Getting R Up and Running

Introduction

We will use R and RStudio for computing. R is an open source statistical language that is increasingly used in the social and physical sciences. RStudio is a "front end" (also called an "IDE") to R that simplifies important aspects, like file management, the installation of new routines in R, and the display of help files. Both R and RStudio run identically under the Mac OSX, Microsoft Windows, and Linux. It will not run on a tablet such as an iPad.

Both R and RStudio are freely available from the web, so if you have access to a laptop or a desktop, you may find it most convenient to download and install the software on those machines. If you do not have access to personal computing, please contact the instructional staff for other options. This document shows you how to download and install R and Rstudio for Windows and for Mac, and gives an overview of what RStudio is.

Getting Started with R

What is R?

The official R project web page describes R as a ' language and environment for statistical computing and graphics'. It can be daunting if you haven't done any programming before but it is worth taking some time to familiarise yourself with the R environment as once you have grasped some of the basics it can be a very useful tool. A wide variety of statistical functions come with the default install and there are many other packages that can be installed if required.

It is very quick and easy to produce graphs with default parameters for a quick view of your data and there are all manner of parameters that can be specified to customise your graphs. R is often used to perform analysis and produce graphs for publication.

Good things about R

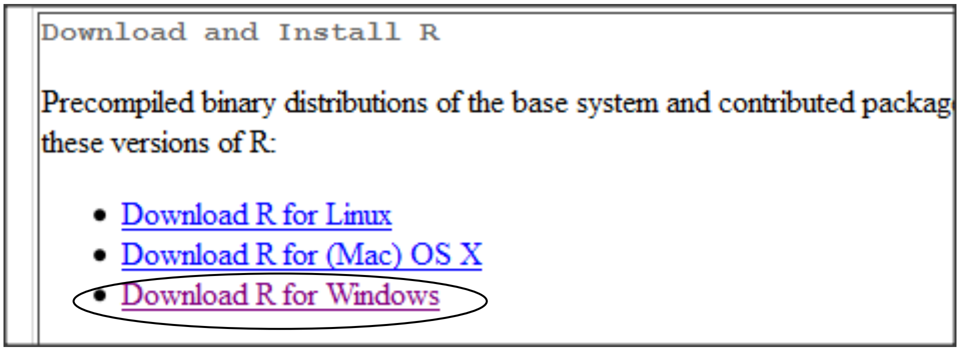
- It's free and handle sophisticated statistical analyses
- It works on all platforms
- It can deal with much larger datasets than Excel for example
- Graphs can be produced to your own specification

Installing R and RStudio

Instructions for downloading and installing R can be found on the R project website http://www.r-project.org/. Versions are available for Windows, Linux and Mac. We also give some detailed instructions below.

Install R and RStudio in Windows

1) Go to http://cran.us.r-project.org/
2) Click on Download R for Windows

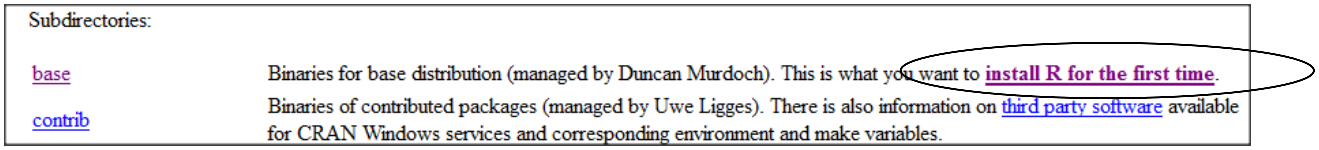


Download and Install R

Precompiled binary distributions of the base system and contributed packag
these versions of R:

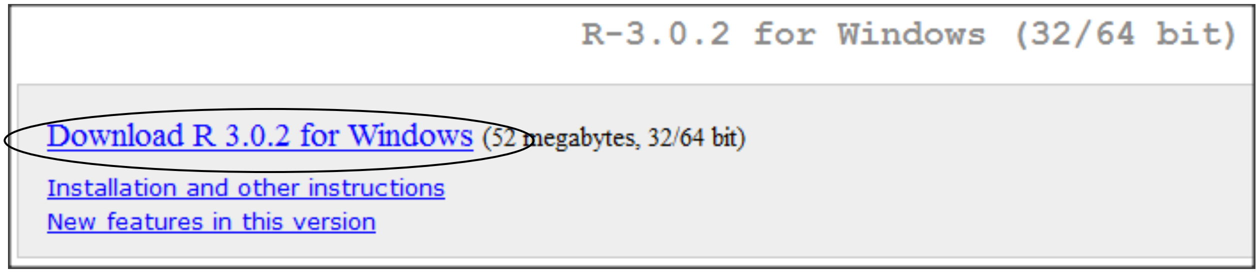
- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

3) Click on "install R for the first time"



Subdirectories:	
base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.

