UCLA Department of Statistics
Statistical Consulting Center

Basic R

Ryan Rosario
ryan@stat.ucla.edu

April 19, 2009

Outline

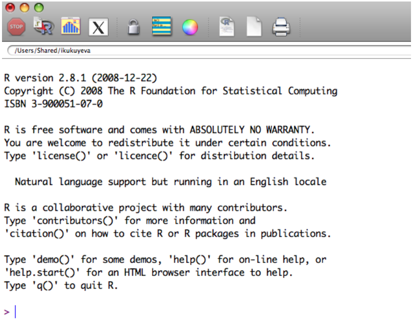
- I. Preliminaries
- II. Variable Assignment
- III. Working with Vectors
- IV. Working with Matrices
- V. From Vectors to Matrices
- VI. More on Handling Missing Data
- VII. The Help System
- VIII. Datasets in R
- IX. Overview of Plots in R
- X. R Environment
- XI. Common Bugs and Fixes
- XII. Online Resources for R
- XIII. Exercises
- XIV. Upcoming Mini-Courses

Part I

Preliminaries

Installing R on Mac

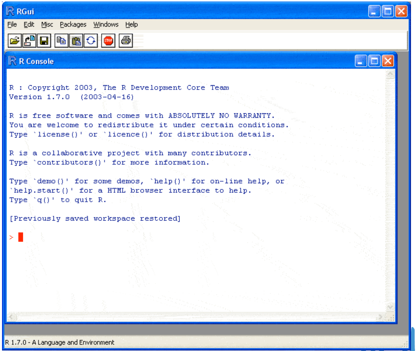
1. Go to `http://cran.r-project.org` and select MacOS X.
2. Select to download the latest version: 2.8.1
3. Install and Open. The R window should look like this:



```
R version 2.8.1 (2008-12-22)
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.

> |
```

Installing R on Windows

1. Go to `http://cran.r-project.org` and select Windows.
2. Select base to install the R system.
3. Click on the large download link. There is other information available on the page.
4. Install and Open. The R window should look like this:



```
R : Copyright 2003, The R Development Core Team
Version 1.7.0  (2003-04-16)

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.

R is a collaborative project with many contributors.
Type `contributors()' for more information.

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

[Previously saved workspace restored]

>
```

Part II

Variable Assignment

Creating Variables I

- To use R as a calculator, type $2+5$ and hit ENTER. (Note how R prints the result.) Your output should look like this:

```
[1] 7
```

- To create variables in R, use either <- or $=$:

```r
# Approach 1
a=5
a
# Approach 2
b<-5
b
```

Creating Variables II

Caution!

Be careful when using <- to compare a variable with a negative number!

```
#Assign a value to a
a<--2
#Is a less than -5?
a<-5
a
[1] 5 #Expected FALSE
```

Creating Variables III

Use spaces so that R will not be confused. It is better to use parentheses instead.

```
a <- 5
a < -2
[1] FALSE
```

Creating Variables IV

Caution!

It is important not to name your variables after existing variables or functions. For example, a bad habit is to name your data frames `data`. `data` is a function used to load some datasets.

If you give a variable the same name as an existing constant, that constant is overwritten with the value of the variable. So, it is possible to define a new value for π.

Creating Variables V

Caution!

On the other hand, if you give a variable the same name as an existing function, R will treat the identifier as a variable if used as a variable, and will treat it as a function when it is used as a function:

```
c <- 2 #typing c yields "2"
c(c,c) #yields a vector containing two 2s.
```

Creating Variables VI

Caution!

As we have seen, you can get away with using the same name for a variable as with an existing function, but you will be in trouble if you give a name to a function and a function with that name already exists.

Part III

Working with Vectors

Creating Vectors I

- Scalars are the most basic vectors.
- To create vectors of length greater than one, use the concatenation function `c()`:

```
d=c(3,4,7); d
```

```
[1] 3 4 7
```

The More You Know...

The semicolon `;` is used to combine multiple statements on one line.

Creating Vectors II

- To create a null vector:

```
x=c(); x
```

```
NULL
```

Creating Vectors III

- Creating a vector with equal spacing, use the sequence function `seq()`:

```
e=seq(from=1, to=3, by=0.5); e
```

```
[1] 1.0 1.5 2.0 2.5 3.0
```

- Creating a vector of a given length, use the repeat function `rep()`:

```
f=rep(NA, 6); f
```

```
[1] NA NA NA NA NA NA
```

Some Useful Vector Functions I

- To find the length of the vector, use `length()`:

```
length(d)
```

```
[1] 3
```

- To find the maximum value of the vector, use the maximum function `max()`:

```
max(d)
```

```
[1] 7
```

Some Useful Vector Functions II

- To find the minimum value of the vector, use the minimum function `min()`:

```
min(d)
```

```
[1] 3
```

- To find the mean of the vector, use `mean()`:

```
mean(d)
```

```
[1] 4.666667
```

Some Useful Vector Functions III

- To sort the vector, use `sort()`:

```
g<-c(2,6,7,4,5,2,9,3,6,4,3)
sort(g, decreasing=TRUE)
```

```
[1] 9 7 6 6 5 4 4 3 3 2 2
```

Caution!

Although `T` and `F` work in place of `TRUE` and `FALSE`, it is not recommended.

Some Useful Vector Functions IV

- To find the unique elements of the vector, use `unique()`:

```
unique(g)
```

```
[1] 2 6 7 4 5 9 3
```

Some Useful Vector Functions V

- Alternatively, to find the elements of the vector that repeat, use duplicated():

```
duplicated(g)
```

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

Some Useful Vector Functions VI

To determine if a value is missing (`NA`), use `is.na`. This is useful for finding missing values and removing them, or doing something else with them.

```r
a <- c(1,2,3,NA,6)
is.na(a)
```

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

But some functions do not tolerate missing values.

