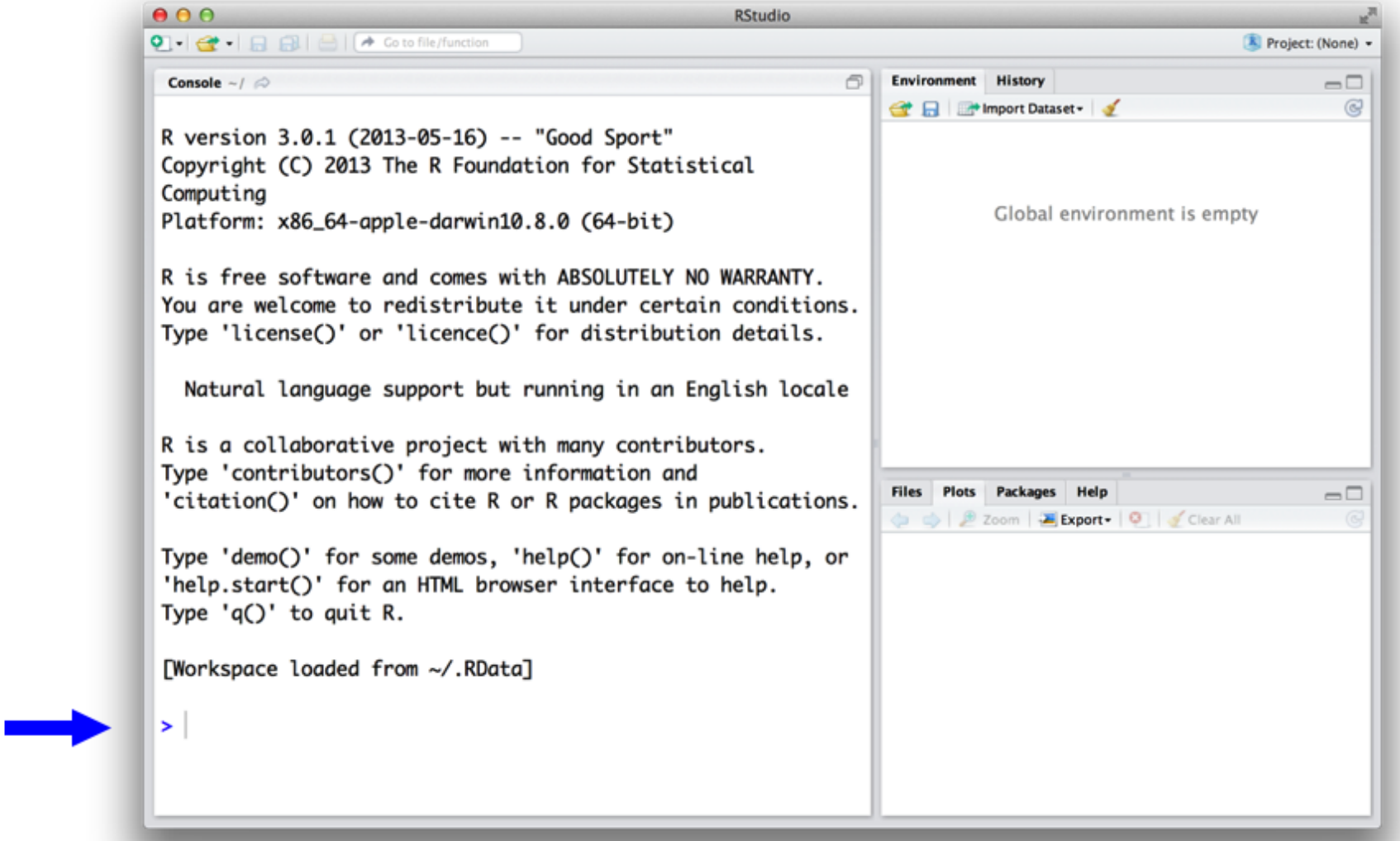
Data Analytics with R

Zahid Asghar, Muhammad Yaseen

Agenda

R language: an open-source language for statistical computing.

This is where R lives in RStudio



R as a Calculator

demo

R as a calculator

R allows all of the standard arithmetic operations.

Addition

```
1 + 2
```

```
## [1] 3
```

Multiplication

```
1 * 2
```

```
## [1] 2
```

Subtraction

```
1 - 2
```

```
## [1] -1
```

Division

```
1 / 2
```

```
## [1] 0.5
```

R as a calculator, cont.

