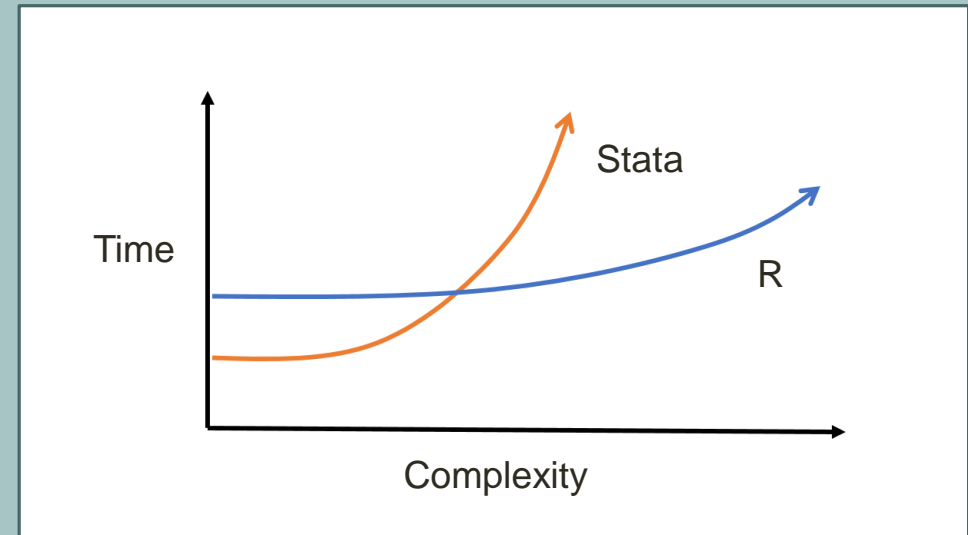


# *INTRO TO R*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)



# *INTRODUCTION*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

## > WHAT IS R?

- R is a statistical programming language (based on S)
- R is open source - researchers develop packages to implement new statistical methods, plots or applications
- R runs on Windows, MacOS and UNIX



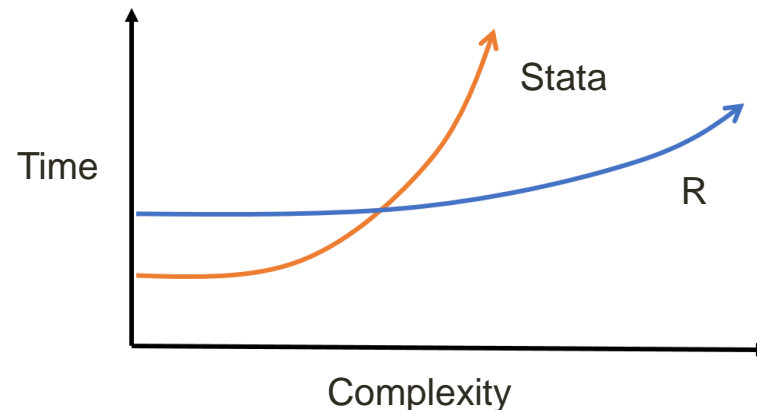
John Chambers



Dirk Eddelbuettel

## > WHY USE R?

- R is free!
- R is flexible
- R is good at handling large datasets and multiple objects
- R has good plotting tools and packages for statistical analysis

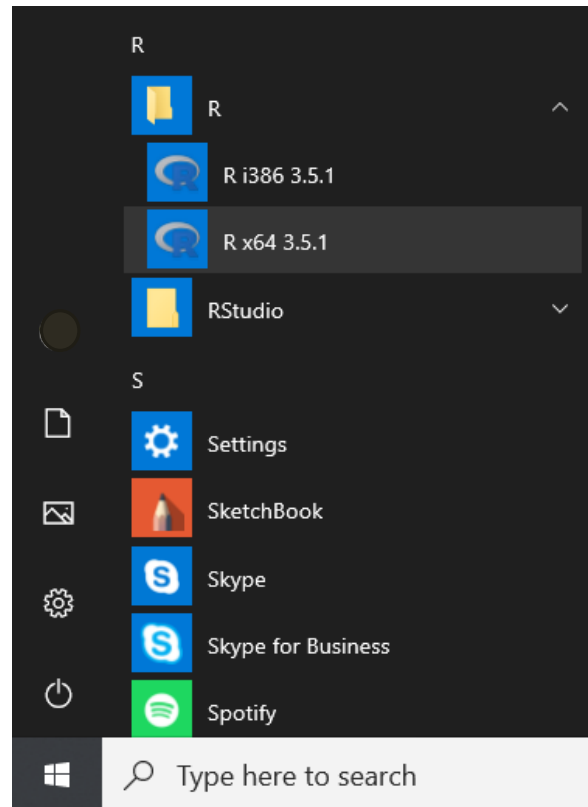


## > DOWNLOADING R

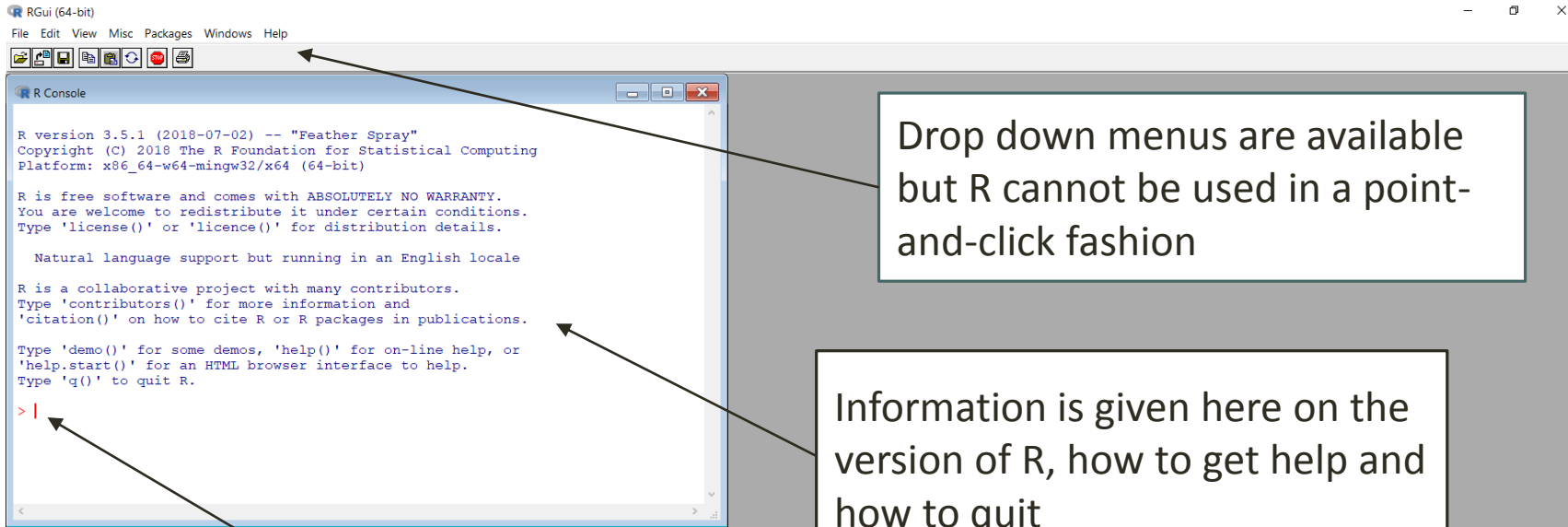
- Visit <https://www.r-project.org> and click “Download R”
- Choose your nearest CRAN mirror (such as <http://www.stats.bris.ac.uk/R>)
- Choose “Download R for [Windows/Mac/Linux]”
- Choose “base” for Windows and click on “Download R 3.5.2 for Windows”
- Choose “R-3.5.2.pkg” for Mac
- Once the .exe (Windows) or .pkg (Mac) have downloaded, run and install

# > OPENING R

- Start → All Programs → R → R x64 3.5.2.R



# > R



The screenshot shows the RGui (64-bit) window. The title bar reads "RGui (64-bit)". The menu bar includes "File", "Edit", "View", "Misc", "Packages", "Windows", and "Help". Below the menu bar is a toolbar with icons for file operations and running code. The main window is the "R Console", which displays the following text:

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 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.

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.

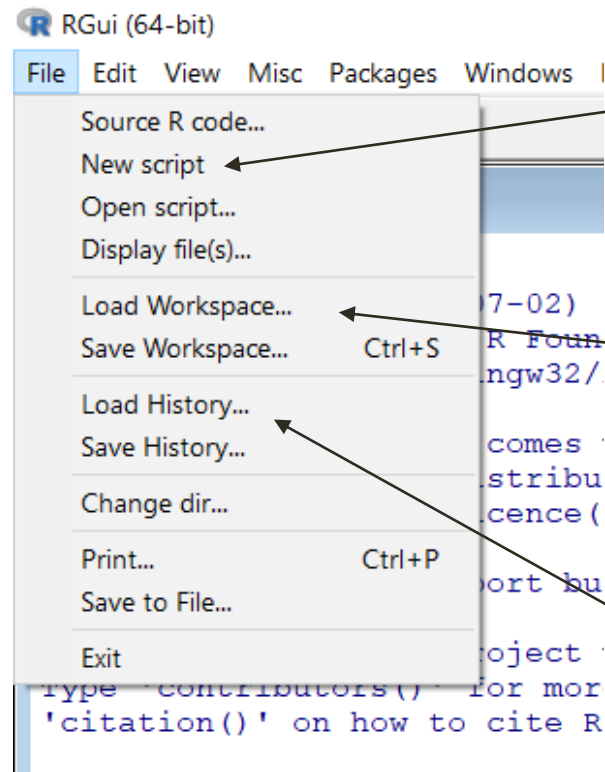
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Three callout boxes with arrows point to specific parts of the interface:

- Drop down menus are available but R cannot be used in a point-and-click fashion**: An arrow points to the menu bar.
- Information is given here on the version of R, how to get help and how to quit**: An arrow points to the text in the console window.
- This is the R command line where we can tell R what to do**: An arrow points to the prompt "> |" in the console window.