Some Useful Vector Functions VII

Caution!

```
mean(a)
[1] NA
mean(a, na.rm=TRUE)
[1] 1.5
```

Some Useful Vector Functions VIII

To get the number of missing values in a vector,

```r
sum(is.na(a))
```

```
[1] 1
```

There are other ways to handle missing values. See `?na.action`.

Some Useful Vector Functions IX

One final common function you can use on vectors (and other objects) is summary.

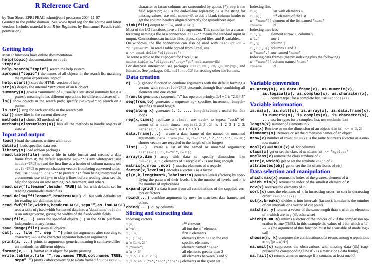
```
summary(a)
```

```
Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
1.00     1.75     2.50     3.00     3.75     6.00
NA's
1.00
```

Some Useful Vector Functions X

There are many, many other functions you can use on vectors.
See your reference card for more information!

Some Useful Vector Functions XI



R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.

help(topic) documentation on topic

?topic id.

help.search("topic") search the help system

apropos("topic") the names of all objects in the search list matching the regular expression "topic"

help.start() start the HTML version of help

str(a) display the internal *str*ucture of an R object

summary(a) gives a "summary" of a, usually a statistical summary but it is *generic* meaning it has different operations for different classes of a

ls() show objects in the search path; specify pat="pat" to search on a pattern

ls.str() str() for each variable in the search path

dir() show files in the current directory

methods(a) shows S3 methods of a

methods(class=class(a)) lists all the methods to handle objects of class a

Input and output

load() load the datasets written with save

data(x) loads specified data sets

library(x) load add-on packages

read.table(file) reads a file in table format and creates a data frame from it; the default separator sep="" is any whitespace; use header=TRUE to read the first line as a header of column names; use as.is=TRUE to prevent character vectors from being converted to factors; use comment.char="" to prevent "#" from being interpreted as a comment; use skip=n to skip n lines before reading data; see the help for options on row naming, NA treatment, and others

read.csv("filename",header=TRUE) id. but with defaults set for reading comma-delimited files

read.delim("filename",header=TRUE) id. but with defaults set for reading tab-delimited files

read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE) read a table of *f*ixed width *f*ormatted data into a 'data.frame'; widths is an integer vector, giving the widths of the fixed-width fields

save(file,...) saves the specified objects (...) in the XDR platform-independent binary format

save.image(file) saves all objects

cat(..., file="", sep=" ") prints the arguments after coercing to character; sep is the character separator between arguments

print(a, ...) prints its arguments; generic, meaning it can have different methods for different objects

format(x,...) format an R object for pretty printing

write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") prints x after converting to a data frame; if quote is TRUE, character or factor columns are surrounded by quotes ("); sep is the field separator; eol is the end-of-line separator; na is the string for missing values; use col.names=NA to add a blank column header to get the column headers aligned correctly for spreadsheet input

sink(file) output to file, until sink()

Most of the I/O functions have a file argument. This can often be a character string naming a file or a connection. file="" means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with description = "clipboard". To read a table copied from Excel, use
x <- read.delim("clipboard")
To write a table to the clipboard for Excel, use
write.table(x,"clipboard",sep="\t",col.names=NA)
For database interaction, see packages RODBC, DBI, RMySQL, RPgSQL, and ROracle. See packages XML, hdf5, netCDF for reading other file formats.

Data creation

c(...) generic function to combine arguments with the default forming a vector; with recursive=TRUE descends through lists combining all elements into one vector

from:to generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"

seq(from,to) generates a sequence by= specifies increment; length= specifies desired length

seq(along=x) generates 1, 2, ..., length(along); useful for for loops

rep(x,times) replicate x times; use each= to repeat "each" element of x each times; rep(c(1,2,3),2) is 1 2 3 1 2 3; rep(c(1,2,3),each=2) is 1 1 2 2 3 3

data.frame(...) create a data frame of the named or unnamed arguments; data.frame(v=1:4,ch=c("a","B","c","d"),n=10); shorter vectors are recycled to the length of the longest

list(...) create a list of the named or unnamed arguments; list(a=c(1,2),b="hi",c=3i);

array(x,dim=) array with data x; specify dimensions like dim=c(3,4,2); elements of x recycle if x is not long enough

matrix(x,nrow=,ncol=) matrix; elements of x recycle

factor(x,levels=) encodes a vector x as a factor

gl(n,k,length=n*k,labels=1:n) generate levels (factors) by specifying the pattern of their levels; k is the number of levels, and n is the number of replications

expand.grid() a data frame from all combinations of the supplied vectors or factors

rbind(...) combine arguments by rows for matrices, data frames, and others

cbind(...) id. by columns

Slicing and extracting data

Indexing vectors

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

Indexing lists

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

Indexing matrices

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

Indexing data frames (matrix indexing plus the following)

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

Variable conversion

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

Variable information

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ... test for type; for a complete list, use methods(is)

length(x) number of elements in x

dim(x) Retrieve or set the dimension of an object; dim(x) <- c(3,2)

dimnames(x) Retrieve or set the dimension names of an object

nrow(x) number of rows; NROW(x) is the same but treats a vector as a one-row matrix

ncol(x) and **NCOL(x)** id. for columns

class(x) get or set the class of x; class(x) <- "myclass"

unclass(x) remove the class attribute of x

attr(x,which) get or set the attribute which of x

attributes(obj) get or set the list of attributes of obj

Data selection and manipulation

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

which.min(x) returns the index of the smallest element of **x**

rev(x) reverses the elements of x

sort(x) sorts the elements of x in increasing order; to sort in decreasing order: rev(sort(x))

cut(x,breaks) divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points

match(x, y) returns a vector of the same length than x with the elements of x which are in y (NA otherwise)

which(x == a) returns a vector of the indices of x if the comparison operation is true (TRUE), in this example the values of i for which x[i] == a (the argument of this function must be a variable of mode logical)

choose(n, k) computes the combinations of *k* events among *n* repetitions $= n!/[(n-k)!k!]$

na.omit(x) suppresses the observations with missing data (NA) (suppresses the corresponding line if x is a matrix or a data frame)

na.fail(x) returns an error message if x contains at least one NA

Comparisons in R

&: in logic, is used to mean AND

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

==: in logic, is used for comparison (double equal sign)

