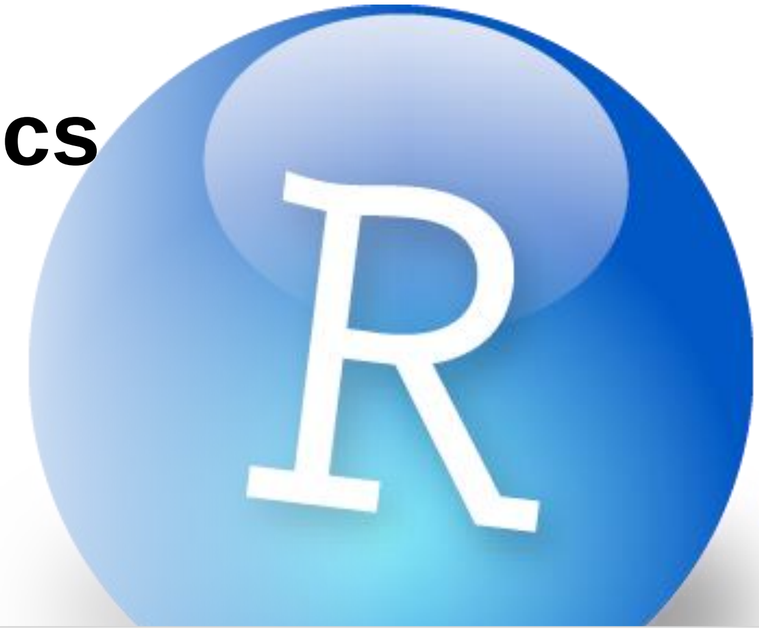
Studio

The R Language Basics

Understand how R stores
and works with data;
R packages



Arthur Petrosian
arthur.petrosian@aua.edu

Spring 2017

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"

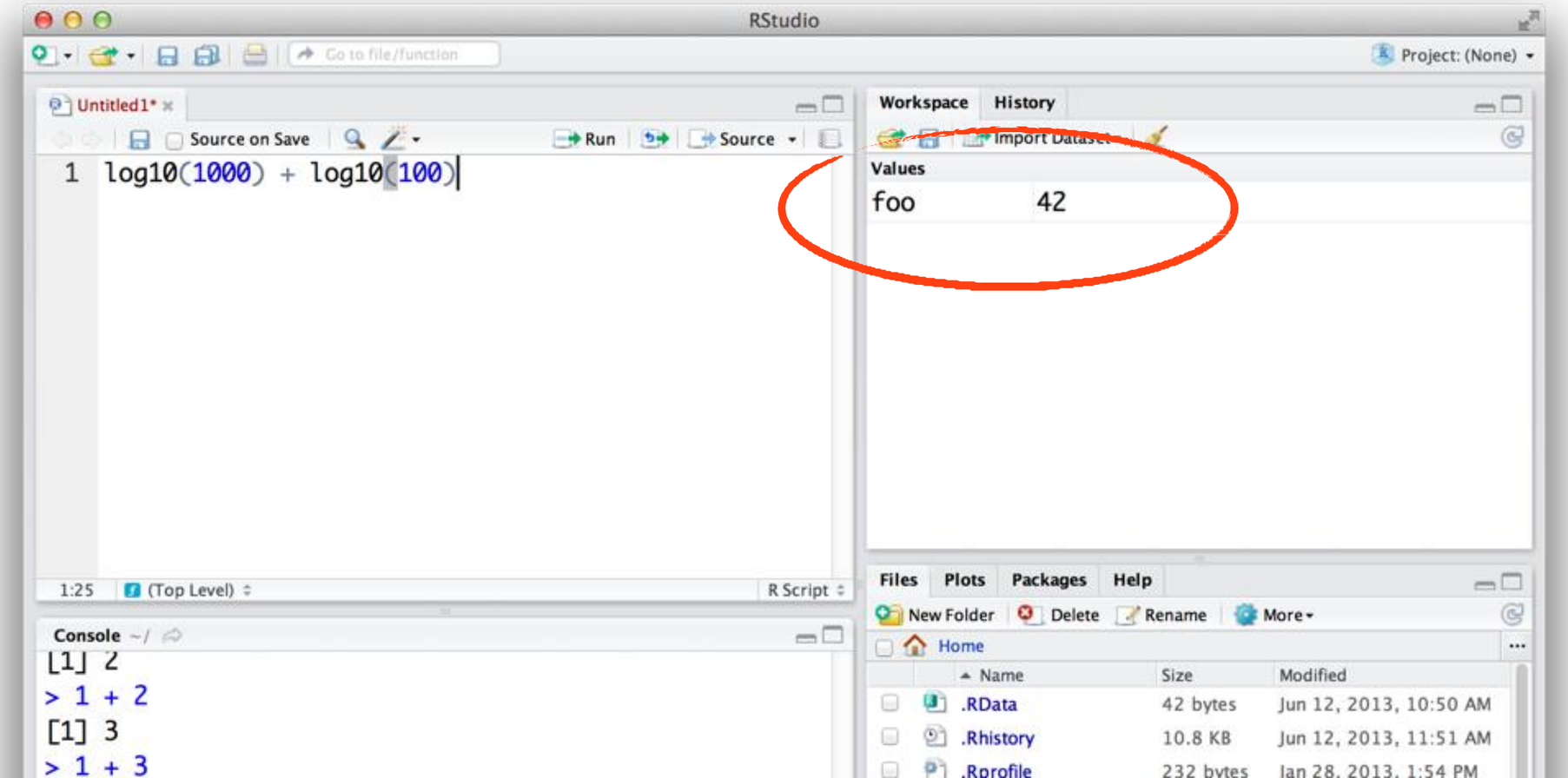
```
foo <- 42
```

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.

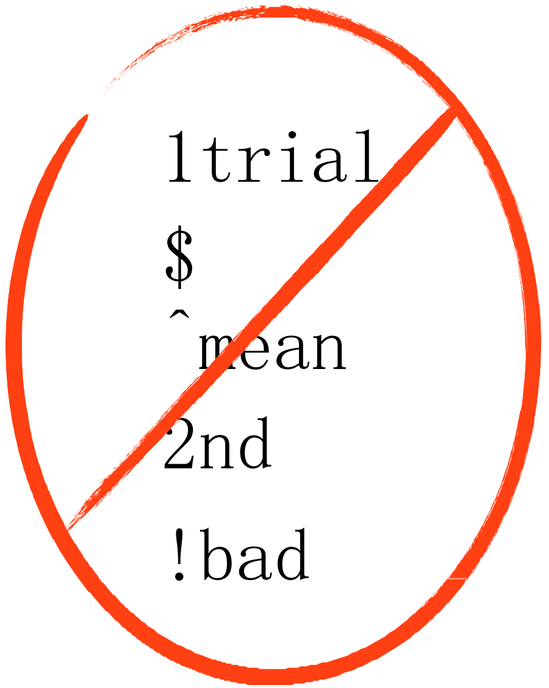
```
foo <- round(3.1415) + 1
foo
# 4

factorial(foo)
# 24
```

Object names

Object names cannot begin with numbers.

Wise to avoid names already in use.

```
a
b
FOO
my_var
.day
```



Capitalization matters

R will treat each of these as a different object

a	A
b	B
sum	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.

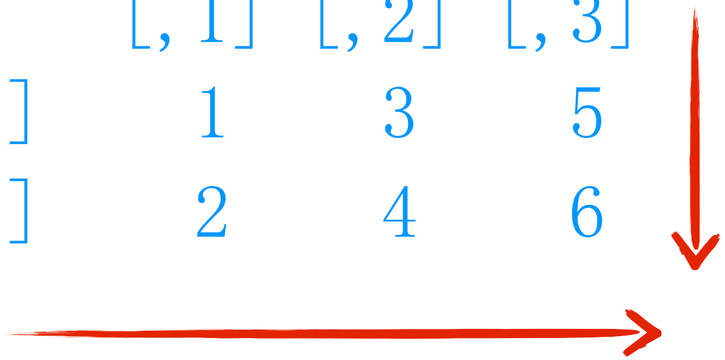
```
mat <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
mat
#      [,1] [,2] [,3]
# [1,]    1    3    5
# [2,]    2    4    6
```

