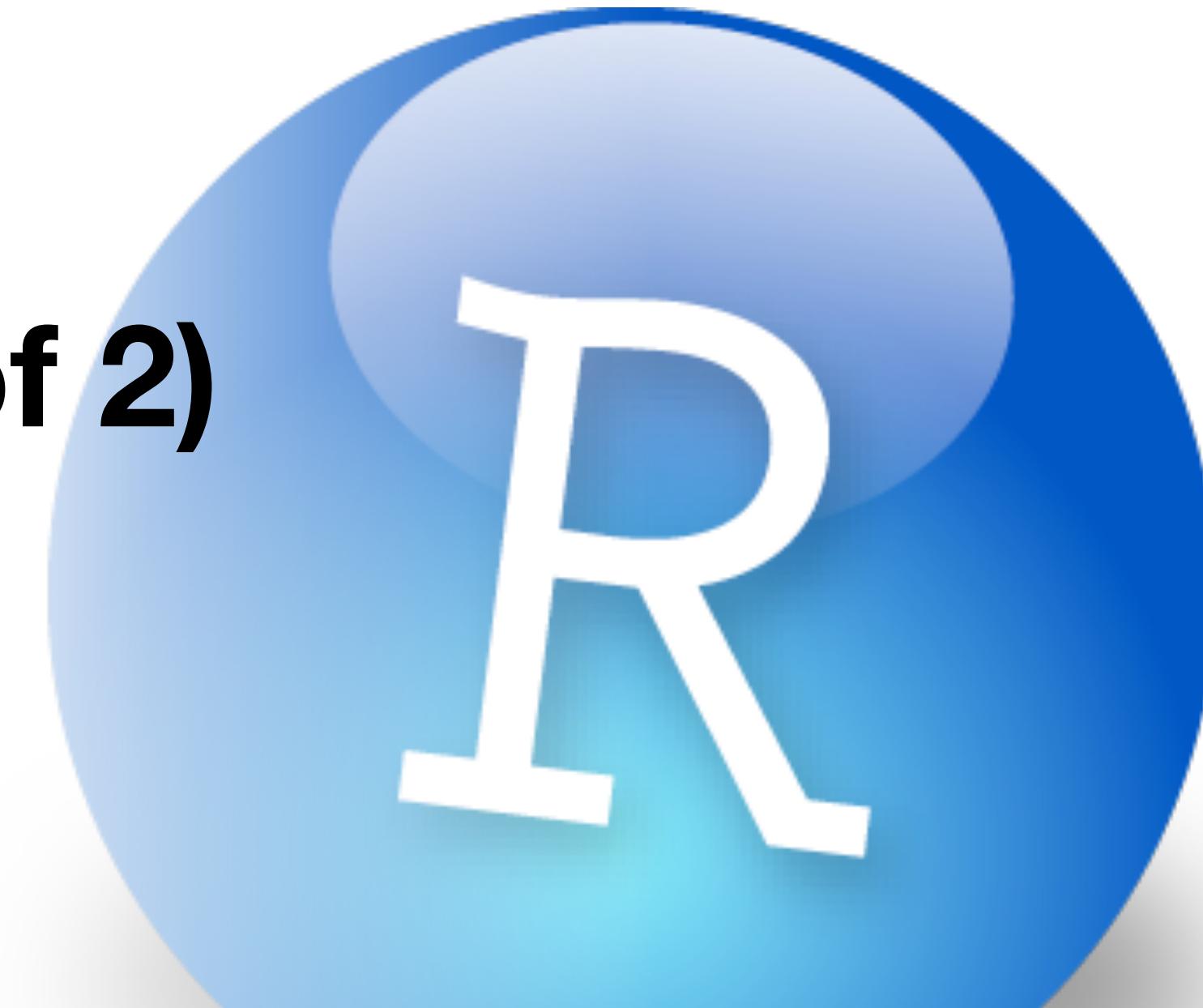


All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

The R Language (1 of 2)

Understand how R stores
and works with data



Garrett Grolemund

Master Instructor, RStudio

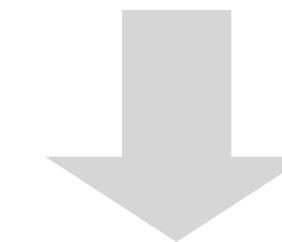
August 2014

1. RStudio basics
2. R basics
3. Data structures
4. Data types

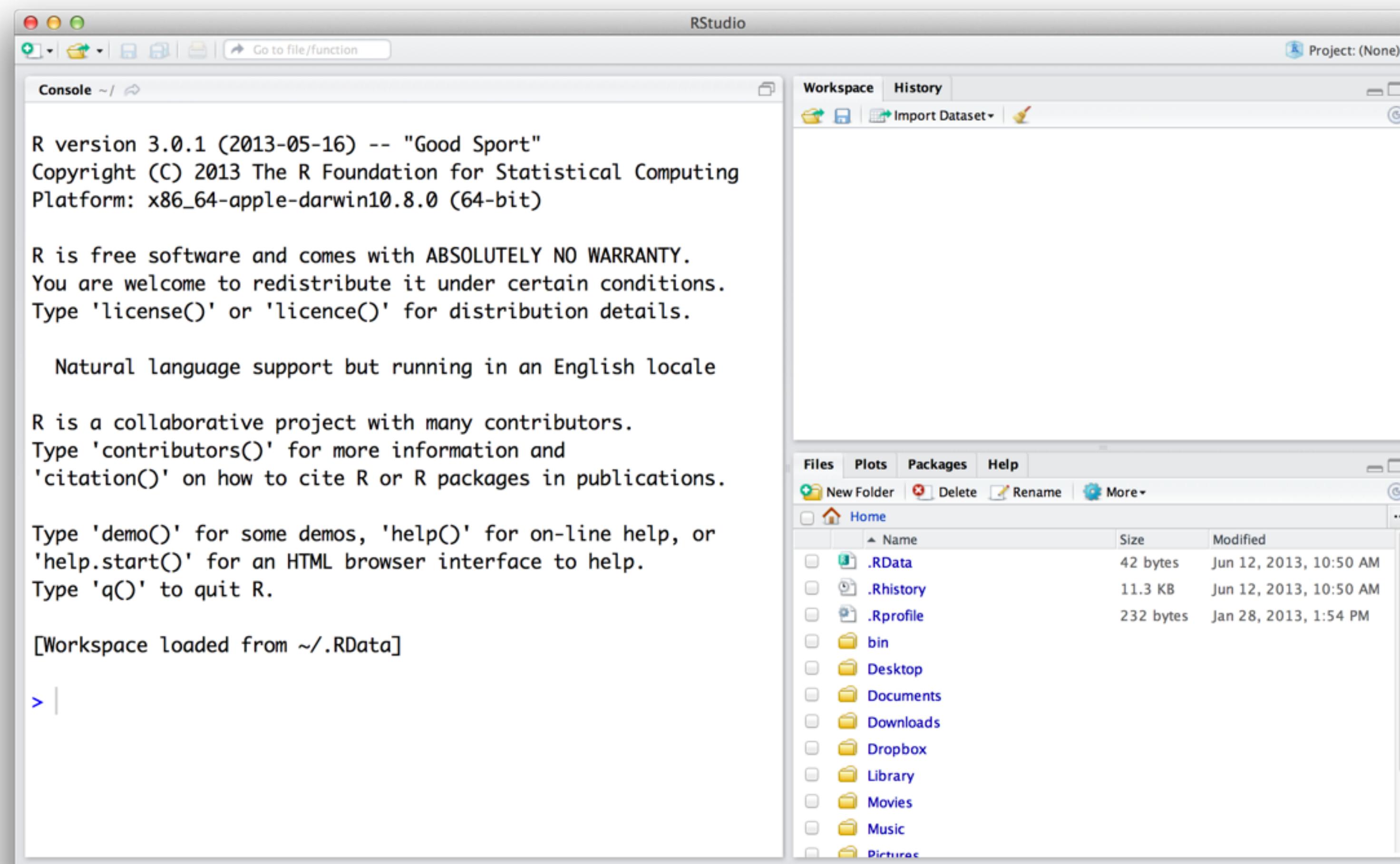
RStudio

Getting started

Open RStudio as you would any program. It's icon looks like this



The RStudio window is divided into 3-4 panes.
Each keeps track of separate information.



R Console

The console gives you a place to execute commands written in the R computer language

R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.

[Workspace loaded from ~/.RData]

> |

Workspace History Import Data

Files Plots Packages

New Folder Delete

Home

.RData .Rhistory .Rprofile bin Desktop Documents Downloads Dropbox Library Movies Music Pictures

Command prompt

Type commands on the line that begins with a > sign (known as the prompt)

The screenshot shows the RStudio interface with the following details:

- Console Tab:** Active tab.
- Header:** Shows "R version 3.0.1 (2013-05-16) -- "Good Sport"" and "Platform: x86_64-apple-darwin10.8.0 (64-bit)".
- Message Area:** Displays the standard R startup message, including license information, natural language support, collaborative project details, and help resources.
- Workspace Area:** Shows a list of files in the current workspace, including ".RData", ".Rhistory", ".Rprofile", "bin", "Desktop", "Documents", "Downloads", "Dropbox", "Library", "Movies", "Music", and "Pictures".
- Bottom Left:** A red circle highlights the command prompt line where the user has typed "`> 1 + 1`".
- Bottom Right:** Copyright notice: "© 2014 RStudio, Inc. All rights reserved."

Output

When you hit enter,
RStudio will run
your command and
display any output
below it

Output →
New prompt →

The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the standard R startup message, followed by a workspace loading message, and then a user input command '1 + 1' followed by its result '[1] 2'. The RStudio sidebar on the right shows the 'Files' tab is active, displaying a list of files and folders in the current workspace.

```
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.

[Workspace loaded from ~/.RData]

> 1 + 1
[1] 2
> |
```

History

As you enter commands, you accumulate a history of past commands

The screenshot shows the RStudio interface. The top bar says "RStudio". The main area has a "Console" tab selected. The console window displays the R startup message, which includes the version (R version 3.0.1 (2013-05-16) -- "Good Sport"), copyright information (Copyright (C) 2013 The R Foundation for Statistical Computing), and platform details (Platform: x86_64-apple-darwin10.8.0 (64-bit)). It also mentions that R is free software with no warranty, encourages redistribution under certain conditions, and provides details about licensing. Below this, it discusses natural language support and running in an English locale, mentions R as a collaborative project with many contributors, and provides information on how to cite R or R packages in publications. It also suggests using 'demo()', 'help()', or 'help.start()' for help, and 'q()' to quit R. A message at the bottom indicates that the workspace was loaded from `~/.RData`. The bottom part of the console shows a command history starting with `> 1 + 1`, followed by `[1] 2`, `> 1 + 2`, `[1] 3`, `> 1 + 3`, `[1] 4`, and a cursor at `>`. To the right of the console is the "Workspace" pane, which lists files and folders in the current workspace, including `.RData`, `.Rhistory`, `.Rprofile`, `bin`, `Desktop`, `Documents`, `Downloads`, `Dropbox`, `Library`, `Movies`, `Music`, and `Pictures`. The "Files" tab is selected in the workspace pane. The bottom right corner of the RStudio window has a footer that reads "© 2014 RStudio, Inc. All rights reserved."

```
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.

[Workspace loaded from ~/.RData]

> 1 + 1
[1] 2
> 1 + 2
[1] 3
> 1 + 3
[1] 4
>
```

Type 'q()' to quit R.

[1]

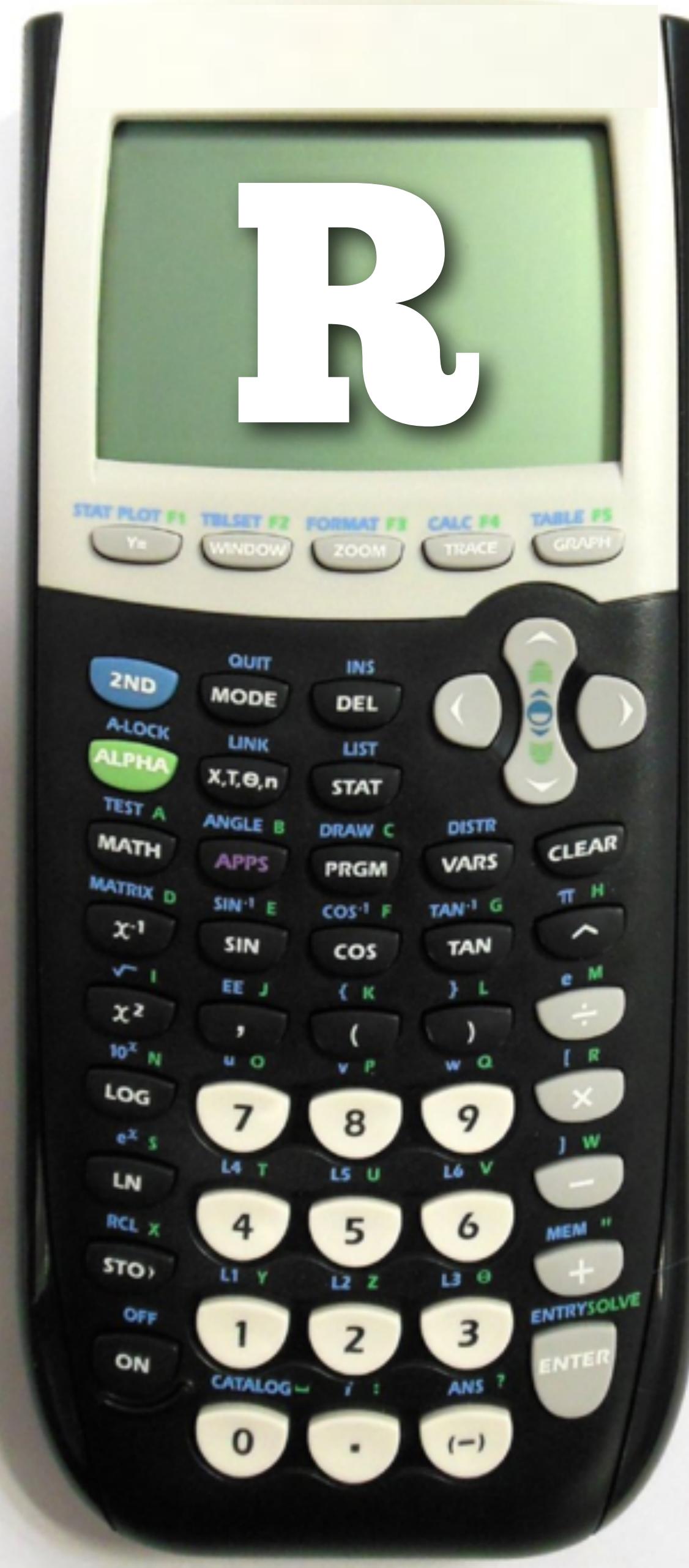
R displays an index next to the output.

Just ignore this.

Somewhat helpful when R returns more than one value in the output.

