

Introduction to R programming I

Today's outline

- Introduction.
 - Motivations of using R.
 - Installation of R, R GUI.
 - R documentation.
 - R packages.
- R programming environment.
 - R console and workspace.
 - Getting help.

- R programming basics:
 - Data types and basic operations of different data types.
 - Control statements and loops.
 - File I/O.

A little history

Data Analysts Captivated by R's Power



Left, Stuart Isett for The New York Times; right, Kieran Scott for The New York Times

R first appeared in 1996, when the statistics professors Robert Gentleman, left, and Ross Ihaka released the code as a free software package.

Motivations of using R

- R is an interactive framework and programming language designed for data and statistical analysis.
- Easy to extend by writing new functions.
- Wide range of statistical and graphical capabilities.
- Easy to learn. Doesn't require deep programming skills.
- Can be well integrated with other programming languages: C/C++, Fortran, Java, Perl, Python, etc.
- Free.

Motivations (cont.)

The New York Times

Business Computing

- "Companies as diverse as Google, Pfizer, Merck, Bank of America, and the InterContinental Hotels Group and Shell use R."
- R is really important to the point that it's hard to overvalue it," said Daryl Pregibon, a research scientist at Google, which uses the software widely. "It allows statisticians to do very intricate and complicated analyses without knowing the blood and guts of computing systems."
- R has really become the second language for people coming out of grad school now, and there's an amazing amount of code being written for it," said Max Kuhn, associate director of nonclinical statistics at Pfizer.
- Close to 1,600 different packages reside on just one of the many Web sites devoted to R, and the number of packages has grown exponentially.

R installation

For Windows:

- 1) Go to <http://www.r-project.org>
- 2) Click on "CRAN" in the left navigation bar. See the list of mirror sites setup by country
- 3) Pick a site in the US such as <http://cran.cnr.berkeley.edu/>
- 4) Click on "Download R for Windows"
- 5) Then click on the "base" link
- 6) Now click on the link at the top of page (e.g. Download R 2.13.1 for Windows)
- 7) After the download completes then double-click the .exe file to initiate the installation. Answer the questions as they are presented.
- 8) To start R, click on Start -> All Programs -> R
- 9) **OPTIONAL:** Consider installing R Studio. While its not a requirement it is a great GUI for R. See <http://rstudio.org/download/desktop>

For Apple OSX:

- 1) Go to <http://www.r-project.org>
- 2) Click on "CRAN" in the left navigation bar. See the list of mirror sites setup by country
- 3) Pick a site in the US such as <http://cran.cnr.berkeley.edu/>
- 4) Click on "Download R for Mac OSX". This will start the download
- 5) After the download completes then double-click the .pkg file to initiate the installation. Answer the questions as they are presented.
- 6) To start R, Click on the icon in the Applications dock. You can also launch a Terminal window and type "R" within it.
- 7) **OPTIONAL:** Consider installing R Studio. While its not a requirement it is a great GUI for R. See <http://rstudio.org/download/desktop>

R GUI

- R is provided with a command line interface (**CLI**),
 - The preferred user interface for power users.
 - However, good knowledge of the language is required so it can be intimidating for beginners.
- R has a default GUI that provides a reasonable amount of menu-drive capability.
- A number of more powerful R GUIs out there to make your life easier: R Commander, R-Studio, JGR, SciViews-R, Tinn-R , Revolution R, etc.

Rstudio

I recommend Rstudio (<http://rstudio.org>).

