

Introduction to 'R'

Rob Cribbie
Department of Psychology
York University



What is R?

- R is a computer language, with exceptional capabilities for statistical analysis and graphics
- R is the open source version of S
 - S was developed at Bell Laboratories in the 1970s, and is commercially sold as S-Plus
- R was initially written in the 1990s by Ross Ihaka and Robert Gentleman, and is now supported by an international R-core team (in addition to thousands of people who contribute to R)



What can R do?

- Handle and store data
- Perform mathematical operations on data points, vectors or matrices
- Perform basic and advanced statistical analyses through either the preloaded functions or user contributed functions
- Can be used as a simple and effective programming language which includes conditionals, loops, if-else, etc. (plus any user contributed or created functions)



R Advantages

- Advantages of R over other software packages:
 - ▶ Free!!
 - ▶ Completely customizable
 - ▶ Active user community
 - ▶ Forces you to think about your analyses
 - ▶ Packaging: users are able to obtain, store and manipulate statistical add ons, called 'functions', which are stored in 'packages'
 - Because of the availability of user contributed 'functions' the gap between the introduction of new statistical procedures in the literature and their availability in R is often much shorter than with other packages



R Disadvantages

- NOT USER FRIENDLY!
 - Steep Learning Curve
- Help files are sometimes difficult to use
- Error messages are sometimes (often?) uninformative
- Working with large data sets (or running simulations) requires a lot of computer speed and memory
 - unless you are very patient :-)



How to Get R

- The R webpage is: <http://www.r-project.org/>
- This webpage contains documentation, manuals, faqs, links to books, a newsgroup, an R search engine, etc.
- To install R (version 2.6.2) on your computer, click on 'CRAN' under the 'Download' heading, and then select a mirror site
 - ▶ Click on 'base', to download the base package
 - ▶ Select the .exe file (the name changes with each new version of R)
 - Note that R is available for Windows, MAC, UNIX, etc.



Running R

- Like any other Windows program, R can be run by clicking on the icon or running R from the 'Programs' menu
- When you start R the first thing you will see is the prompt (`>`), which is R's way of saying "Go Ahead ... Do something"
- If you see a "+" in place of the prompt that means that your last command was not complete
- Don't forget that R is case sensitive!



Getting Help with R

- A lot of valuable information including links to a mailing list, an 'R search' website, and other helpful information can be found through CRAN
 - ▶ <http://cran.r-project.org/>
- I have also found that 'googling' the topic of interest, followed by r, can be effective
 - ▶ Example: google 'anova r'
- If you know the r function you are interested in, you can get help on it in R by typing
 - ▶ `>?function`
 - ▶ Example: `>?t.test`



Packages and Functions

- One of the tricky parts to learning R is understanding what functions are automatically available, and how to access functions that are not automatically available
- Functions are R commands, that often have specific defaults and options
 - ▶ 't.test' is an R function that computes a t-test
- However, 't.test' is also part of a package called 'statistics'
 - ▶ The 'statistics' package is automatically installed when you install R and is automatically loaded each time you start R



Packages and Functions, cont'd

- Because the 'statistics' package is automatically loaded each time you start R, all of the functions available in the 'statistics' package are available at any time
- Other packages are installed when you install R, but are not loaded each time you start R
 - The package 'foreign', which has functions for reading external data sets is installed when you install R, but is not loaded when you start R
- Other packages need to be both installed and loaded before they are accessible



Loading Installed Packages

- What if we try the following command:
 - ▶ `>histogram(x)`
 - ▶ Error: could not find function "histogram"
- Why do we get an error?
 - ▶ The function “histogram” is part of the package “lattice”, which is part of the default installation but is not loaded each time you start R
 - ▶ To load a new package we can use the drop down menus or just type
 - `>library(lattice)`
 - ▶ To get help with any of the commands or packages type either
 - `>help(lattice)` or `>?lattice`