# > R MENUS



A script is a text editor for your code (like a do file in Stata)

A workspace includes all the objects which are in R's memory at a given time. These can include data or results. To see what is in the workspace type `ls()`

The History includes all the commands and results which are displayed during an R session



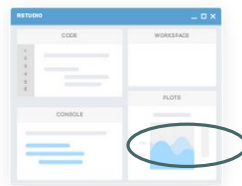
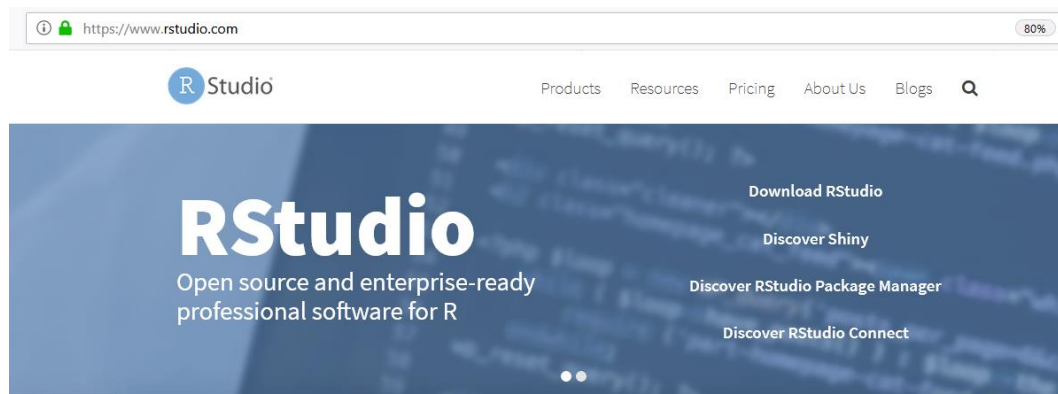
## > RSTUDIO

- RStudio makes R easier to use
- It includes a code editor, debugging & visualization tools
- <https://www.rstudio.com>