Powerful productivity tools

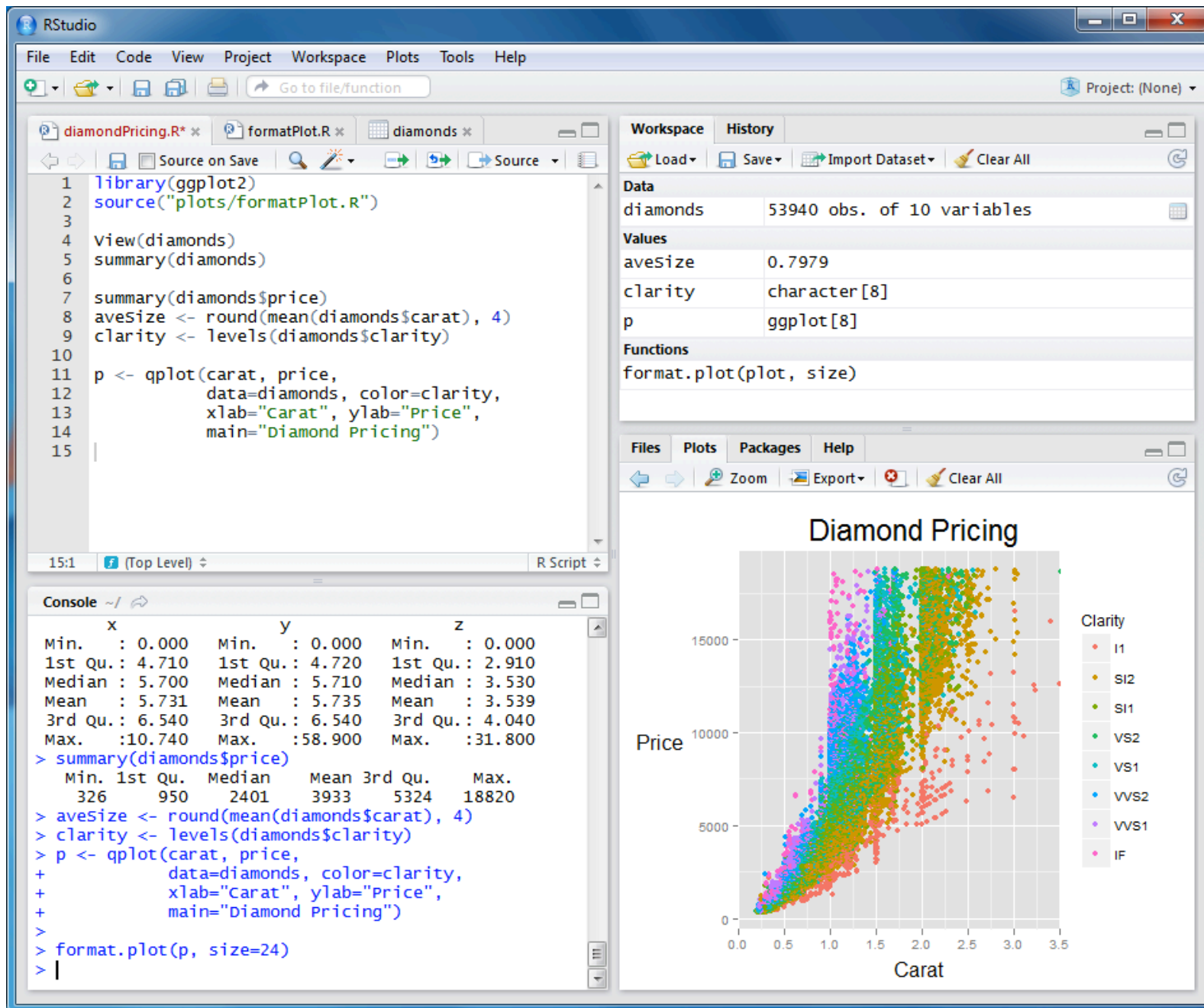
- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Easily manage multiple working directories using projects
- Quickly navigate code using typeahead search and go to definition

An IDE built for R

- Workspace browser and data viewer
- Plot history, zooming, and flexible image and PDF export
- Integrated R help and documentation
- Sweave authoring including one-click PDF preview
- Searchable command history

Open and compatible

- Works with any version of R (2.11.1 or greater)
- Runs on Windows, Mac, Linux, and even over the web using RStudio Server
- Integrated with Git and Subversion for version control
- Free and open source (AGPLv3 license)














R documentation

- There are numerous materials and books for R. I'd start with the free resources. Next slide provide a list of them.
- The official R website (www.r-project.org) provide a list of R manuals. But these are a little difficult to read for beginners.
- I always recommend: “***R for beginners***” by Paradis.
- There are also mailing lists for R related questions.