Installing Packages

- Many packages are not installed as part of the default installation of R
 - ▶ The reason for this is that there are more than a thousand packages (and this number is growing fast) and there are probably only a small subset of those that will ever be of interest to you
- To install a new package in R, click on “Packages” and then “Install package(s)...”
- If you know the name of the package you want to install you can just use:
 - ▶ `install.packages("package_name")`



Creating and Submitting Commands in R

- All commands can be entered at the R prompt (>)
- However, when you are working with a large string of commands, creating programs or functions, or saving your commands, it is usually easier to use a script editor
- Examples of script editors are:
 - ▶ R editor
 - click on 'File', then 'New Script' and press "ctrl-R" to submit highlighted commands
 - ▶ Notepad
 - ▶ Tinn-R



How to get Tinn-R

- Tinn-R is a free and efficient replacement for the basic script editor provided by R
 - ▶ Tinn-R makes it very easy to submit commands to R, and can also help you with your R syntax if you forget the details of a function (press 'ctrl-D')
- To download Tinn-R to your computer go to:
 - ▶ <http://sourceforge.net/projects/tinn-r>
 - ▶ Click on "Download Tinn-R", then "Tinn-R Setup"
 - ▶ Next, click on the file 'Tinn-R_1.19.4.7_setup.exe' which will start the download
 - ▶ Update: The most recent version of Tinn-R has had some bugs, so if you have trouble you may want to wait for an update



The R Commander

- John Fox, a sociology professor at McMaster University created a GUI for running basic commands in R
 - ▶ To obtain the GUI first install and activate the Rcmdr package
 - `install.packages("Rcmdr", dependencies=TRUE)`
 - `library(Rcmdr)`
- We are not using the R commander for this session for two reasons:
 - ▶ I am not very familiar with the R commander
 - ▶ In my opinion, it takes us away from the “statistical exploration” that R creates



R as a Fancy Calculator

- ▶ `> 2+(2*3/6)`
- ▶ `[1] 3`
- ▶ `> sqrt(2)`
- ▶ `[1] 1.414214`
- ▶ `> factorial(5)`
- ▶ `[1] 120`
- ▶ `> log(10)`
- ▶ `[1] 2.302585`
- ▶ `> round(3.8)`
- ▶ `[1] 4`
- ▶ `> floor(3.8)`
- ▶ `[1] 3`



R as a Fancy Calculator, cont'd

- `> pi`
 - `[1] 3.141593`
- `> 1:10`
 - `[1] 1 2 3 4 5 6 7 8 9 10`
- `> rep(0,10)`
 - `[1] 0 0 0 0 0 0 0 0 0 0`
- `> seq(0,10,2)`
 - `[1] 0 2 4 6 8 10`
- `> 1+3:10`
 - `[1] 4 5 6 7 8 9 10 11`



R working with Vectors

- > `x<-c(3,4,2,3,5,6)`
 - Note that the '`<-`' combination is used as a “gets” command
- > `y<-c(5,7,3,3,4,7)`
- > `mean(x)`
- [1] 3.833333
- > `var(y)`
- [1] 3.366667
- > `median(x)`
- [1] 3.5
- > `sd(y)`
- [1] 1.834848
- > `shapiro.test(x)`
- Shapiro-Wilk normality test
- data: x
- $W = 0.958$, $p\text{-value} = 0.8043$



R working with Vectors, cont'd

```
- > x<-rnorm(10,mean=5, sd=2)
- > x
- [1] 5.778666 2.143754 7.489185 5.508185 2.708508
  9.040165 3.875893 3.246373 6.384725 3.515293
- > sum(x)
- [1] 49.69075
- > sum(x[1:3])
- [1] 15.41161
- > sort(x)
- [1] 2.143754 2.708508 3.246373 3.515293 3.875893
  5.508185 5.778666 6.384725 7.489185 9.040165
- > rev(sort(x))[1:2]
- [1] 9.040165 7.489185
- > length(x)
- [1] 10
```