R allows all of the standard arithmetic operations.

Exponents

```
2 ^ 3
```

```
## [1] 8
```

Parentheses for Order of Ops.

```
2 ^ 3 + 1
```

```
## [1] 9
```

```
2 ^ (3 + 1)
```

```
## [1] 16
```

Saving Objects

When you type a line of code (here: math) at the console and press `enter`, R will:

1. Run the code / evaluate the expression
2. *Print* the output to the console

If instead you want R to

1. Run the code / evaluate the expression
2. Save the output to an *object*

You can use the **assignment operator**.

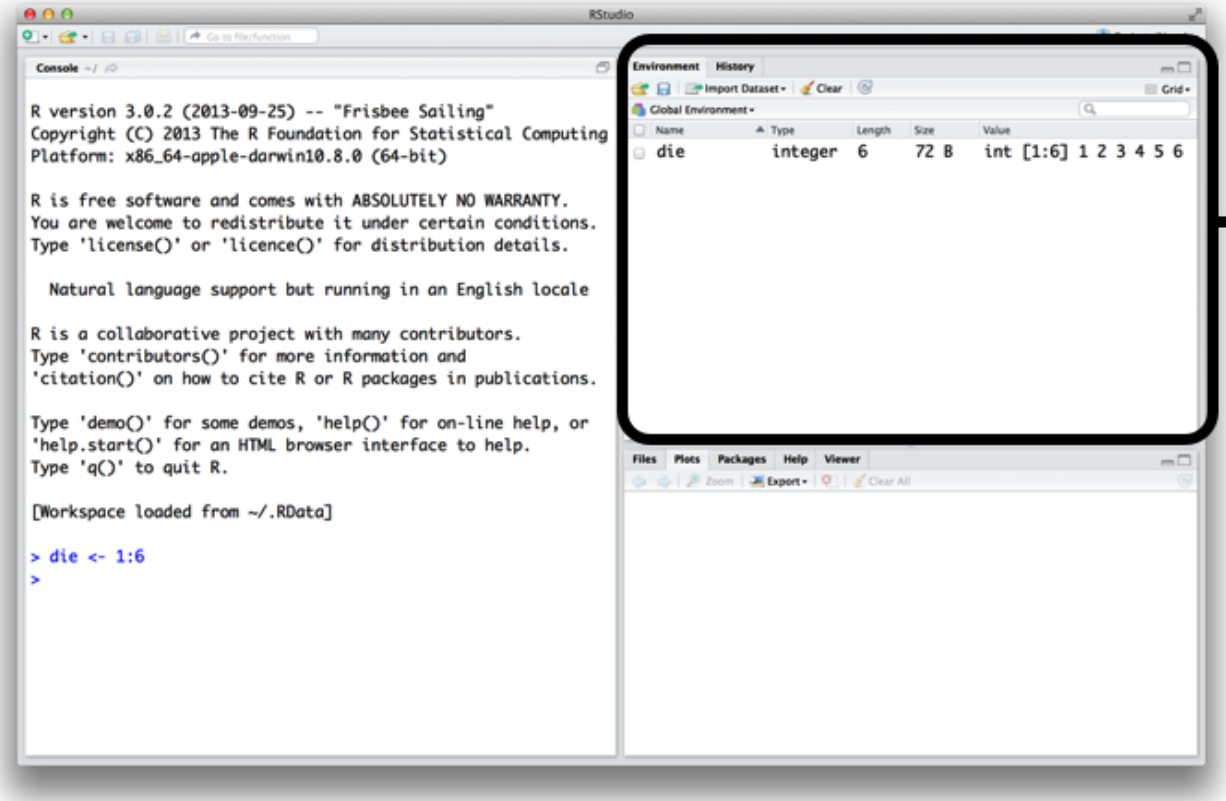
```
<-
```

(less than symbol, hyphen)

Saving objects

demo

Saving objects



The environment pane

Naming objects

In order to be recognized as a valid object name, you have to certain conventions; namely, the object name should begin with a letter.