4) Click on "Download R 3.0.2 for Windows"



R-3.0.2 for Windows (32/64 bit)

Download R 3.0.2 for Windows (52 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

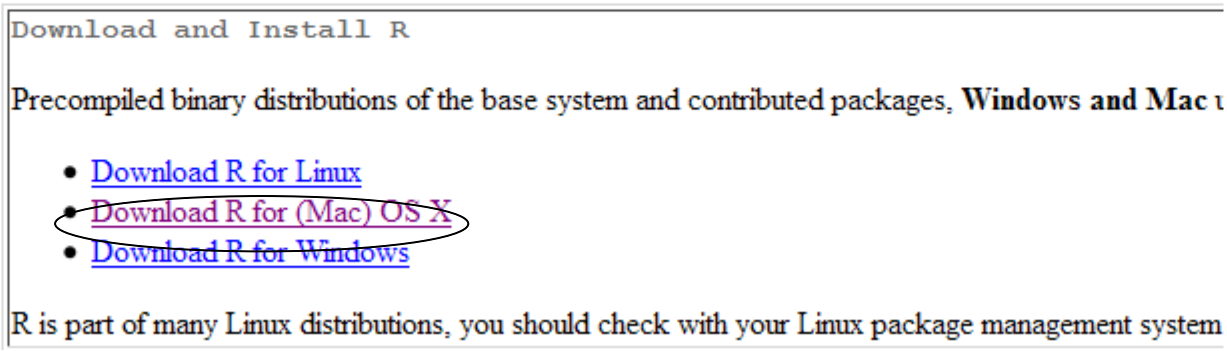
5) Run this executable file. When prompted, leave all default settings in the installation options.

6) Now that R is installed, it is time to install Rstudio

7) Download RStudio from http://rstudio.org/download/desktop and install it. Leave all default settings in the installation options.

Install R and RStudio in Mac OS X

1) Go to http://cran.us.r-project.org/
2) Click on Download R for (Mac) OS X



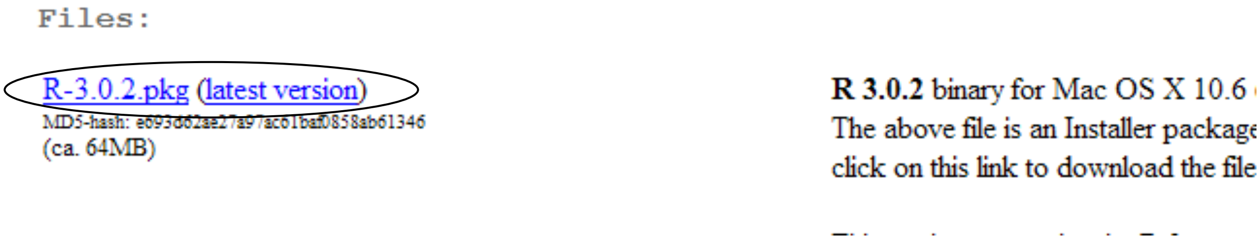
Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac**

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system

3) Click on R-3.0.2.pkg (latest version) (the version number might have changed since we wrote this).



Files:

R-3.0.2.pkg (latest version)
MD5-hash: e693d62ae27a97ac61baf0858ab61346
(ca. 64MB)

R 3.0.2 binary for Mac OS X 10.6
The above file is an Installer package
click on this link to download the file

4) Now that R is installed, it is time to install RStudio

5) Download RStudio from http://rstudio.org/download/desktop and install it. Leave all default settings in the installation options. Install RStudio by dragging the application icon to your Applications folder.