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

==: in logic, is used for comparison (double equal sign)

>: in logic, is used to mean GREATER THAN

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

==: in logic, is used for comparison (double equal sign)

>: in logic, is used to mean GREATER THAN

<: in logic, is used to mean LESS THAN

Comparisons in R

$\&$: in logic, is used to mean AND

$|$: in logic, is used to mean OR

$==$: in logic, is used for comparison (double equal sign)

$>$: in logic, is used to mean GREATER THAN

$<$: in logic, is used to mean LESS THAN

$< -$ OR $=$: for variable assignment

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

$==$: in logic, is used for comparison (double equal sign)

$>$: in logic, is used to mean GREATER THAN

$<$: in logic, is used to mean LESS THAN

$< -$ OR $=$: for variable assignment

?: for use with a function in R, to determine what arguments to use, examples and background information

Comparisons in R

&: in logic, is used to mean AND

|: in logic, is used to mean OR

==: in logic, is used for comparison (double equal sign)

>: in logic, is used to mean GREATER THAN

<: in logic, is used to mean LESS THAN

< − OR =: for variable assignment

?: for use with a function in R, to determine what arguments to use, examples and background information

- Example: `?mean`

Subsetting with Vectors I

- To find out what is stored in a given element of the vector, use `[ ]`:

```
d[2]
```

```
[1] 4
```

- To see if the elements of a vector equal a certain number, use `==`:

```
d==3
```

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

Subsetting with Vectors II

- To see if any of the elements of a vector do not equal a certain number, use `!=`:

```
d!=3
```

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

- To obtain the element number of the vector when a condition is satisfied, use `which()`:

```
which(d==4)
```

```
[1] 2
```

To store the result, type: `a=which(d==4); a`.

Subsetting with Vectors III

- We can also tell R what we *do not* want when subsetting by using the minus – sign. To obtain everything but the 2nd element,

```
d <- seq(1,10,2)
d[-2]
```

```
[1] 1 5 7 9
```

Subsetting with Vectors IV

There is no function to remove an element from a vector. Instead, we use this subsetting by exclusion.

```
d <- seq(1,10,2)
d <- d[-2]
```

Subsetting with Vectors V

We can use subsetting to explicitly tell R what observations we want to use. To get all elements of d greater than or equal to 2,

```
d[d >= 2]
```

```
[1] 3 5 7 9
```

- R will return values of d where the expression within brackets is TRUE. Think of these statements as: ”give me all d such that d $\geq$ 2.”
  We will see more sophisticated (realistic) examples in the exercises...

Part IV

Working with Matrices

Creating Matrices I

To create a matrix, use the `matrix()` function:

```
mat<-matrix(10:15, nrow=3, ncol=2); mat
```

```
     [,1] [,2]
[1,]   10   13
[2,]   11   14
[3,]   12   15
```

Some Useful Matrix Functions I

To add two matrices, use +

```
mat+mat
```

```
     [,1] [,2]
[1,]   20   26
[2,]   22   28
[3,]   24   30
```

Some Useful Matrix Functions II

To find the transpose of a matrix, use `t()`:

```
t(mat)
```

```
     [,1] [,2] [,3]
[1,]   10   11   12
[2,]   13   14   15
```

To find the dimensions of a matrix, use `dim()`:

```
dim(mat)
```

```
[1] 3 2
```

Some Useful Matrix Functions III

To multiply two matrices, use `%*%`.
Note: If you use `*` instead, you will be performing matrix multiplication element-wise.

```
mat%*%t(mat)
```

```
     [,1] [,2] [,3]
[1,]  269  292  315
[2,]  292  317  342
[3,]  315  342  369
```

Alternatively, we can find the rows and columns of the matrix, by `nrow()` and `ncol()`:

Some Useful Matrix Functions IV

```r
nrow(mat); ncol(mat)
```

Subsetting with Matrices I

- To see what is stored in the first element of the matrix, use
`[ ]`:

```
mat[1,1]
```

```
[1] 10
```

- To see what is stored in the first row of the matrix:

```
mat[1,]
```

```
[1] 10 13
```

Subsetting with Matrices II

- To see what is stored in the second column of the matrix:

```
mat[, 2]
```

```
[1] 13 14 15
```

- To extract elements 1 and 3 from the second column, use `c()` and `[ ]`:

```
mat[c(1,3), 2]
```

```
[1] 13 15
```

Part V

From Vectors to Matrices

Creating Matrices from Vectors I

- To stack two vectors, one below the other, use `rbind()`:

```
mat1<-rbind(d,d); mat1
```

```
  [,1] [,2] [,3]
d    3    4    7
d    3    4    7
```

- To stack two vectors, one next to the other, use `cbind()`:

```
mat2<-cbind(d,d); mat2
```

Creating Matrices from Vectors II

```
     d d
[1,] 3 3
[2,] 4 4
[3,] 7 7
```

Part VI

More on Handling Missing Data

Missing Data in Matrices I

- Start by creating a matrix with missing data:

```
h=matrix(c(NA,3,1,7,-8,NA), nrow=3,
    ncol=2, byrow=TRUE); h
```

```
     [,1] [,2]
[1,]   NA    3
[2,]    1    7
[3,]   -8   NA
```