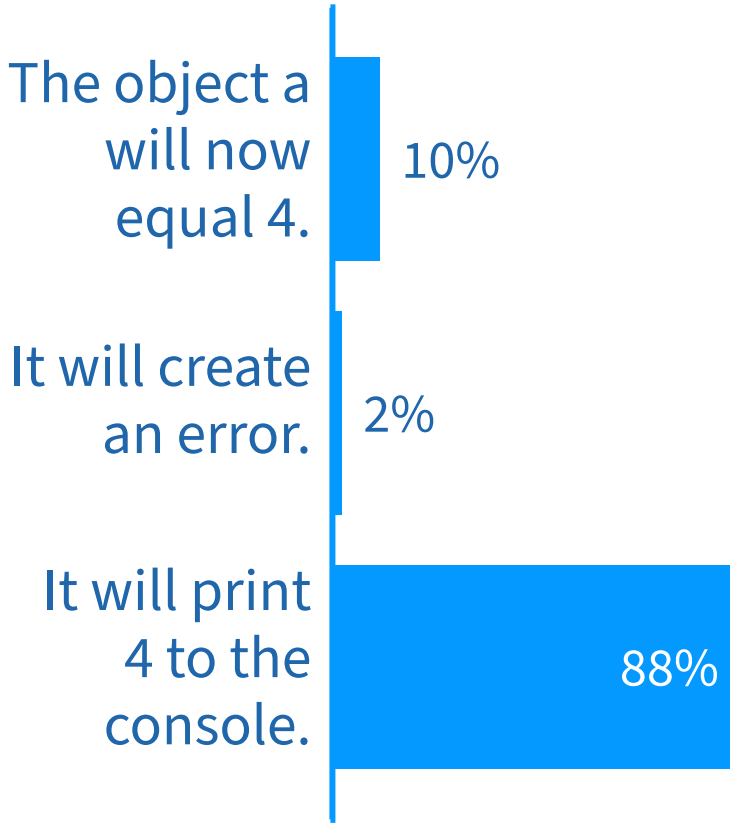
good names	names that cause errors
a	1trial
b	$
FOO	^mean
my_var	my var

What will happen here?

```
a <- 1 + 2
a + 1
```

Answer at PollEv.com/zahidasghar349.

What will happen here?



Powered by Poll Everywhere

What will happen here?

```
a <- 1 + 2
a + 1
```

```
## [1] 4
```

To update an object, you can overwrite it by assigning it to the same name.

```
a <- 1 + 2
a <- a + 1
a
```

```
## [1] 4
```

Creating a Vector

One of the core data structure in R is the *vector*, a one-dimensional ordered set of values. You can create a vector using a function called `c( )` for *combine* or *concatenate*.

demo

Creating a Vector

One of the core data structure in R is the *vector*, a one-dimensional ordered set of values. You can create a vector using a function called `c( )` for *combine* or *concatenate*.

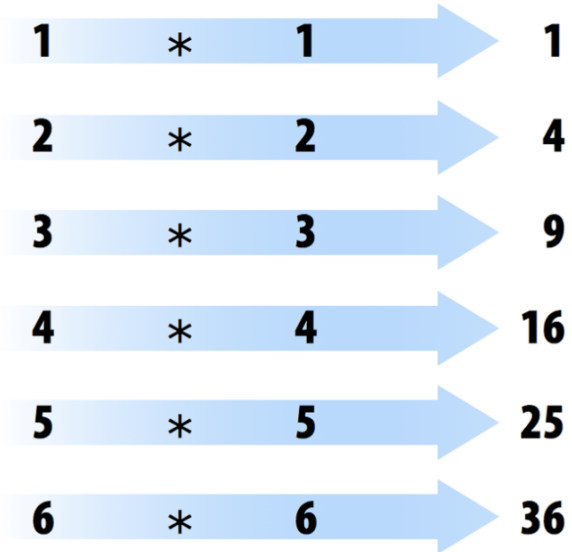
```
c(1, 2, 3)
```

```
## [1] 1 2 3
```

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

Math on Vectors

Arithmetic on vectors happens element-by-element.




```
a <- c(1, 2, 3, 4, 5, 6)
b <- c(1, 2, 3, 4, 5, 6)
a * b
```

```
## [1]  1  4  9 16 25 36
```

Functions

You can do operations on R objects by using *functions*.