[Workspace loaded from

```
> 1 + 1  
[1] 2  
> 1 + 2  
[1] 3  
> 1 + 3  
[1] 4  
>
```



R is like a fancy calculator
on your computer

5 + 5

10

4 - 1

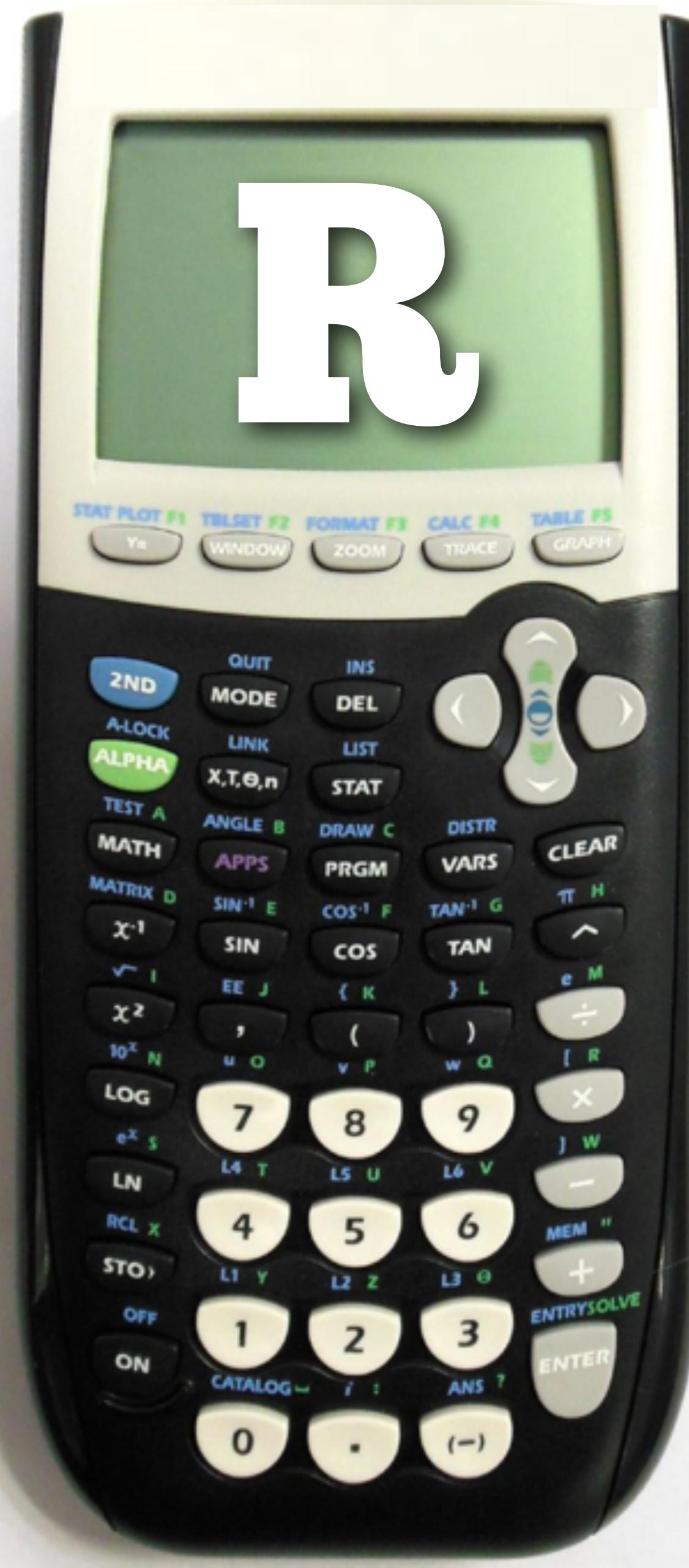
3

1 * 2

2

4 ^ 2

16



It can do algebra

a <- 1

b <- 2

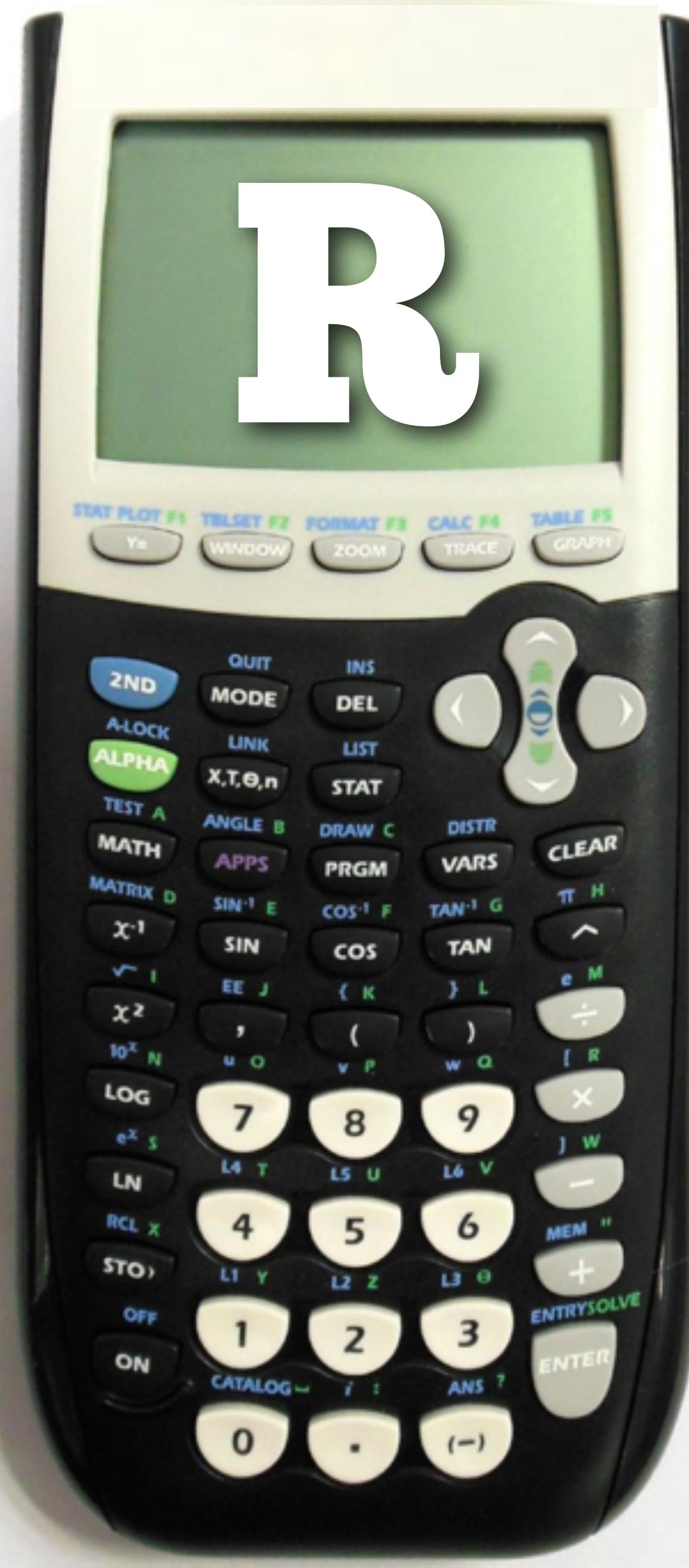
a + b

3

A <- 3

a + b - A

0



And it has functions that let you do more sophisticated manipulations

round(3.1415)

3

factorial(3)

6

sqrt(9)

3

3! = 3 x 2 x 1

square root

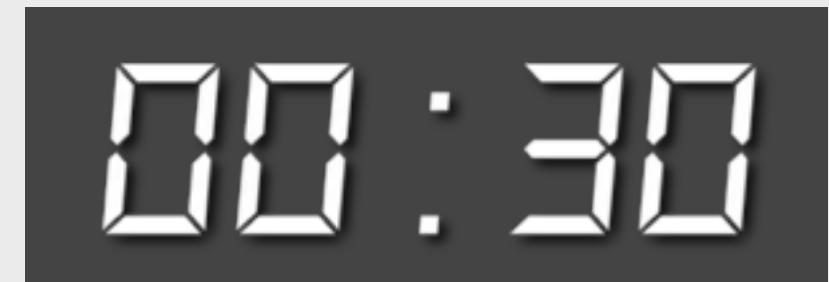
Your turn

What do you think this will return?
You have 30 seconds.

Your turn

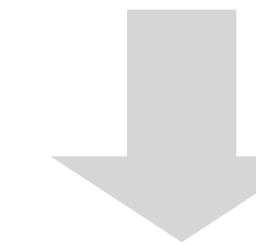
What do you think this will return?
You have 30 seconds.

```
factorial(round(2.0015) + 1)
```

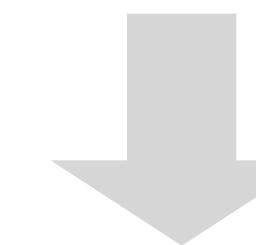


R always works from the innermost parenthesis to the outermost (just like a calculator).

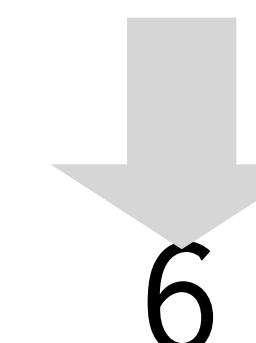
```
factorial(round(2.0015) + 1)
```



```
factorial(2 + 1)
```



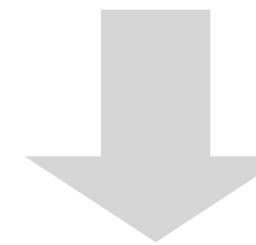
```
factorial(3)
```



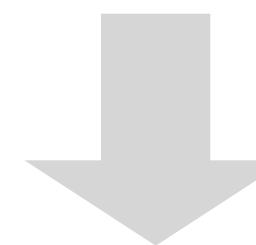
6

R always works from the innermost parenthesis to the outermost (just like a calculator).

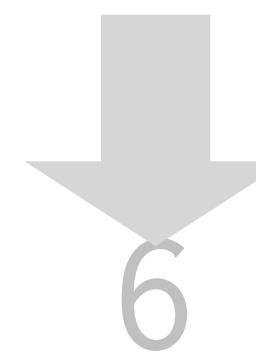
```
factorial(round(2.0015) + 1)
```



```
factorial(2 + 1)
```



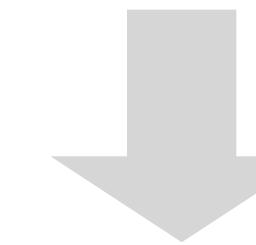
```
factorial(3)
```



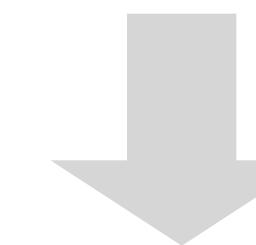
6

R always works from the innermost parenthesis to the outermost (just like a calculator).

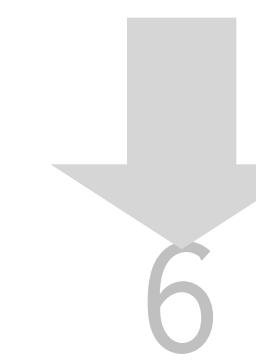
factorial(round(2.0015) + 1)



factorial(2 + 1)



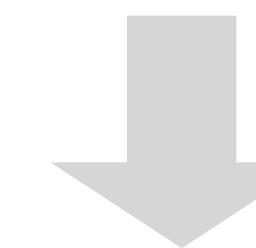
factorial(3)



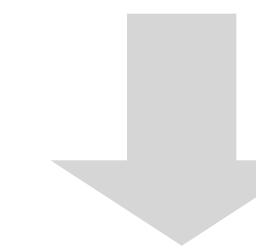
6

R always works from the innermost parenthesis to the outermost (just like a calculator).

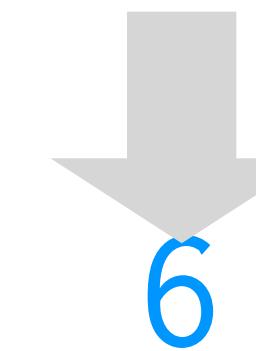
factorial(round(2.0015) + 1)



factorial(2 + 1)



factorial(3)



$$3! = 3 \times 2 \times 1$$

+ prompt

If your prompt turns into a "+", R thinks you haven't finished your previous command.

Either finish the command, or press escape.

```
Type 'demo()' for some demos, 'he  
'help.start()' for an HTML browser.  
Type 'q()' to quit R.
```

```
[Workspace loaded from ~/.RData]
```

```
> 1 + 1  
[1] 2  
> 1 + 2  
[1] 3  
> 1 + 3  
[1] 4  
> factorial(round(3.1415) + 1  
+
```



+ prompt

If your prompt turns into a "+", R thinks you haven't finished your previous command.

Either finish the command, or press escape.

```
?help.start() for an HTML browser  
Type 'q()' to quit R.
```

```
[Workspace loaded from ~/.RData]
```

```
> 1 + 1  
[1] 2  
> 1 + 2  
[1] 3  
> 1 + 3  
[1] 4  
> factorial(round(3.1415) + 1  
+ )  
[1] 24  
>
```

Your turn

Open RStudio and try the following tasks:

1. Pick a number and add 2 to it
2. Multiply the result by 3
3. Subtract 6 from the result of step 2
4. Divide the result of step 3 by 3

```
10 + 2
```

```
# 12
```

```
12 * 3
```

```
# 36
```

```
36 - 6
```

```
# 30
```

```
30 / 3
```

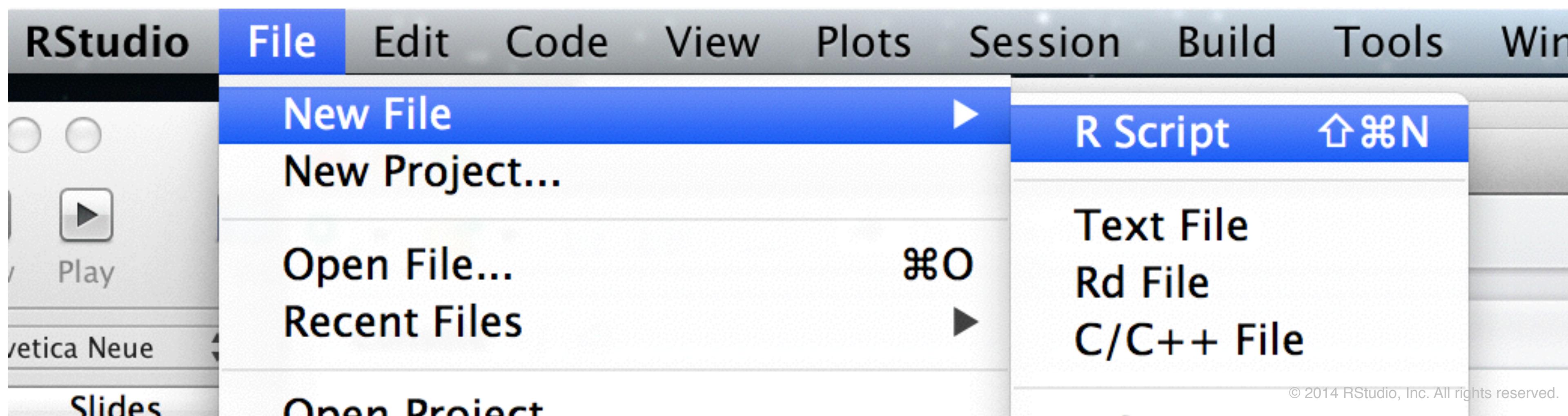
```
# 10
```

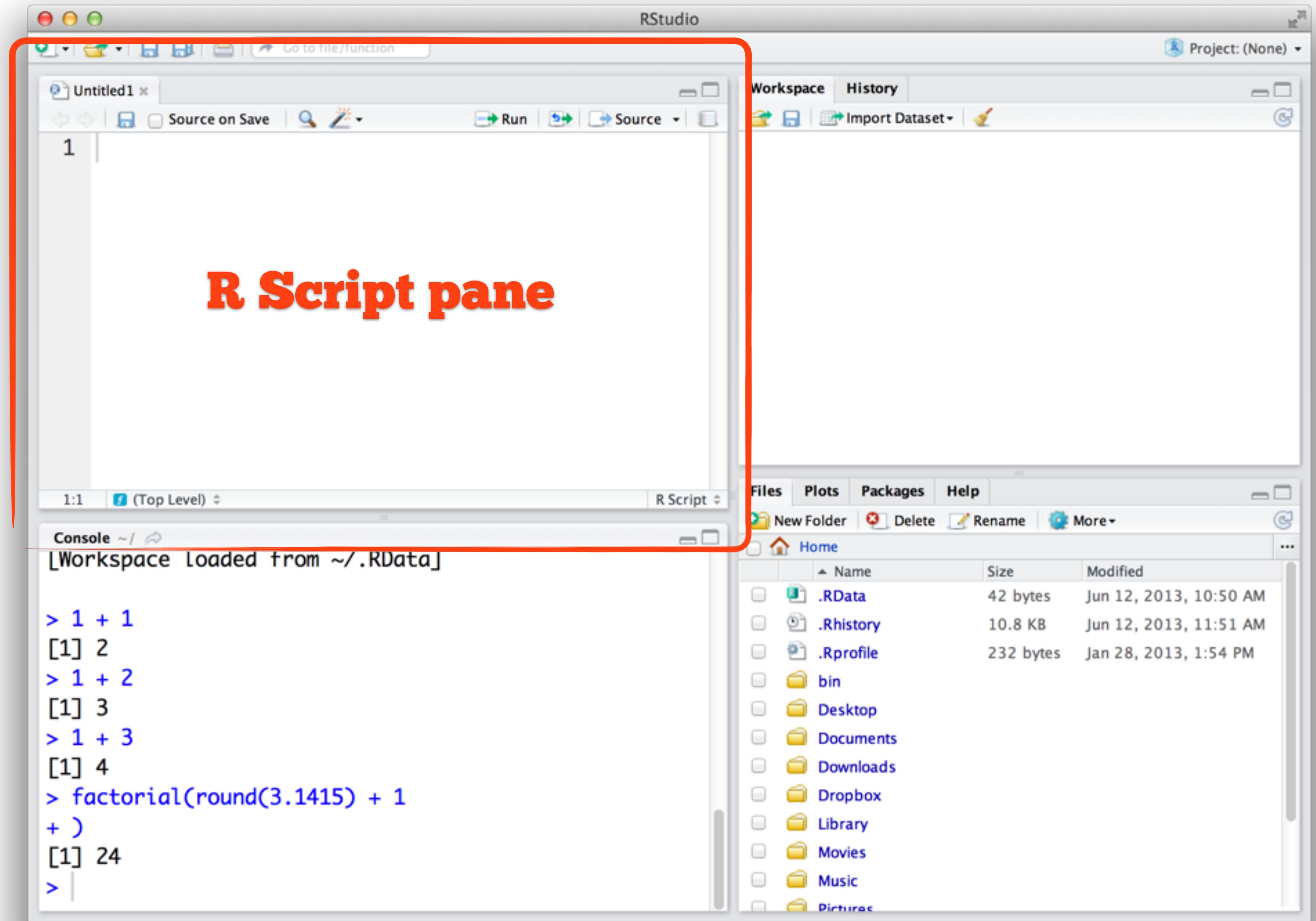
Workflow

R scripts

It is easier to compose your code in an R script than in the command line.

