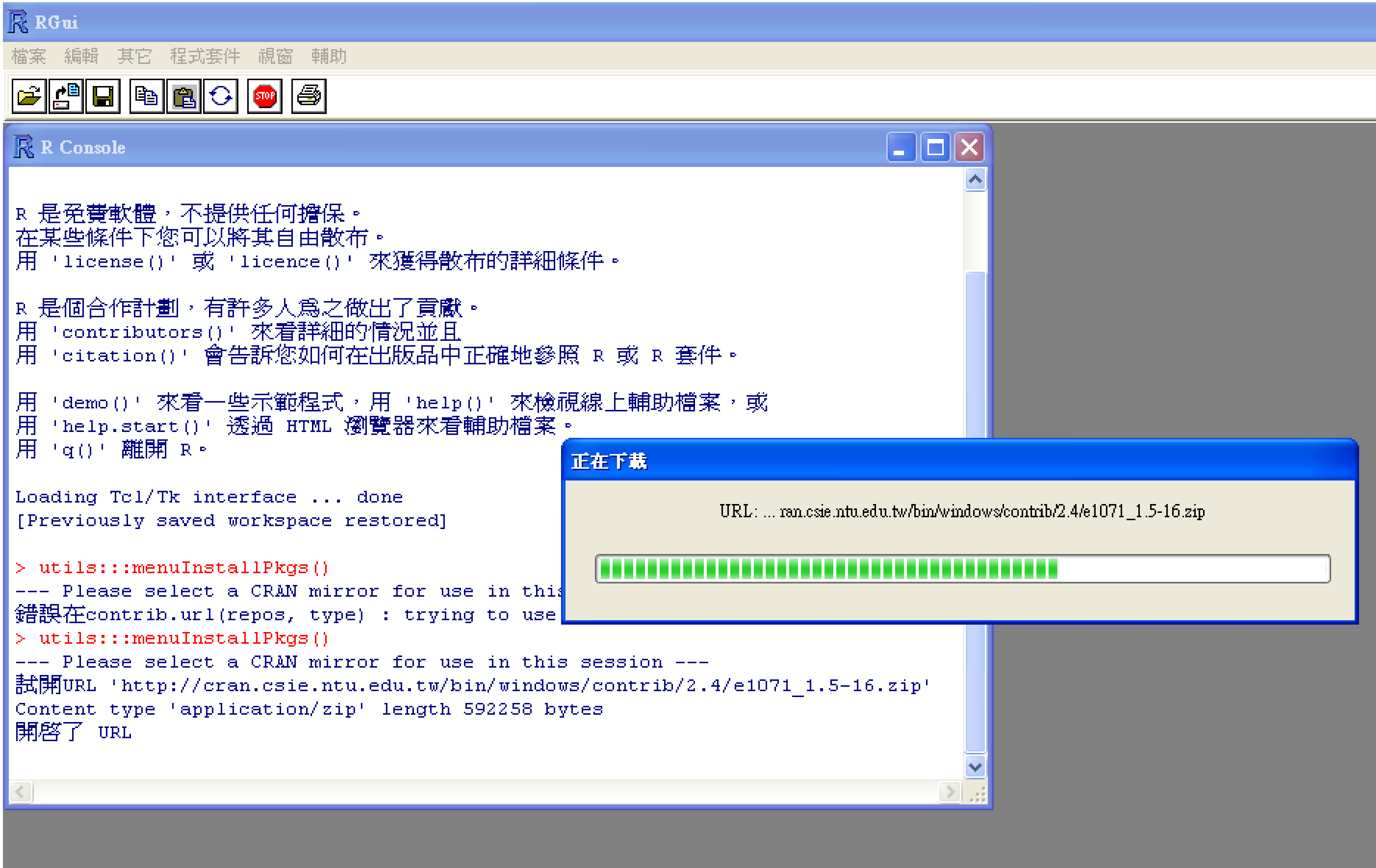
Histogram (2)

- Example: Density and Histogram
  - X=rnorm(500)
  - hist(X, probability=TRUE,main="Density and Histogram of Normal (0,1)")
  - lines(density(X),col="blue")



Package of R

Installing


RGui
檔案 編輯 其它 程式套件 視窗 輔助
R Console
R 是免費軟體，不提供任何擔保。
在某些條件下您可以將其自由散布。
用 'license()' 或 'licence()' 來獲得散布的詳細條件。
R 是個合作計劃，有許多人為之做出了貢獻。
用 'contributors()' 來看詳細的情況並且
用 'citation()' 會告訴您如何在出版品中正確地參照 R 或 R 套件。
用 'demo()' 來看一些示範程式，用 'help()' 來檢視線上輔助檔案，或
用 'help.start()' 透過 HTML 瀏覽器來看輔助檔案。
用 'q()' 離開 R。
Loading Tcl/Tk interface ... done
[Previously saved workspace restored]
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this
錯誤在contrib.url(repos, type) : trying to use
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this session ---
試開URL 'http://cran.csie.ntu.edu.tw/bin/windows/contrib/2.4/e1071_1.5-16.zip'
Content type 'application/zip' length 592258 bytes
開啟了 URL
正在下載
URL: ... ran.csie.ntu.edu.tw/bin/windows/contrib/2.4/e1071_1.5-16.zip

Load Package **e1071**