# > DOWNLOADING RSTUDIO

- <https://www.rstudio.com>



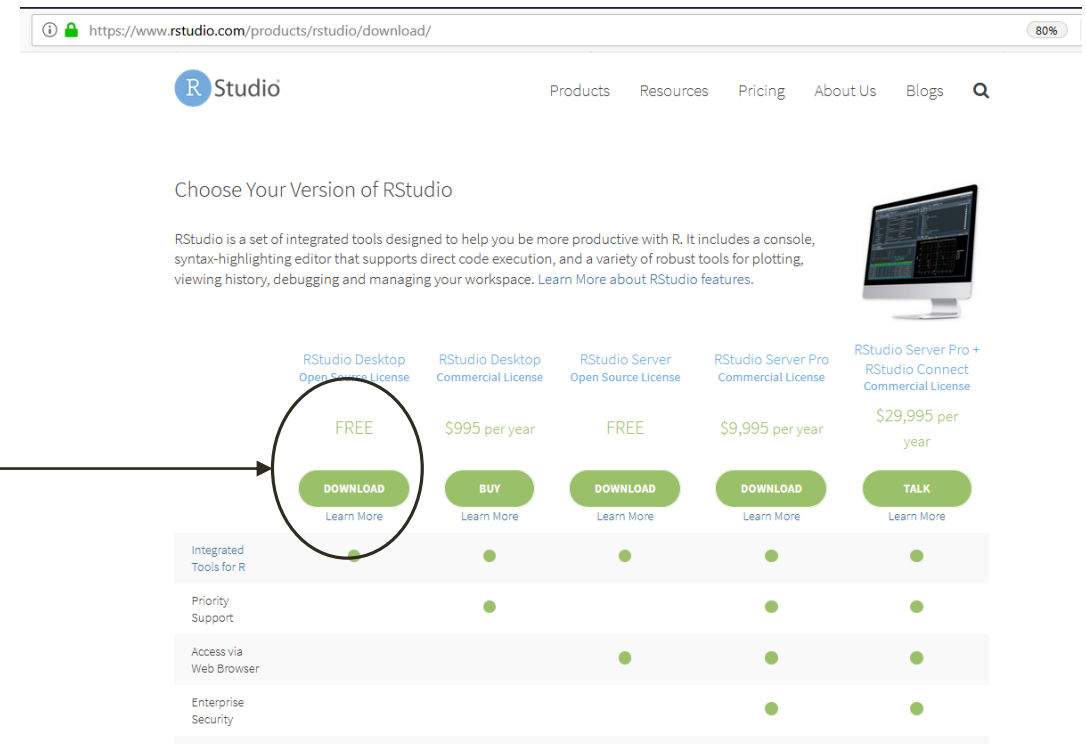
RStudio



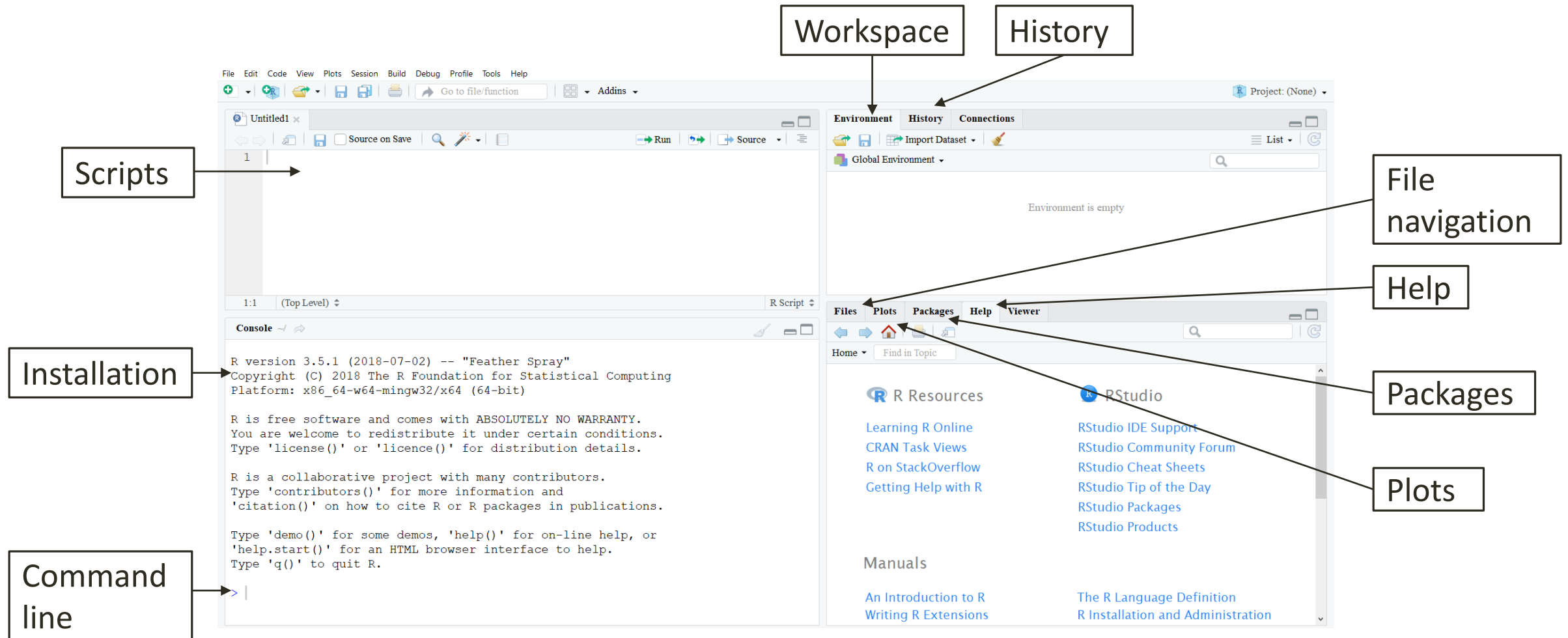
Shiny



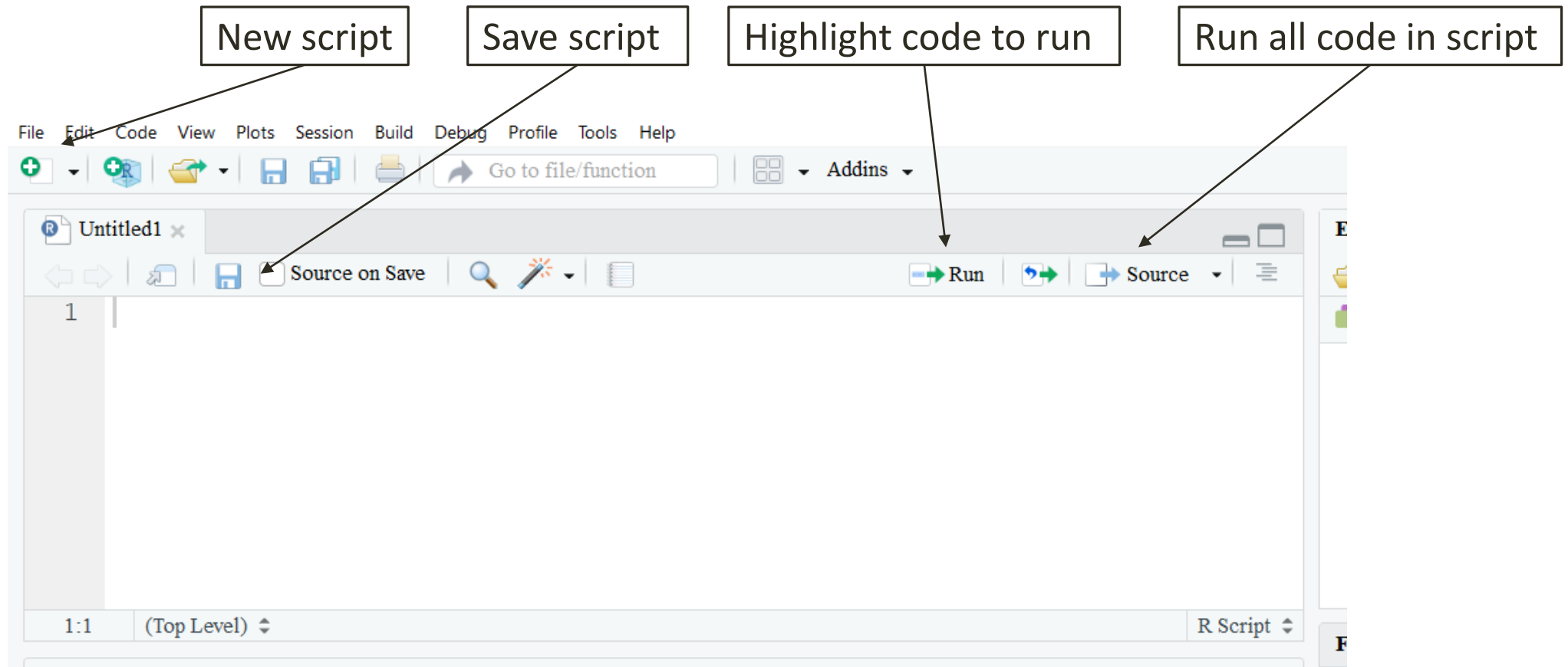
R Packages

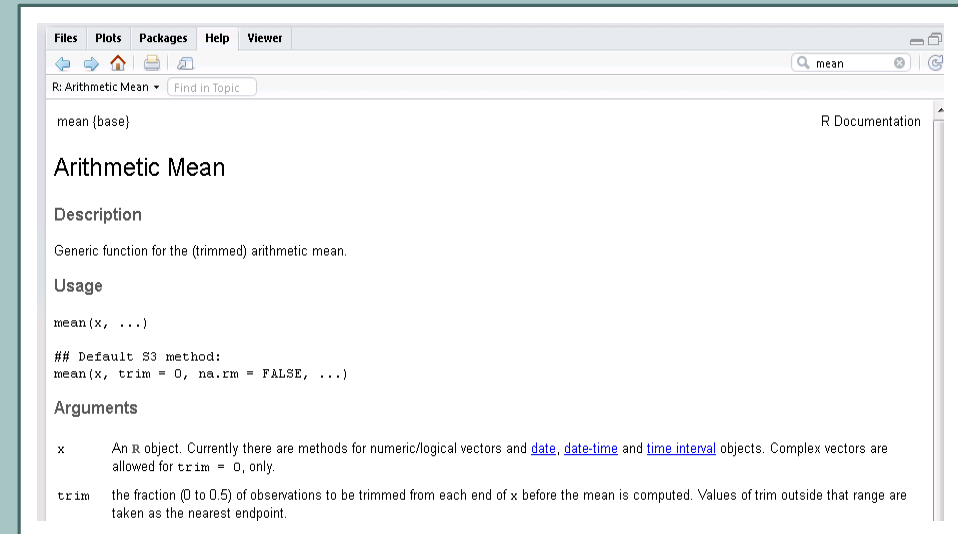


# > USING RSTUDIO



# > R SCRIPTS





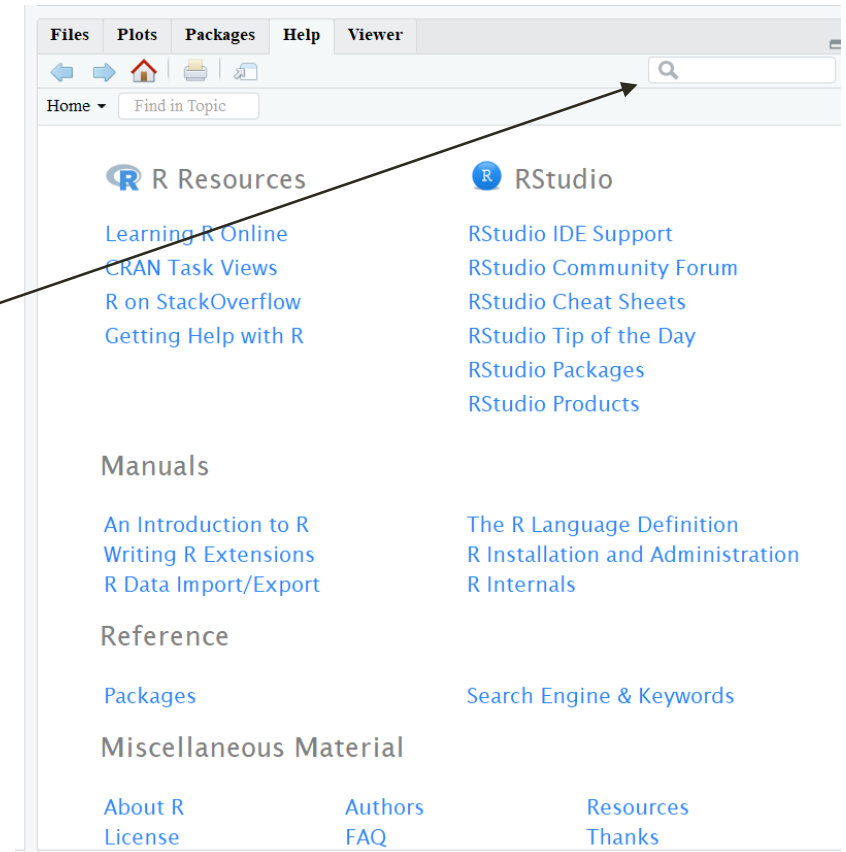
*GETTING HELP*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

# > HELP

- Use the command *help(topic)* or *?topic*
- In R this will open a web browser
- In RStudio this will open the Help tab  
can also use the search bar)



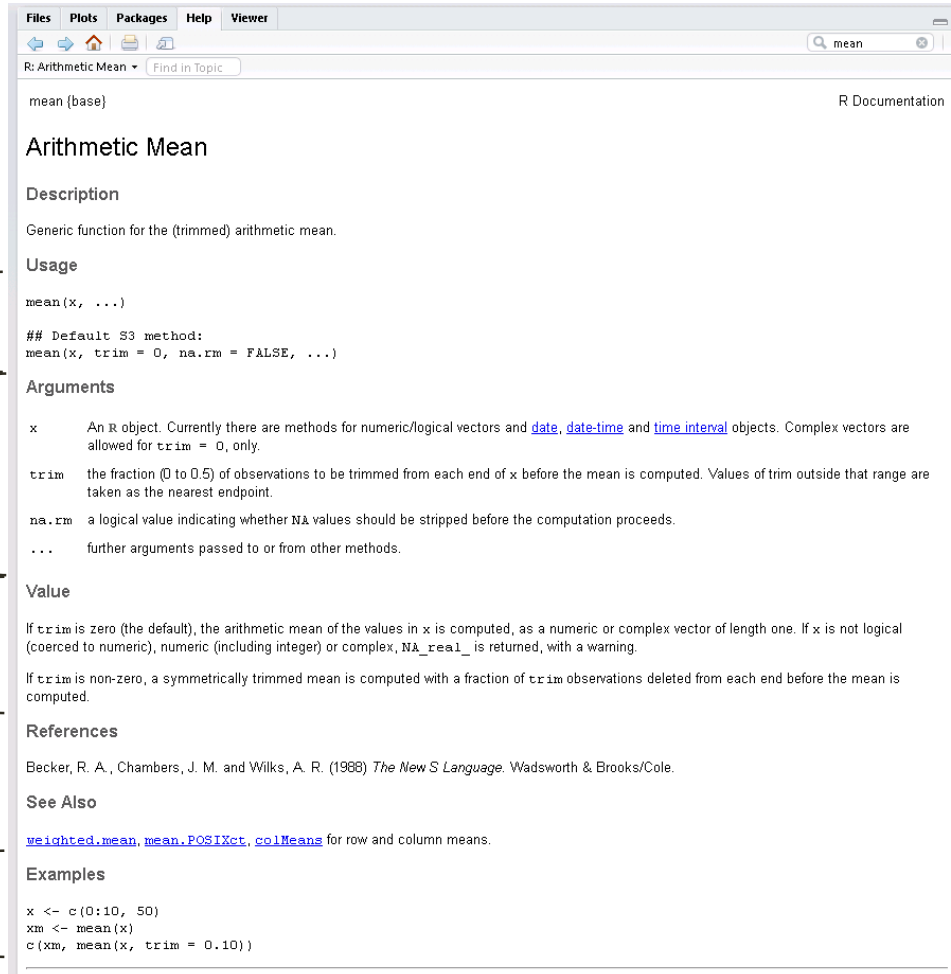
# > HELP(MEAN)

Input and options  
of *mean*

Arguments of  
*mean*

The output of  
*mean*

Examples



The screenshot shows the R Documentation page for the `mean` function. The page is titled "Arithmetic Mean" and includes sections for Description, Usage, Arguments, Value, References, See Also, and Examples. The `mean` function is described as a generic function for the (trimmed) arithmetic mean. The Usage section shows the function signature `mean(x, ...)` and the default S3 method `mean(x, trim = 0, na.rm = FALSE, ...)`. The Arguments section lists the parameters: `x` (An R object), `trim` (the fraction of observations to be trimmed), `na.rm` (a logical value indicating whether NA values should be stripped), and `...` (further arguments). The Value section explains that if `trim` is zero, the arithmetic mean is computed, and if `trim` is non-zero, a symmetrically trimmed mean is computed. The References section cites Becker, R. A., Chambers, J. M., and Wilks, A. R. (1988) *The New S Language*. The See Also section lists `weighted.mean`, `mean.POSIXct`, and `colMeans`. The Examples section shows the following code: 

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

mean (base)

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)
```

## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)

### Arguments

**x** An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

**trim** the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

**na.rm** a logical value indicating whether NA values should be stripped before the computation proceeds.

**...** further arguments passed to or from other methods.

### Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

### Examples

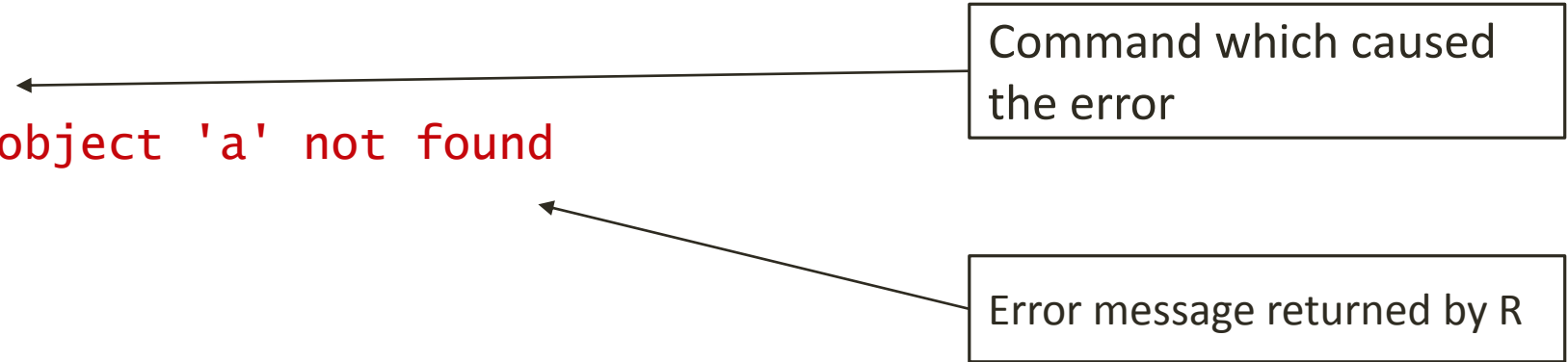
```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

## > ERROR MESSAGES

- R provides informative errors

```
> 10*a  
Error: object 'a' not found
```

Command which caused  
the error



Error message returned by R



# > ONLINE HELP

- R website: [www.r-project.org](https://www.r-project.org)

Directs you towards user forums for R

Directs you towards ways to get help in R

Several introductory manuals are available

List of FAQs

The screenshot shows the R Project website with several annotations. On the left, four boxes point to specific links in the navigation menu: 'Directs you towards user forums for R' points to 'Search'; 'Directs you towards ways to get help in R' points to 'Get Involved: Mailing Lists'; 'Several introductory manuals are available' points to 'Manuals'; and 'List of FAQs' points to 'FAQs'. The website content includes the R logo, a navigation menu with links like 'Download', 'R Project', 'R Foundation', 'Help With R', and 'Documentation', and a main section titled 'The R Project for Statistical Computing' with sub-sections 'Getting Started' and 'News'.

https://www.r-project.org

**The R Project for Statistical Computing**

### Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

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.