To open an R script, go to File>New File>R Script in the toolbar.





Common workflow

1. Write code in an R script

The screenshot shows the RStudio interface. On the left, the 'Script' tab of the 'Untitled1' file is open, displaying the following R code:

```
1 log10(1000) + log10(100)
```

An orange arrow points from the text 'Common workflow' to the code in the script editor. On the right, the 'Console' tab is active, showing the output of the code:

```
1:25 [Top Level] R Script
```

Console output:

```
[Workspace loaded from ~/.RData]

> 1 + 1
[1] 2
> 1 + 2
[1] 3
> 1 + 3
[1] 4
> factorial(round(3.1415) + 1
+ )
[1] 24
>
```

The 'Files' panel on the right lists various R workspace files and folders.

Common workflow

1. Write code in an R script
2. Run code in console with Run

The screenshot shows the RStudio interface. In the top-left, there's a script editor window titled "Untitled1" containing the R code:

```
1 log10(1000) + log10(100)
```

A red circle highlights the "Run" button in the toolbar above the script editor. A red arrow points from the bottom of the script editor down to the "Console" tab in the bottom-left, which displays the following R session history:

```
1:25 (Top Level) R Script  
Console ~/  
> 1 + 1  
[1] 2  
> 1 + 2  
[1] 3  
> 1 + 3  
[1] 4  
> factorial(round(3.1415) + 1  
+ )  
[1] 24  
> log10(1000) + log10(100)  
[1] 5  
>
```

The right side of the interface shows the "Workspace" and "History" panes, and the bottom-right shows the file browser.

Common workflow

1. Write code in an R script
2. Run code in console with Run
3. Save R Script when finished

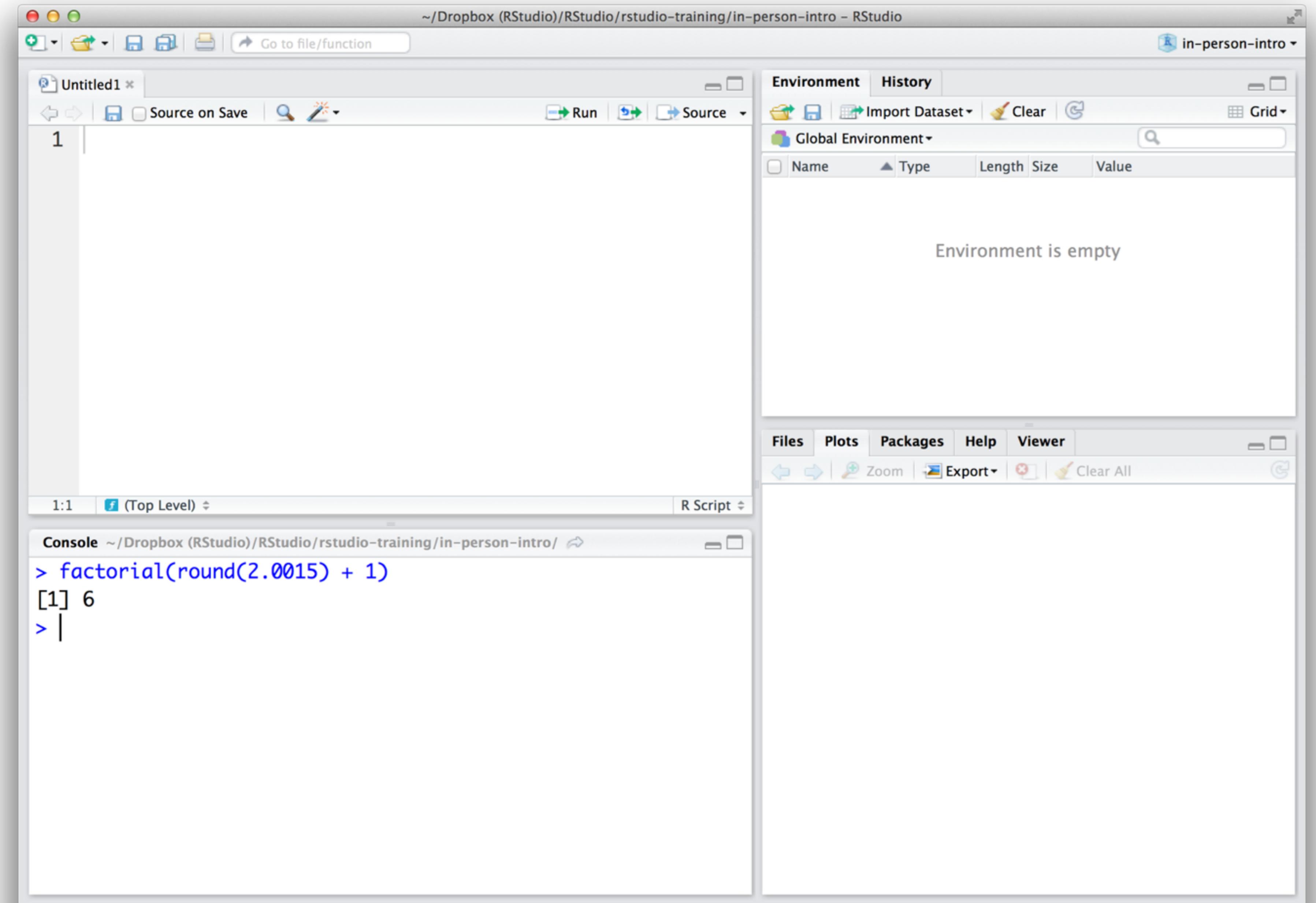
The screenshot shows the RStudio interface with the following elements:

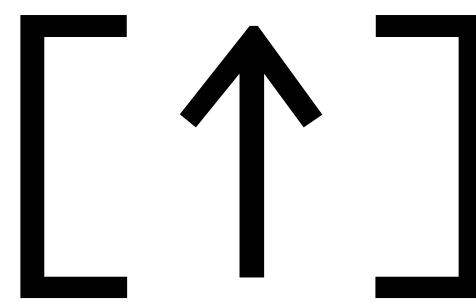
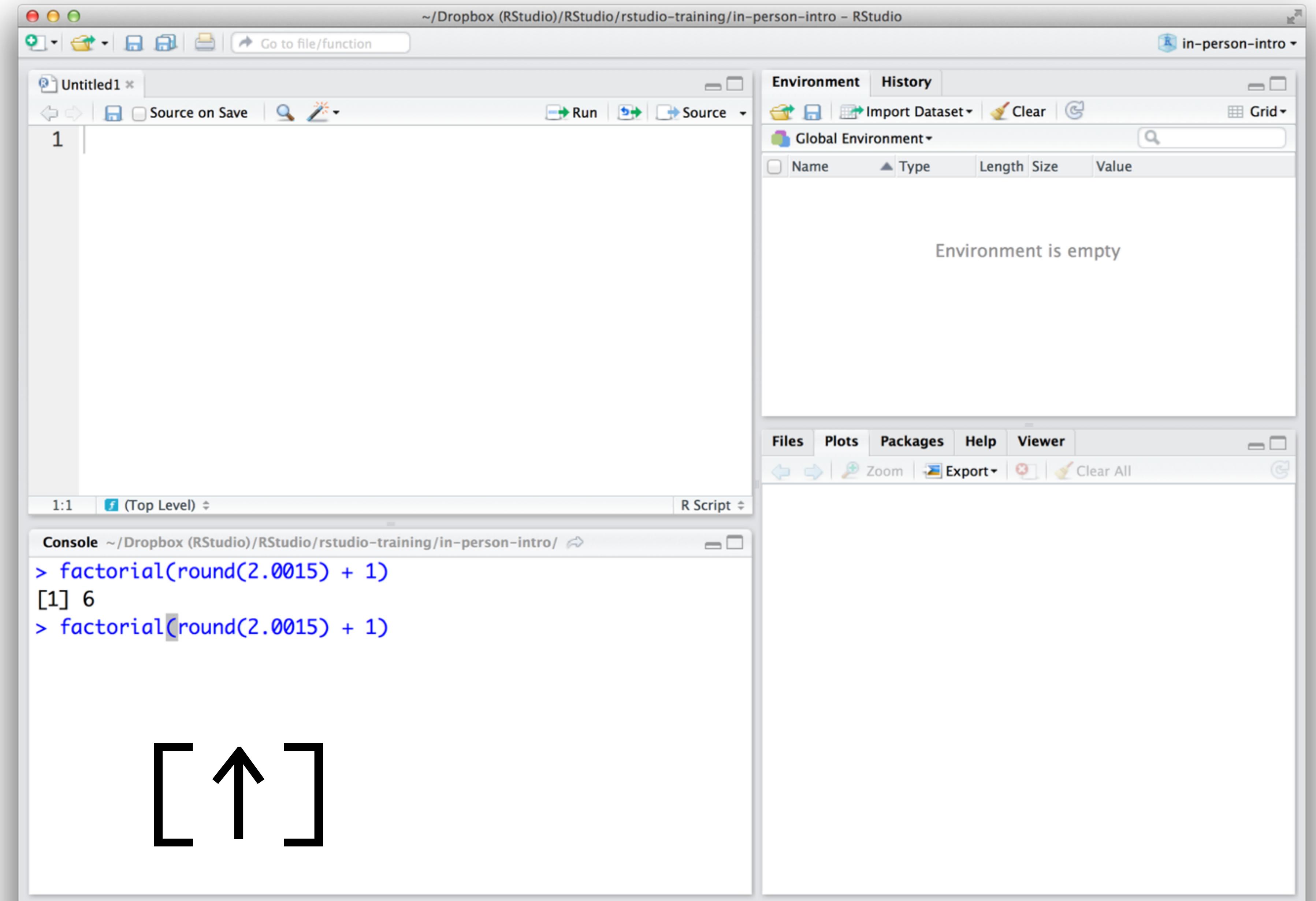
- Top Bar:** Shows the RStudio logo and menu items.
- File Tab:** Untitled1* is selected.
- Toolbar:** Includes icons for New File, Open, Save, Print, Go to file/function, Source on Save (circled in red), Run, Source, and Help.
- Code Editor:** Displays the R code:

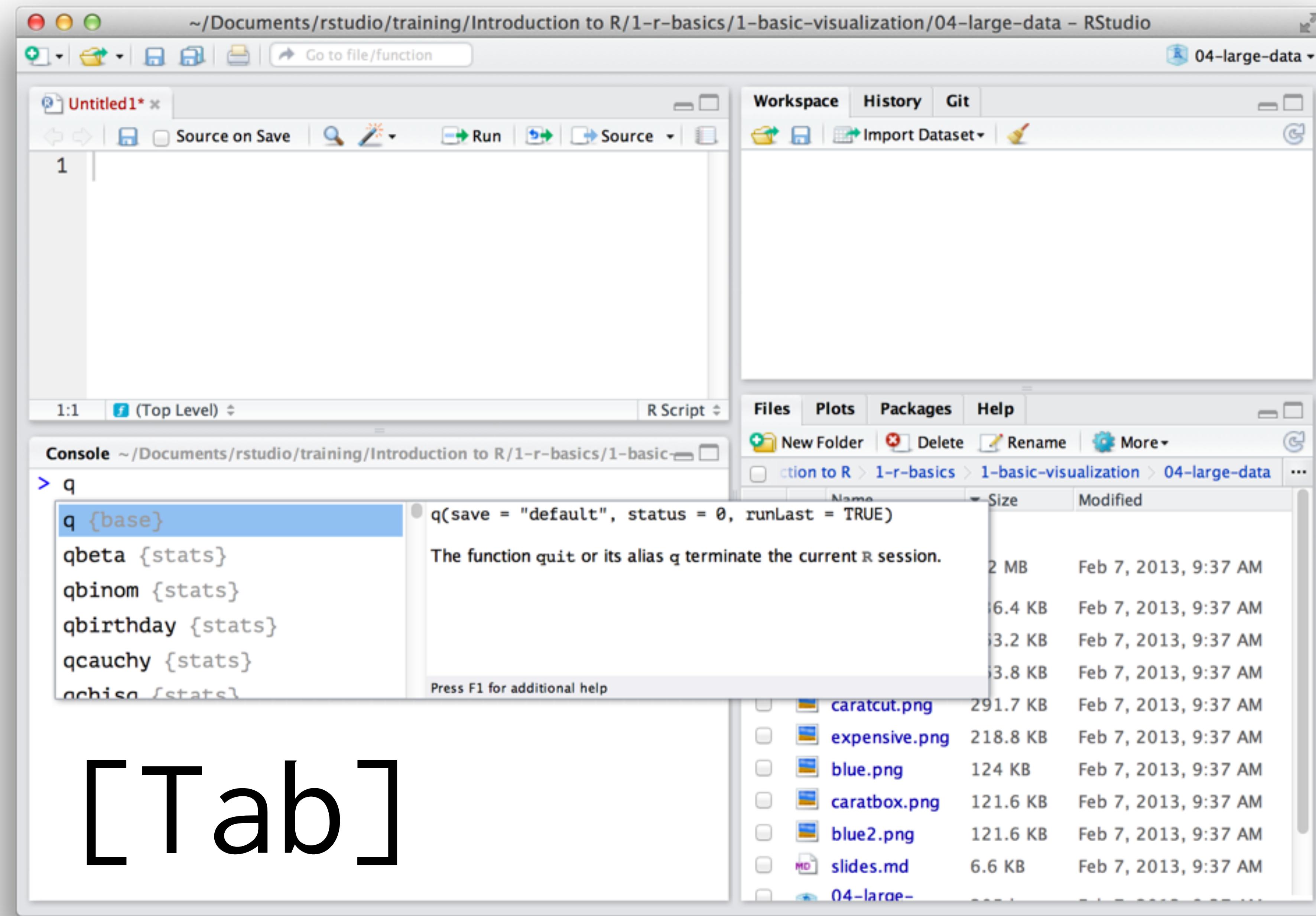
```
1 log10(1000) + log10(100)
```
- Console:** Shows the output of the R code:

```
1:25 (Top Level) R Script
> 1 + 1
[1] 2
> 1 + 2
[1] 3
> 1 + 3
[1] 4
> factorial(round(3.1415) + 1
+ )
[1] 24
> log10(1000) + log10(100)
[1] 5
>
```
- File Explorer:** Shows the current directory structure under Home:
 - .RData
 - .Rhistory
 - .Rprofile
 - bin
 - Desktop
 - Documents
 - Downloads
 - Dropbox
 - Library
 - Movies
 - Music
 - Pictures

Shortcuts







The screenshot shows the RStudio interface with the following details:

- Code Editor (Left Panel):** An R script named "Untitled1" is open. The code contains several `qplot` commands and a single-line comment. The last command, `qplot(mpg, wt, data = mtcars, colour = cyl)`, is highlighted with a blue selection bar.
- File Browser (Right Panel):** The "Files" tab is selected. The sidebar shows the directory path: "action to R > 1-r-basics > 1-basic-visualization > 04-large-data". The main area lists files with their names, sizes, and modification dates. The files listed are:

Name	Size	Modified
..		
04-large-data.html	4.2 MB	Feb 7, 2013, 9:37 AM
overplot.png	936.4 KB	Feb 7, 2013, 9:37 AM
transparent.png	863.2 KB	Feb 7, 2013, 9:37 AM
small.png	463.8 KB	Feb 7, 2013, 9:37 AM
caratcut.png	291.7 KB	Feb 7, 2013, 9:37 AM
expensive.png	218.8 KB	Feb 7, 2013, 9:37 AM
blue.png	124 KB	Feb 7, 2013, 9:37 AM
caratbox.png	121.6 KB	Feb 7, 2013, 9:37 AM
blue2.png	121.6 KB	Feb 7, 2013, 9:37 AM
slides.md	6.6 KB	Feb 7, 2013, 9:37 AM
04-large-		
- Bottom Left Overlay:** A large black bracket containing the text "[Cmd + Up]" is overlaid on the bottom-left corner of the screen, indicating a keyboard shortcut for navigating through the file list.