Functions

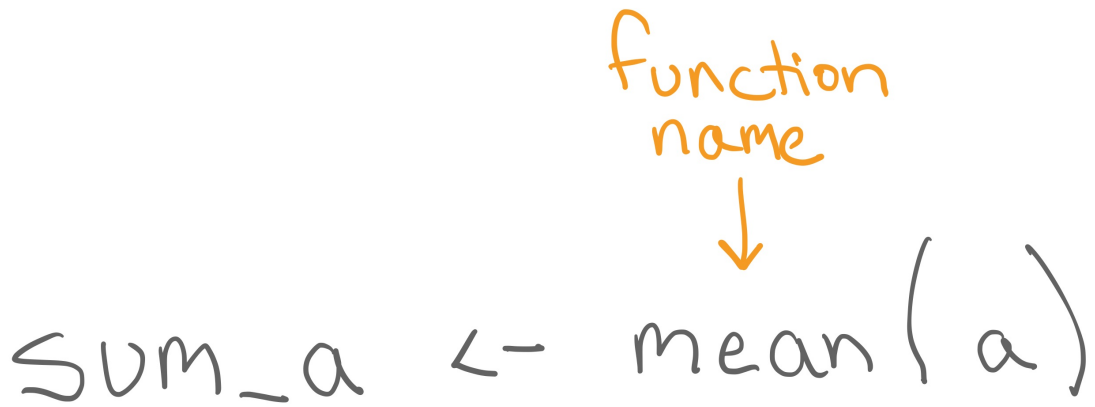
You can do operations on R objects by using *functions*.

```
sum_a <- mean(a)
```

Functions

You can do operations on R objects by using *functions*.

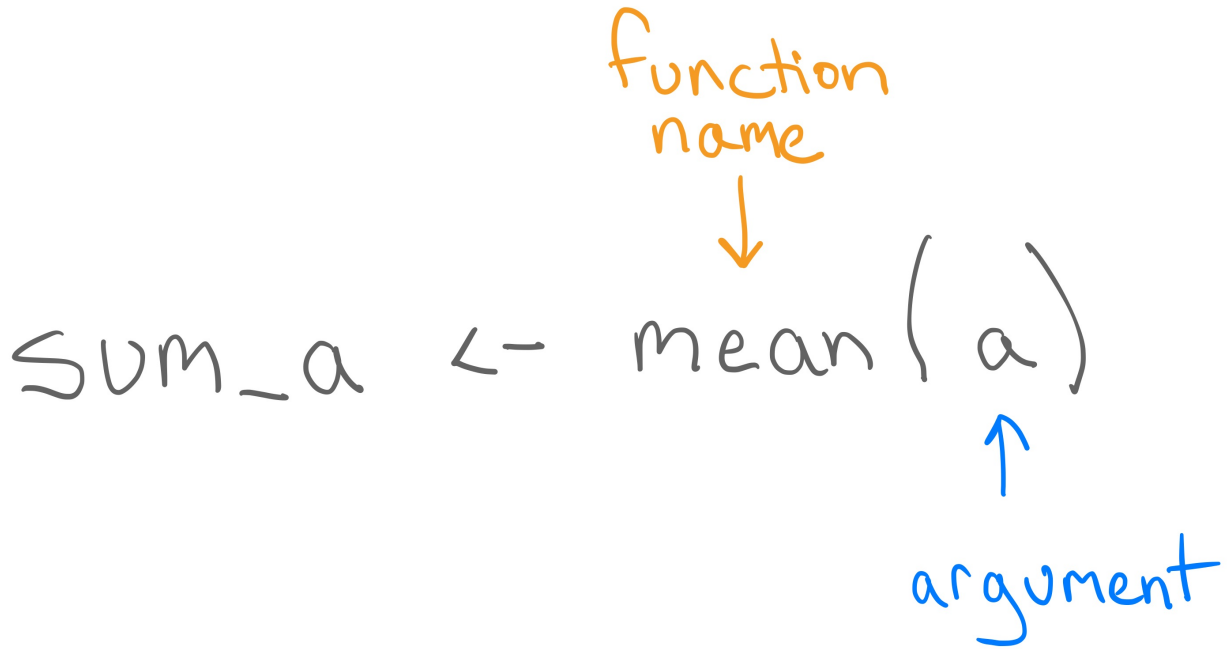
> Each function has a name followed by parentheses.



Functions

You can do operations on R objects by using *functions*.