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

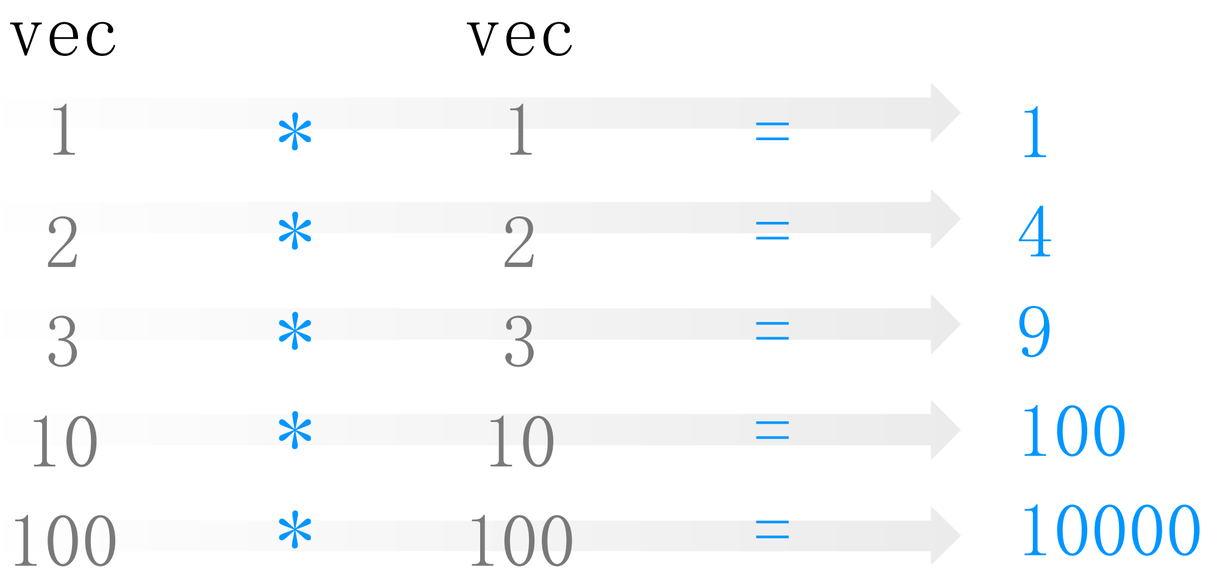
```
matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
#      [,1] [,2] [,3]
# [1,]    1    3    5
# [2,]    2    4    6
```

```
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	286
18	Jul 31, 1969	Richard Nixon	FALSE	
19	Aug 31, 1969	Richard Nixon	FALSE	
20	Sep 30, 1969	Richard Nixon	FALSE	
21	Oct 31, 1969	Richard Nixon	FALSE	
22	Nov 30, 1969	Richard Nixon	FALSE	

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(0.00001)
# "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

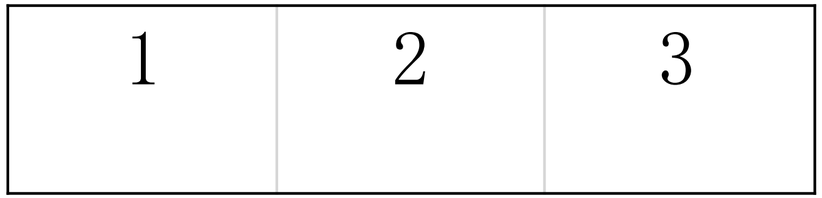
Make a vector that contains the number 1, the letter R, and the logical TRUE.

What class of data is the vector?

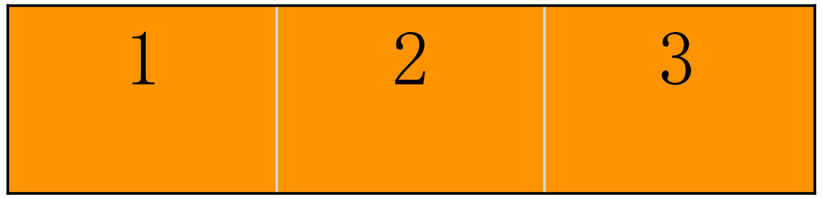
```
vec <- c(1, "R", TRUE)
class(vec)
# "character"

vec
# "1"    "R"    "TRUE"