Free R documentations

Free resources (PDFs, websites, tutorials):

[ed

- The R Inferno - Patrick Burns. Download the PDF http://www.burns-stat.com/pages/Tutor/R_inferno.pdf 
- The R Programmer Wiki Book. See Wiki at http://en.wikibooks.org/wiki/R_Programming/ 
- Introduction to Probability and Statistics Using R - Jay Kerns. Download from <http://ipsur.org/index.html> 
- Statistics with R - Vincent Zoonekynd. Website is at http://zoonek2.free.fr/UNIX/48_R/all.html 
- Lattice Multivariate Data Visualization with R - Deepayan Sarkar <http://lmdvr.r-forge.r-project.org/figures/figures.html> 
- Contributed R Information (A collection of free guides/handouts) <http://cran.r-project.org/other-docs.html> 
- Rtips - Paul Johnson. <http://pj.freefaculty.org/R/Rtips.html#toc-Subsection-2.1> 
- simpleR - Using R for Introductory Statistics - John Verzani - <http://www.unt.edu/rss/class/splus/Verzani-SimpleR.pdf> 
- Do it yourself introduction to R - University of North Texas http://www.unt.edu/rss/class/Jon/R_SC/ 
- R Bloggers - A news aggregate site for R <http://www.r-bloggers.com/> 
- R Journal - An open access, refereed journal of the R project for statistical computing. It features short to medium length articles covering topics that might be of interest to users or developers of R. <http://journal.r-project.org/index.html> 

R packages

- In a nutshell, an R package is a collection of R functions and data, with necessary documentation.
- R package is (after a short learning phase) a convenient way to maintain collections of R functions and data sets.
- In fact, all R functions are delivered in packages.
- Packages can be dynamically loaded and unloaded on runtime and hence only occupy memory when actually used.
- The R “base system” includes following packages: stats, graphics, grDevices, utils, datasets, methods, base. They are automatically loaded when starting R.

Where to obtain R packages

- There are several repositories where one can obtain R packages, including:
 - CRAN: The Comprehensive R Archive Network. This is the “official” R package distribution center.
 - Bioconductor: for bioinformatics related R packages.
 - R-Forge: similar to CRAN but better. It provides an “ecosystem” for R package developments, including version control system and forums/mailing list.
 - Researchers’ website, for example, mine.

CRAN (Comprehensive R Archive Network)

Contributed Packages

Available Packages

Currently, the CRAN package repository features 4342 available packages.

[Table of available packages, sorted by date of publication](#)

[Table of available packages, sorted by name](#)

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual [R Installation and Administration \[PDF\]](#) (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, 30 views are available.

Package Check Results

All packages are tested regularly on machines running [Debian GNU/Linux](#), [Fedora](#) and Solaris. Packages are also checked under MacOS X and Windows, but typically only on 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](#).

Bioconductor



Search:

Home

Install

Help

Developers

About

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, [610 software packages](#), and an active user community. Bioconductor is also available as an [Amazon Machine Image \(AMI\)](#).



Use Bioconductor for...

➤ **Microarrays**

Import Affymetrix, Illumina, Nimblegen, Agilent, and other platforms. Perform quality assessment, normalization, differential expression, clustering, classification, gene set enrichment, genetical genomics and other workflows for expression, exon, copy number, SNP, methylation and other assays. Access GEO, ArrayExpress, Biomart, UCSC, and other community resources.

➤ **Variants**

Read and write VCF files. Identify structural location of variants and compute amino acid coding changes for non-synonymous variants. Use SIFT and PolyPhen database packages to predict consequence of amino acid coding changes.

➤ **Annotation**

Use microarray probe, gene, pathway, gene ontology, homology and other annotations. Access GO, KEGG, NCBI, Biomart, UCSC, vendor, and other sources.