Missing Data in Matrices II

- To see if any of the elements of a vector are missing use `is.na()`:

```r
is.na(h)
```

```
      [,1]  [,2]
[1,]  TRUE FALSE
[2,] FALSE FALSE
[3,] FALSE  TRUE
```

Missing Data in Matrices III

- To see how many missing values there are, use `sum()` and `is.na()` (TRUE=1, FALSE=0):

```
sum(is.na(h))
```

```
[1] 2
```

- To obtain the element number of the matrix of the missing value(s), use `which()` and `is.na()`:

```
which(is.na(h))
```

```
[1] 1 6
```

Missing Data in Matrices IV

- To keep only the rows without missing value(s), use `na.omit()`

```
na.omit(h)
```

```
              [,1] [,2]
[1,]     1     7
attr(,"na.action")
[1] 1 3
attr(,"class")
[1] "omit"
```

Part VII

The Help System

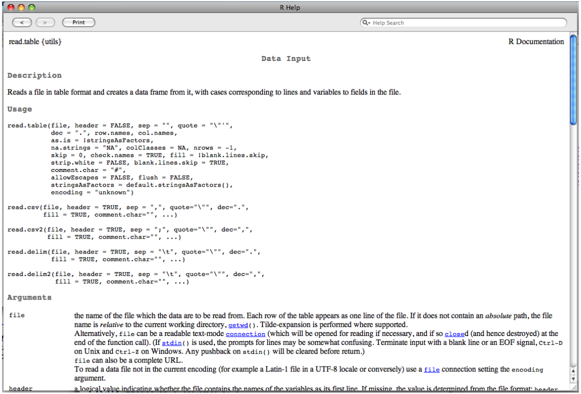
Help with a Function I

- To get help with a function in R, use `?` followed by the name of the function.

```
?read.table
```

- By the way, `help(function_name)` also works.

Help with a Function II



read.table {utils}   R Documentation

Data Input

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           encoding = "unknown")

read.csv(file, header = TRUE, sep = ",", quote="\"", dec=".",
         fill = TRUE, comment.char="", ...)

read.csv2(file, header = TRUE, sep = ";", quote="\"", dec=",",
          fill = TRUE, comment.char="", ...)

read.delim(file, header = TRUE, sep = "\t", quote="\"", dec=".",
           fill = TRUE, comment.char="", ...)

read.delim2(file, header = TRUE, sep = "\t", quote="\"", dec=",",
            fill = TRUE, comment.char="", ...)
```

Arguments

`file` the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, `getwd()`. Tilde-expansion is performed where supported.
Alternatively, `file` can be a readable text-mode `connection` (which will be opened for reading if necessary, and if so `close`d (and hence destroyed) at the end of the function call). (If `stdin()` is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, `Ctrl-D` on Unix and `Ctrl-Z` on Windows. Any pushback on `stdin()` will be cleared before return.)
`file` can also be a complete URL.
To read a data file not in the current encoding (for example a Latin-1 file in a UTF-8 locale or conversely) use a `file` connection setting the `encoding` argument.

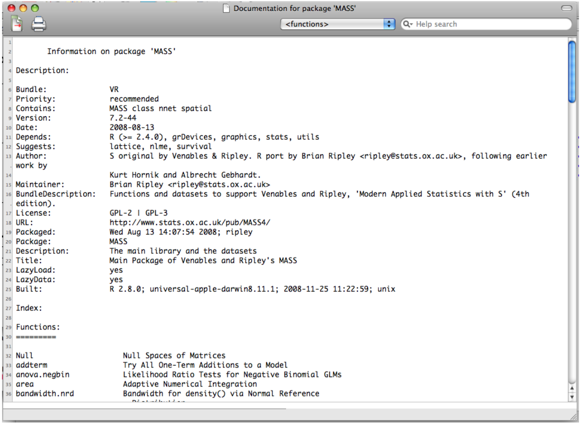
`header` a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: `header`

Help with a Package I

To get help with a package, use `help(package="name")`.

```
help(package="MASS")
```

Help with a Package II



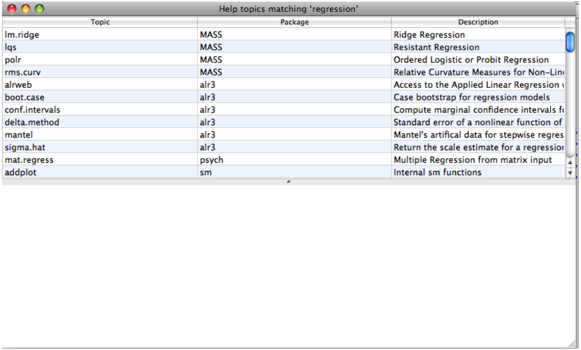
```
       Information on package 'MASS'

Description:

Bundle:              VR
Priority:            recommended
Contains:            MASS class nnet spatial
Version:             7.2-44
Date:                2008-08-13
Depends:             R (>= 2.4.0), grDevices, graphics, stats, utils
Suggests:            lattice, nlme, survival
Author:              S original by Venables & Ripley. R port by Brian Ripley <ripley@stats.ox.ac.uk>, following earlier
work by
                     Kurt Hornik and Albrecht Gebhardt.
Maintainer:          Brian Ripley <ripley@stats.ox.ac.uk>
BundleDescription:   Functions and datasets to support Venables and Ripley, 'Modern Applied Statistics with S' (4th
edition).
License:             GPL-2 | GPL-3
URL:                 http://www.stats.ox.ac.uk/pub/MASS4/
Packaged:            Wed Aug 13 14:07:54 2008; ripley
Package:             MASS
Description:         The main library and the datasets
Title:               Main Package of Venables and Ripley's MASS
LazyLoad:            yes
LazyData:            yes
Built:               R 2.8.0; universal-apple-darwin8.11.1; 2008-11-25 11:22:59; unix