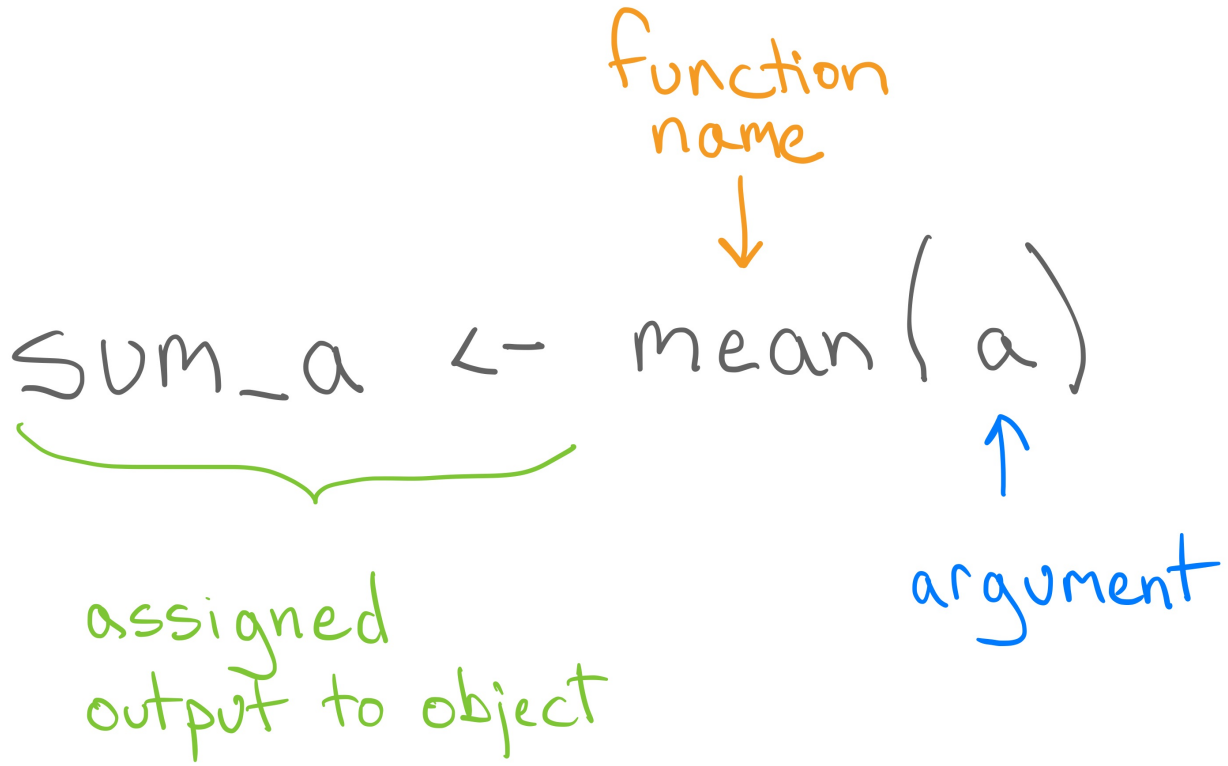
> Inside the parenthesis you put the object you'd like to operate on, called the *argument* of the function.



Functions

You can do operations on R objects by using *functions*.

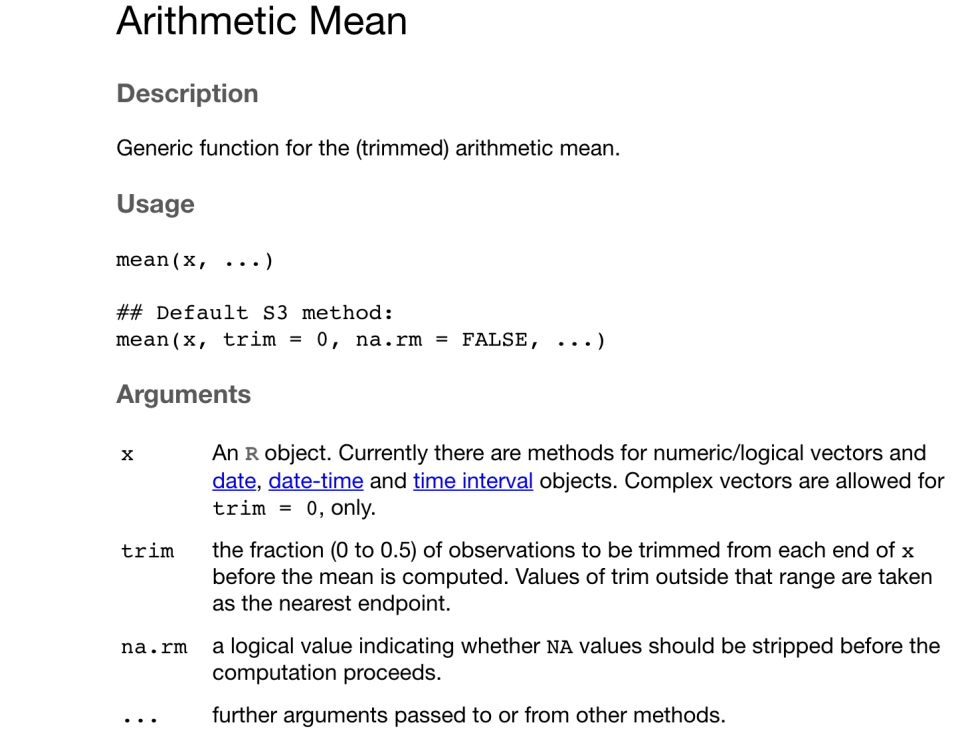
> When you provide the argument to the function and run it, you can capture the output with an object (or print it to the console).