Untitled1* ~/Documents/rstudio/training/Introduction to R/1-r-basics/1-basic-visualization/04-large-data - RStudio

Go to file/function 04-large-data

Untitled1* Source on Save Run Source

1 library(ggplot2)

1:17 (Top Level) R Script

Console ~/Documents/rstudio/training/Introduction to R/1-r-basics/1-basic-visualization/04-large-data

>

Workspace History Git

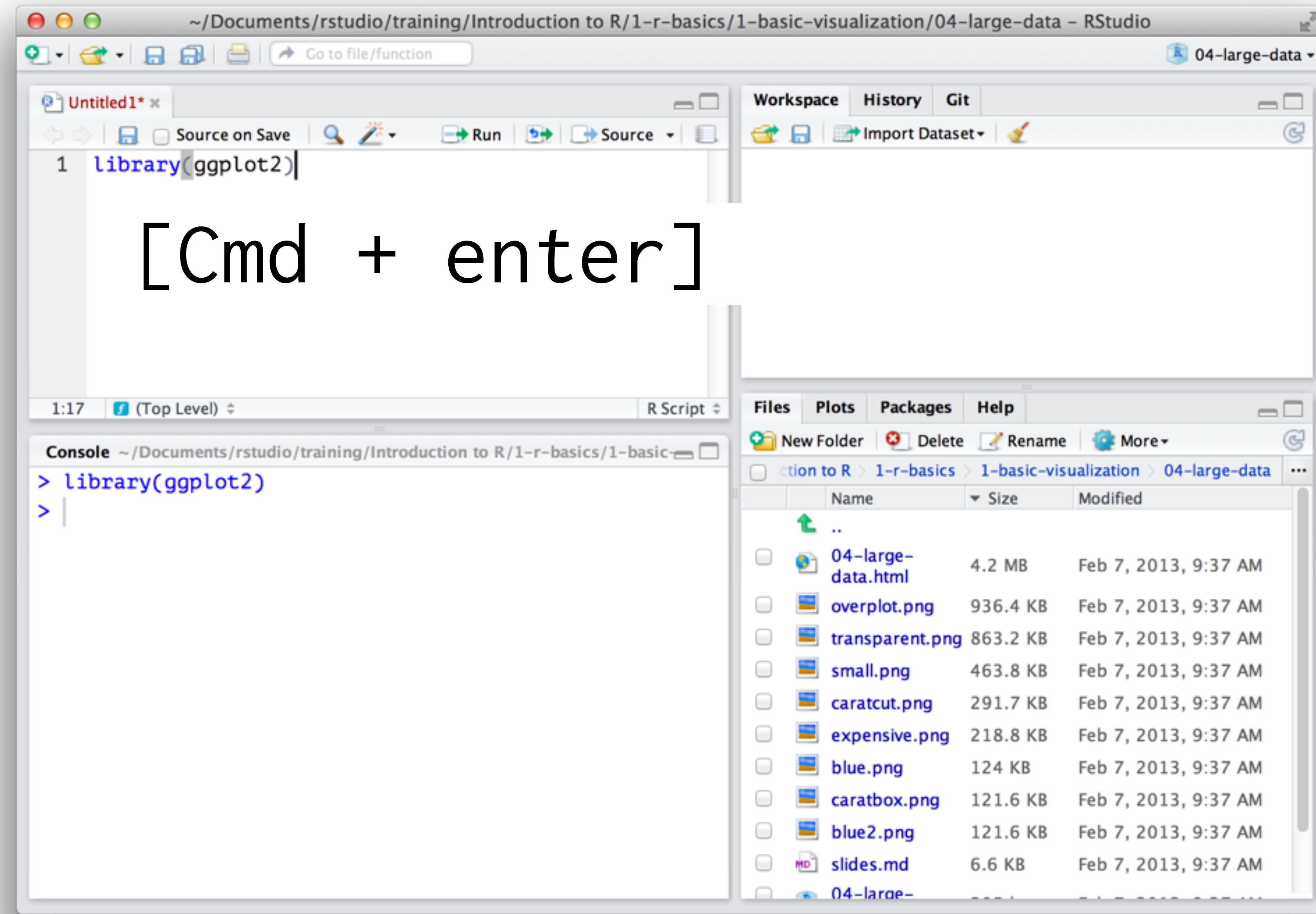
Import Dataset

Files Plots Packages Help

New Folder Delete Rename More

action to R > 1-r-basics > 1-basic-visualization > 04-large-data ...

	Name	Size	Modified
..			
	04-large-data.html	4.2 MB	Feb 7, 2013, 9:37 AM
	overplot.png	936.4 KB	Feb 7, 2013, 9:37 AM
	transparent.png	863.2 KB	Feb 7, 2013, 9:37 AM
	small.png	463.8 KB	Feb 7, 2013, 9:37 AM
	caratcut.png	291.7 KB	Feb 7, 2013, 9:37 AM
	expensive.png	218.8 KB	Feb 7, 2013, 9:37 AM
	blue.png	124 KB	Feb 7, 2013, 9:37 AM
	caratbox.png	121.6 KB	Feb 7, 2013, 9:37 AM
	blue2.png	121.6 KB	Feb 7, 2013, 9:37 AM
	slides.md	6.6 KB	Feb 7, 2013, 9:37 AM
	04-large-		



[Cmd + enter]

R objects

Save information as an R object with the greater than sign followed by a minus, e.g, an arrow: <-

```
foo <- 42
```

Save information as an R object with the greater than sign followed by a minus, e.g, an arrow: <-

name of new
object

```
foo <- 42
```

Save information as an R object with the greater than sign followed by a minus, e.g, an arrow: <-

assignment
operator,
"gets"

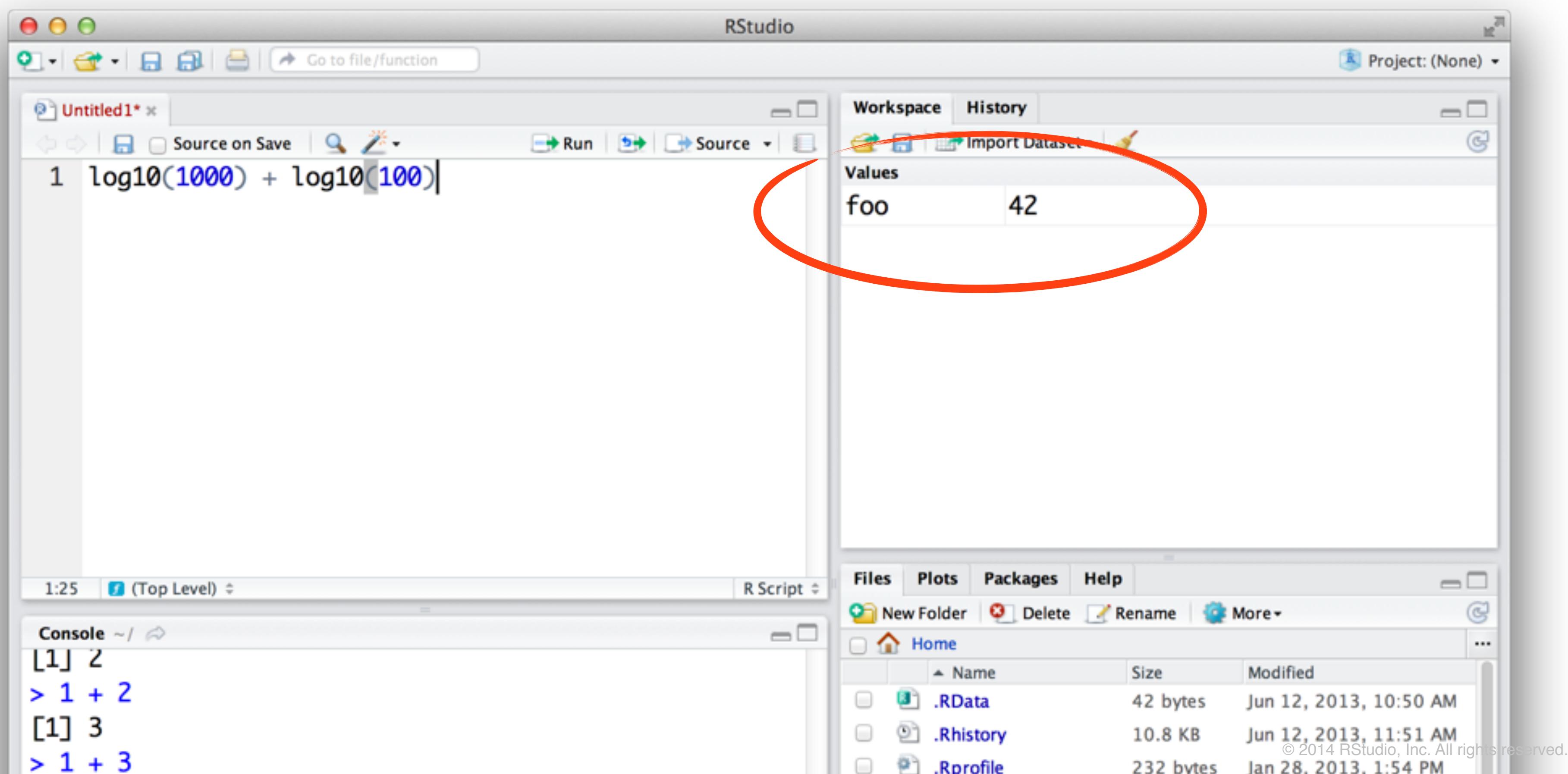
```
foo <- 42
```

Save information as an R object with the greater than sign followed by a minus, e.g, an arrow: <-

information to
store in the
object

```
foo <- 42
```

When you create an R object, you'll see it appear in your workspace pane



Common R workflow

Save output of one function as an R object
to use in a second function.

```
foo <- round(3.1415) + 1
```

```
foo
```

```
# 4
```

```
factorial(foo)
```

```
# 24
```

Object names

Object names cannot begin with numbers.

Wise to avoid names already in use.

a
b
F00
my_var
.day



Capitalization matters

R will treat each of these as a different object

a

b

sum

A

B

SUM

rm

You can remove an object with rm

```
foo  
# 4  
  
rm(foo)  
foo  
# Error: object 'foo' not found
```

This can be useful if you overwrite an object that comes with R

```
pi  
# 3.141593
```

```
pi <- 1  
pi  
# 1
```

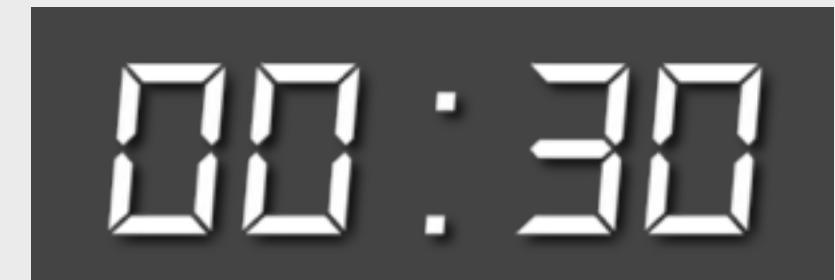
```
rm(pi)  
pi  
# 3.141593
```

Data structures

Warm up

Look at the R object `WorldPhones` (by typing its name in the console).

What is inside of `WorldPhones`?



WorldPhones

	N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
1951	45939	21574	2876	1815	1646	89	555
1956	60423	29990	4708	2568	2366	1411	733
1957	64721	32510	5230	2695	2526	1546	773
1958	68484	35218	6662	2845	2691	1663	836
1959	71799	37598	6856	3000	2868	1769	911
1960	76036	40341	8220	3145	3054	1905	1008
1961	79831	43173	9053	3338	3224	2005	1076

You can save more than a single number in an object by creating a *vector*, *matrix*, or *array*.

vectors

Combine multiple elements into a one dimensional array.

Create with the `c` function.

```
vec <- c(1, 2, 3, 10, 100)
```

```
vec
```

```
# 1 2 3 10 100
```

vectors

multiple elements stored in a one dimensional array.

Create with the c function.

```
vec <- c(1, 2, 3, 10, 100)
```

```
vec
```

```
# 1   2   3   10 100
```



matrices

multiple elements stored in a two dimensional array.

Create with the `matrix` function.

```
mat <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)  
mat  
#      [,1] [,2] [,3]  
# [1,]    1    3    5  
# [2,]    2    4    6
```

matrices

Combine multiple elements into a two dimensional array.

Create with the `matrix` function.

```
mat <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
```

```
mat
```

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



vector of elements to
go in the matrix

```
matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)  
#           [,1] [,2] [,3]  
# [1,]      1     3     5  
# [2,]      2     4     6
```

number of rows for
matrix

```
matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)  
#           [,1] [,2] [,3]  
# [1,]     1     3     5  
# [2,]     2     4     6
```

```
matrix(c(1, 2, 3, 4, 5, 6), nrow = 3)
#           [,1] [,2]
# [1,]      1     4
# [2,]      2     5
# [3,]      3     6
```

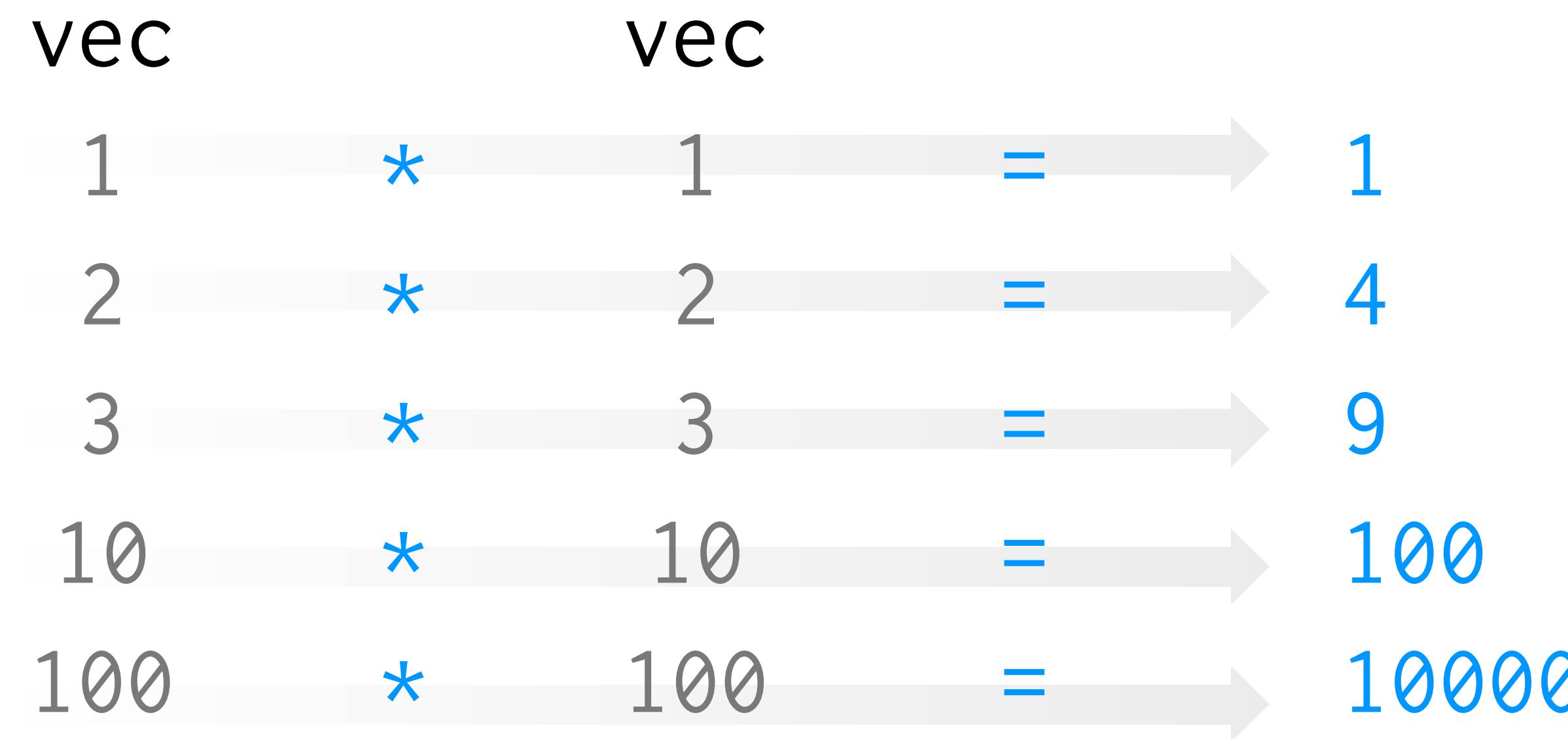
Math: element-wise

```
vec + 4  
# 5   6   7   14 104
```

```
vec * 4  
# 4   8   12  40 400
```

```
vec * vec  
# 1     4     9    100 10000
```

```
vec * vec  
# 1     4     9    100 10000
```



Matrix multiplication

```
vec %*% vec # inner
```

```
#      [,1]  
# [1,] 10114
```

```
vec %o% vec # outer
```

```
#      [,1] [,2] [,3] [,4] [,5]  
# [1,]    1    2    3   10   100  
# [2,]    2    4    6   20   200  
# [3,]    3    6    9   30   300  
# [4,]   10   20   30  100  1000  
# [5,]  100  200  300 1000 10000
```

```
mat
```

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

```
t(mat)
```

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

arrays

Combine multiple elements into an array
that has three or more dimensions.

Create with the array function.

```
array(c(1, 2, 3, 4, 5, 6), dim = c(2, 2, 3))
```

arrays

Combine multiple elements into an array
that has three or more dimensions.