R working with Vectors, cont'd

```
- > vec<-3:12
- > vec
- [1] 3 4 5 6 7 8 9 10 11 12
- > vec[5]
- [1] 7
- > vec[-5]
- [1] 3 4 5 6 8 9 10 11 12
- > vec<7
- [1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE
    FALSE FALSE FALSE
- > any(vec>12)
- [1] FALSE
- > all(vec>2)
- [1] TRUE
```



R working with Vectors, cont'd

```
- > x<-rchisq(10,3)
- > x
- [1] 0.6526930 1.0386719 0.3574053 10.2467790
    1.5565259 0.7233155 3.8427101 3.1414846
    0.3029174 1.8367612
- > rx<-rank(x)
- > rx
- [1] 3 5 2 10 6 4 9 8 1 7
- > sample(x,3)
- [1] 1.0386719 0.7233155 0.3574053
- > sample(x,replace=T)
- [1] 0.3029174 0.7233155 1.8367612 3.8427101
    1.8367612 3.1414846 3.8427101 3.8427101 1.0386719
    1.0386719
```



R working with Matrices

```
- > x<-matrix(c(1,4,5,2,4,5,6,2),nrow=2)
```

```
- > x
```

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

```
- [1,]  1  5  4  6
```

```
- [2,]  4  2  5  2
```

```
- > mean(x[,3])
```

```
- [1] 4.5
```

```
- > rowSums(x)
```

```
- [1] 16 13
```

```
- > x<-rbind(x,apply(x,2,mean))
```

```
- > x
```

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

```
- [1,] 1.0 5.0 4.0  6
```

```
- [2,] 4.0 2.0 5.0  2
```

```
- [3,] 2.5 3.5 4.5  4
```



R working with Data Sets

- You can enter your data directly into R if you wish

```
- > x<-c(1,1,1,1,2,2,2,2)
- > y<-c(3,4,5,2,6,6,5,4)
- > data<-cbind(x,y)
- > data<-data.frame(data)
- > data
  x y
1 1 3
2 1 4
3 1 5
4 1 2
5 2 6
6 2 6
7 2 5
8 2 4
```



R working with Data Sets, cont'd

- However, it is usually not necessary to bind columns, or even create the dataset (unless you want to save it)
 - ▶ `> iv<-c(1,1,1,1,2,2,2,2)`
 - ▶ `> dv<-rnorm(8)`
 - ▶ `> dv`
 - `[1] 1.4671261 -1.0878009 -1.0487529 -0.5214934`
`1.9040232 0.5586128 2.8512241`
 - `[8] -1.0348406`
 - ▶ `> dv[iv==1]`
 - `[1] 1.4671261 -1.0878009 -1.0487529 -0.5214934`
 - ▶ `> mean(dv[iv==1])`
 - `[1] -0.2977303`



R Working with Data Sets, Cont'd

The Problem of “=” and “==”

- `> x<-c(2,5,4,2,8,6)`
- `> y<-c(1,1,1,2,2,2)`
 - ▶ `> x[y=1]`
– [1] 2
 - ▶ `> x[y==1]`
– [1] 2 5 4
 - ▶ `> x[y>1]`
– [1] 2 8 6
- Note that `x[y==1]` represents ‘all x where y is equal to 1’



R working with Data Sets cont'd

- You can also read in data files from other statistical software packages
- To read in a data set from SPSS there are multiple options
 - Use the 'read.spss' command (not recommended as it is very fussy with variable names, formats,...)
 - Convert the SPSS file to a stata file (.dta) and then read it into R with the `read.dta(file=" ")` command (which is part of the 'foreign' package)(recommended)
 - Convert the SPSS file to a comma separated file (.csv) and then open the file using `read.csv(file=" ")`
 - Convert the SPSS file to a tab delimited file (.) and then open the file using `read.table(file=" ", header=T)`