# What is R doing?
```

Vector

Vector

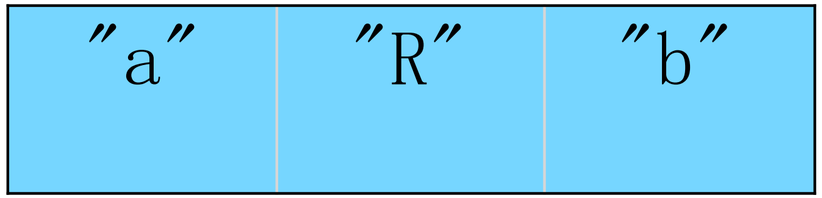


Vector



numeric

Vector



character

Vector

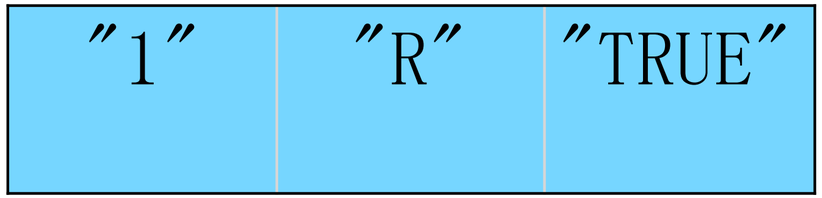
logical

Vector

1	″R″	TRUE

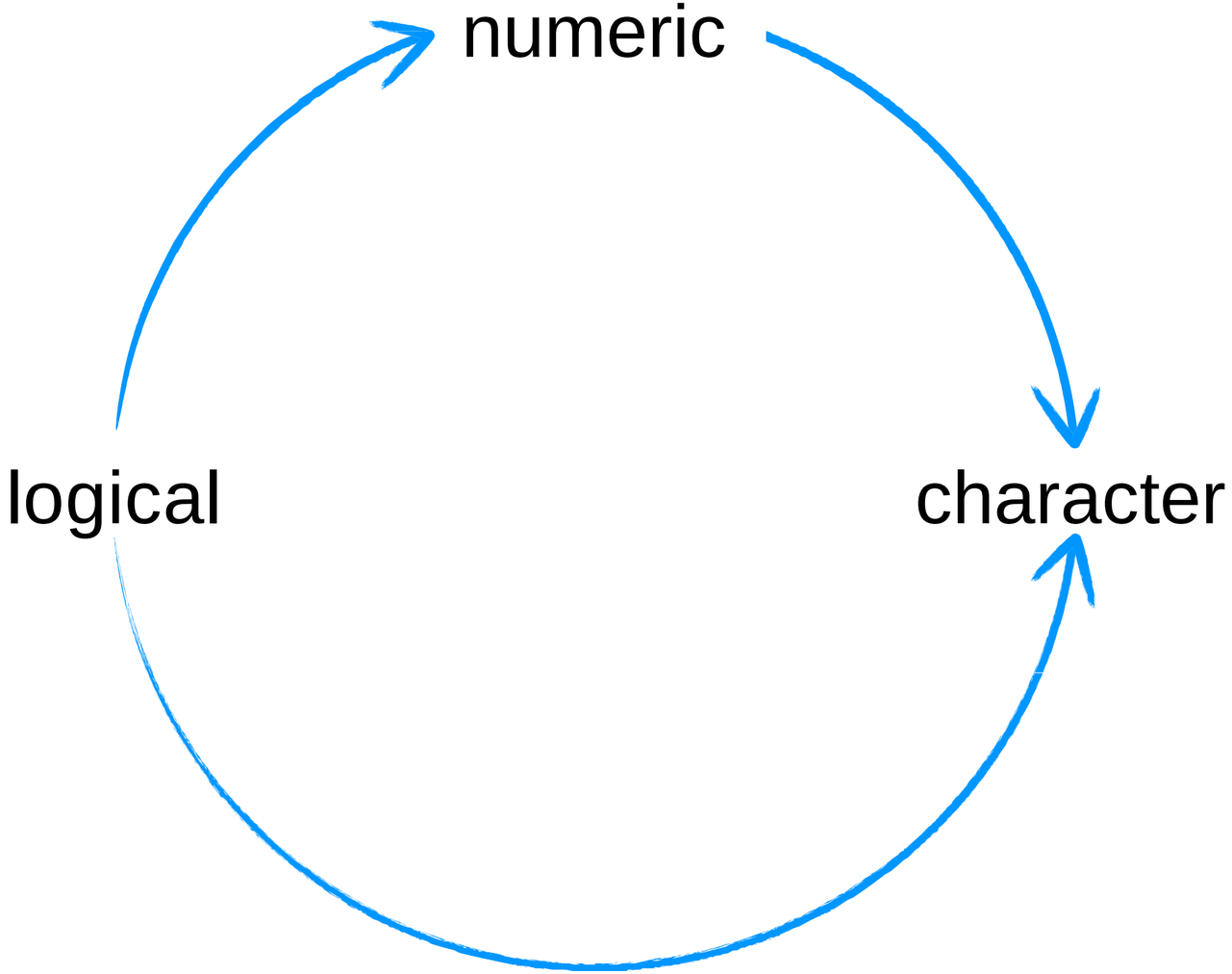
?

Vector



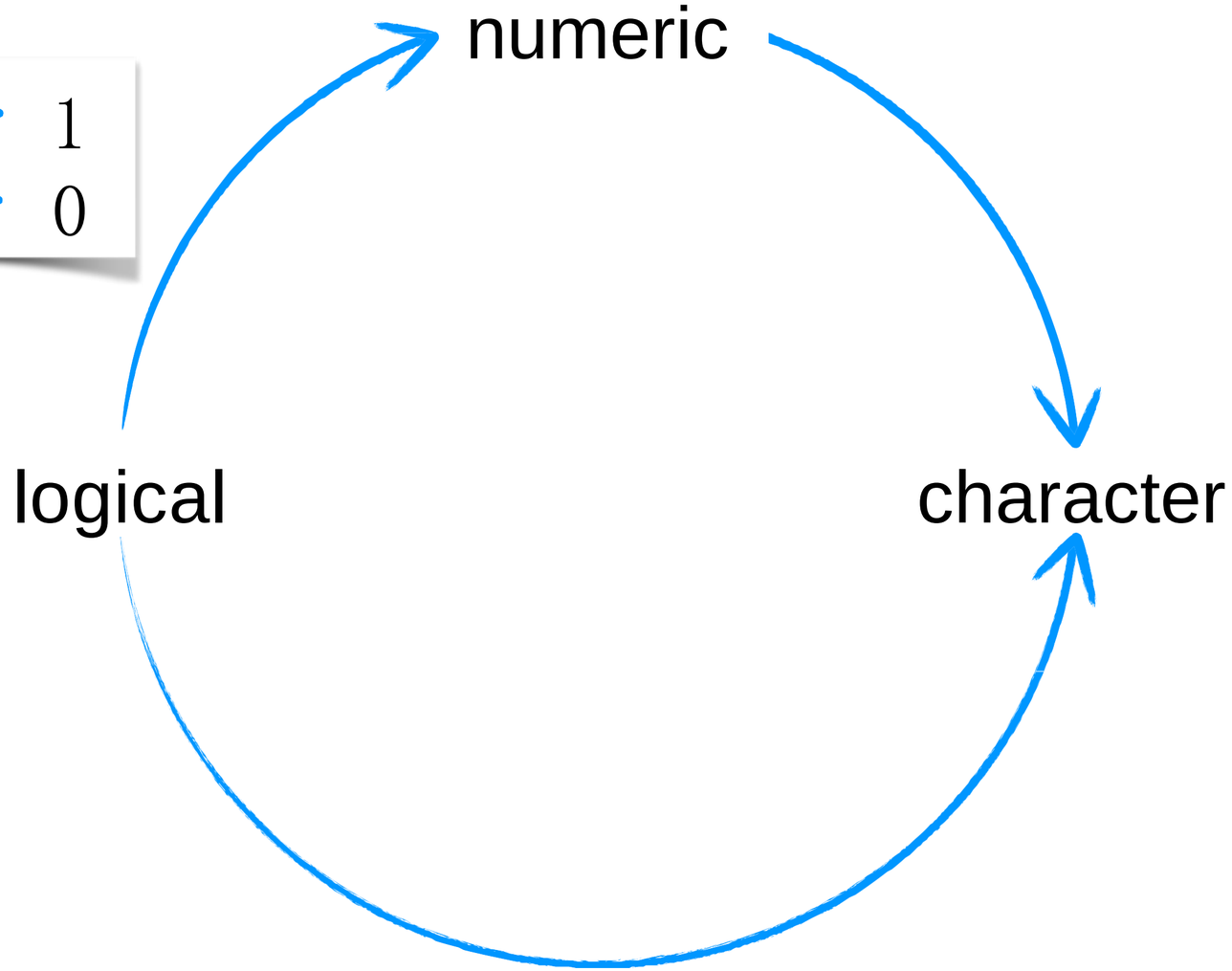
character

coercion



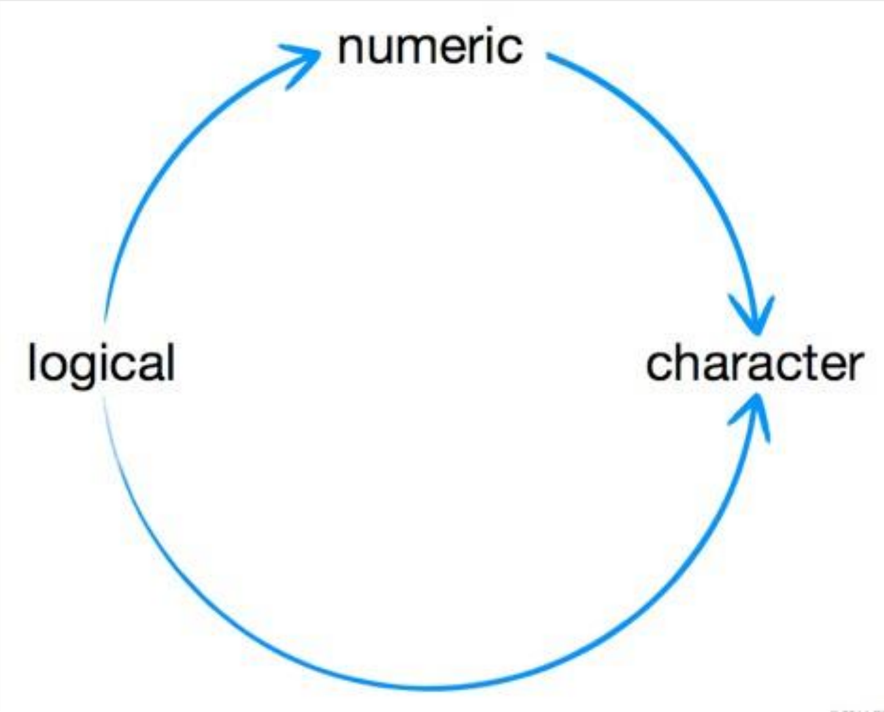
coercion

TRUE → 1
FALSE → 0



Quiz

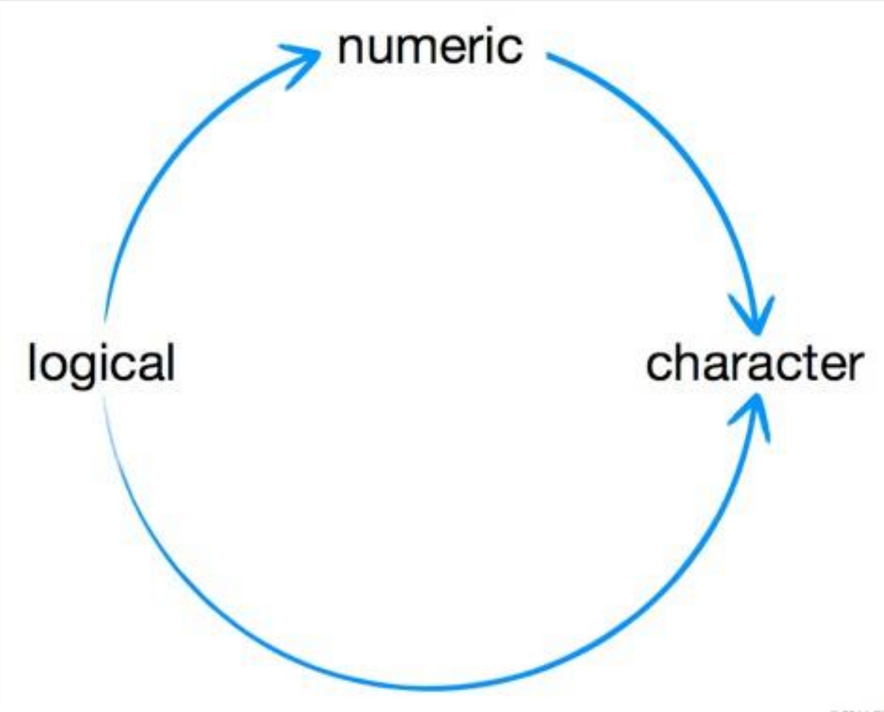
What type of data will result?