Create with the array function.

```
array(c(1, 2, 3, 4, 5, 6), dim = c(2, 2,
```

We won't focus
on arrays

Data types

Warm up

	A	B	C	D
1	date	president	democrat	unemploy
2	Mar 31, 1968	Lyndon Johnson	TRUE	2709
3	Apr 30, 1968	Lyndon Johnson	TRUE	2740
4	May 31, 1968	Lyndon Johnson	TRUE	2938
5	Jun 30, 1968	Lyndon Johnson	TRUE	2883
6	Jul 31, 1968	Lyndon Johnson	TRUE	2768
7	Aug 31, 1968	Lyndon Johnson	TRUE	2686
8	Sep 30, 1968	Lyndon Johnson	TRUE	2689
9	Oct 31, 1968	Lyndon Johnson	TRUE	2715
10	Nov 30, 1968	Lyndon Johnson	TRUE	2685
11	Dec 31, 1968	Lyndon Johnson	TRUE	2718
12	Jan 31, 1969	Richard Nixon	FALSE	2692
13	Feb 28, 1969	Richard Nixon	FALSE	2712
14	Mar 31, 1969	Richard Nixon	FALSE	2758
15	Apr 30, 1969	Richard Nixon	FALSE	2713
16	May 31, 1969	Richard Nixon	FALSE	2816
17	Jun 30, 1969	Richard Nixon	FALSE	2868
18	Jul 31, 1969	Richard Nixon	FALSE	2868
19	Aug 31, 1969	Richard Nixon	FALSE	2868
20	Sep 30, 1969	Richard Nixon	FALSE	2868
21	Oct 31, 1969	Richard Nixon	FALSE	2868
22	Nov 30, 1969	Richard Nixon	FALSE	2868
...				

What types of data appear
in this spreadsheet?

data types

Like Excel, Numbers, etc., R can recognize different types of data.

We'll look at four basic types:

- numbers
- character strings (text)
- logical
- factor

numeric

Any number, no quotes.

Appropriate for math.

```
1 + 1
```

```
3000000
```

```
class(0.0001)
```

```
# "numeric"
```

character

Any symbols surrounded by quotes.

Appropriate for words, variable names,
messages, any text.

```
"hello"
```

```
class("hello")
```

```
# "character"
```



```
"hello" + "world"  
# Error
```

```
nchar("hello")  
# 5
```

```
paste("hello", "world")  
# "hello world"
```

Warm up

Which of these are numbers? **What
are the others?** How can you tell?

1

"1"

"one"



logical

TRUE or FALSE

R's form of binary data. Useful for logical tests.

```
3 < 4
```

```
# TRUE
```

```
class(TRUE)
```

```
# "logical"
```

```
class(T)
```

```
# "logical"
```

factor

R's form of categorical data. Saved as an integer with a set of labels (e.g. levels).

```
fac <- factor(c("a", "b", "c"))
fac
```

```
# a b c
# Levels: a b c
```

```
class(fac)
# factor
```

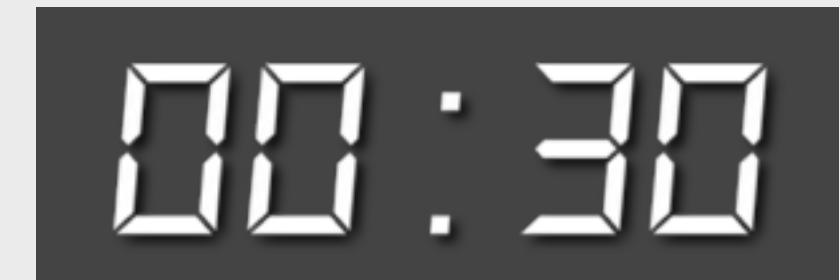
Quiz

`x <- c(1, 2, 3)`

What is the difference between these?

`x`

`"x"`



Type	Examples
numeric	0, 1, -2, 3.1415, 0.0005
character	"gender", "date", "31"
logical	TRUE, FALSE
factor	a c c b Levels: a b c

Aside: dates and times

Surprisingly difficult for computers!

We won't cover, but I recommend the following resources

<http://www.r-statistics.com/2012/03/do-more-with-dates-and-times-in-r-with-lubridate-1-1-0/>

<http://www.jstatsoft.org/v40/i03/>

Your turn

Make a vector that contains the number 1,
the letter R, and the logical TRUE.

What class of data is the vector?

```
vec <- c(1, "R", TRUE)  
class(vec)  
# "character"
```

```
vec  
# "1"      "R"      "TRUE"
```

```
# What is R doing?
```

Vector



Vector

1	2	3
---	---	---

Vector

1	2	3
---	---	---

numeric

Vector

"a"	"R"	"b"
-----	-----	-----

character

Vector

TRUE	TRUE	TRUE
------	------	------

logical

Vector

1	"R"	TRUE
---	-----	------

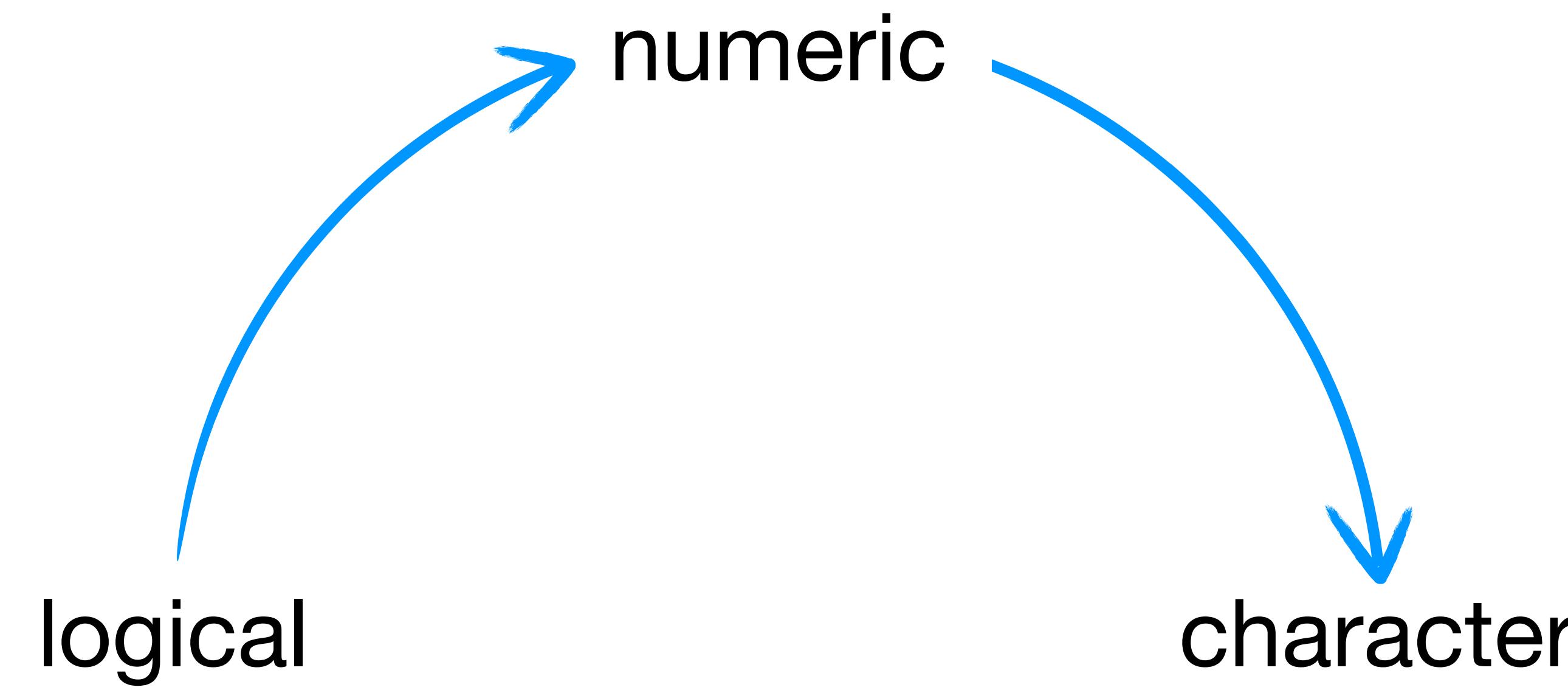
?

Vector

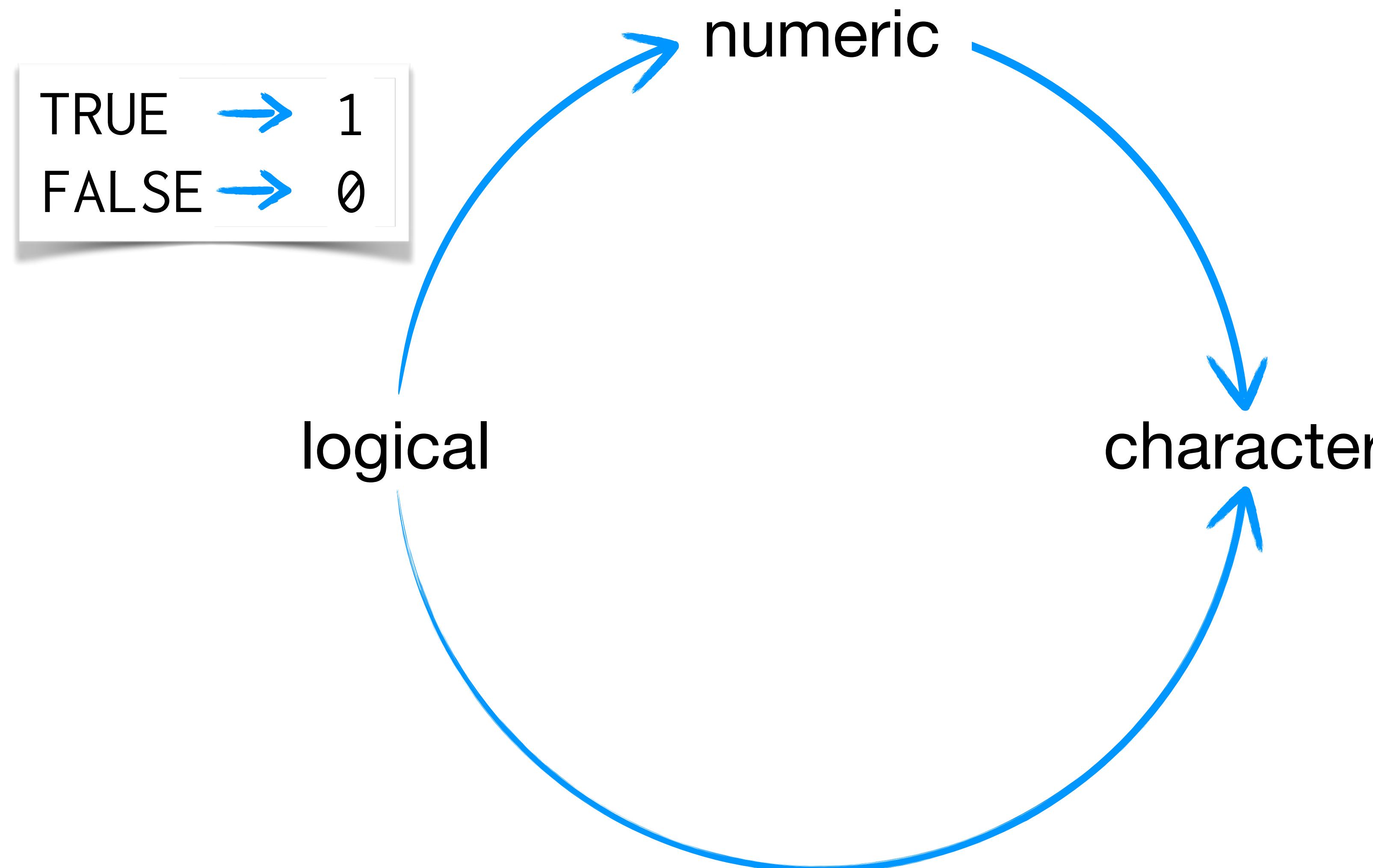
"1"	"R"	"TRUE"
-----	-----	--------

character

coercion

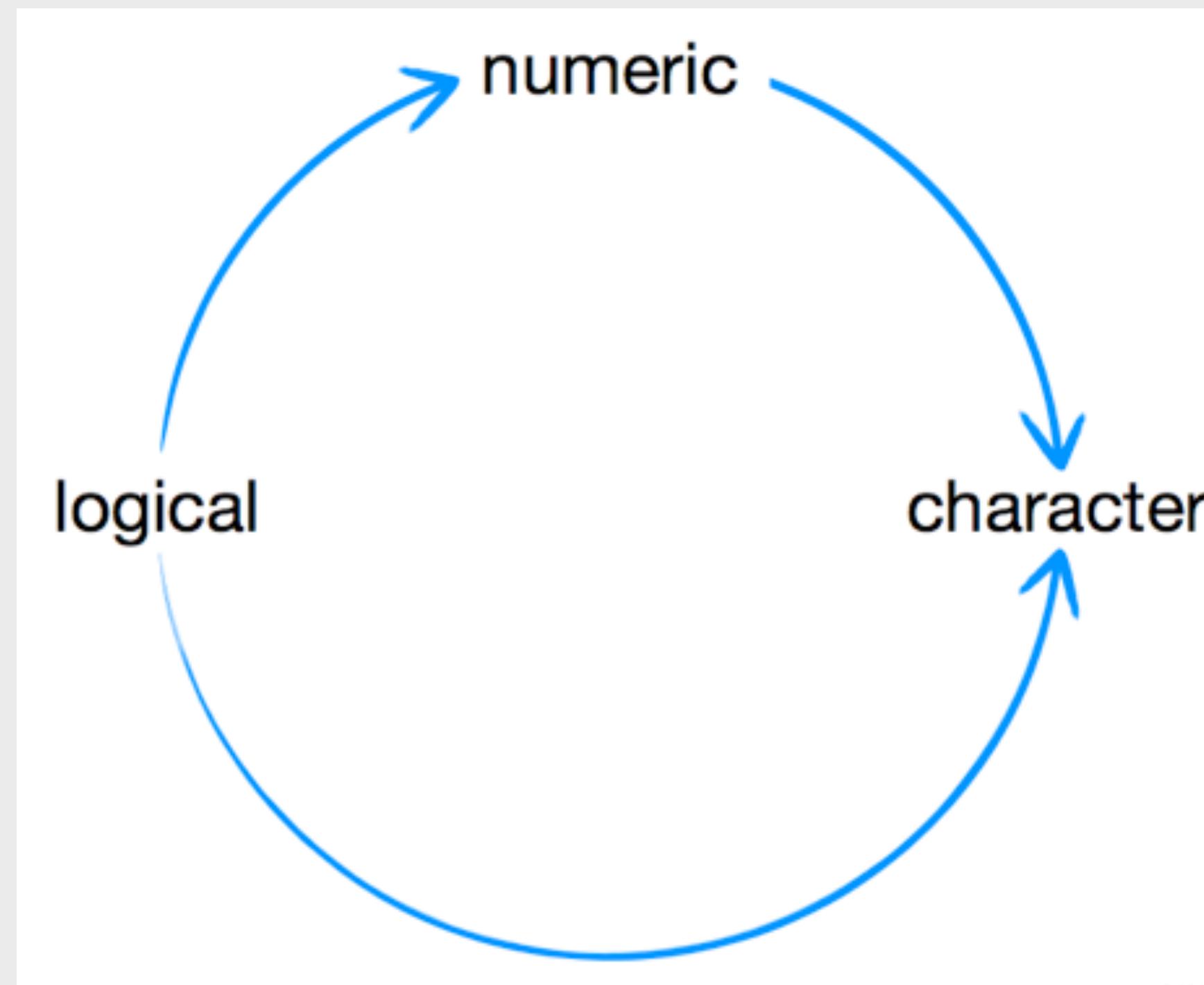


coercion



Quiz

What type of data will result?