Index:

Functions:
=========

Null                    Null Spaces of Matrices
addterm                 Try All One-Term Additions to a Model
anova.negbin            Likelihood Ratio Tests for Negative Binomial GLMs
area                    Adaptive Numerical Integration
bandwidth.nrd           Bandwidth for density() via Normal Reference
                          Distribution
```

Searching for Help I

To get help with a package, use `help.search()`.

help.search(”regression”)

Searching for Help II



Help topics matching 'regression'

Topic	Package	Description
lm.ridge	MASS	Ridge Regression
lqs	MASS	Resistant Regression
polr	MASS	Ordered Logistic or Probit Regression
rms.curv	MASS	Relative Curvature Measures for Non-Lin
alrweb	alr3	Access to the Applied Linear Regression w
boot.case	alr3	Case bootstrap for regression models
conf.intervals	alr3	Compute marginal confidence intervals fo
delta.method	alr3	Standard error of a nonlinear function of
mantel	alr3	Mantel's artifical data for stepwise regres
sigma.hat	alr3	Return the scale estimate for a regression
mat.regress	psych	Multiple Regression from matrix input
addplot	sm	Internal sm functions

Part VIII

Datasets in R

Data from the Internet I

When downloading data from the internet, use `read.table()`. In the arguments of the function:

- `header:` if TRUE, tells R to include variables names when importing
- `sep:` tells R how the entires in the data set are separated
  - `sep=",":` when entries are separated by COMMAS
  - `sep="\t":` when entries are separated by TAB
  - `sep=" ":` when entries are separated by SPACE

Data from the Internet II

```
df<-read.table("http://www.stat.ucla.edu
/~rosario/scc/textbooks.csv",
header=TRUE, sep=",")
```

Importing Data from Your Computer I

1. Check what folder R is working with now:

```r
getwd()
```

2. Tell R in what folder the data set is stored (if different from (1)). Suppose your data set is on your desktop:

```r
setwd("~/Desktop")
```

3. Now use the `read.table()` command to read in the data, substituting the name of the file for the website.

Using Data Available in R I

To use a data set available in one of the R packages, install that package (if needed).

Load the package into R, using the `library()` function.

```
library(alr3)
```

Extract the data set you want from that package, using the `data()` function. In our case, the data set is called `UN2`.

```
data(UN2)
```

Working with Datasets in R I

To use the variable names when working with data, use `attach()`:

```
data(UN2)
attach(UN2)
```

After the variable names have been "attached", to see the variable names, use `names()`:

```
names(UN2)
```

To see the descriptions of the variables, use `?`:

```
?UN2
```

Working with Datasets in R II

After modifying variables, use `detach()` and `attach()` to save the results:

```
# Make a copy of the data set
UN2.copy<-UN2
detach(UN2)
attach(UN2.copy)
# Change the 10th observation for logFertility
UN2.copy[10, 2]<-999
```

Working with Datasets in R III

To get an overview of the data sets and its variables, use the `summary()` function:

```
# Check that the change has been made
summary(UN2)
summary(UN2.copy)
```

Working with Datasets in R IV

Caution!

Avoid using `attach()` if possible. Many strange things can occur if you accidentally attach the same data frame multiple times, or forget to `detach`. Instead, you can refer to a variable using `$`. To access the `Locality` variable in data frame `UN2`, use `UN2$Locality`.
"`attach` at your own risk!"

Working with Datasets in R V

To get the mean of all the variables in the data set, use `mean()`:

```
mean(df,na.rm=TRUE)
```

Working with Datasets in R VI

```
> mean(df,na.rm=TRUE)
Author               Title          Department
       NA                  NA                  NA
Amazon.List.Price   Amazon.Sale.Price  Barnes.Noble.Price
       46.83325            42.48925            47.78401
Textbooks.com.Price  Borders.Price
       88.71250  52.38154
Warning messages:
1: In mean.default(X[[1L]], ...) :
  argument is not numeric or logical: returning NA
....
```

Working with Datasets in R VII

To get the correlation matrix of all the (numerical) variables in the data set, use `cor()`:

```
cor(df[c(4,5), c(4,5)])
```

```
                  Amazon.List.Price Amazon.Sale.Price
Amazon.List.Price                 1                -1
Amazon.Sale.Price                -1                 1
```

Can obtain the variance-covariance matrix using `var`.

Part IX

Overview of Plots in R

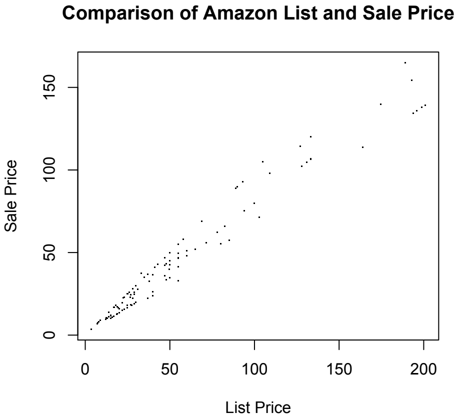
Creating Plots in R I

To make a plot in R, you can use `plot()`:

```
plot(Amazon.Sale.Price~Amazon.List.Price,
        main="Comparison of Amazon List and Sale Price",
        xlab="List Price",
        ylab="Sale Price",pch='.')
```

Creating Plots in R I

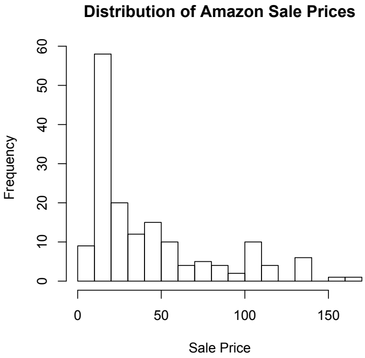
Creating Plots in R II



Creating Plots in R I

To make a histogram in R, you can use hist():

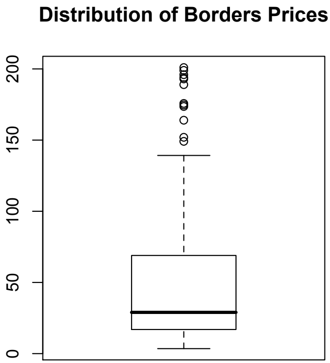
```
hist(Amazon.Sale.Price,main="Distribution of Amazon Sale Prices",xlab="Sale Price",br=20)
```



Creating Plots in R II

To make a boxplot in R, you can use boxplot():

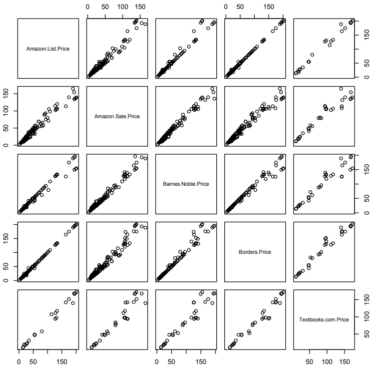
```
boxplot(Borders.Price, main="Distribution of Borders Prices")
```



Creating Plots in R III

To make a scatterplot for all the (numerical) variables, you can use pairs():

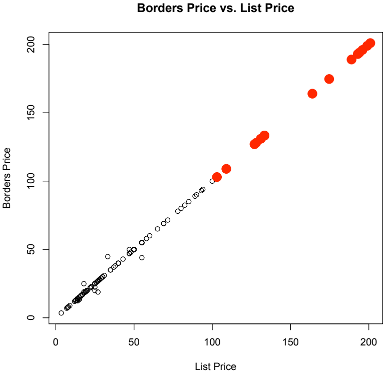
```r
pairs(df[, c(4:8)])
```



Creating Plots in R IV

To add more points to an existing plot, use `points()`. Here, we add a red dot for books with list cost > $100.

```
plot(Borders.Price~Amazon.List.
    Price, xlab="List Price",
    ylab="Borders Price",main="
    Borders Price vs. List Price
    ")
points(Borders.Price[Borders.
    Price > 100],Amazon.List.
    Price[Borders.Price>100],
    col="red", pch=19, cex=2)
```



Creating Plots in R V

Caution!

Once a plot is constructed using `plot`, it cannot be modified. To overlay things on a rendered plot, use one of the following

1. `abline` - add a line with slope *b*, intercept *a* or horizontal/vertical.
2. `points` - add points.
3. `lines` - add lines.

Saving Plots as a PDF I

Note: The files will be saved in the folder specified with `setwd()`.

To save a plot in R as a PDF, you can use `pdf()`:

```r
pdf("myniftyplot.pdf", width=6, height=6, onefile=FALSE)
boxplot(Borders.Price, main="Distribution of Borders Prices")
dev.off()
```

Part X

R Environment

Exploring R Objects I

- To see the names of the objects available to be saved (in your current workspace), use `ls()`.

```r
ls()
```

```
[1] "UN2" "a" "b" "d" "data" "e" "f" "h" "mat1" "mat2"
```

- To remove objects from your workspace, use `rm()`.

```r
rm(d)
ls()
```

```
[1] "UN2" "a" "b" "data" "e" "f" "h" "mat1" "mat2"
```

Exploring R Objects II

- To remove *all* the objects from your workspace, type:

```
rm(list=ls())
ls()
```

```
character(0)
```

Saving and Loading R Objects I

- To save (to the current directory) all the objects in the workspace, use `save.image()`.

```r
save.image("basicR.RData")
```

- To load (from the current directory), use `load()`.

```r
load("basicR.RData")
```

Saving and Loading R Objects I

- To save (to the current directory) a **single** object in the workspace, use `save()`.

```r
save(df,file="textbooks.RData")
```

- To load (from the current directory), use `load()`.

```r
load("textbooks.RData")
```

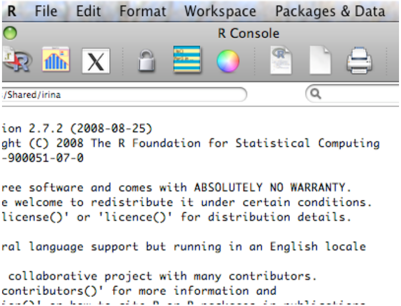
Exporting R Objects to Other Formats I

- To save (to the current directory) certain objects in the workspace to be used in Excel, use `write.csv()`.

```
write.csv(df, file="textbooks.csv")
```

Saving R Commands I

- To see all of the commands you typed in an R session, click on the Yellow and Green Tablet



```
ion 2.7.2 (2008-08-25)
ght (C) 2008 The R Foundation for Statistical Computing
-900051-07-0

ree software and comes with ABSOLUTELY NO WARRANTY.
e welcome to redistribute it under certain conditions.
license()' or 'licence()' for distribution details.

ral language support but running in an English locale

 collaborative project with many contributors.
contributors()' for more information and
ion()' on how to cite R or R packages in publications
```

Saving R Commands II

- To save all of the commands you typed in an R session, use:

```
savehistory(file="history.log")
```

Saving R Commands III

- Alternatively, use a `.r` file to store your commands.
  1. Go to: File -> New Document
  2. Type your commands
  3. Save the file as `"code.r"`
  4. Go back to the R Console
  5. To run all the commands, use:

```
source("code.r")
```

The More You Know...

Use the # sign to write comments in your code. Use them!

Part XI

Common Bugs and Fixes

`Error: syntax error`

Possible causes:

- Incorrect spelling (of the function, variable, etc.)
- Including a "+" when copying code from the Console
- Having an extra parenthesis at the end of a function
- Having an extra bracket when subsetting

Trailing +

Possible causes:

- Not closing a function call with a parenthesis
- Not closing brackets when subsetting
- Not closing a function you wrote with a squiggly brace

You can escape this sticky situation by hitting the ESCAPE key to exit your command.

`Error in ... : requires numeric matrix/vector arguments`

Possible causes:

1. Objects are data frames, not matrices
2. Elements of the vectors are characters

Possible solutions:

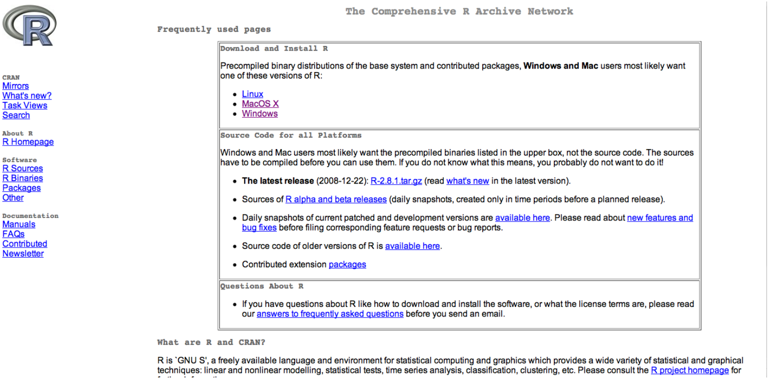
1. Coerce (a copy of) the data set to be a matrix, with the `as.matrix()` command
2. Coerce (a copy of) the vector to have numeric entries, with the `as.numeric()` command

Part XII

Online Resources for R

CRAN

`http://cran.stat.ucla.edu/`



The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

Frequently used pages

Download and Install R

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

- Linux
- MacOS X
- Windows

Source Code for all Platforms

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

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

Questions About R

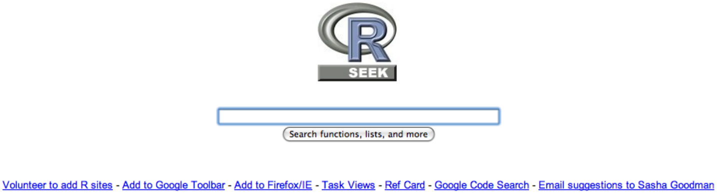
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

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

R-Seek Search Engine

`http://www.rseek.org`


R
SEEK
Search functions, lists, and more
Volunteer to add R sites - Add to Google Toolbar - Add to Firefox/IE - Task Views - Ref Card - Google Code Search - Email suggestions to Sasha Goodman

UCLA Statistics Bootcamp Resources

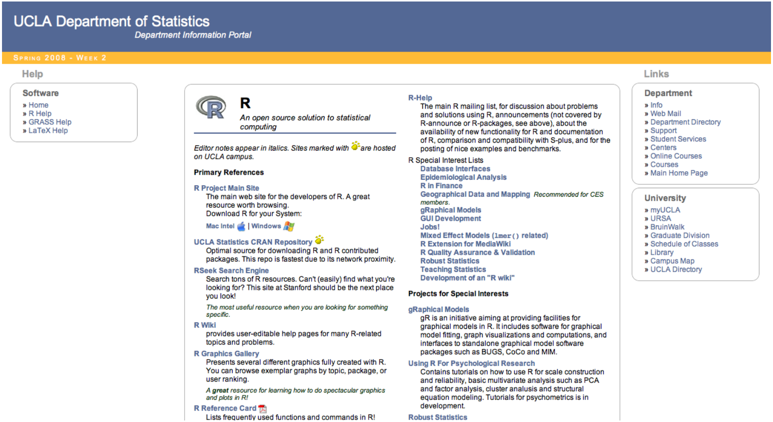
R Bootcamp is a day-long version of this mini-course. Handouts and datasets from Bootcamp 2007 and 2008 can be found on my website:

`http://www.stat.ucla.edu/ rosario/boot08/`

`http://www.stat.ucla.edu/ rosario/boot/`

UCLA Statistics Information Portal

http://info.stat.ucla.edu/grad/



UCLA Department of Statistics
Department Information Portal

SPRING 2008 - WEEK 2

Help

Software
- » Home
- » R Help
- » GRASS Help
- » LaTeX Help

R
An open source solution to statistical computing

Editor notes appear in italics. Sites marked with ● are hosted on UCLA campus.

Primary References

R Project Main Site
The main web site for the developers of R. A great resource worth browsing.
Download R for your System:
Mac Intel | Windows

UCLA Statistics CRAN Repository ●
Optimal source for downloading R and R contributed packages. This repo is fastest due to its network proximity.

RSeek Search Engine
Search tons of R resources. Can't (easily) find what you're looking for? This site at Stanford should be the next place you look!

The most useful resource when you are looking for something specific.

R Wiki
provides user-editable help pages for many R-related topics and problems.

R Graphics Gallery
Presents several different graphics fully created with R. You can browse exemplar graphs by topic, package, or user ranking.

*A **great** resource for learning how to do spectacular graphics and plots in R!*

R Reference Card
Lists frequently used functions and commands in R!

R-Help
The main R mailing list, for discussion about problems and solutions using R, announcements (not covered by R-announce or R-packages, see above), about the availability of new functionality for R and documentation of R, comparison and compatibility with S-plus, and for the posting of nice examples and benchmarks.

R Special Interest Lists
- Database Interfaces
- Epidemiological Analysis
- R in Finance
- Geographical Data and Mapping *Recommended for CES members.*