➤ **High Throughput Assays**

Import, transform, edit, analyze and visualize flow cytometric, mass spec, HTqPCR, cell-based, and other assays.

R-forge



Project

[Log In](#) | [New Account](#)

[Home](#)

[My Page](#)

[Projects](#)

What are R and R-Forge?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R-project homepage](#) for further information.

R-Forge offers a central platform for the development of R packages, R-related software and further projects. It is based on [FusionForge](#) offering easy access to the best in SVN, daily built and checked packages, mailing lists, bug tracking, message boards/forums, site hosting, permanent file archival, full backups, and total web-based administration.

A Platform for the Whole R Community

In order to get the most out of R-Forge you'll need to [register as a site user](#) and then [login](#). This will allow you to participate fully in all we have to offer, e.g., you may [register your project](#). Of course, you may also browse the site without registration, but will only have limited access to some features. For details see the documentation.

Documentation

- [Short Introduction](#): Stefan Theußl and Achim Zeileis. Collaborative software development using R-Forge. *The R Journal*, 1(1):9-14, May 2009. URL <http://journal.R-project.org/> [[bib](#)] [[pdf](#)] [[local copy](#)](Official R-Forge citation)
- [User's Manual](#): [[pdf](#)] (*Detailed technical documentation*)

If you experience any problems or need help you can submit a [support request to the R-Forge team](#) or write an email to R-Forge@R-Project.org.
Thanks... and enjoy the site.

Tag Cloud

Bayesian **Bioinformatics** Multivariate
Regression **R** biostatistics classification clustering data
mining ecology finance mixed effect models mixed
model model estimation multivariate optimization
spatial **spatial data** spatial methods spatio-
temporal time series visualization

R-Forge Statistics

Hosted Projects: **1,477**
Registered Users: **6,054**

Most Active This Week

(100.0%) **R.* packages**
(99.3%) **vegan - Community Ecology Package**
(98.6%) **NMF - Nonnegative Matrix Factorization**
(97.9%) **iHELP**
(97.2%) **Bayesian AuTomated Metabolite Analyser**

A few remarks for R packages

- There's no guarantee for bug free (as for all free software).
- The major packages (base, stat, util, etc.) should work fine.
- There are a lot of badly written packages.
- Packages are updated often (especially Bioconductor packages), and backward compatibility is not guaranteed. This means sometimes your old codes stop working with the new package.