`c(5, "two")`

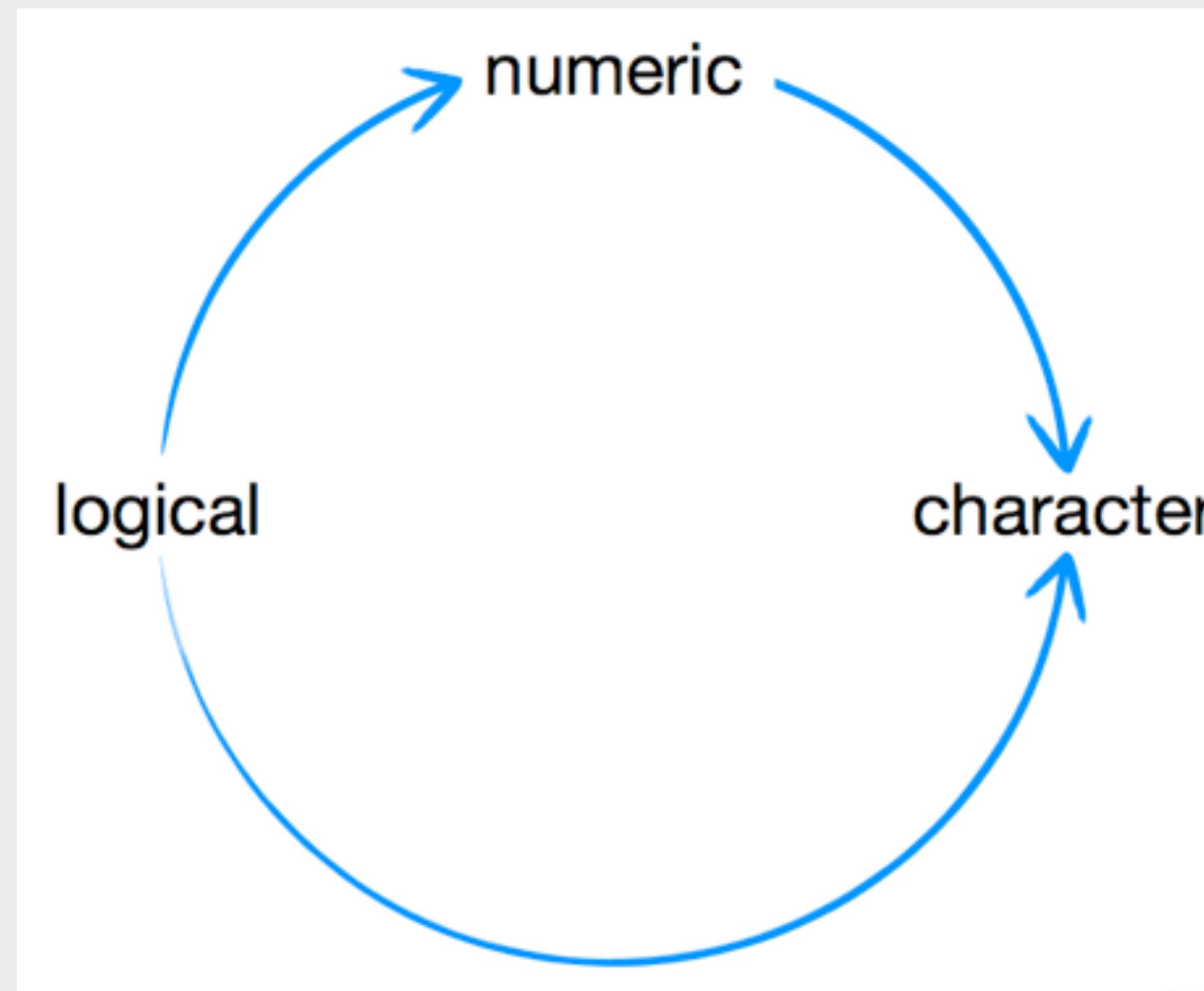
`c(TRUE, "a")`

`c(1, "TRUE")`

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")
character`

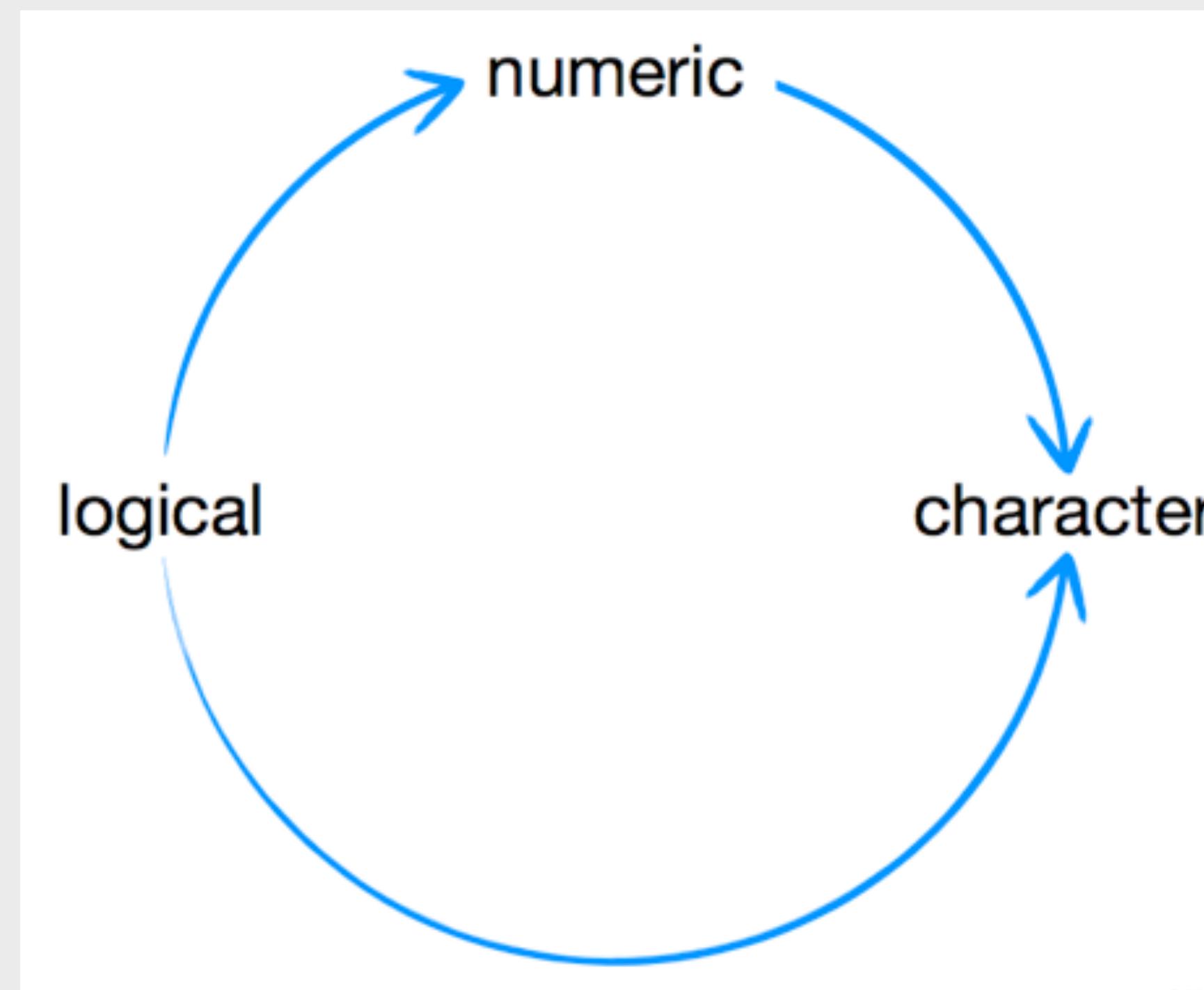
`c(TRUE, "a")`

`c(1, "TRUE")`

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")`
character

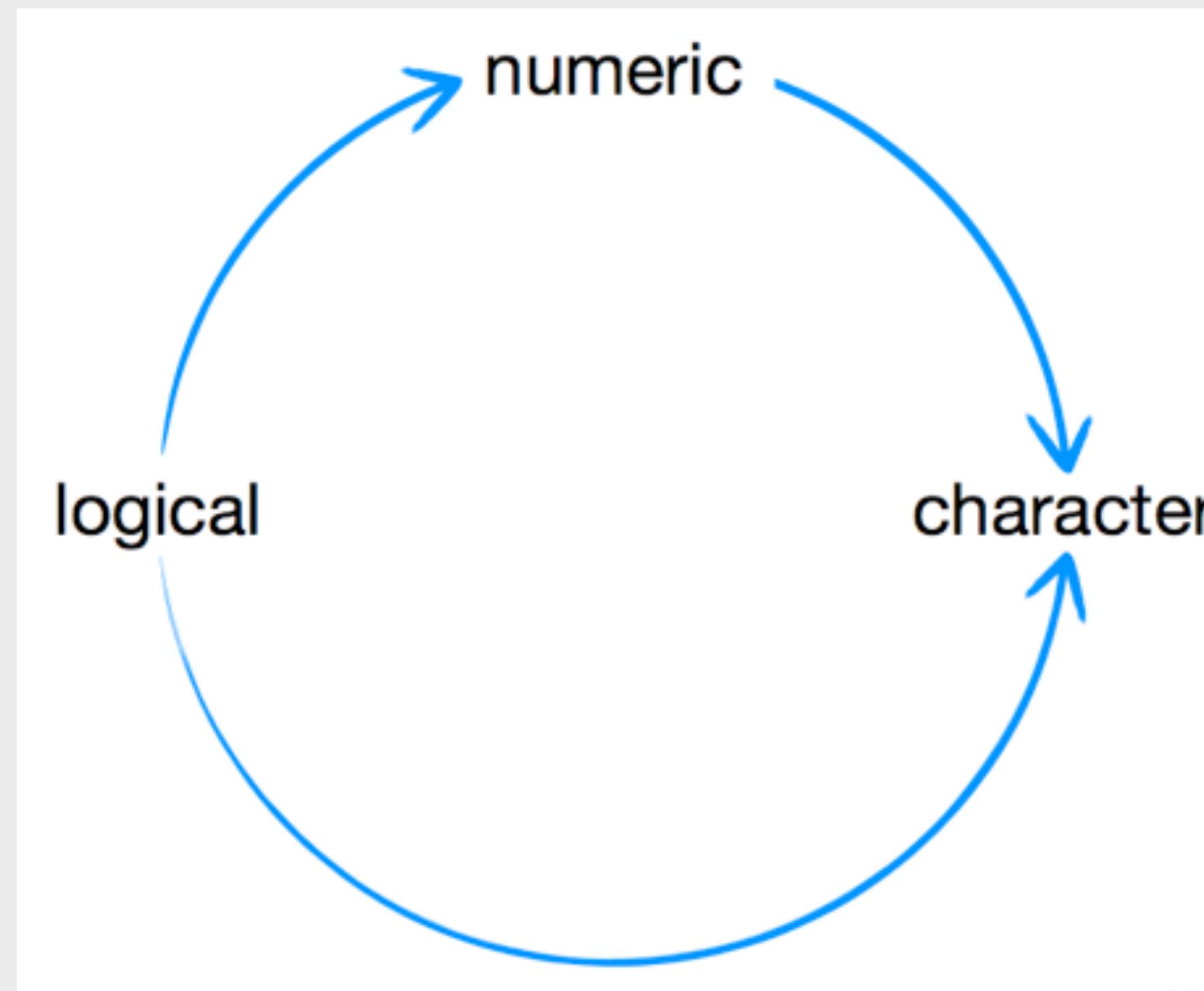
`c(TRUE, "a")`
character

`c(1, "TRUE")`

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")`
character

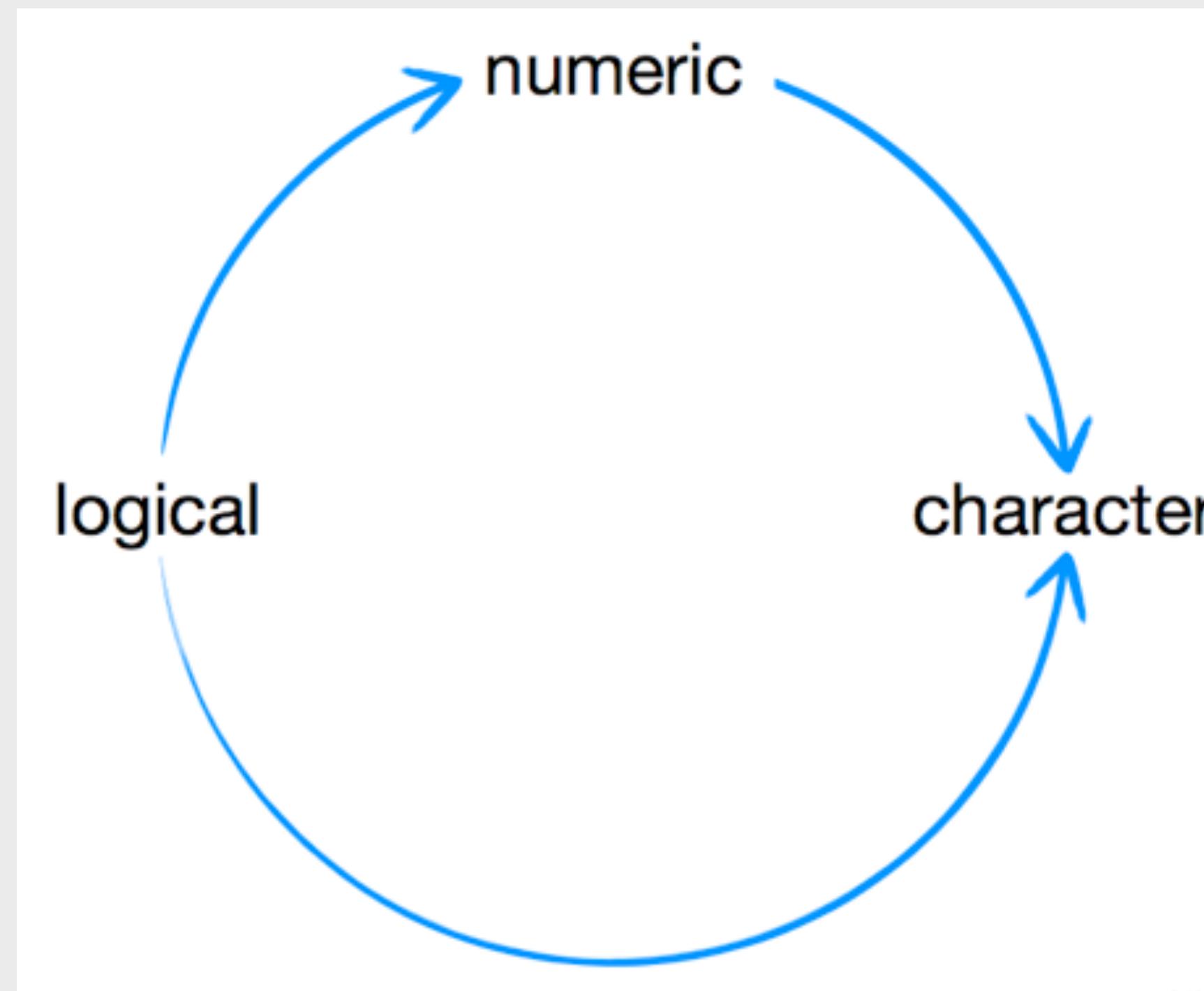
`c(TRUE, "a")`
character

`c(1, "TRUE")`
character

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")`
character

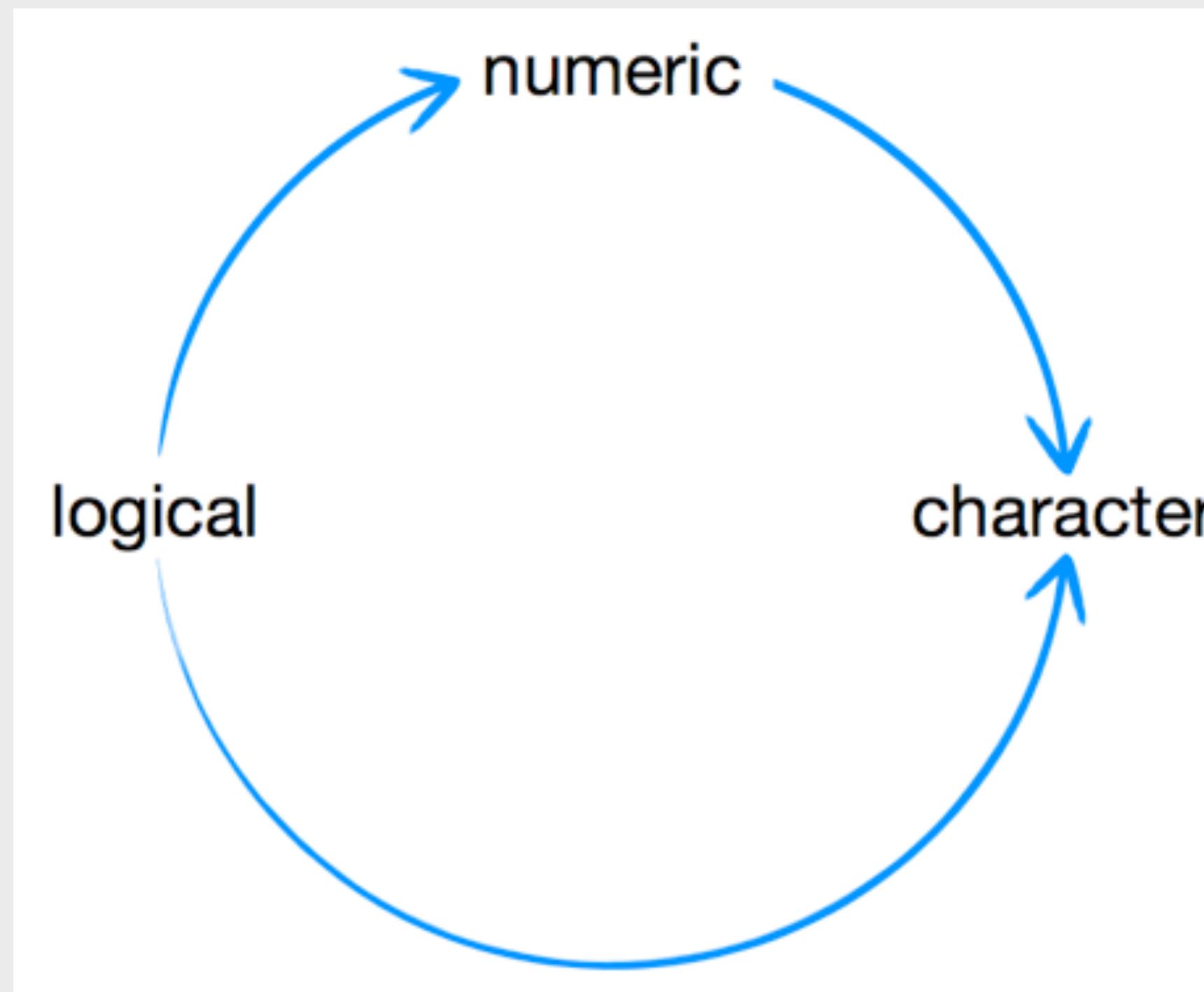
`c(TRUE, "a")`
character

`c(1, TRUE)`
character

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")`
character

`c(TRUE, "a")`
character

`c(1, "TRUE")`
character

`TRUE + 5`
numeric

manual coercion

function	coerces data to
as.numeric	numeric
as.character	character
as.logical	logical
as.factor	factor

```
as.numeric("1")
```

```
as.character(TRUE)
```

Matrix

1	"R"	TRUE
2	"S"	FALSE
3	"T"	TRUE

?