### News

- [R version 3.5.2 \(Eggshell Igloo\)](#) has been released on 2018-12-20.
- The R Foundation Conference Committee has released a [call for proposals](#) to host useR! 2020 in North America.
- You can now support the R Foundation with a renewable subscription as a [supporting member](#)
- The R Foundation has been awarded the Personality/Organization of the year 2018 award by the professional association of German market and social researchers.

### News via Twitter

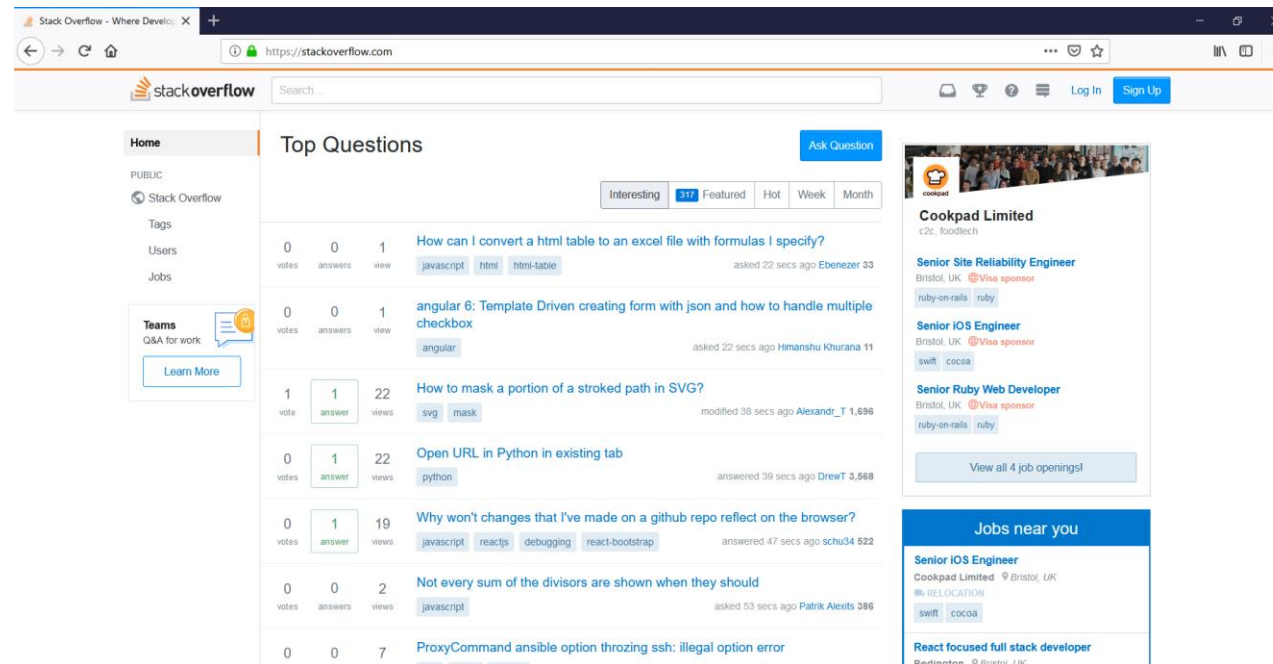
**The R Foundation**  
@\_R\_Foundation  
We welcome [@gdequeiroz](#), [@edzerpebesma](#) and [@henrikbengtsson](#), elected as ordinary members of the R Foundation in recognition of their services to the R community.  
Oct 26, 2018

The R Foundation Retweeted

**useR! 2019**  
@UseR2019\_Conf  
17-12-18 ;  
01-19 ;  
15-02-19 ;

# > FORUMS

- Mailing list archive and forum <http://r.789695.n4.nabble.com>
- Stack Overflow <https://stackoverflow.com>
- Google!





```
> x <- 11:20
> x [1] 11 12 13 14 15 16 17 18 19 20
>
>
> x[3]
[1] 13
>
>
> x[-2]
[1] 11 13 14 15 16 17 18 19 20
```

*BASICS*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

## > BRACKETS

- `()` are used by R functions and contain the options or arguments required to run that function
- `[]` are used to pick out elements, rows or columns of R objects such as matrices and vectors
- `{}` are used to hold a series of commands, e.g. within a user written function or a loop

# > VECTORS

```
> x <- 11:20
```

```
> x
```

```
[1] 11 12 13 14 15 16 17 18 19 20
```

```
>
```

```
>
```

```
> x[3]
```

```
[1] 13
```

```
>
```

```
>
```

```
> x[-2]
```

```
[1] 11 13 14 15 16 17 18 19 20
```

```
>
```

```
>
```

```
> x[1:3]
```

```
[1] 11 12 13
```

The 3<sup>rd</sup> element of x

All except the 2<sup>nd</sup> element of x

The first 3 elements of x

## > VECTOR ELEMENTS

```
> x[c(1, 5, 7)]  
[1] 11 15 17
```

The 1<sup>st</sup>, 5<sup>th</sup> and 7<sup>th</sup> elements of x

```
>
```

```
>
```

```
> x[-c(1, 5, 7)]  
[1] 12 13 14 16 18 19 20
```

All except the 1<sup>st</sup>, 5<sup>th</sup> and 7<sup>th</sup> elements of x

```
>
```

```
>
```

```
> x[x>13]  
[1] 14 15 16 17 18 19 20
```

Those elements of x which are greater than 13

# > SEQUENCES

```
> x<-seq(0, 100, by=5)
> x
[1] 0 5 10 15 20 25 30 35 40 45 50
[12] 55 60 65 70 75 80 85 90 95 100
```

x is a sequence from 0 to 100, in steps of 5

```
>
>
> length(x)
```

Returns the length of x, i.e. how many elements are in x

```
[1] 21
>
>
> x<-seq(0, 100, length=5)
> x
[1] 0 25 50 75 100
```

x is a sequence from 0 to 100 which has 5 elements (i.e. has length 5)

## > REPLICATE ELEMENTS

```
> x<-c(1, 1, 1, 1, 1)
```

x is the combination of 1 given five times

```
> x
```

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

```
>
```

```
>
```

```
> x<-rep(1, 5)
```

x is the number 1 replicated 5 times, using the *rep* function

```
> x
```

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



## > CREATING DATA

```
> y<-c(rep("A", 2), rep("B", 2), "C")
```

```
> y
```

```
[1] "A" "A" "B" "B" "C"
```

```
>
```

```
>
```

```
> y<-c(rep(c("A", "B"), each=2), "C")
```

```
> y
```

```
[1] "A" "A" "B" "B" "C"
```

```
>
```

```
>
```

```
> y<-c(rep(c("A", "B"), times=2), "C")
```

```
> y
```

```
[1] "A" "B" "A" "B" "C"
```

y is the combination of the letter A replicated 2 times, the letter B replicated 2 times and the letter C

*rep* can also be used to replicated combinations of letters, either using *each* or *times*

# > COMBINING DATA

```
> z<-c(x, y)
```

Combine x and y as a vector z

```
> z
```

```
[1] "1" "1" "1" "1" "1" "A" "B" "A" "B" "C"
```

```
> z<-cbind(x, y)
```

Combine x and y as columns into a matrix z

```
> z
```

```
      x    y  
[1,] "1" "A"  
[2,] "1" "B"  
[3,] "1" "A"  
[4,] "1" "B"  
[5,] "1" "C"
```

```
> z<-rbind(x, y)
```

Combine x and y as rows into a matrix z

```
> z
```

```
      [,1] [,2] [,3] [,4] [,5]  
x "1"  "1"  "1"  "1"  "1"  
y "A"  "B"  "A"  "B"  "C"
```

# > MATRICES

```
> z<-matrix(c(x, y), nrow=5)
```

```
> z
```

```
      [,1] [,2]  
[1,] "1"  "A"  
[2,] "1"  "B"  
[3,] "1"  "A"  
[4,] "1"  "B"  
[5,] "1"  "C"
```

```
> d_z<-dim(z)
```

```
> d_z
```

```
[1] 5 2
```

```
> t_z<-t(z)
```

```
> t_z
```

```
      [,1] [,2] [,3] [,4] [,5]  
[1,] "1"  "1"  "1"  "1"  "1"  
[2,] "A"  "B"  "A"  "B"  "C"
```

Explicitly create a matrix z, combining x and y and specifying 5 rows. Note that R fills matrices column-wise unless specified using *byrow=TRUE*.