```
c(5, "two")

c(TRUE, "a")

c(1, "TRUE")

TRUE + 5
```

Quiz

What type of data will result?



`c(5, "two")`
character

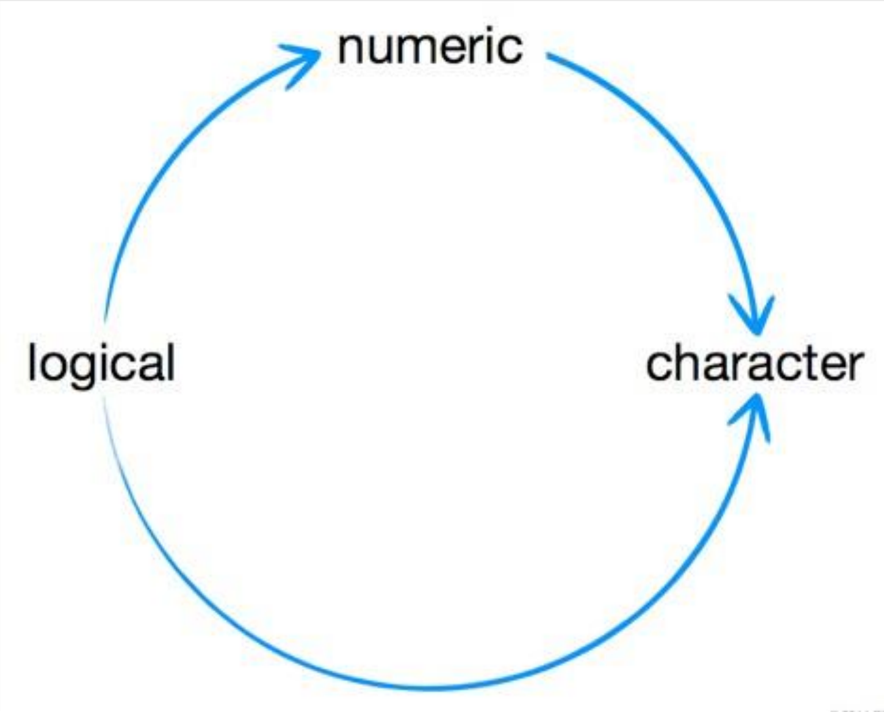
`c(TRUE, "a")`

`c(1, "TRUE")`

`TRUE + 5`

Quiz

What type of data will result?



`c(5, "two")`
character

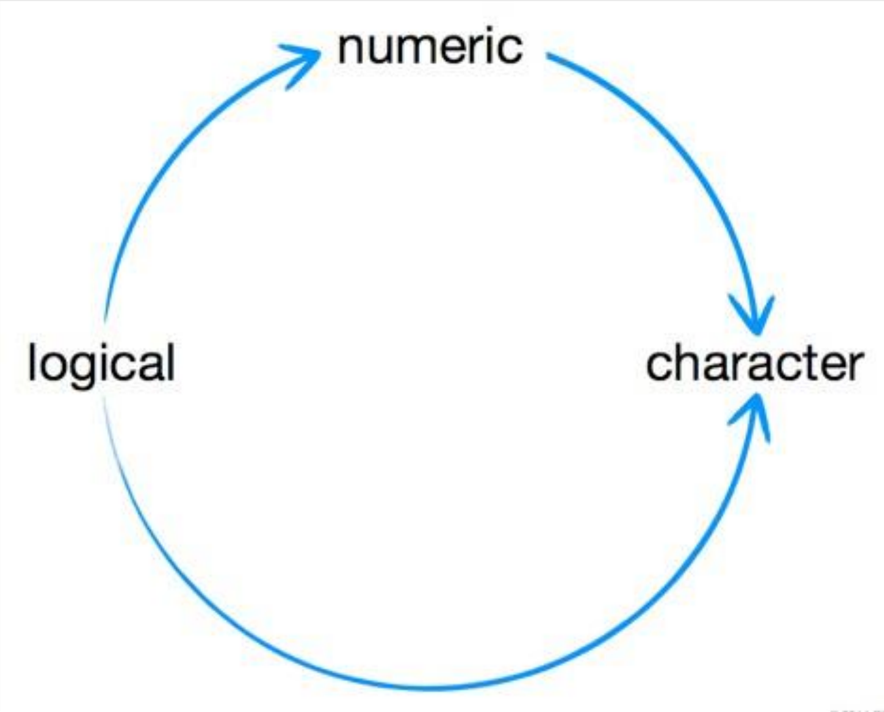
`c(TRUE, "a")`
character

`c(1, "TRUE")`

`TRUE + 5`

Quiz

What type of data will result?




```
c(5, "two")
```
character

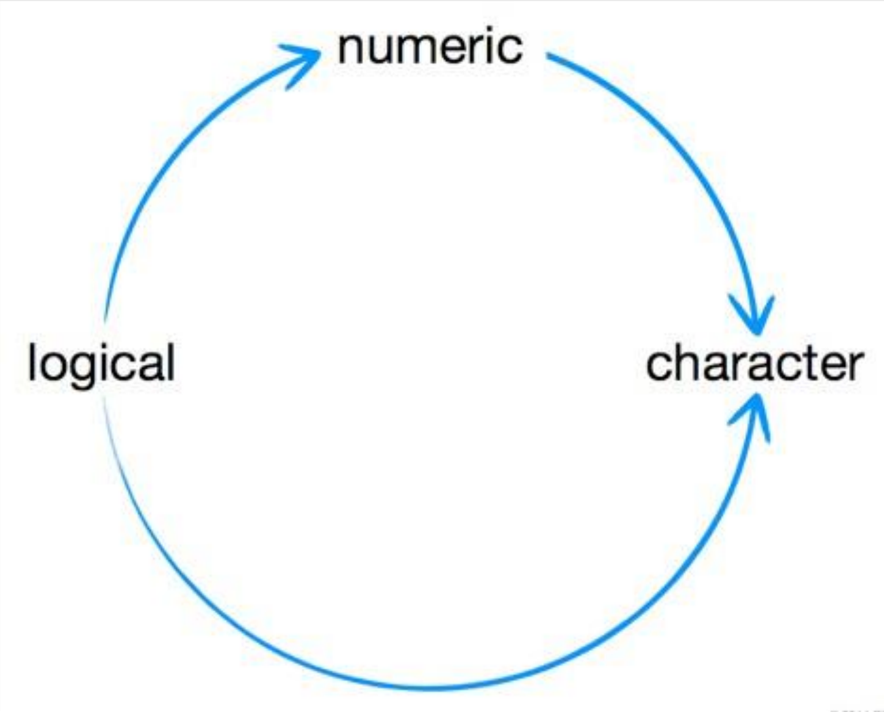
```
c(TRUE, "a")
```
character

```
c(1, "TRUE")
```
character

```
TRUE + 5
```

Quiz

What type of data will result?




```
c(5, "two")
```
character

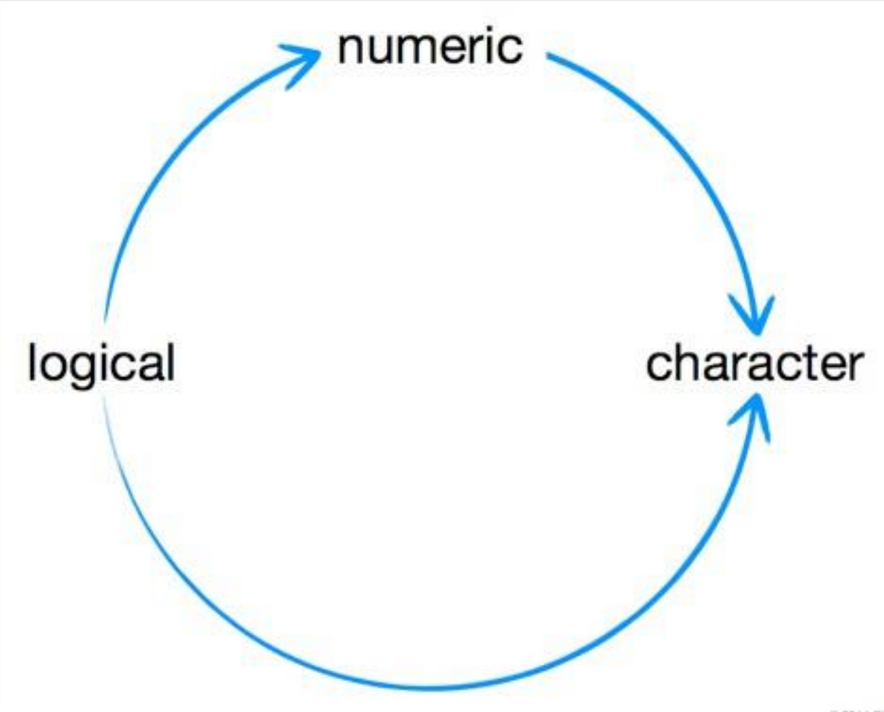
```
c(TRUE, "a")
```
character

```
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

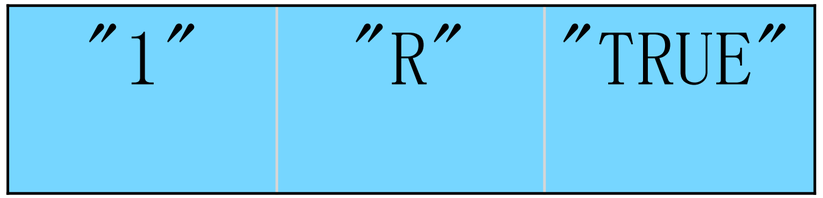
lists

A list is a one dimensional group of R objects.