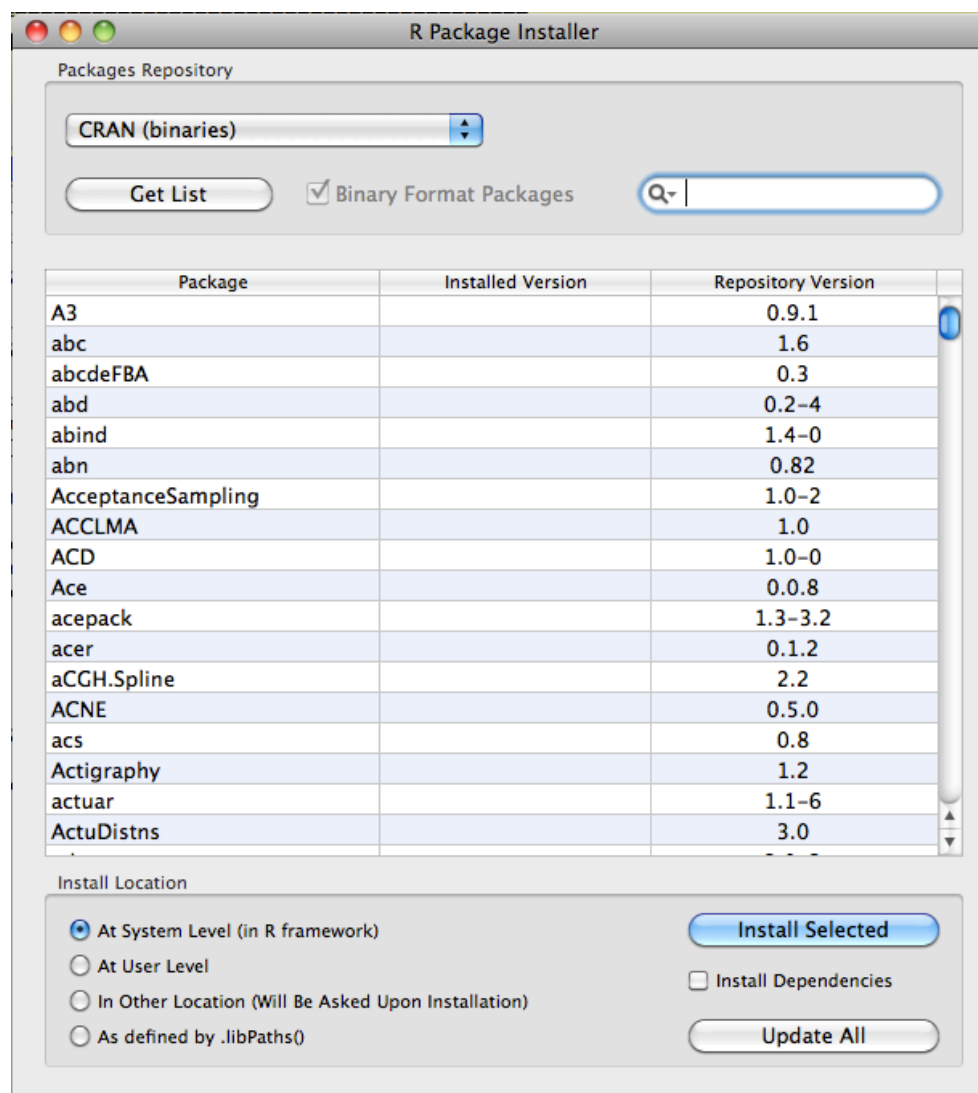
Package Installation

- Can be done in different ways:
 - Use GUI.
 - In R console.
 - Use command line.
- Using the GUI or R console is easier. It will determine the correct platform and R version, also install dependence packages.
- It's tricky to use command line to install from **package source**. But sometimes one has to.

Install a new R package - using the GUI

On Mac:

- Under “Packages & Data” menu, click “Package Installer” and get a dialog window.
- Choose a repository and specify the location, then install.



On Windows:

For CRAN packages:

- Click “Packages” menu, then select “Install package(s) ...”. If you do this the first time, you will be asked to choose a CRAN mirror set. Select “USA(TN)”.
- Click the package you want to install, then click OK. Again if it’s the first time, you’ll be asked to create a personal library, just click OK.

For packages not on CRAN. You will have to have the package source as a zip or gz file.

- Click “Packages” menu, then select “Install package(s) from local zip files ...”.

Install a new R package - in R console

For packages from CRAN, use the “**install.packages**” function to install.

```
> install.packages("ggplot2")  
also installing the dependencies 'digest', 'gtable', 'reshape2', 'proto'
```

```
trying URL 'http://cran.r-project.org/bin/macosx/leopard/contrib/2.15/  
digest_0.6.3.tgz'
```

```
Content type 'application/x-gzip' length 161113 bytes (157 Kb)
```

```
opened URL
```

```
=====
```

```
downloaded 157 Kb
```

```
trying URL 'http://cran.r-project.org/bin/macosx/leopard/contrib/2.15/  
gtable_0.1.2.tgz'
```

```
Content type 'application/x-gzip' length 60865 bytes (59 Kb)
```

```
opened URL
```

```
=====
```

Install a new R package - in command line

- If you are familiar with Linux style command line, you can obtain the package code (usually in a .tar.gz format), and issue the following command to install:

```
R CMD INSTALL pkg_1.0.tar.gz
```

- There's no compelling reason to do it this way unless you really like command line (like me).

R programming environment

- R console:
 - An interactive, command-line window, where you type your commands and see the text results. (Graphs appear in a separate window.)
 - **Up** and **down arrow keys** scroll through your command history.