What are the dimensions of matrix z (rows then columns)

Transpose the matrix z

# > MATRIX ELEMENTS

```
> z  
      [,1] [,2]  
[1,] "1"  "A"  
[2,] "1"  "B"  
[3,] "1"  "A"  
[4,] "1"  "B"  
[5,] "1"  "C"
```

Element in the 1<sup>st</sup> row and 2<sup>nd</sup> column  
of z

```
>  
> z[1, 2]  
[1] "A"
```

```
>  
> z[2,]  
[1] "1" "B"
```

All elements in the 2<sup>nd</sup> row of z

```
>  
> z[, -2]  
[1] "1" "1" "1" "1" "1"
```

Remove the second column

# > MATRIX ROWS & COLUMNS

```
> z[1:3,]  
      [,1] [,2]  
[1,] "1"  "A"  
[2,] "1"  "B"  
[3,] "1"  "A"
```

First 3 rows of z

```
>  
> z[c(1, 4),]  
      [,1] [,2]  
[1,] "1"  "A"  
[2,] "1"  "B"
```

1<sup>st</sup> and 4<sup>th</sup> rows of z

```
>  
> z[-c(1, 4),]  
      [,1] [,2]  
[1,] "1"  "B"  
[2,] "1"  "A"  
[3,] "1"  "C"
```

Remove the 1<sup>st</sup> and 4<sup>th</sup> rows of z

# > DATA FRAMES

```
> z <- data.frame(x=x, y=y)
```

Create a data frame with variables x and y

```
> z[1:3,]
```

```
  x y
```

```
1 1 A
```

```
2 1 A
```

```
3 1 B
```

```
> z$x
```

Use \$ to access the x column of z

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

# > LISTS

```
> x <- 1
```

```
> y <- diag(2)
```

```
> w <- list(x=x, y=y)
```

← Create an array of data using R lists

```
> w
```

```
$x
```

← Use \$ to access the a component of w

```
[1] 1
```

```
$y
```

```
      [,1] [,2]
```

```
[1,]     1     0
```

```
[2,]     0     1
```

# > FUNCTIONS

- Are commands
- Are of the form: *function(input, options)*
- Examples: *help, cor, lm*
- Users can write their own functions, e.g. a function for multiplication:

```
> multiply <- function(a, b){  
+   res <- a*b  
+   return(res)  
+ }  
> multiply(2,3)  
[1] 6
```



# > LOGICAL STATEMENTS

Command	Operator
And	&
Or	
Not	!
Is equal to	==
Is not equal to	!=
Is less than	<
Is less than or equal to	<=
Is greater than	>
Is greater than or equal to	>=

## > CLASS STATEMENTS

Command	Operator
Are any values in x missing?	is.na(x)
Are any values in x not missing?	!is.na(x)
Is the variable x numeric?	is.numeric(x)
Is the variable x a string?	is.character(x)
Is the variable x a factor?	is.factor(x)
Coerce the variable x to be a factor	as.factor(x)

## > OTHER USEFUL FUNCTIONS

Command	Operator
Addition, subtraction, multiplication and division	+, -, *, /
Matrix operations: transpose, inverse, multiplication	t, solve, %*%
List R objects in session	ls
Class, length and dimensions	class, length, dim
Paste	paste
Row and column names (matrices)	rownames, colnames
Row and column names (data frames)	row.names, names
Combine, row bind and column bind	c, rbind, cbind
Subset dataset	subset



```
> read.csv("bmi.csv")
```

	id	age	bmi	sex	diet
1	1	32.0	25.0	M	0
2	2	35.0	32.0	M	0
3	3	41.0	27.0	M	0
4	4	29.0	29.0	M	0
5	5	33.5	28.0	M	1
6	6	33.2	29.1	M	1
7	7	32.9	29.4	M	1
8	8	32.6	29.7	F	1
9	9	32.3	30.0	F	1

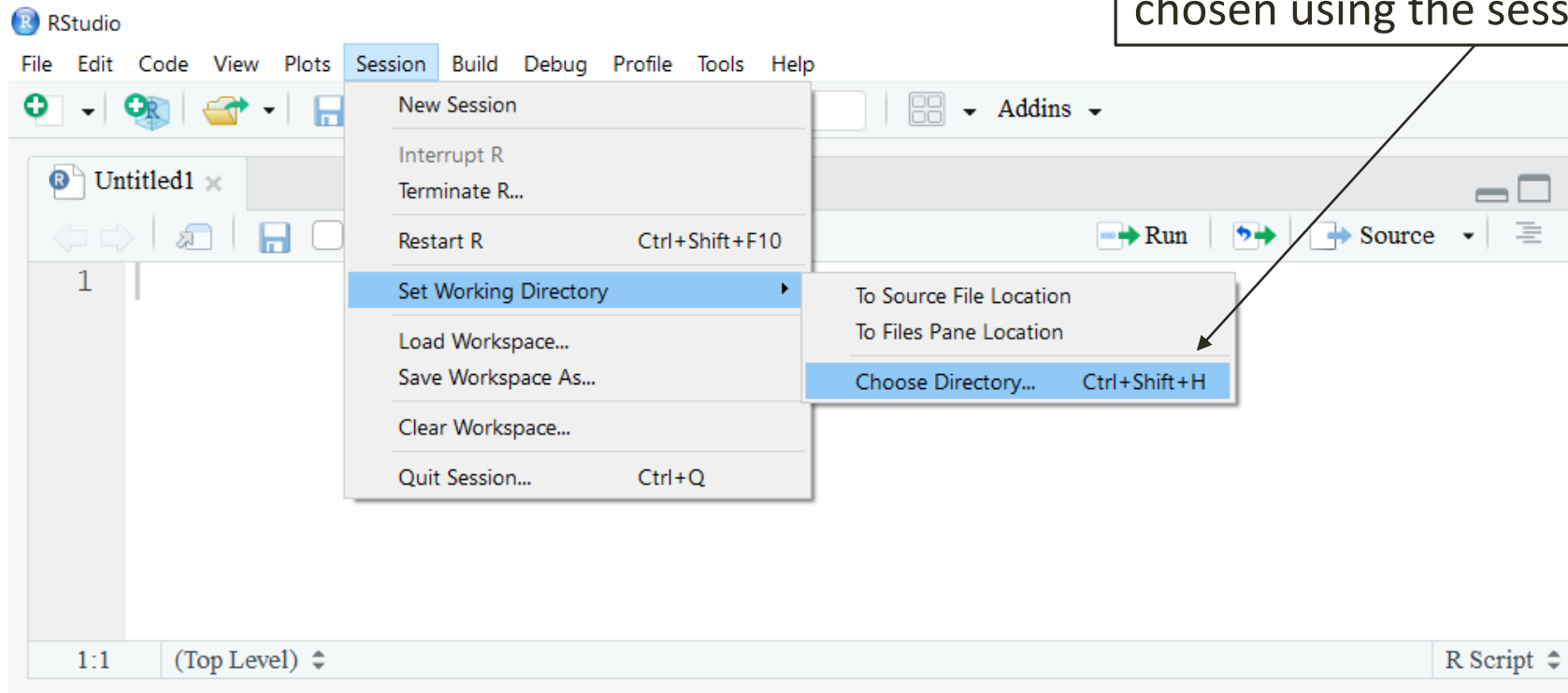
## *READING AND WRITING DATA*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

# > SETTING THE WORKING DIRECTORY

> `setwd("O:/Documents")`



## > READING DATA - CSV

```
> read.csv("bmi.csv")
```

	id	age	bmi	sex	diet
1	1	32.0	25.0	M	0
2	2	35.0	32.0	M	0
3	3	41.0	27.0	M	0
4	4	29.0	29.0	M	0
5	5	33.5	28.0	M	1
6	6	33.2	29.1	M	1
7	7	32.9	29.4	M	1
8	8	32.6	29.7	F	1
9	9	32.3	30.0	F	1
10	10	32.0	30.3	F	1

...

```
> data_bmi<-read.csv("bmi.csv")
```

```
> data_bmi<-read.table("bmi.csv", header=T, sep=",")
```

*read.csv* will read and display the bmi data but not store it

To allow us to use the bmi data, we assign the .csv file to an object called `data_bmi`

*read.table* can also read in CSV files and is more generic than *read.csv*, e.g. can open tab-delimited text files

# > USING THE DATA

```
> data_bmi[, 3]
[1] 25.0 32.0 27.0 29.0 28.0 29.1 29.4 29.7 30.0 30.3
[11] 30.6 30.9 31.2 31.5 31.8 32.1 32.4 32.7 33.0 33.3
```

Access the 3<sup>rd</sup> column of  
data\_bmi data

```
>
> data_bmi[, "bmi"]
[1] 25.0 32.0 27.0 29.0 28.0 29.1 29.4 29.7 30.0 30.3
[11] 30.6 30.9 31.2 31.5 31.8 32.1 32.4 32.7 33.0 33.3
```

Access the bmi column  
from data\_bmi

```
>
>
> data_bmi$bmi
[1] 25.0 32.0 27.0 29.0 28.0 29.1 29.4 29.7 30.0 30.3
[11] 30.6 30.9 31.2 31.5 31.8 32.1 32.4 32.7 33.0 33.3
```

Use \$ to access the bmi  
column of data\_bmi

## > WRITING DATA - CSV

```
> data_bmi<-read.csv("bmi.csv")
```

Read in the .csv and  
assign to object data\_bmi

```
> data_bmi[1:5,]
```

	id	age	bmi	sex	diet
1	1	32.0	25.0	M	0
2	2	35.0	32.0	M	0
3	3	41.0	27.0	M	0
4	4	29.0	29.0	M	0
5	5	33.5	28.0	M	1

```
> write.csv(data_bmi, "data_bmi.csv", row.names=F, quote=F)
```

```
> write.table(data_bmi, "data_bmi.csv", row.names=F, quote=F, sep=",")
```

Save the object data\_bmi to a .csv  
file at the given path



## > READING & WRITING DATA - STATA

- The *foreign* package allows R to read and write Stata 11 and 12 datasets
- There are other packages that read and write datasets for Stata 13, 14 and 15
- *read.dta* and *write.dta* are the Stata dataset equivalents of *read.csv* and *write.csv*



```
> cor(data_bmi$bmi, data_bmi$age)
[1] -0.56091
>
>
> lm(bmi~age, data=data_bmi)
Call:
lm(formula = bmi ~ age, data = data_bmi)

Coefficients:
(Intercept)          age
   44.5249       -0.4419
```

# *STATISTICAL ANALYSIS*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

# > BASIC FUNCTIONS

The diagram illustrates the relationship between R functions and statistical measures. On the left, a series of R commands are listed, each followed by its output. On the right, a vertical column of boxes contains the names of the statistical measures. Arrows point from each box to the corresponding R function in the code on the left.