Create lists with `list`

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

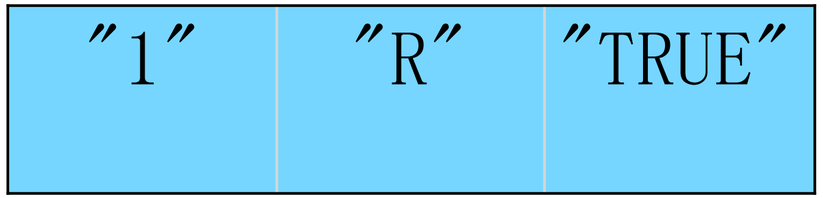
Vector



character

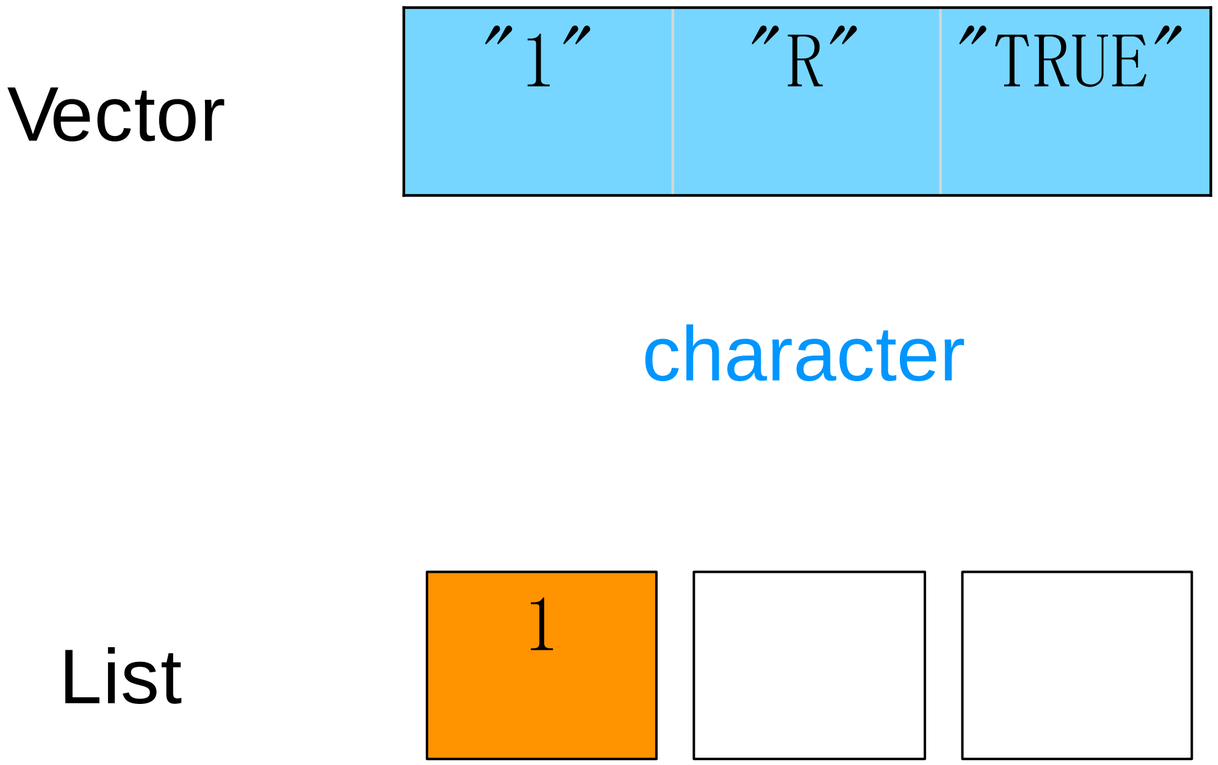
List

Vector

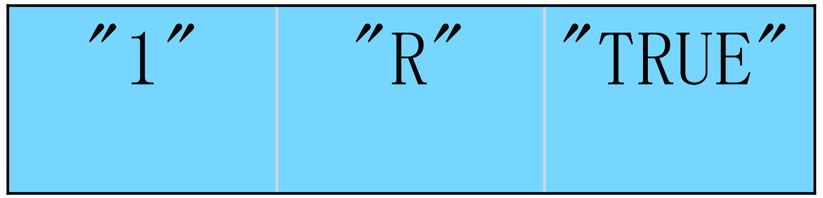


character

List

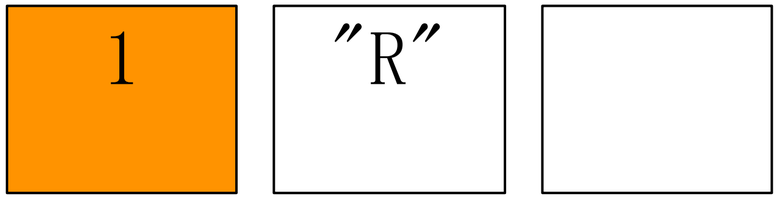

Vector
"1"
"R"
"TRUE"
character
List
1
numeric

Vector



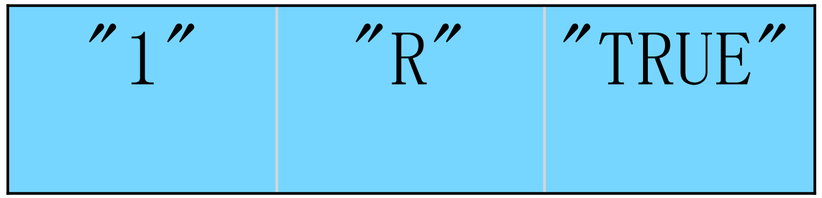
character

List



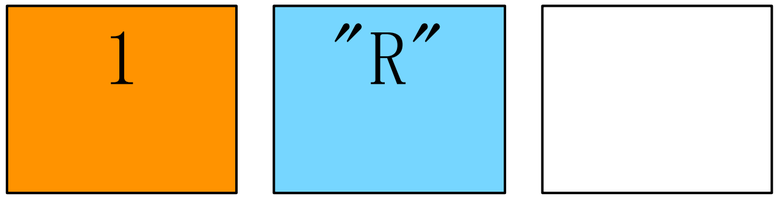
numeric

Vector



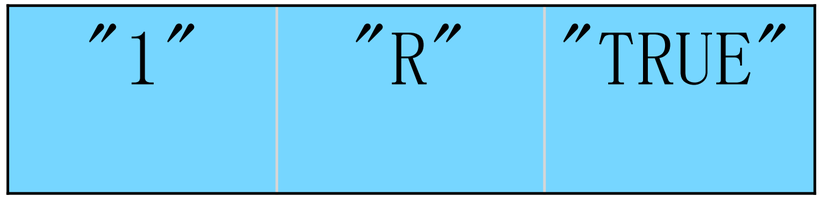
character

List



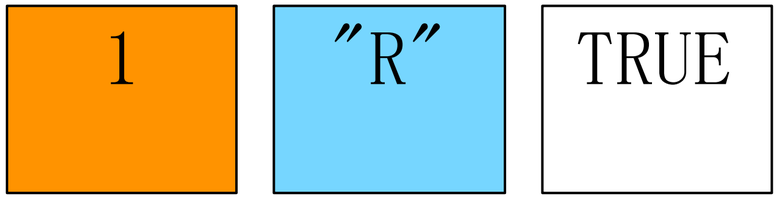
numeric character

Vector



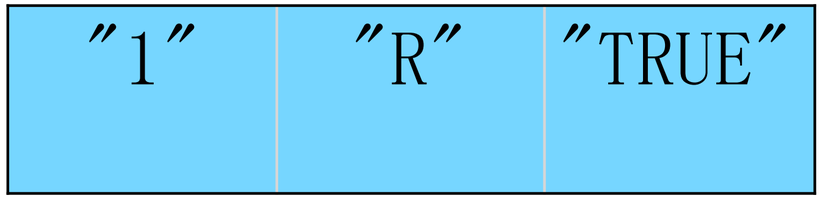