R programming environment (cont.)

- R workspace:
 - Your current R working environment which includes any user-defined objects (vectors, matrices, data frames, lists, functions).
 - At the end of an **R** session, the user can save an image of the current workspace that is automatically reloaded the next time **R** is started (NOT recommended).

Getting in and out of R

- To start R, just click the icon or type “R” in command line.
- To quit, type “q()” in the console.
 - You’ll be asked whether to save the workspace image. My usual answer is no. It’s not a good practice to save the whole workspace.

Getting help

- R has built-in assistance that is very useful.
- Type the following in R console: `help.start()`

Statistical Data Analysis



Manuals

[An Introduction to R](#)
[Writing R Extensions](#)
[R Data Import/Export](#)

[The R Language Definition](#)
[R Installation and Administration](#)
[R Internals](#)

Reference

[Packages](#)

[Search Engine & Keywords](#)

Miscellaneous Material

[About R](#)
[License](#)
[NEWS](#)

[Authors](#)
[Frequently Asked Questions](#)
[User Manuals](#)

[Resources](#)
[Thanks](#)
[Technical papers](#)

Getting help (cont.)

- Help can be obtained at R console:

```
help(function_name) # Get help on function
?function_name     # Equivalent to the above
args(function_name) # See what arguments the function accepts
example(function_name) # See an example
?mean
example(mean)
```

- Search help by keyword:

```
help.search("time series")
??"time series"
```

R language basics – data types

- The primary variable classes include: numeric, logical, character, factor, and dates.
- The variable types include scalar, vector, matrix, data frame, and list.
 - For example, there can be vector/matrices of numeric, logical, or characters.
 - Data frame and list can contain variable of different classes.
- It's important to know how to manipulate these and, if necessary, convert among them.

Basic R data type - numeric

- Represents numbers.
- If we assign a number to a variable x as follows, x will be of numeric type.

```
> aa = 5  
> class(aa)  
[1] "numeric"
```

- Its worth pointing out that there is a distinction between integers and real values.

```
> aa2=as.integer(aa)  
> class(aa2)  
[1] "integer"
```

Numeric operators

- Arithmetic operators for numeric variables include $+$ $-$ $*$ $/$ $^$ $\% \%$.
 - $\% \%$ means compute the remainder.
 - See “?Arithmetic” for more details.

```
> 2+3
```

```
[1] 5
```

```
> 5^2 ## power
```

```
[1] 25
```

```
> 10%%3 ## remainder
```

```
[1] 1
```


Basic R data type - character

- Represent strings.

```
> aa = "a string"  
> class(aa)  
[1] "character"
```

- `as.character()` converts other data type to character.

```
> aa=3.14  
> class(aa)  
[1] "numeric"  
> aa2=as.character(aa)  
> aa2  
[1] "3.14"  
> class(aa2)  
[1] "character"
```

Basic R data type - logical

- Represent logical values (TRUE/FALSE), often returned from comparison between variables.

```
> aa=4>5
```

```
> aa
```

```
[1] FALSE
```

```
> class(aa)
```

```
[1] "logical"
```

```
> aa = "a string"=="a string"
```

```
> aa
```

```
[1] TRUE
```

```
> class(aa)
```

```
[1] "logical"
```

Logical operators

- Standard logical operations are "&" (and), "|" (or), and "!" (negation).

```
> u = TRUE; v = FALSE
```

```
> u & v      # u AND v
```

```
[1] FALSE
```

```
> u | v      # u OR v
```

```
[1] TRUE
```

```
> !u         # negation of u
```

```
[1] FALSE
```

Other data types

- There are some other basic data types:
 - date: represent the date.
 - complex: represent the complex numbers.
- We won't discuss these in details because they are less useful in our work.