Getting familiar with the R console

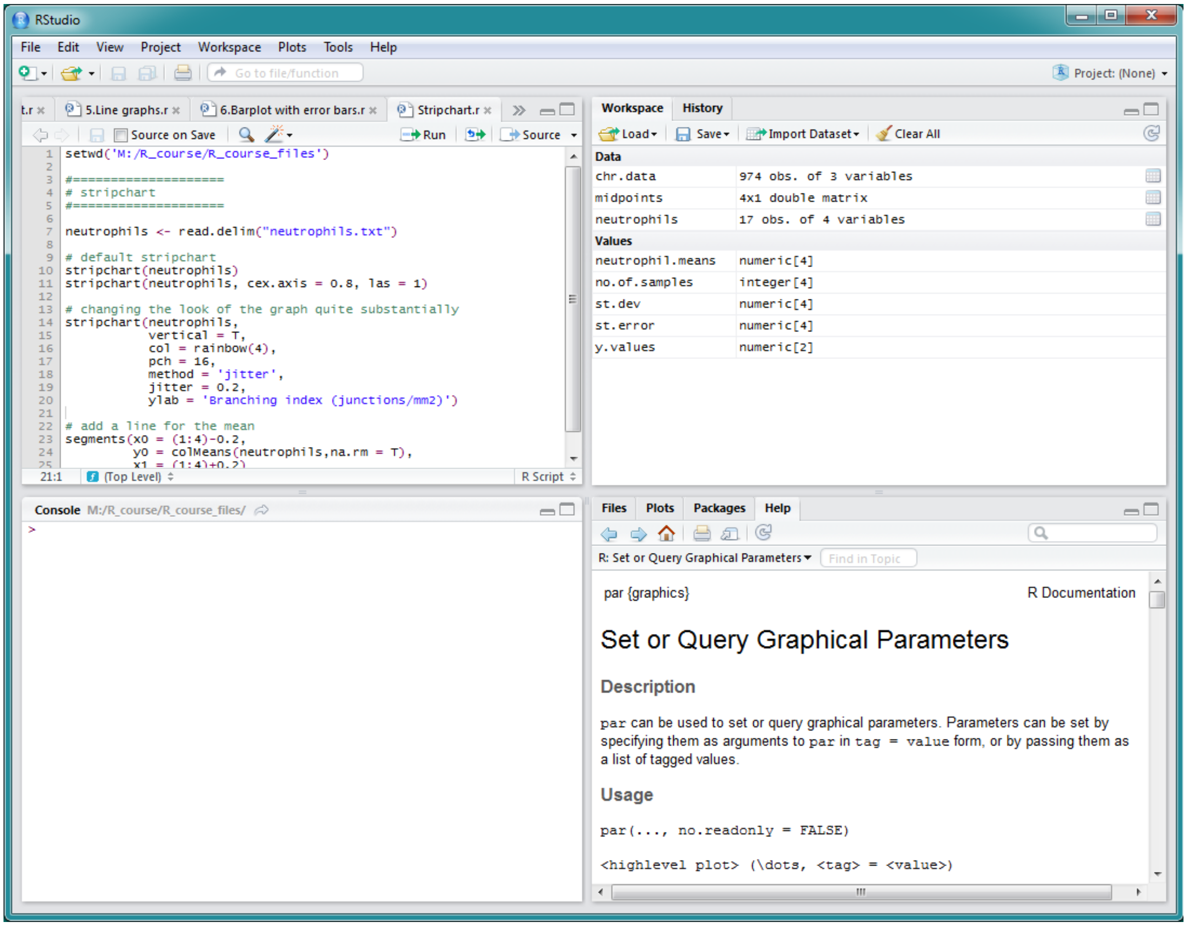
R is a command line environment, this means that you type in an instruction and R interprets this and either stores the result or writes it back on the screen for you. You can run R in a very simple command shell window, but for all practical purposes it's much easier to use a dedicated piece of software that makes it easy to work within the R environment.

There are several different R IDEs (integrated development environments) around, but by far the most common one is RStudio and this is what we're going to be using for this course. Let's take a look at what it offers.

Open RStudio. The default layout is shown in the image below. You should see the following elements:

- Top left panel is the text editor. Commands can be sent into the console from here.
- Bottom left is the R console. You can type directly into here.
- Top right is the workspace and history. History keeps a record of the last commands entered, this is searchable. The workspace tab shows all the R objects (variables and data frames).
- Bottom right is where graphs are plotted and help topics are shown.