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

What if you want different data types in the same object?

lists and data frames

lists and *data frames* generalize vectors and matrices to allow multiple types of data

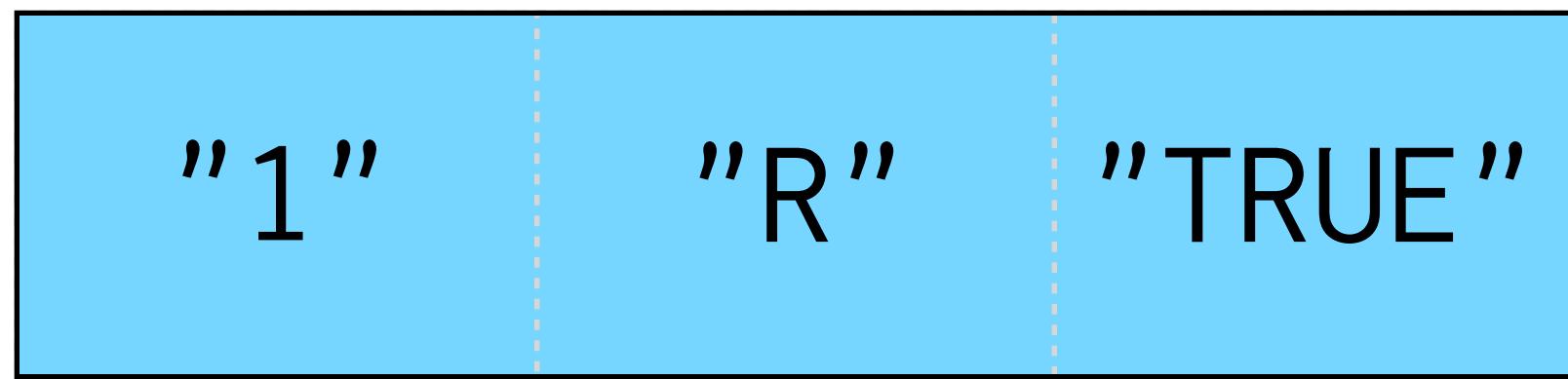
lists

A list is a one dimensional group of R objects.

Create lists with `list`

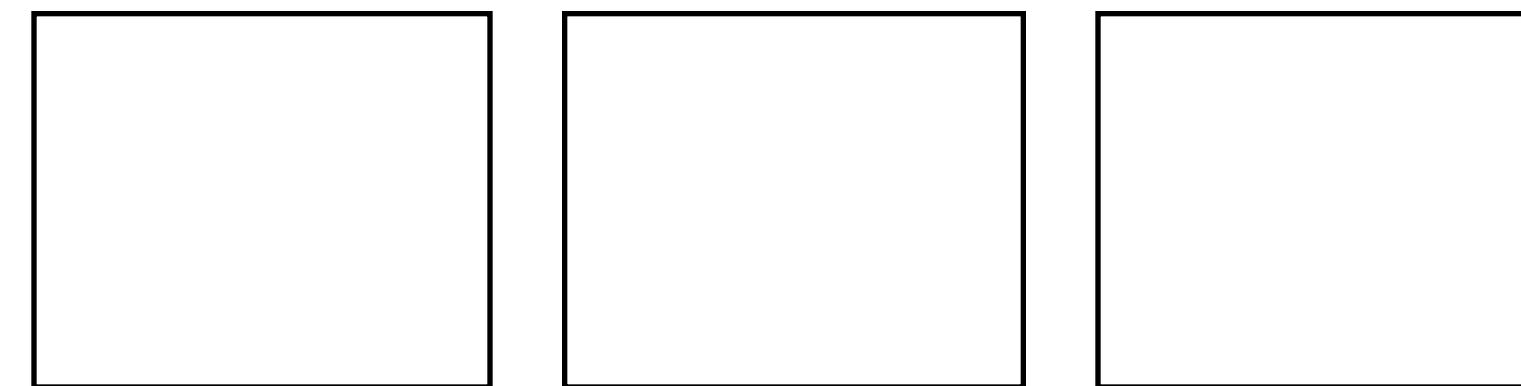
```
lst <- list(1, "R", TRUE)  
class(lst)  
# "list"
```