- gRaphical Models
- GUI Development
- Jobs!
- Mixed Effect Models (`lmer()` related)
- R Extension for MediaWiki
- R Quality Assurance & Validation
- Robust Statistics
- Teaching Statistics
- Development of an "R wiki"

Projects for Special Interests

gRaphical Models
gR is an initiative aiming at providing facilities for graphical models in R. It includes software for graphical model fitting, graph visualizations and computations, and interfaces to standalone graphical model software packages such as BUGS, CoCo and MIM.

Using R For Psychological Research
Contains tutorials on how to use R for scale construction and reliability, basic multivariate analysis such as PCA and factor analysis, cluster analusis and structural equation modeling. Tutorials for psychometrics is in development.

Robust Statistics

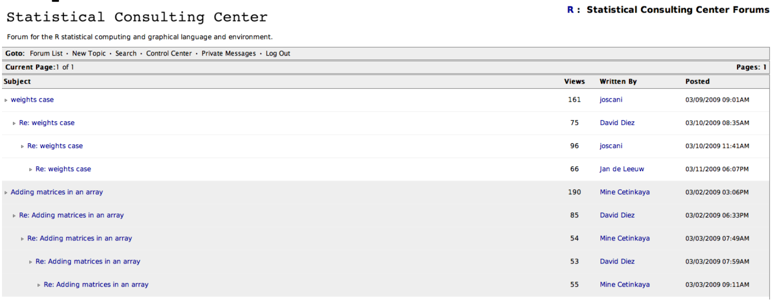
Links

Department
- » Info
- » Web Mail
- » Department Directory
- » Support
- » Student Services
- » Centers
- » Online Courses
- » Courses
- » Main Home Page

University
- » myUCLA
- » URSA
- » BruinWalk
- » Graduate Division
- » Schedule of Classes
- » Library
- » Campus Map
- » UCLA Directory

UCLA Statistical Consulting Center E-consulting and Walk-in Consulting

http://scc.stat.ucla.edu



Statistical Consulting Center

R : Statistical Consulting Center Forums

Forum for the R statistical computing and graphical language and environment.

Goto: Forum List • New Topic • Search • Control Center • Private Messages • Log Out

Current Page: 1 of 1 **Pages:** 1

Subject	Views	Written By	Posted
▸ weights case	161	joscani	03/09/2009 09:01AM
▸ Re: weights case	75	David Diez	03/10/2009 08:35AM
▸ Re: weights case	96	joscani	03/10/2009 11:41AM
▸ Re: weights case	66	Jan de Leeuw	03/11/2009 06:07PM
▸ Adding matrices in an array	190	Mine Cetinkaya	03/02/2009 03:06PM
▸ Re: Adding matrices in an array	85	David Diez	03/02/2009 06:33PM
▸ Re: Adding matrices in an array	54	Mine Cetinkaya	03/03/2009 07:49AM
▸ Re: Adding matrices in an array	53	David Diez	03/03/2009 07:59AM
▸ Re: Adding matrices in an array	55	Mine Cetinkaya	03/03/2009 09:11AM

Part XIII

Exercises

Exercises in R

1. Load in the UCLA textbook price comparison data from `http://www.stat.ucla.edu/~rosario/scc/textbooks.csv` It is a CSV file with a header. Find the average list price of textbooks for courses in Management. (Hint: `summary(data$Department)`)
2. Plot the Amazon list price vs. the Amazon sales price and Amazon list price vs. Barnes & Noble price on the same plot. Use different plotting symbols for Amazon and Barnes and Noble. Add a dashed line representing the location on the plot where the sales price of a book is equal to the list price. Highlight in red those Amazon books that differ from the list price by more than 25%. Label the plot and the axes.

Solution to the Exercises

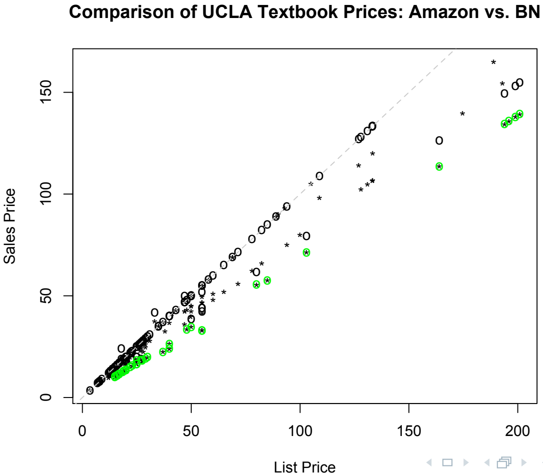
Exercise 1 Solution

```r
mean(Amazon.List.Price[Department == "XMANG"],
    na.rm=TRUE)
```

Solution Exercise 2

```r
plot(Amazon.Sale.Price~Amazon.List.Price,pch='
    *',main="Comparison of UCLA Textbook
    Prices: Amazon vs. BN",xlab="List Price",
    ylab="Sales Price")
points(Barnes.Noble.Price~Amazon.List.Price,
    pch='o')
abline(a=0,b=1,lty=2,col="grey")
good.deals <- which(abs((Amazon.List.Price-
    Amazon.Sale.Price)/Amazon.List.Price) >=
    0.25)
points(Amazon.Sale.Price[good.deals]~Amazon.
    List.Price[good.deals],pch='o',col="green"
    )
```

Solution Exercise 2



Part XIV

Upcoming Mini-Course

Upcoming Mini-Courses

- Thursday (April 16) and Tuesday (April 21):
  - Migrating to R for SAS/SPSS/Stata Users
- For a schedule of all mini-courses offered please visit *http://scc.stat.ucla.edu/mini-courses*.

We Want to Hear From You!

Please complete our online survey. Your feedback will help us improve our mini-course series. We greatly appreciate your time.

`http://scc.stat.ucla.edu/survey`