Getting help

One of the most useful commands in R is ?.

```
?mean
```



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.

Look for:

- Which arguments are available
- What kind of output the function will return (called the "Value")
- Look at the examples at the bottom

Multiple arguments

```
a <- c(-5, 2, 3, 4, 5, 6, 7, 8, 9, 100)
mean(x = a, trim = .1)
```

```
## [1] 5.5
```

A few notes:

1. Functions can take multiple arguments separated by a comma: `,`
2. Arguments have names that you can specify or leave blank
3. Some arguments have default values that will be used if you specify something different

So many functions...

There are many functions in R, most of them named sensibly.

```
a <- c(1, 2, 3, 4)
mean(a)
```

```
## [1] 2.5
```

```
sum(a)
```

```
## [1] 10
```

```
length(a)
```

```
## [1] 4
```

Functions on vectors

Many of them will operate element-by-element if given a vector argument.

```
a <- c(1, 2, 3, 4)
sqrt(a)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```
log(a)
```

```
## [1] 0.0000000 0.6931472 1.0986123 1.3862944
```

```
abs(a)
```

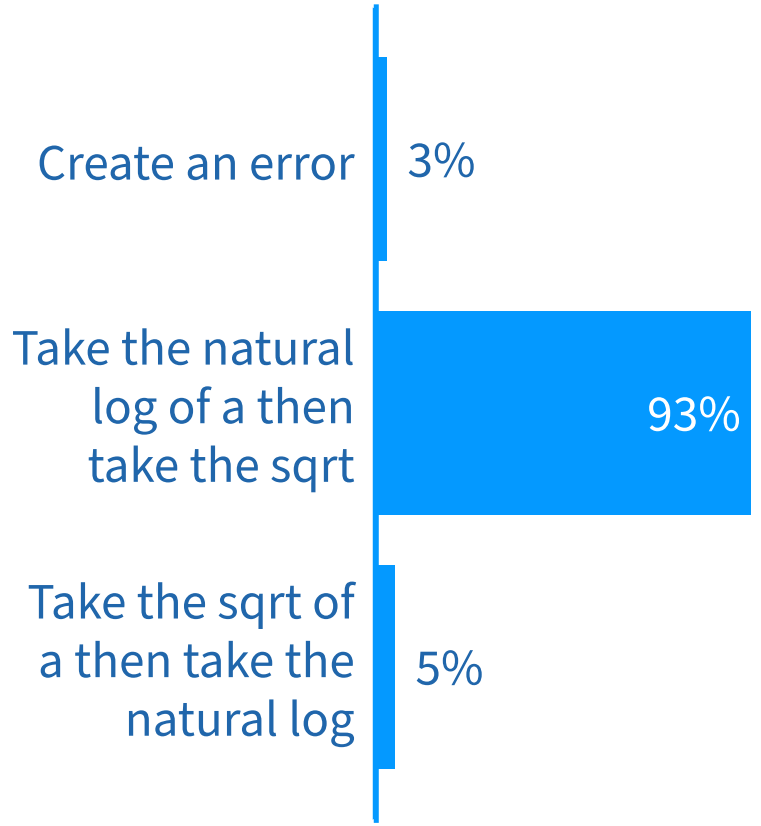
```
## [1] 1 2 3 4
```

What will happen here?

```
a <- c(1, 2, 3, 4)
sqrt(log(a))
```

Answer at PollEv.com/zahidasghar349.