R Function	Statistical Measure
<code>&gt; mean(data_bmi\$bmi)</code> [1] 30.45	Mean
<code>&gt; median(data_bmi\$bmi)</code> [1] 30.75	Median
<code>&gt; sd(data_bmi\$bmi)</code> [1] 2.124419	Standard deviation
<code>&gt; min(data_bmi\$bmi)</code> [1] 25	Minimum
<code>&gt; max(data_bmi\$bmi)</code> [1] 33.3	Maximum
<code>&gt; quantile(data_bmi\$bmi, probs=0.25)</code> 25% 29.325	Lower quartile

# > SUMMARISE DATA

```
> summary(data$bmi)
```

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
25.00 29.32 30.75 30.45 32.02 33.30
```

*summary* can be used on a variable  
or the whole dataset

```
>
```

```
> summary(data)
```

id	age	bmi	sex	diet
min. : 1.00	Min. :29.00	Min. :25.00	F:13	Min. :0.00
1st Qu.: 5.75	1st Qu.:30.12	1st Qu.:29.32	M: 7	1st Qu.:0.00
Median :10.50	Median :31.55	Median :30.75		Median :1.00
Mean :10.50	Mean :31.85	Mean :30.45		Mean :0.55
3rd Qu.:15.25	3rd Qu.:32.67	3rd Qu.:32.02		3rd Qu.:1.00
Max. :20.00	Max. :41.00	Max. :33.30		Max. :1.00

```
>
```

```
> table(data$sex)
```

```
 F  M
13  7
```

# > CORRELATION AND REGRESSION

```
> cor(data$bmi, data$age)
```

Pearson correlation  
between bmi and age

```
[1] -0.56091
```

```
>
```

```
>
```

```
> model <- lm(bmi~age, data=data)
```

Linear regression for outcome  
bmi and exposure age

```
> model
```

```
Call:
```

```
lm(formula = bmi ~ age, data = data)
```

```
Coefficients:
```

```
(Intercept)      age  
  44.5249      -0.4419
```

Intercept and slope estimates from  
the regression

# > SUMMARISE REGRESSION MODELS

```
> summary(model)
```

```
Call:
```

```
lm(formula = bmi ~ age, data = data)
```

*summary* gives you more information on the regression model

```
Residuals:
```

```
Min 1Q Median 3Q Max
```

```
-5.3837 -0.4604 0.3349 0.9627 2.9420
```

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	44.5249	4.9131	9.063	3.97e-08	***
age	-0.4419	0.1537	-2.875	0.0101	*

```
---
```

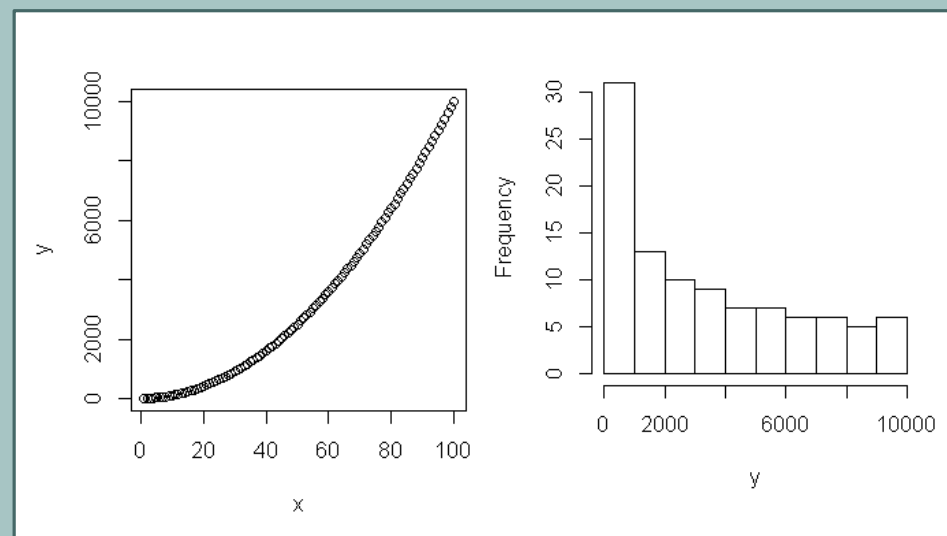
```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## > USING RESULTS

```
> names(model)
[1] "coefficients" "residuals" "effects" "rank"
[5] "fitted.values" "assign" "qr" "df.residual"
[9] "xlevels" "call" "terms" "model"
> names(summary(model))
[1] "call" "terms" "residuals" "coefficients"
[5] "aliased" "sigma" "df" "r.squared"
[9] "adj.r.squared" "fstatistic" "cov.unscaled"
> model$coefficients
(Intercept)          age
44.524864    -0.441911
```

*names* function allows us to see the various components within “model”

The \$ command allows us to extract a component within “model”