R working with Data Sets, cont'd

- Opening files from other statistical software packages works the same way
- To browse your directories for a file use the 'file.choose()' option (with a read statement)
 - ▶ `newdata<-read.table(file.choose(), header=T)`
- Also note that if you are specifying the exact file name, that the slashes are backward to Windows
 - ▶ `> c<-read.table(file="C:/My Documents/Robs Work/SCS/R Course/testdata2.dat", header=T)`



R working with Data Sets, cont'd

- Once you have a data set active in R you can perform commands on that data
 - `> c<-read.table(file="C:/My Documents/Robs Work/SCS/R Course/testdata2.dat", header=T)`
 - `> names(c)`
 - `[1] "commsize" "sex" "recycle"`
 - `> attach(c)`
 - `> summary(c)`
 - | commsize | sex | recycle |
|-------------|-------------|----------------|
| Min. :1.0 | Min. :1.0 | Min. : 5.655 |
| 1st Qu.:1.0 | 1st Qu.:1.0 | 1st Qu.:16.432 |
| Median :2.0 | Median :1.5 | Median :19.878 |
| Mean :2.5 | Mean :1.5 | Mean :19.240 |
| 3rd Qu.:4.0 | 3rd Qu.:2.0 | 3rd Qu.:22.670 |
| Max. :4.0 | Max. :2.0 | Max. :29.039 |



R working with Data Sets, cont'd

```
- > mean(c$recycle[c$sex==1 & c$commsize==2])
- [1] 20.89839
- > var(recycle[sex==2 & commsize==4])
- [1] 25.05803
- > tapply(recycle,list(commsize,sex),mean)
-      1      2
- 1 22.11109 22.34478
- 2 20.89839 20.39652
- 3 17.80242 18.89766
- 4 17.01439 14.85440
- > tapply(recycle,list(commsize,sex),var)
-      1      2
- 1 5.819652 11.093372
- 2 8.430797 9.129323
- 3 13.697746 23.506278
- 4 36.760121 25.058034
```



R working with Data Sets, cont'd

- Let's say we wanted to compare the variances of the 8 cells ... this is one unsophisticated way:
 - ▶ `comsex<-interaction(commsize,sex)`
 - `> comsex`
 - `[1] 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.2 1.2 1.2`
`1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2`
`2.1 2.1 2.1`
 - Levels: 1.1 2.1 3.1 4.1 1.2 2.2 3.2 4.2
 - `> bartlett.test(recycle,comsex)`
 - ▶ Bartlett test of homogeneity of variances
 - data: recycle and comsex
 - Bartlett's K-squared = 19.9885, df = 7, p-value = 0.005595



R working with Data Sets, cont'd

- An even better variance equality test:
 - > `install.packages("lawstat")`
 - > `library(lawstat)`
 - > `levene.test(recycle,comsex)`
 - > Classical Levene's test based on the absolute deviations from the mean
 - data: recycle
 - Test Statistic = 3.1136, p-value = 0.004664

 - > `levene.test(c$recycle,c$comsex, option="trim.mean")`
 - Modified Robust Levene-type test based on the absolute deviations from the trimmed mean
 - data: c\$recycle
 - Test Statistic = 2.588, p-value = 0.01596



R working with Data Sets, cont'd

- What if wanted to verify that the distributions in each cell are normal in shape
 - `> shapiro.test(recycle[comsex==1])`
 - Shapiro-Wilk normality test
 - data: `c$recycle[c$comsex == 1]`
 - $W = 0.9671$, $p\text{-value} = 0.8571$

 - `> shapiro.test(c$recycle[c$comsex==2])`
 - Shapiro-Wilk normality test
 - data: `c$recycle[c$comsex == 2]`
 - $W = 0.9874$, $p\text{-value} = 0.9913$



Writing Functions

- One of the main advantages of R is its flexibility
- For example, R makes it very easy to write your own functions
 - Here is a function to take the mean of a set of observations
 - ```
>robsmean<-function (x) {
 result<-sum(x)/length(x)
 return(result)
}
```
  - ▶ 

```
>robsmean(c(3,2,4))
```

    - [1] 3