Vector

- A **vector** is a sequence of data elements of the same basic type.
- vector is created using “c” command. Or “:” for a sequence of integers.

```
> c(2, 3, 5)
```

```
[1] 2 3 5
```

```
> 1:5
```

```
[1] 1 2 3 4 5
```

```
> c("aa", "bb", "cc")
```

```
[1] "aa" "bb" "cc"
```

```
> c(TRUE, FALSE, TRUE)
```

```
[1] TRUE FALSE TRUE
```

Basic vector operations - combine

- Combine two vectors: using “c”:

```
> a = c(1,3)
> b = c(10, 12, 15)
> c(a, b)
[1] 1 3 10 12 15
```

- When two vectors have different data types, some conversions are performed automatically.

```
> a=c(1,3)
> b=c("a", "b")
> c(a,b)
[1] "1" "3" "a" "b"
```

Basic vector operations - indexing

- Use square bracket. The index can be specified in different ways:
 - Numeric index:

```
> a=c(10,20,30,40)
> a[c(1,3)]
[1] 10 30
```
 - Logical index:

```
> a[c(TRUE, TRUE, TRUE, FALSE)]
[1] 10 20 30
```
 - There are other ways. But won't be covered in this lecture.

Matrix

- A **matrix** is a collection of data elements arranged in a two-dimensional rectangular layout. Matrix construction:

- Use “matrix” function:

```
> A = matrix(c(1,3,5, 2,4,6), ncol=2)
```

```
> A
```

```
      [,1] [,2]  
[1,]    1    2  
[2,]    3    4  
[3,]    5    6
```

- Use “cbind” or “rbind” to create from vectors.

```
> a=c(1,3,5); b=c(2,4,6)
```

```
> cbind(a,b)
```

```
      a b  
[1,] 1 2  
[2,] 3 4  
[3,] 5 6
```


Basic matrix operations - combine

- Combine two matrices: using “cbind” or “rbind”:

```
> a=matrix(c(1,3, 2,4), ncol=2)
> b=matrix(c(11,13, 12,14), ncol=2)
```

```
> a
```

```
      [,1] [,2]
[1,]     1     2
[2,]     3     4
```

```
> b
```

```
      [,1] [,2]
[1,]    11    12
[2,]    13    14
```

```
> cbind(a, b)
```

```
      [,1] [,2] [,3] [,4]
[1,]     1     2    11    12
[2,]     3     4    13    14
```

```
> rbind(a,b)
```

```
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]    11    12
[4,]    13    14
```

Basic matrix operations - Indexing

- Matrix can be indexed by row or column, using similar ways for indexing (numeric or logical).

```
> a=matrix(c(1,3, 5, 2,4,6), ncol=2)
> a[c(1,3), ]
      [,1] [,2]
[1,]    1    2
[2,]    5    6
> a[,2]
[1] 2 4 6
> a[1,2]
[1] 2
```

Basic matrix operations - transpose

- Use “t” function:

```
> a=matrix(c(1,3, 5, 2,4,6), ncol=2)
```

```
> a
```

	[,1]	[,2]
[1,]	1	2
[2,]	3	4
[3,]	5	6

```
> t(a)
```

	[,1]	[,2]	[,3]
[1,]	1	3	5
[2,]	2	4	6

Basic matrix operations - multiplication

- Use “%*%” to multiply two matrices:

```
> a=matrix(c(1,3, 5, 2,4,6), ncol=2)
```

```
> a2=t(a)
```

```
> a
```

```
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
```

```
> a2
```

```
      [,1] [,2] [,3]
[1,]     1     3     5
[2,]     2     4     6
```

```
> a2 %*% a
```

```
      [,1] [,2]
[1,]    35    44
[2,]    44    56
```

Basic matrix operations - inversion

- Use “solve” to invert a square matrix:

```
> a=matrix(c(1,2,3,4), nrow=2)
> a2=solve(a)
```

```
> a %*% a2
      [,1] [,2]
[1,]     1     0
[2,]     0     1
> a2 %*% a
      [,1] [,2]
[1,]     1     0
[2,]     0     1
```