# *PLOTS*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

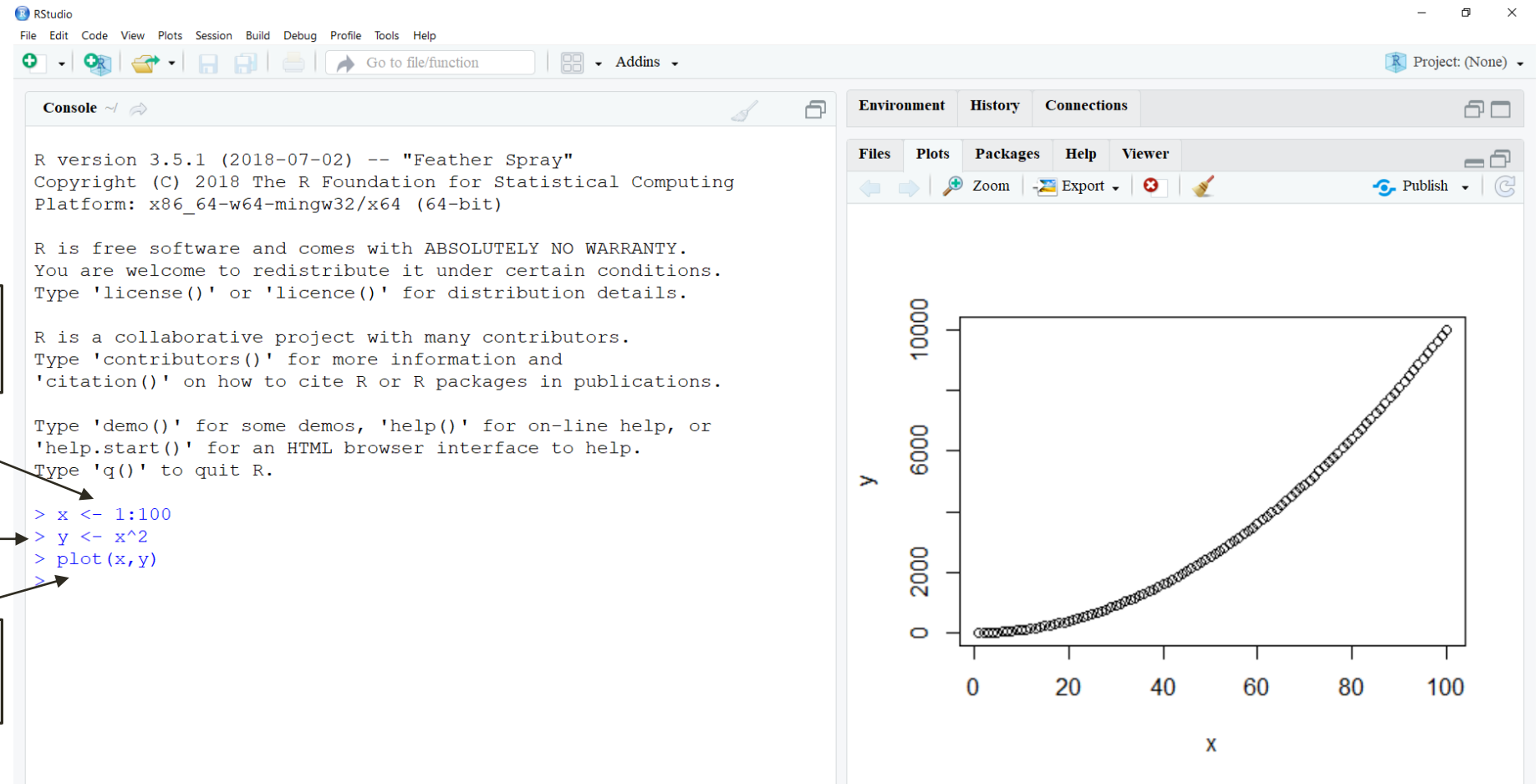


# > SCATTERPLOT

x is a vector of the numbers 1 to 100

y is x squared

plot(x,y) produces a scatterplot

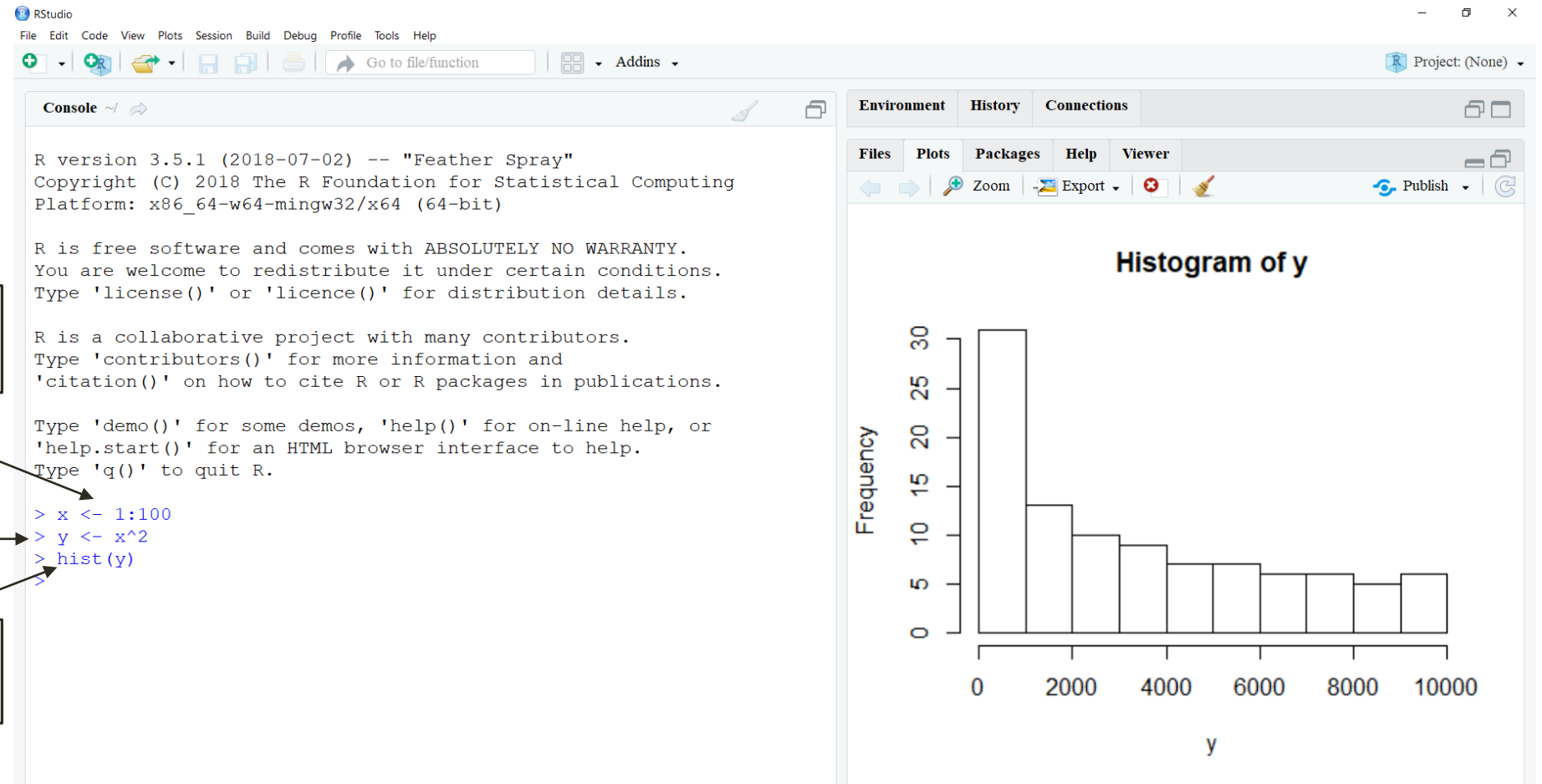


# > HISTOGRAM

x is a vector of the numbers 1 to 100

y is x squared

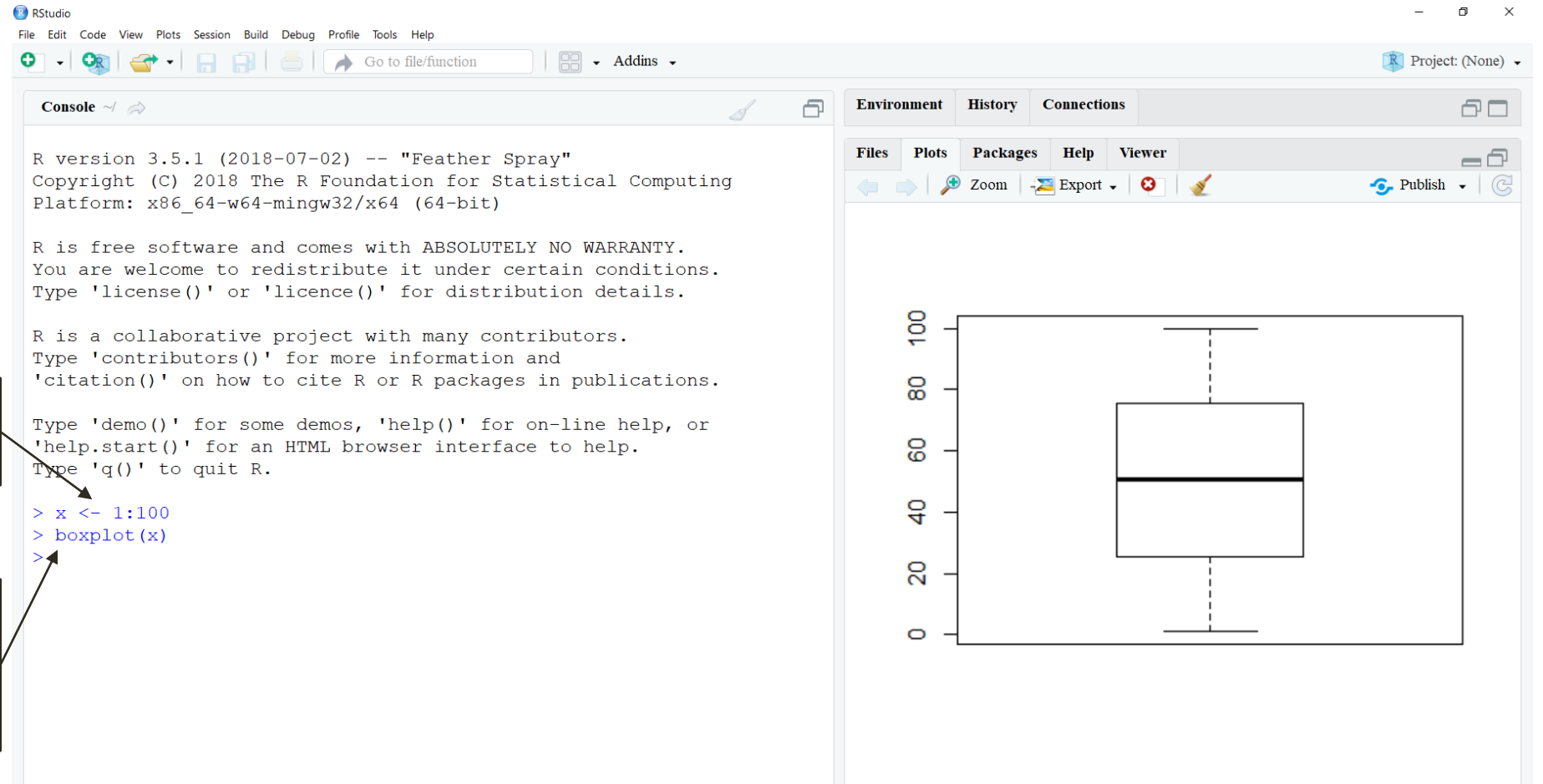
hist(x,y) produces a histogram of y



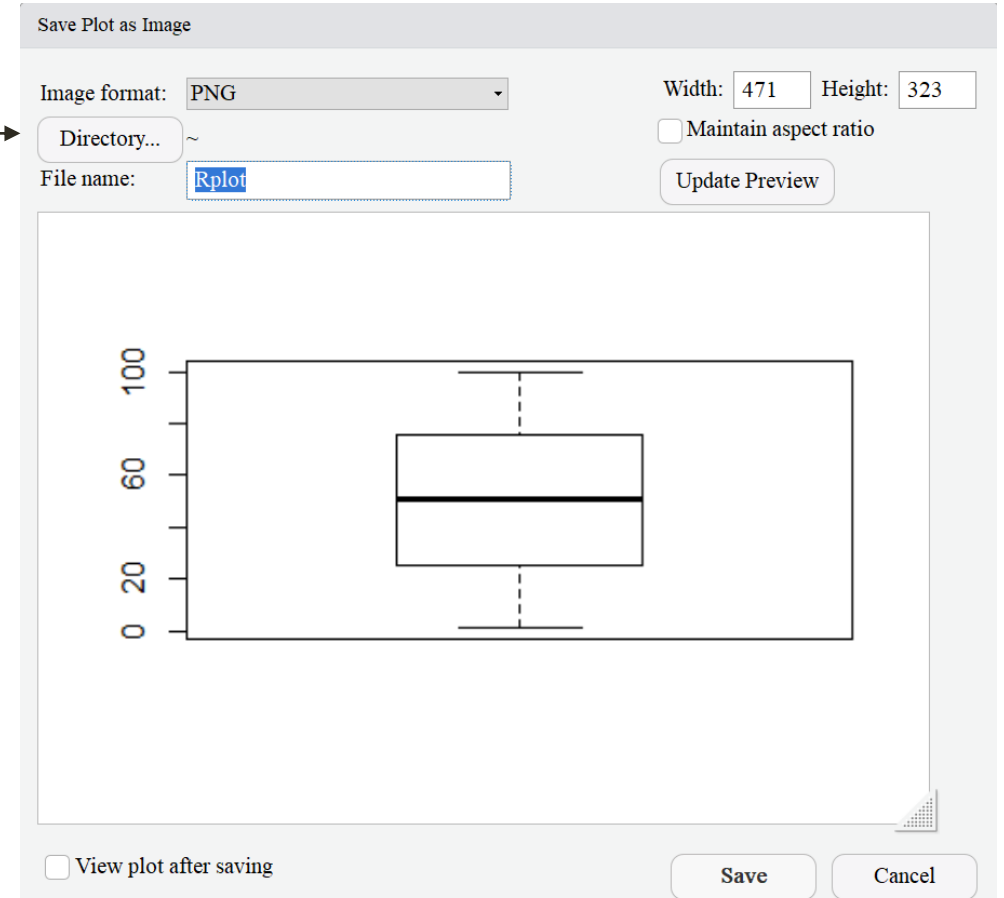
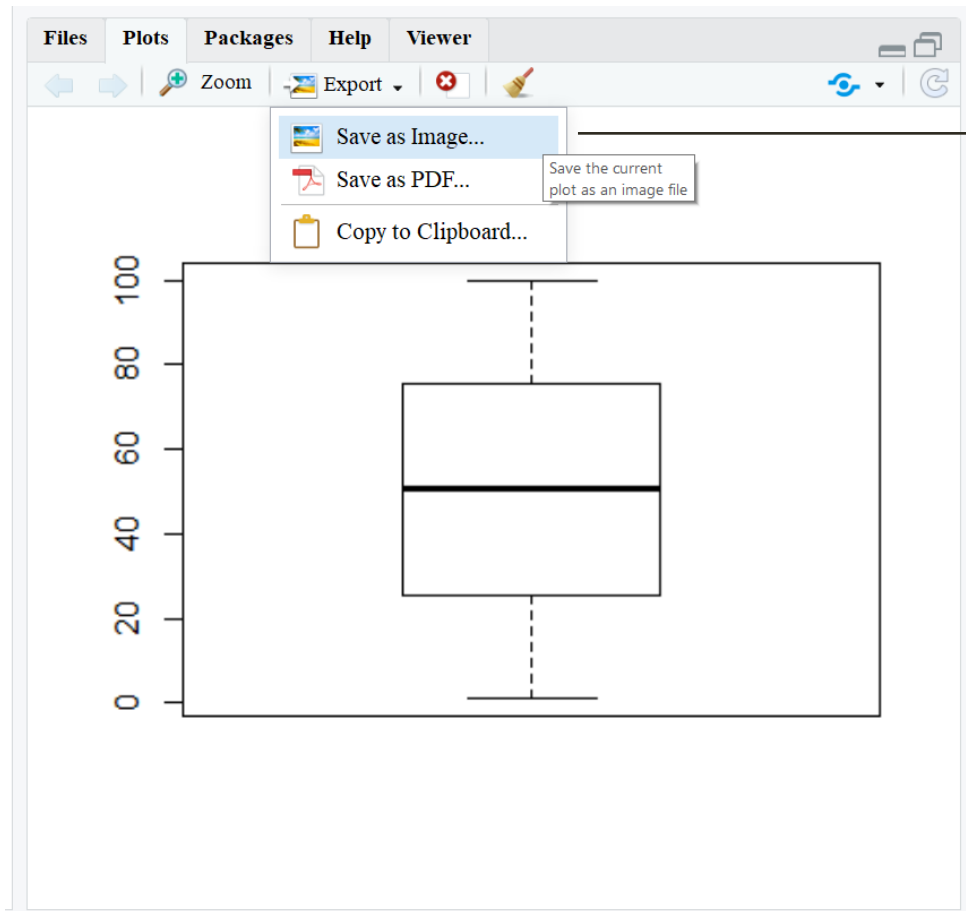
# > BOXPLOT

x is a vector of the  
numbers 1 to 100

boxplot(x)  
produces a box  
plot of x



# > SAVING PLOTS - DIRECTLY



## > SAVING PLOTS - INLINE

```
> x<-1:100
>
> png(filename="example_boxplot.png", width=10, height=10,
units="cm", res=500)
>
> boxplot(x)
>
> dev.off()
null device
      1
```