Vector

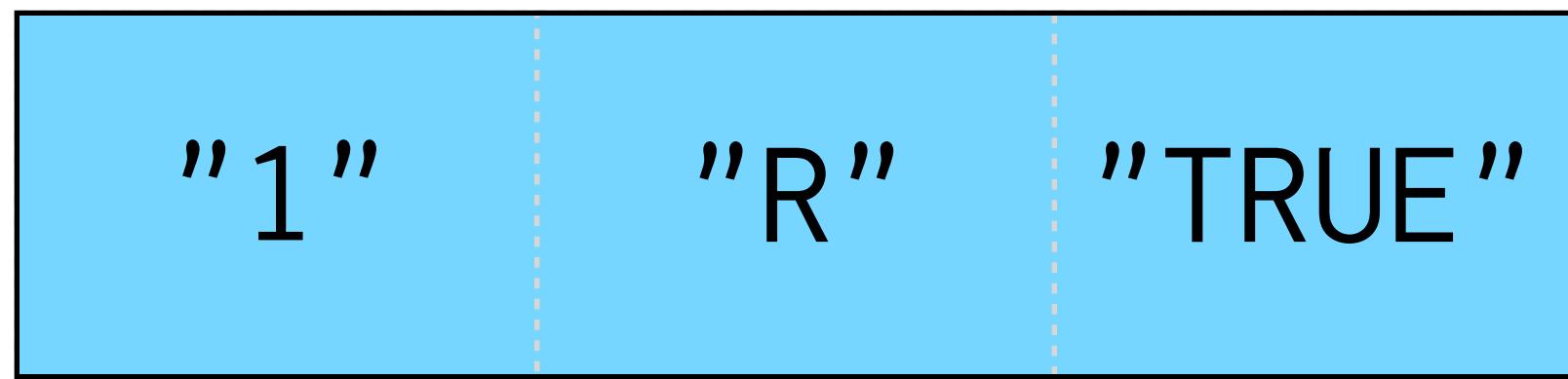


character

List

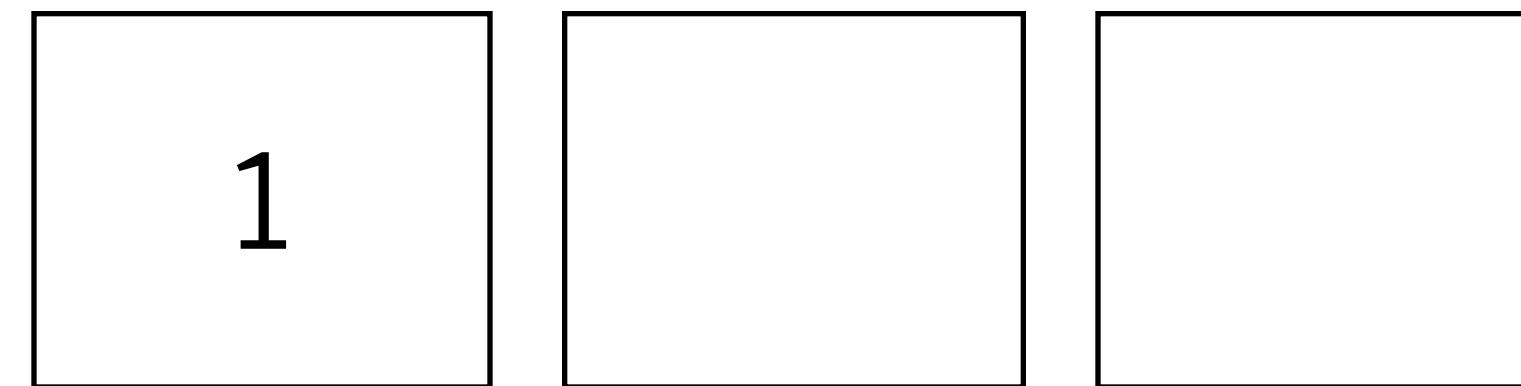


Vector

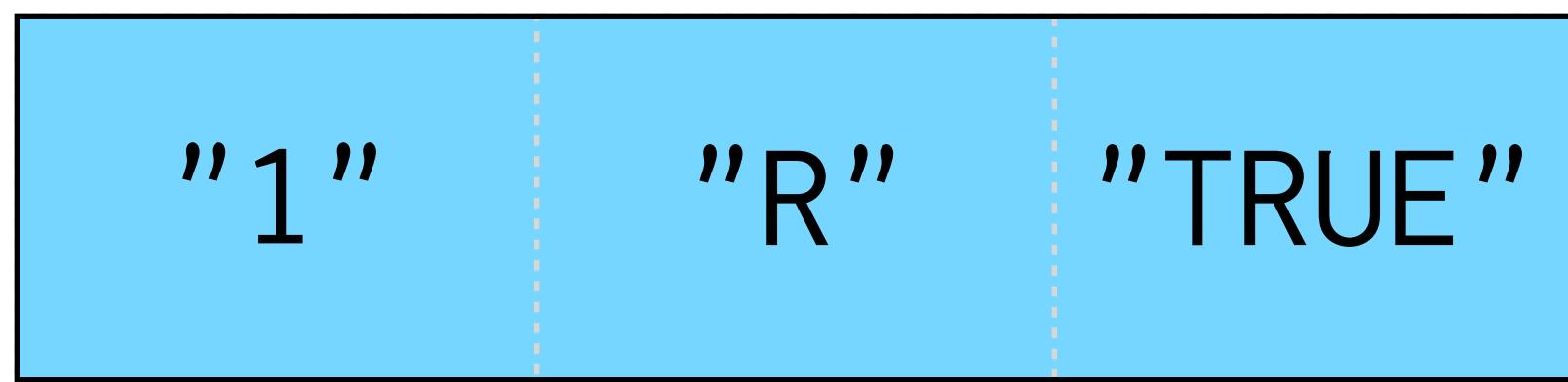


character

List

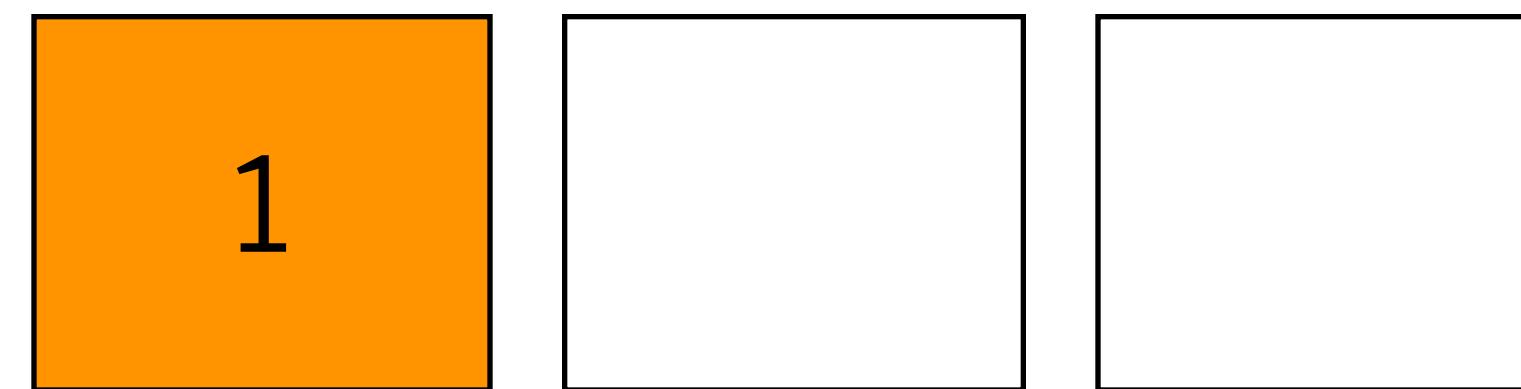


Vector



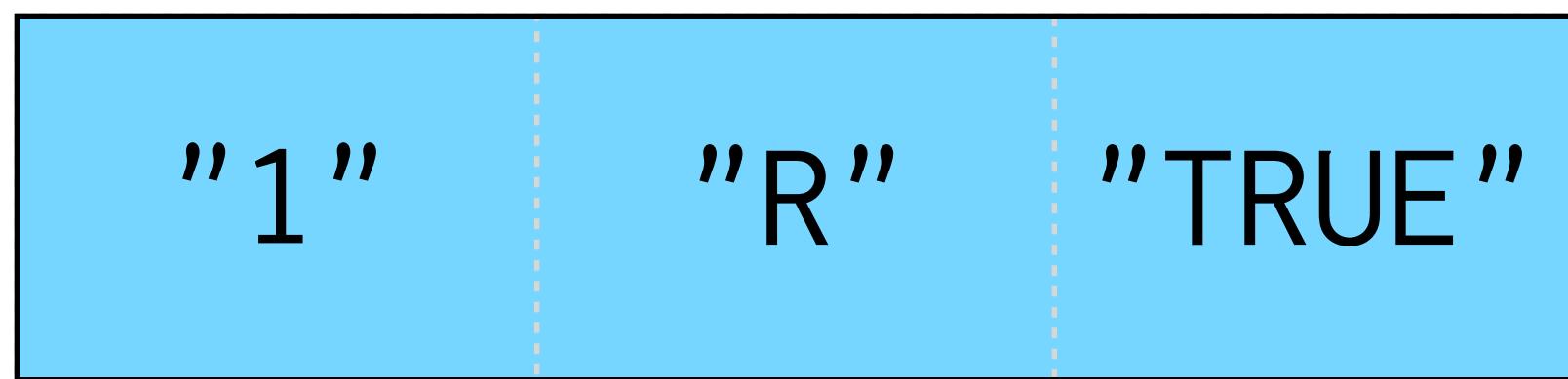
character

List



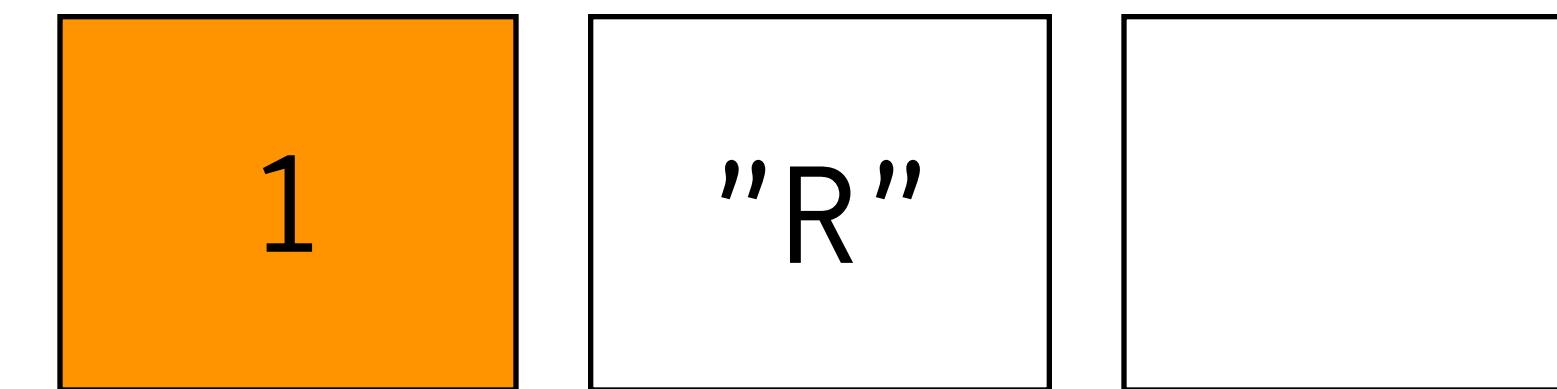
numeric

Vector



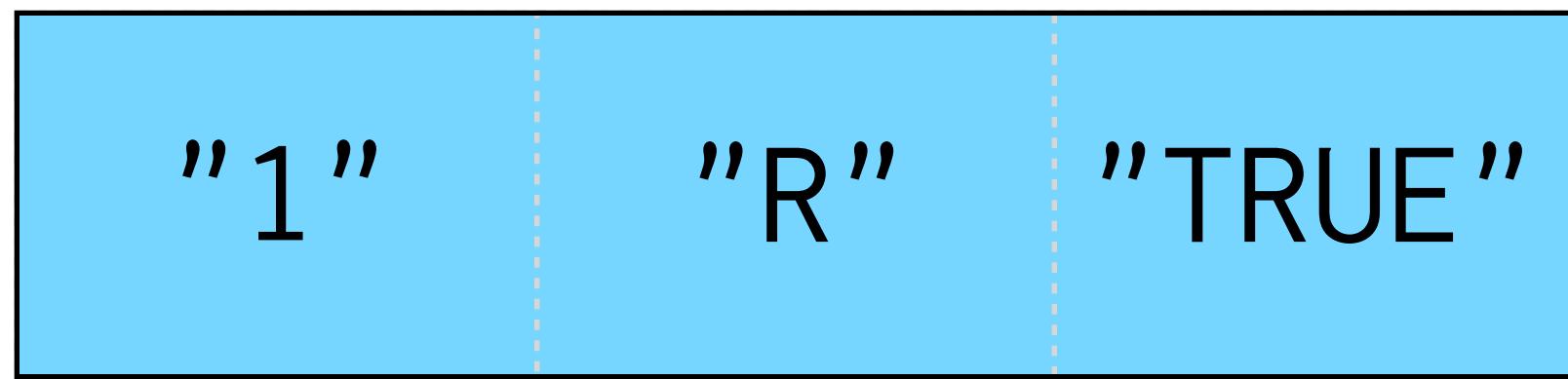
character

List



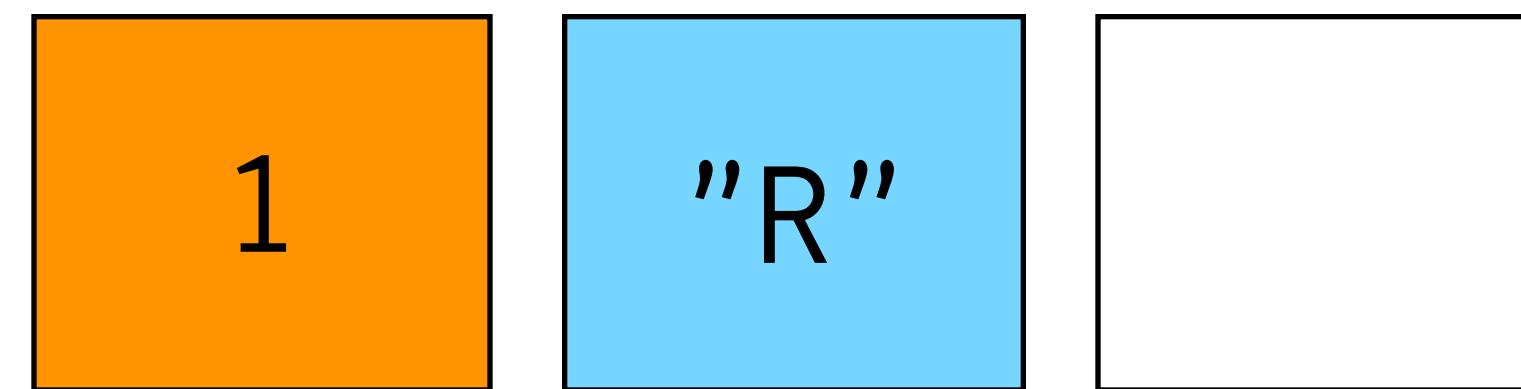
numeric

Vector



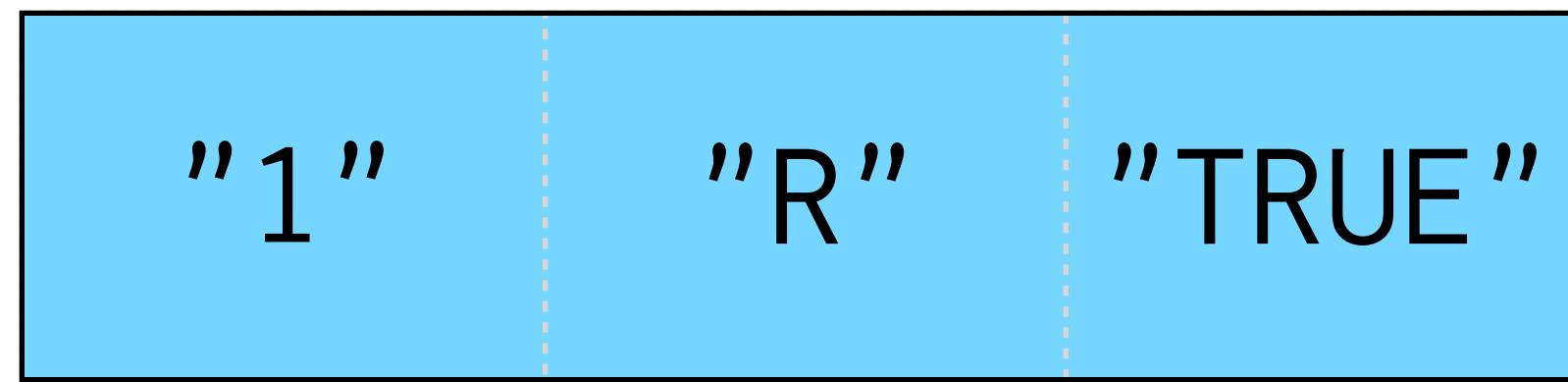
character

List



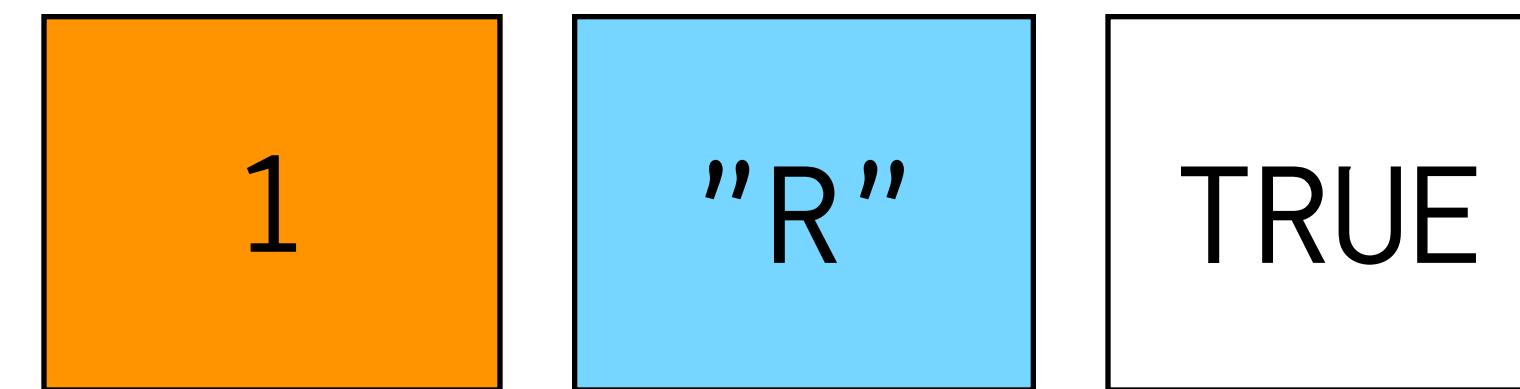
numeric character

Vector



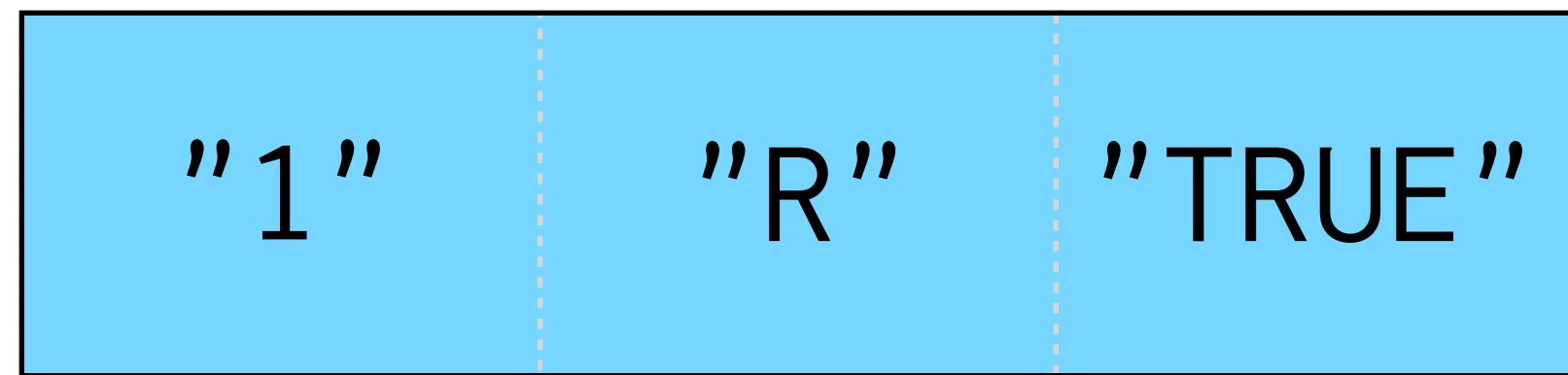
character

List



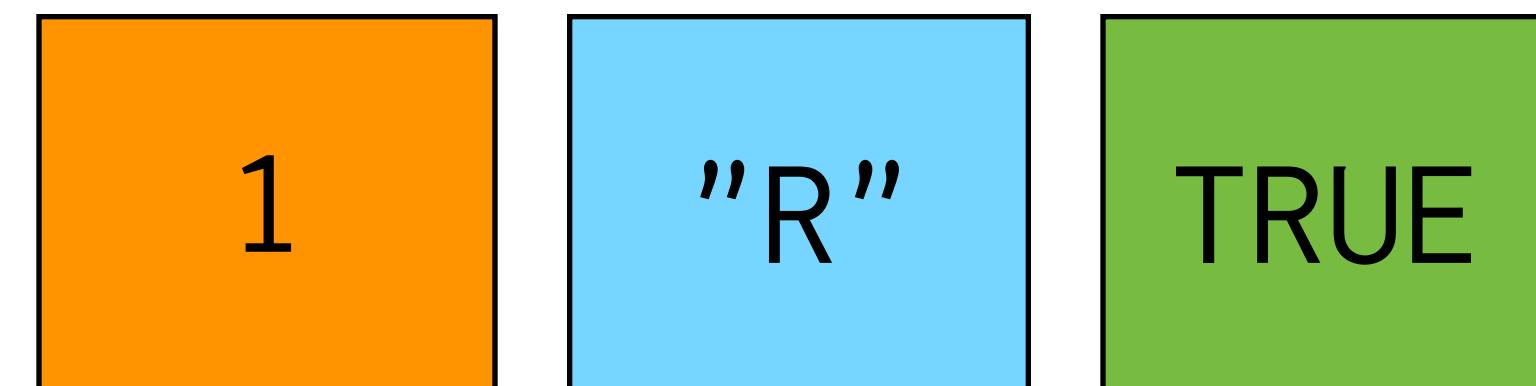
numeric character

Vector



character

List



numeric character logical

The elements of a list can be anything. Even vectors or other lists.

```
list(c(1, 2), TRUE, c("a", "b", "c"))
```

List

c(1, 2)

TRUE

c("a", "b", "c")

data frame

A data frame is a two dimensional group of R objects.

Each column in a data frame can be a different type

```
df <- data.frame(c(1, 2, 3),  
                  c("R", "S", "T"), c(TRUE, FALSE, TRUE))  
  
class(df)  
  
# "data.frame"
```

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1		
2		
3		

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1		
2		
3		

numeric

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1	"R"	
2	"S"	
3	"T"	

numeric

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1	"R"	
2	"S"	
3	"T"	

numeric

character

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1	"R"	TRUE
2	"S"	FALSE
3	"T"	TRUE

numeric

character

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

1	"R"	TRUE
2	"S"	FALSE
3	"T"	TRUE

numeric

character

logical

names

You can name the elements of a vector, list, or data frame when you create them.

```
nvec <- c(one = 1, two = 2, three = 3)
```

```
nvec  
# one  two three  
#   1    2     3
```

```
nlst <- list(one = 1, two = 2,  
             many = c(3, 4, 5))
```

```
nlst  
# $one  
# [1] 1  
#  
# $two  
# [1] 2  
#  
# $many  
# [1] 3 4 5
```

```
ndf <- data.frame(numbers = c(1, 2, 3),  
                   letters = c("R", "S", "T"),  
                   logic = c(TRUE, FALSE, TRUE))
```

```
ndf  
#   numbers letters logic  
# 1       1        R    TRUE  
# 2       2        S   FALSE  
# 3       3        T    TRUE
```

You can also see and set the names with `names`

```
names(ndf)
# [1] "numbers" "letters" "logic"
```

```
names(nvec)
# [1] "one"    "two"    "three"
```

```
names(nvec) <- c("uno", "dos", "tres")
nvec
# uno   dos  tres
# 1     2    3
```

single type

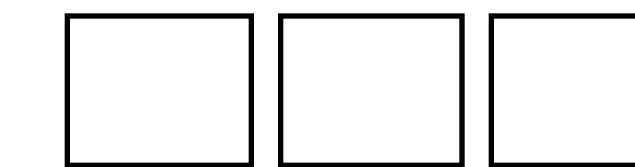
multiple types

1D

Vector

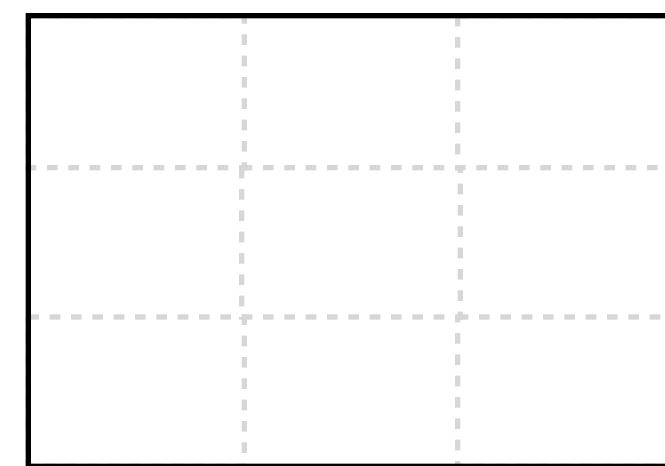


List

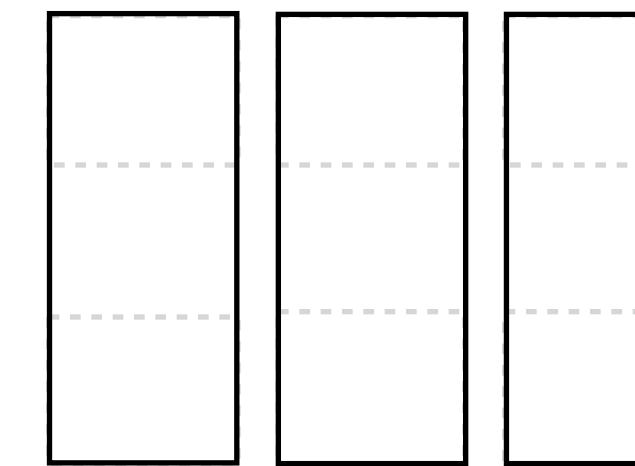


2D

Matrix

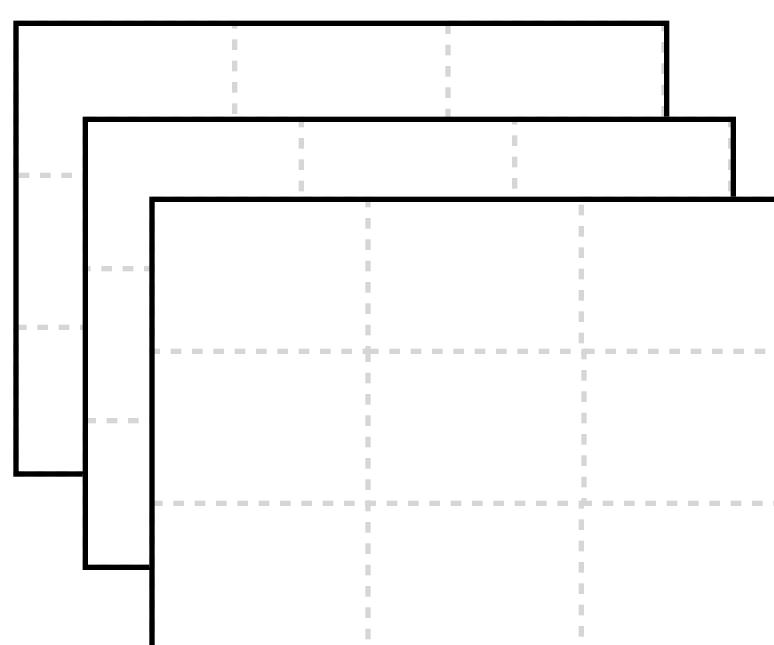


Data frame



nD

Array



helper functions for data structures

	create	change to	check	get names	get dimensions
vector	c, vector	as.vector	is.vector	names	length
matrix	matrix	as.matrix	is.matrix	rownames, colnames	dim, nrow, ncol
array	array	as.array	is.array	dimnames	dim
list	list	as.list	is.list	names	length
data frame	data.frame	as.data.frame	is.data.frame	names	dim, nrow, ncol