character

List



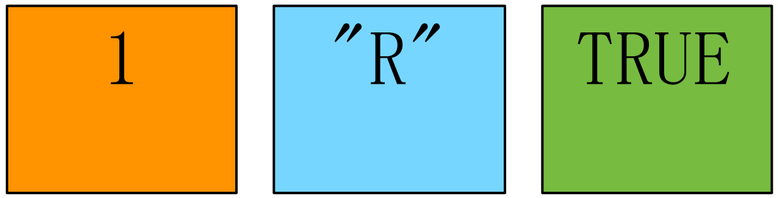
numeric character

Vector



character

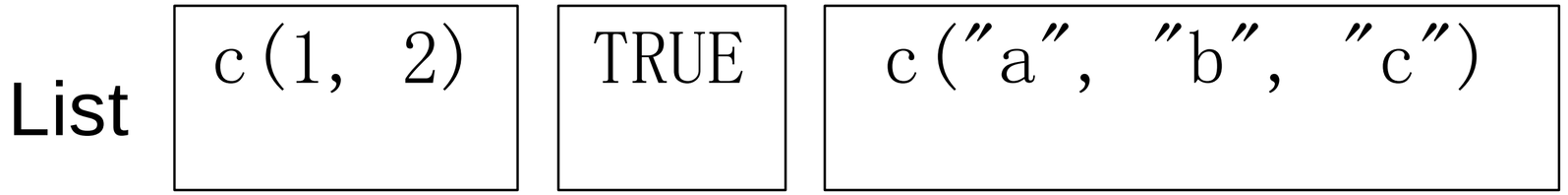
List



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"))
```



data frame

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

Each column in a data frame can be a different type

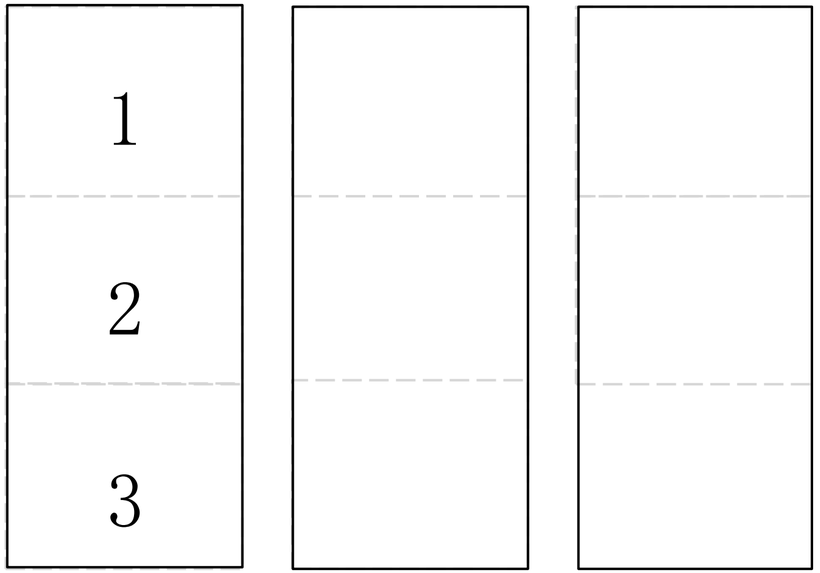
```
df <- data.frame(c(1, 2, 3),
 c("R","S","T"), c(TRUE, FALSE, TRUE))
class(df)
# "data.frame"
```

Matrix

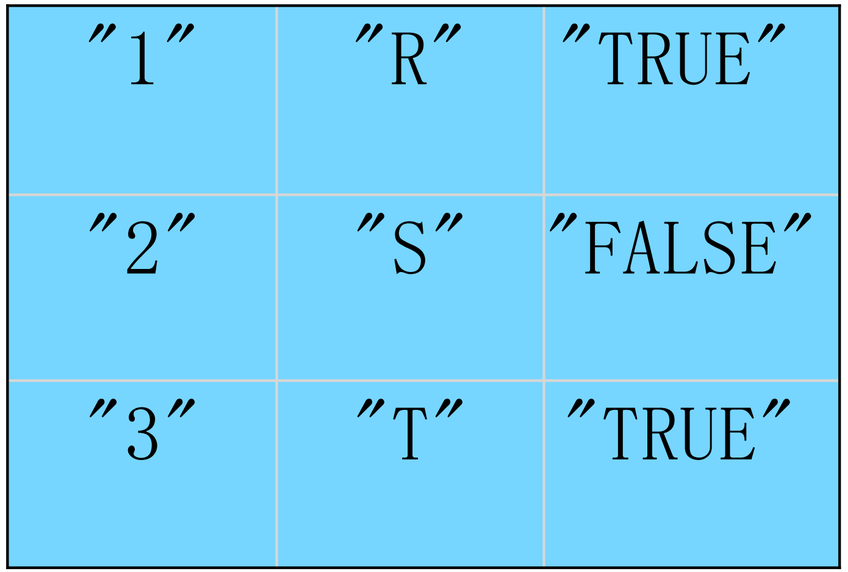
"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

data frame

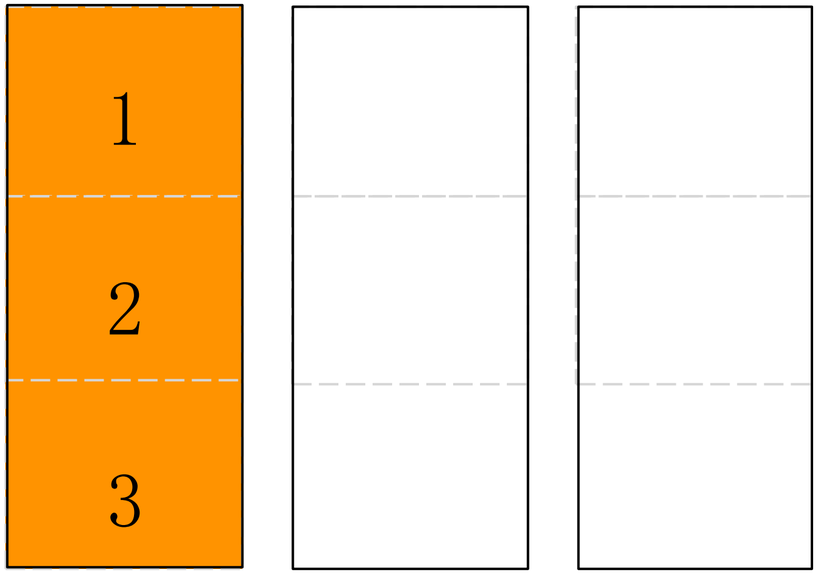