List

- Generic vector containing other objects (of whatever types). Create using “list” function:

```
> n = matrix(c(1,2,3,4), nrow=2)
> s = c("aa", "bb", "cc", "dd", "ee")
> l = c(TRUE, FALSE, TRUE, FALSE, FALSE)
> x = list(n, s, l)
> x
[[1]]
      [,1] [,2]
[1,]    1    3
[2,]    2    4

[[2]]
[1] "aa" "bb" "cc" "dd" "ee"

[[3]]
[1] TRUE FALSE TRUE FALSE FALSE
```

List operation - combine

- Use “c” command:

```
> n = matrix(c(1,2,3,4), nrow=2)
> s = c("aa", "bb", "cc", "dd", "ee")
> l1 = list(n, s)
> l2 = list(c(TRUE, FALSE, TRUE, FALSE, FALSE))
> l=c(l1, l2)
> l
[[1]]
      [,1] [,2]
[1,]    1    3
[2,]    2    4

[[2]]
[1] "aa" "bb" "cc" "dd" "ee"

[[3]]
[1] TRUE FALSE TRUE FALSE FALSE
```

List operation - indexing

- Use single square bracket “[]” to get back another list:
- Use double square bracket “[[]]” to access an individual element.

```
> n = matrix(c(1,2,3,4), nrow=2)
> s = c("aa", "bb", "cc", "dd", "ee")
> l = c(TRUE, FALSE, TRUE, FALSE, FALSE)
> x = list(n, s, l)
>
> x[2]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
> x[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
```


- Elements in a list can be assigned a name, then they can be accessed by name using “\$” or “[[]]”:

```
> names(x) = c("a matrix", "strings", "logicals")
> x$strings
[1] "aa" "bb" "cc" "dd" "ee"
> x[["logicals"]]
[1] TRUE FALSE TRUE FALSE FALSE
```

- Names can also be assigned to vectors and matrix. We skipped the details.

Data frame

- A data frame is used for storing data tables. It is a list of vectors of equal length. It can be created using “data.frame” function.

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> l = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, l)
> df
```

	n	s	l
1	2	aa	TRUE
2	3	bb	FALSE
3	5	cc	TRUE

Data frame operations

- The operations of data frame are similar to those for both matrix and list:
 - Combine using “c”, “cbind”, “rbind”.
 - Index by row or column using numeric or logical indices.
 - Access column by “\$”.

R basic control statements

- R control statement is very similar to other languages (C/Java/Matlab), including:
 - If-else:
`if (cond) expr`
`if (cond) expr1 else expr2`
 - for: `for (var in seq) expr`
 - while: `while (cond) expr`
- Here, `expr` can be multiple (compound) statements by enclosing them in braces `{ }`.

File I/O: read in from files

- Read in text files (TAB or comma delimited) in R: `read.table` is a powerful function.
- Other (more advanced functions available): `scan`, `readLines`, `readBin`, etc. We won't cover those in this class.
- Read the 'R Data Import/Export' manual for details.

read.table example

```
> url = "http://stevie42.bitbucket.org/bios545r/DATA.DIR/hsb2.csv"
> data = read.table(url, header=TRUE, sep=",")
```

```
> head(data)
```

	id	female	race	ses	schtyp	prog	read	write	math	science	socst
1	70	0	4	1	1	1	57	52	41	47	57
2	121	1	4	2	1	3	68	59	53	63	61
3	86	0	4	3	1	1	44	33	54	58	31
4	141	0	4	3	1	3	63	44	47	53	56
5	172	0	4	2	1	2	47	52	57	53	61
6	113	0	4	2	1	2	44	52	51	63	61

```
> class(data)
```

```
[1] "data.frame"
```

File I/O: write to text file

- `write.table` function writes a data frame to a text file. The input will be converted into a data frame if it is not.

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> l = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, l)
> write.table(df, file="df.txt", sep="\t", quote=FALSE,
row.names=FALSE)
```

Other I/O functions

- “source” function runs a script in the current session:
> source("myscript.R")
- “save” function saves one or several R object to specified file. The objects can then be loaded back using “load” function.
> save(df, file="df.RData")
> load("df.RData")