Opens a device in which to save the plot (in this case a .png file)

Create plot

Once the plot is complete we close the device



```
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from
'package:base':

      format.pval, round.POSIXt,
      trunc.POSIXt, units
```

***R PACKAGES***

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)

# > INSTALLING PACKAGES

```
> install.packages("Hmisc")
```

Install packages

Installing package into 'O:/Documents/R/win-library/3.5'

(as 'lib' is unspecified)

also installing the dependencies 'backports', 'scales', 'checkmate',  
'viridisLite', 'survival', 'ggplot2', 'htmlTable', 'viridis'

·  
·  
·

package 'Hmisc' successfully unpacked and MD5 sums checked

The downloaded binary packages are in < path >

# > LOADING PACKAGES

```
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2
```

Command to load the *Hmisc* package into our current R session

```
Attaching package: 'Hmisc'
```

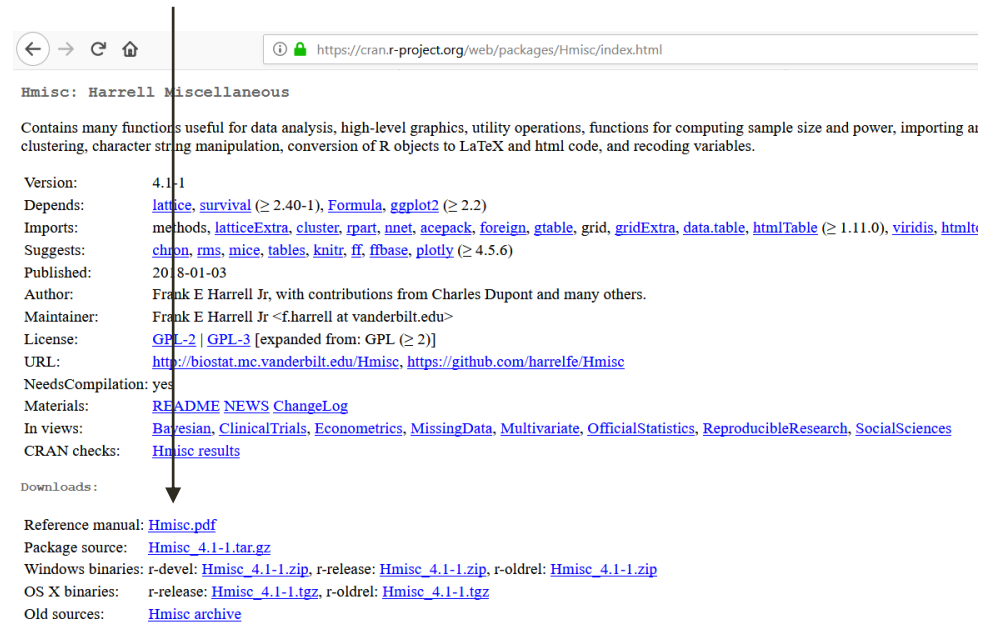
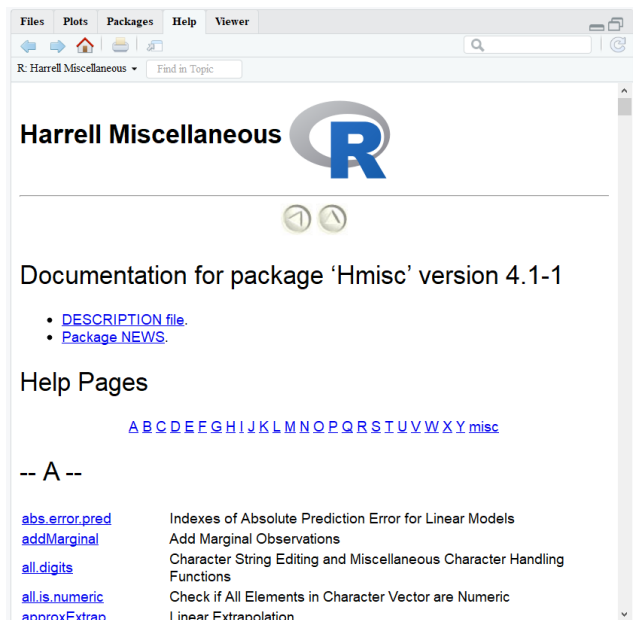
The following objects are masked from 'package:base':

```
format.pval, round.POSIXt, trunc.POSIXt, units
```



# > HELP FOR PACKAGES

- The *help* function can be used to open the documentation for an R package, e.g. `help(package="Hmisc")`
- Search the package list on CRAN <https://cran.r-project.org>



# > LEARN R - SWIRL

```
> install.packages("swirl")
```

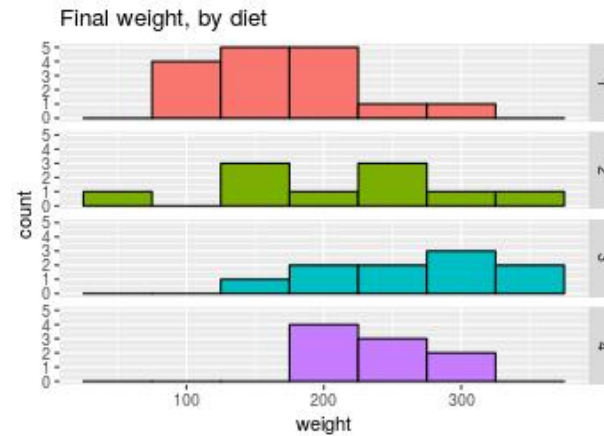
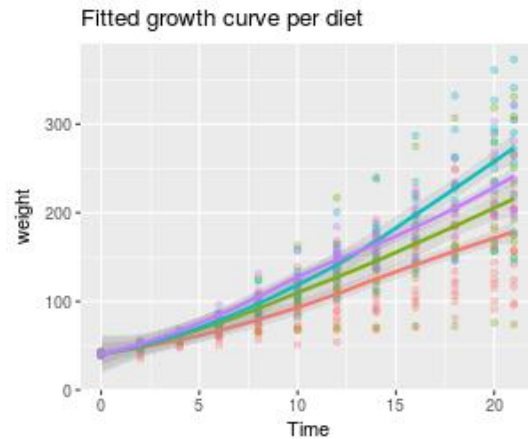
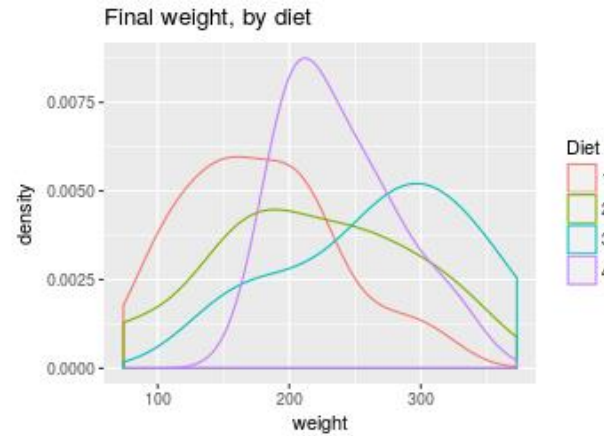
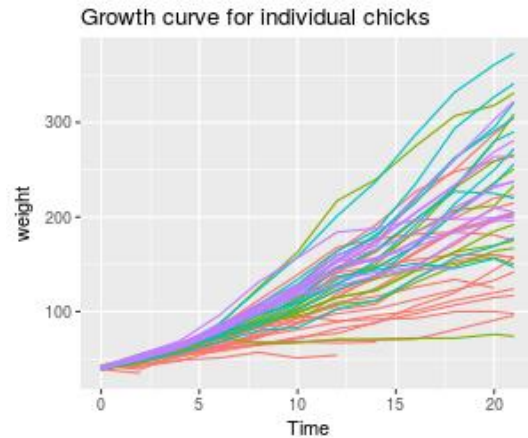
```
> library(swirl)
```

```
| Hi! Type swirl() when you are ready to begin.
```



*swirl* (<http://swirlstats.com/students.html>)  
allows you to learn R within R itself

# > PLOTS - GGPLOT2



ggplot2  
(<http://ggplot2.tidyverse.org>)  
allows you to create  
publication quality plots

# > NOTEBOOK INTERFACE - R MARKDOWN

## Reusable Version in rCharts

As I mentioned above, this visualization works well with any cumulative growth time series, so let's apply it to the `managers` dataset supplied by the `PerformanceAnalytics` package.

### Get Data and Transform

```
1. #get the data and convert to a format that we would expect from melted xts
2. #will be typical
3. #also original only uses a single value (val) and not other
4. require(reshape2)
5. require(PerformanceAnalytics)
6.
7. data(managers)
8. managers <- na.omit(managers)
9. managers.melt <- melt(
10.   data.frame( index( managers ), coredata(cumprod( managers+1 )*100 ) ),
11.   id.vars = 1
12. )
13. colnames(managers.melt) <- c("date", "manager", "val")
14. managers.melt[, "date"] <- format(managers.melt[, "date"], format = "%Y-%m-%d")
```

### Draw The Graph

```
1. require(rCharts)
2. p2 <- rCharts$new()
3. p2$setLib('libraries/widgets/ny_t_home')
4. p2$setTemplate(script = "libraries/widgets/ny_t_home/layouts/ny_t_home.html")
5.
6. p2$set(
7.   description = "This data comes from the managers dataset included in the R package PerformanceAnalytics.",
8.   data = managers.melt,
9.   groups = "manager"
10. )
11. cat(noquote(p2$html()))
```

If you bought `HAM1` around `Sep. 2002` it would be worth **96 percent more** today.

Price changes from **Sept 2002**... **...to Dec 2006**



*rmarkdown*

(<http://rmarkdown.rstudio.com>)

provides an interface to put together narrative text, code and output

# > WEB APPS - SHINY



*shiny*  
(<http://shiny.rstudio.com>)  
allows you to create  
interactive web applications



Harriet Mills



Andrew Simpkin

## *ACKNOWLEDGMENTS*

James Staley

[james.staley@bristol.ac.uk](mailto:james.staley@bristol.ac.uk)