Matrix



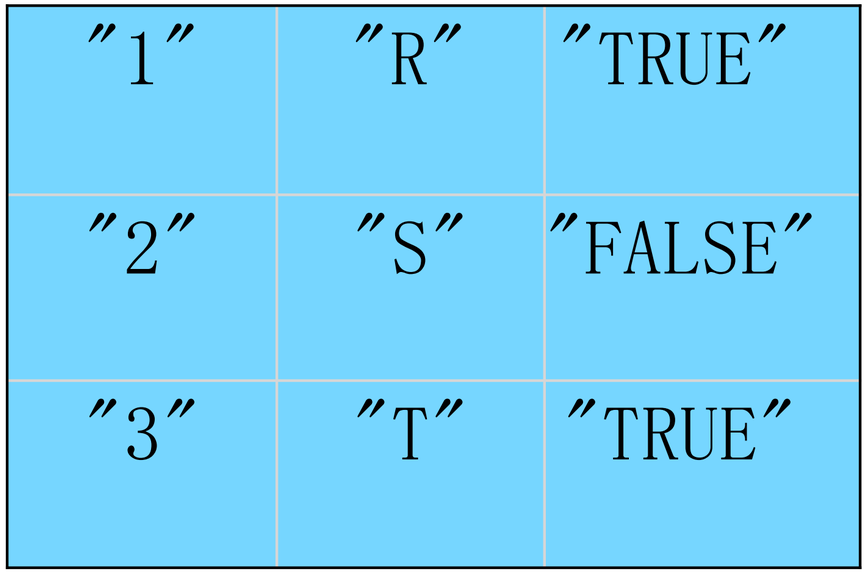
character

data frame



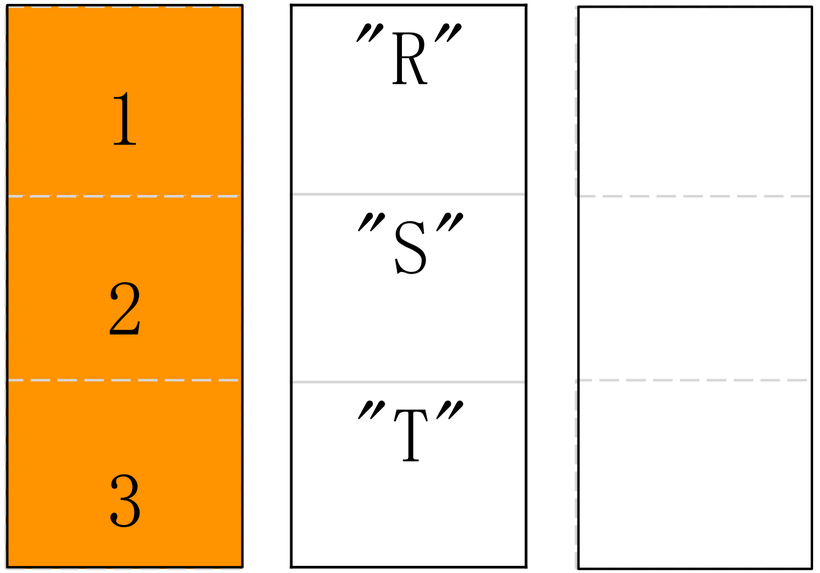
numeric

Matrix



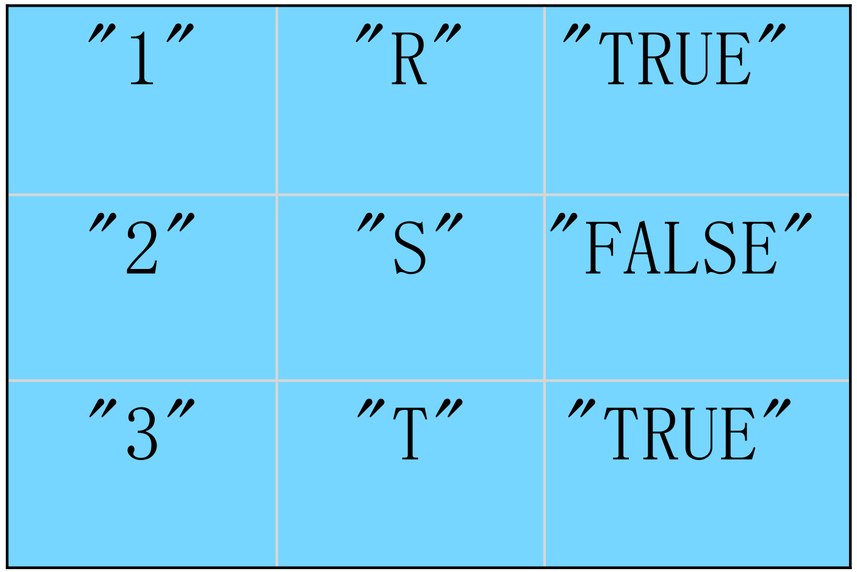
character

data frame



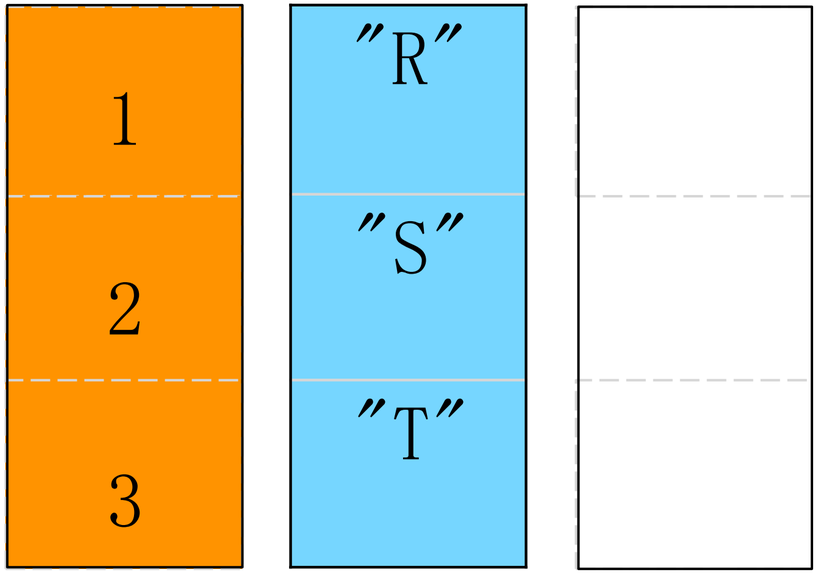
numeric

Matrix



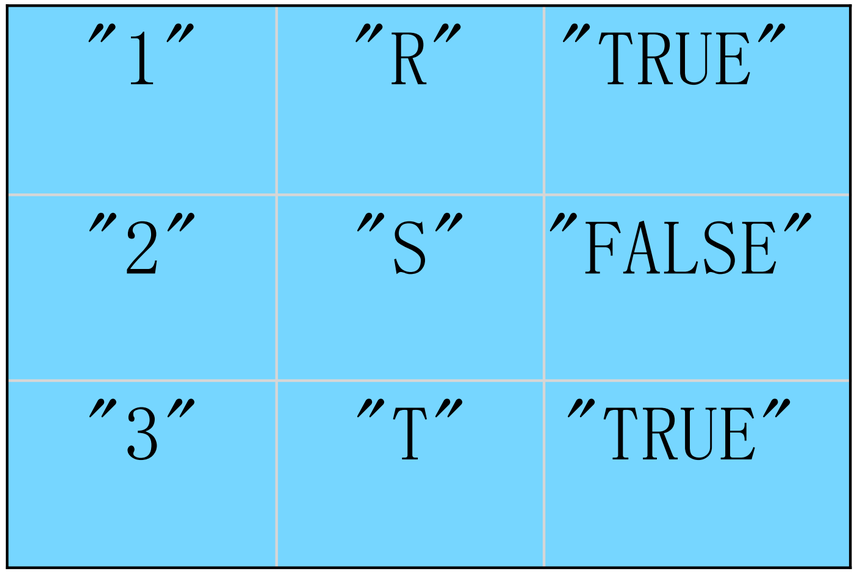
character

data frame



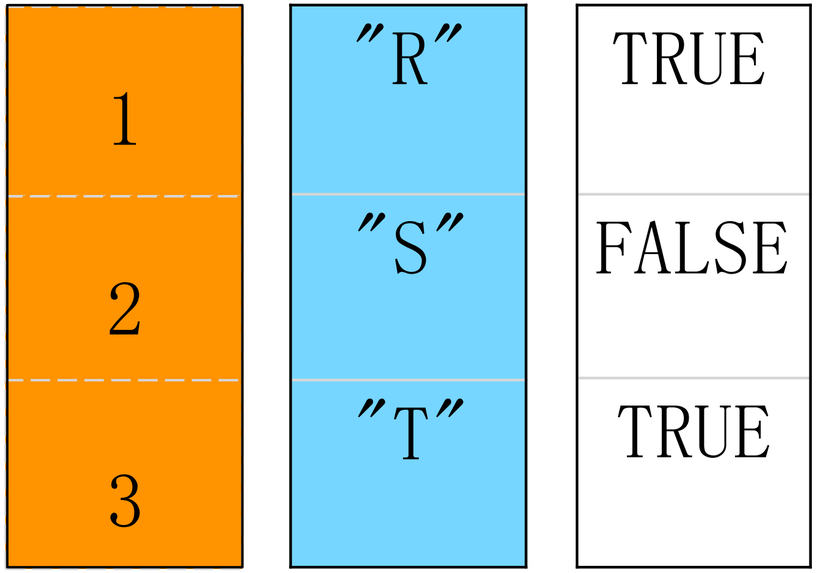
numeric character

Matrix



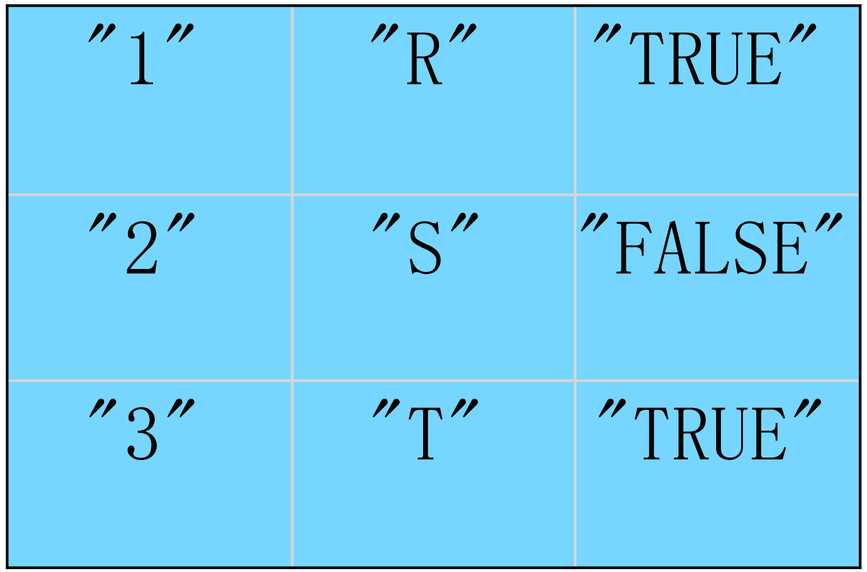
character

data frame



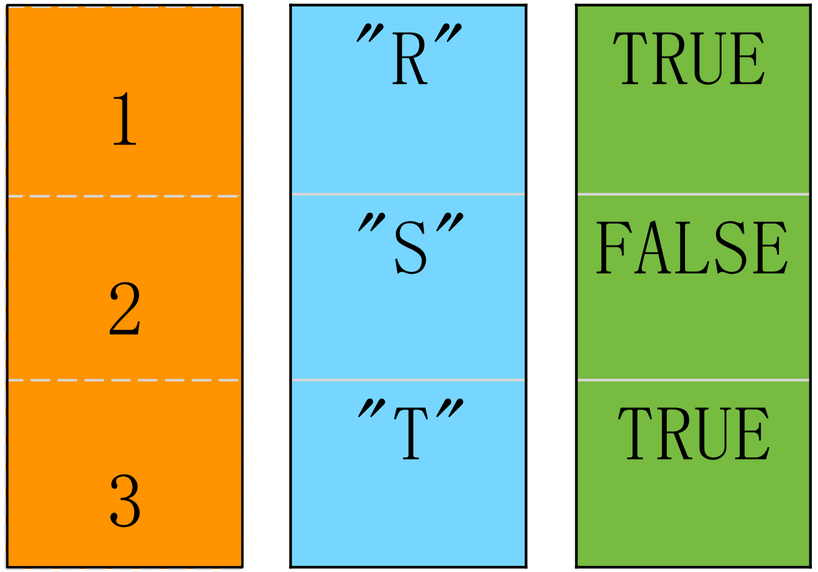
numeric character

Matrix



character

data frame



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
```