The panels can be rearranged if desired.



```
setwd('M:/R_course/R_course_files')

#====================
# stripchart
#====================

neutrophils <- read.delim("neutrophils.txt")

# default stripchart
stripchart(neutrophils)
stripchart(neutrophils, cex.axis = 0.8, las = 1)

# changing the look of the graph quite substantially
stripchart(neutrophils,
           vertical = T,
           col = rainbow(4),
           pch = 16,
           method = 'jitter',
           jitter = 0.2,
           ylab = 'Branching index (junctions/mm2)')

# add a line for the mean
segments(x0 = (1:4)-0.2,
         y0 = colMeans(neutrophils,na.rm = T),
         x1 = (1:4)+0.2)
```

Data	
chr.data	974 obs. of 3 variables
midpoints	4x1 double matrix
neutrophils	17 obs. of 4 variables
Values	
neutrophil.means	numeric[4]
no.of.samples	integer[4]
st.dev	numeric[4]
st.error	numeric[4]
y.values	numeric[2]

```
>
```

par {graphics} R Documentation

Set or Query Graphical Parameters

Description

par can be used to set or query graphical parameters. Parameters can be set by specifying them as arguments to par in tag = value form, or by passing them as a list of tagged values.

Usage

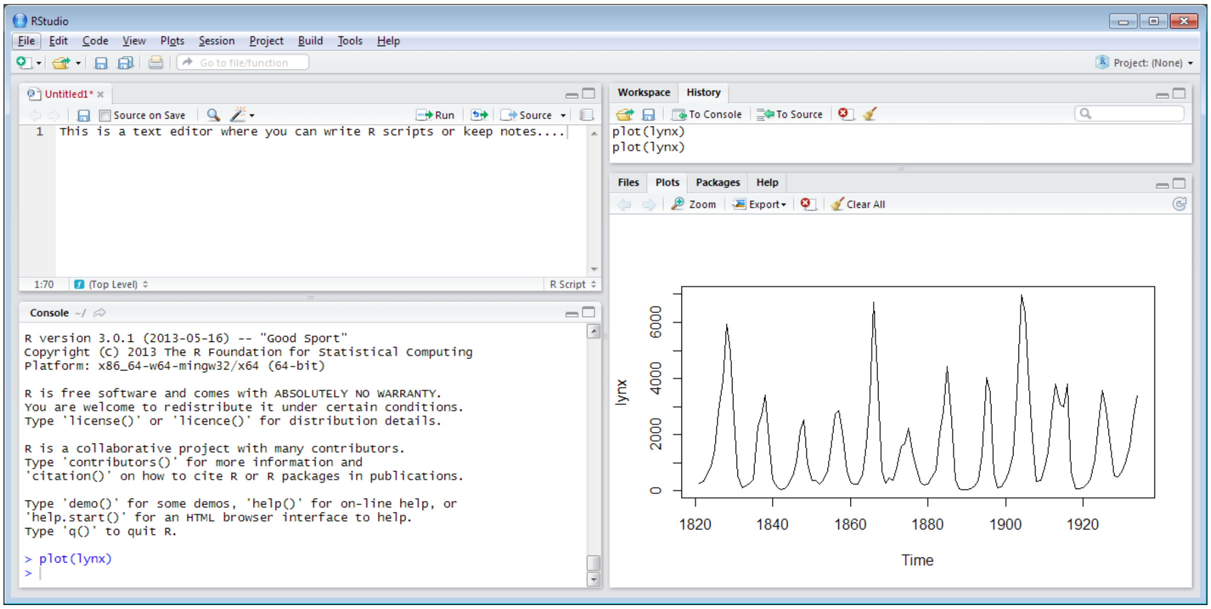
```
par(..., no.readonly = FALSE)

<highlevel plot> (\dots, <tag> = <value>)
```

Lots of data can be imported, manipulated and saved in an R session all at once, and though it won't always be visible on the screen, there are various ways of viewing and manipulating it. Note the top-right window of the above. This lists different collections of data currently loaded into R. This collection of things constitutes your "workspace," consisting of all the data and stored results you are currently working with. This window is a good way to browse what you have. You can import data using the "Import Dataset" command as well. "Clear all" will get rid of everything currently loaded. You can also save and load entire workspaces, although we recommend instead writing scripts, discussed below, that you re-run when you want to start working on a project again.

The actual R session is the console window at the bottom left of the R-studio window. This is where the actual computation is done in R. All of the other windows are either keeping records of what you've done or showing you the result of previous commands.