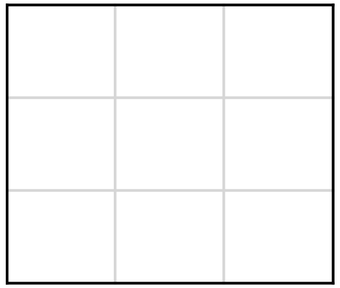
```
ndf <- data.frame(numbers = c(1, 2, 3),
                  letters = c("R","S","T"),
                  logic = c(TRUE, FALSE, TRUE))
```

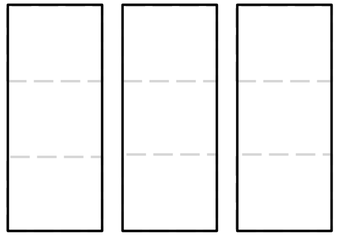
```
ndf
#   numbers letters logic
# 1       1       R  TRUE
# 2       2       S FALSE
# 3       3       T  TRUE
```

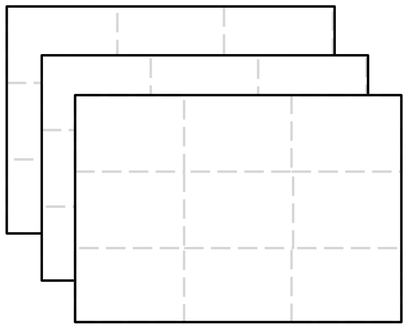
You can also see and set the names with `names`

```
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

R Packages

R Packages

A collection of code and functions written for the R language.

Usually focuses on a specific task or problem.

Most of the useful R applications appear in packages.

Start RStudio

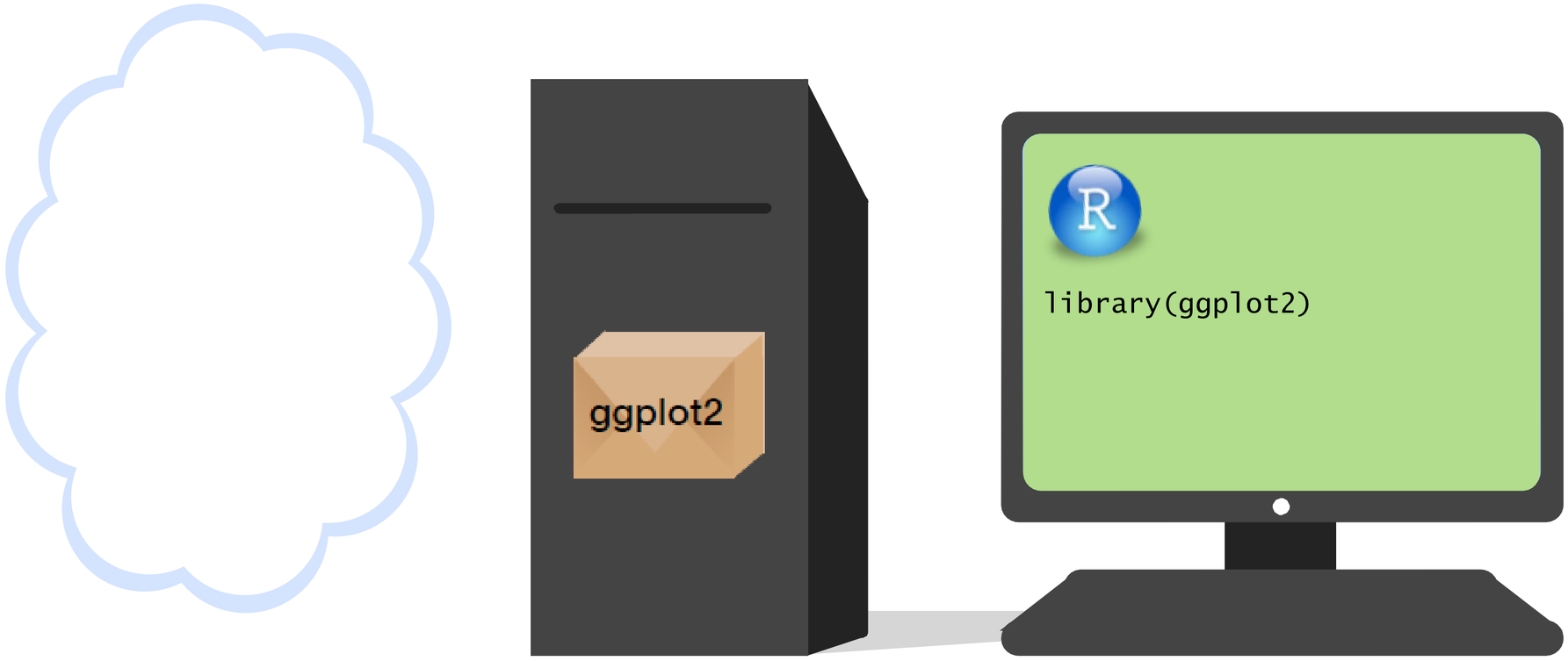
Install your package with

`install.packages("ggplot2")`

(ONE TIME)

Load your package with
`library(ggplot2)`

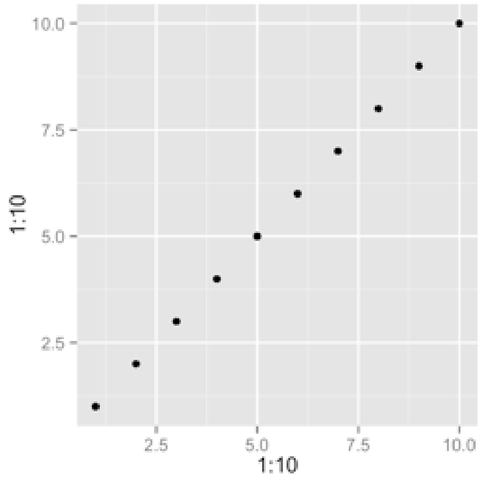
(EVERY TIME)



```
qplot(1:10, 1:10)
## Error: could not find function "qplot"
```

```
library(ggplot2)
qplot(1:10, 1:10)
```

```
##
```



You cannot use a function in a package until you load the package

Package summary

1. Download the package with
   `install.packages("name")`
   - You only have to do this once
   - You should be connected to the internet

2. Load the package with
   `library("name")`
   - You have to do this each time you start an R session.

There are over 5100 R packages

Your Turn

We're going to use the ggplot2, maps, RColorBrewer, and scales packages.

Load them with

```
library("ggplot2")
library("maps")
library("RColorBrewer")
```

Note: If you have not yet installed them, you'll need to run `install.packages(c("ggplot2", "maps", "RColorBrewer"))` first.