When you type a command into the console it is evaluated by the R interpreter. Think of an interpreter as an entity that takes your command, follows it, and gives you back some result. If the result of this is a value (e.g., number, sequence of numbers, bit of text) then it will be printed in the console.

If it is a plot, it will show up on the plot window, as shown in the following screenshot.



```
This is a text editor where you can write R scripts or keep notes....
```

```
plot(lynx)
plot(lynx)
```

```
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-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.

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.

> plot(lynx)
>
```

R Console, R Editor and R Graphics device open within the RStudio IDE

The above screenshot also shows a text editor in which you can write anything you like including notes, though it would generally be used to create a script (a script is a file with a bunch of R commands. Stata "do" files are scripts). You can make a new document by selecting File => New => RScript from the menu bar.

Importantly, as well as issuing commands one at a time by typing directly into the console, you can also send commands to the R Console from the R Editor by selecting a command or a line of text and selecting Ctrl + Enter, or by copying and pasting. You can have R run all the commands in your window by clicking on the “Source” button. This is a great way to document a specific analysis in a file, and then replicate it as needed whenever you want.

Using files to store your R commands allows you to recall how you did things in the past. Most R coding is basically copying and adapting old code. So start generating a library of old code you understand.

In the console you can scroll through previous command that have been entered by using the up arrow (↑) on the keyboard.

As already mentioned, to tell R what you want it to do you must enter commands into the console. The text editor can be ignored for the moment and we'll type directly into the console.

The `>` symbol shows that R is ready for something to be entered. The console can work just like a calculator. Type `8+3` and press return. It doesn't matter whether there are spaces between the values or not.

```
> 8 + 3
[1] 11
```

The answer is printed in the console as above. We'll come on to what the `[1]` means at the end of this section.

```
> 27 / 5
[1] 5.4
```

These calculations have just produced output in the console - no values have been saved.

To save a value, it can be assigned to a name. `‟=″` is generally used as the assignment operator, though ‘’<-‘’ is sometimes used as well. For now we'll use x, y and z as names of variables (i.e., named objects that hold information), though more informative names can be used as discussed later in this section.

```
> x = 8 + 3
```

If R has performed the command successfully you will not see any output, as the value of 8 + 3 has been saved to the variable called `x`. You can access and use this variable at any time and can print the value of `x` into the console.

```
> x
[1] 11
```

Create another variable called `y`.

```
> y = 3
```

Now that values have been assigned to `x` and `y` they can be used in calculations.

```
> x + y
[1] 14
> x * y
[1] 33

> z = x * y
> z
[1] 33
```

Warning: R is case sensitive so x and X are not the same. If you try to print the value of X out into the console an error will be returned as X has not been used so far in this session.

```
> X
Error: object 'X' not found
```

To check what variables you have created, enter `ls()` or `objects()` into the console or look at the 'workspace' tab in RStudio

If you use the same variable name as one that you have previously used then R will overwrite the previous information with the new information.

```
> y = 3
> y
[1] 3

> y = 12
> y
[1] 12
```

See other documents for more extensive exploration of using R.

R help

R has comprehensive help pages that are very useful once you have familiarised yourself with the layout. Information about a function (for example `read.table`) can be accessed by typing the following into the console:

```
help(read.table)
```

or

```
?read.table
```

This should include information about parameters that can be passed to the function, and at the bottom of the page should be examples that you can run which can be very useful.

If you don't know the function name that you're after, e.g. for finding out the standard deviation, try

```
help.search("deviation")
```

or

```
??deviation
```

The main thing these pages tell you is what you should give to a function you are calling. These things (called “parameters”) can radically change how a function operates.

The course’s Piazza forum is an excellent place to turn to if you get stuck. But when asking for help, please follow the following posting guidelines:

You can always try searching the internet but remember that 'R' in a general search isn't always very good at returning relevant information, so try and include as much information as possible. You can also try http://www.rseek.org/, which will return more R specific information. But **be warned**, the information on the Internet can be confusing and cryptic, so when first learning R you might be better off sticking to local resources.

Acknowledgements

This document is a mildly edited version of a document generated by David Harrington and his TF staff for Stat 102 in Spring, 2015. Edits by Luke Miratrix and his TF staff for S-043/Stat-151 in Fall, 2015.