What will happen here?



Powered by Poll Everywhere

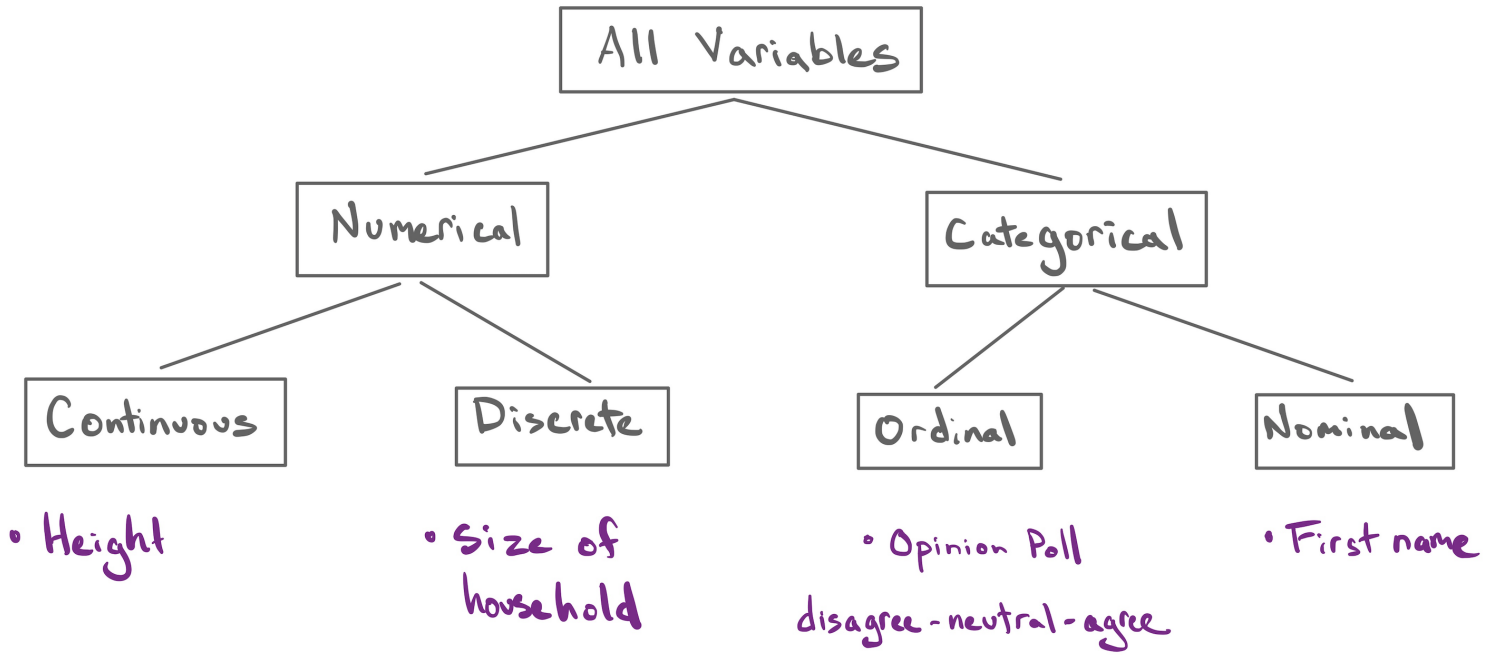
What will happen here?

```
a <- c(1, 2, 3, 4)
sqrt(log(a))
```

```
## [1] 0.0000000 0.8325546 1.0481471 1.1774100
```

Functions can be *nested* and will evaluate from the inside out.

The Taxonomy of Data



Every vector in R has a *type* that defines the way that it is stored and a *class* that determines how it will behave.

Common classes of vectors: `numeric`

You can ask R for the class of an object using the `class()` function.

```
a <- c(1, 2, 3)
class(a)
```

```
## [1] "numeric"
```

`numeric` objects map directly to *numeric variables*, with no strong distinction between continuous and discrete.

- You may also see related types such as `integer` and `dbl` (for double precision).

Common classes of vectors: `character`

`character` vectors contain *strings*, identified by being enclosed in quotation marks.

```
greetings <- c("hello", "hola", "ni hao", "heisann")
greetings
```

```
## [1] "hello"   "hola"    "ni hao"  "heisann"
```

```
class(greetings)
```

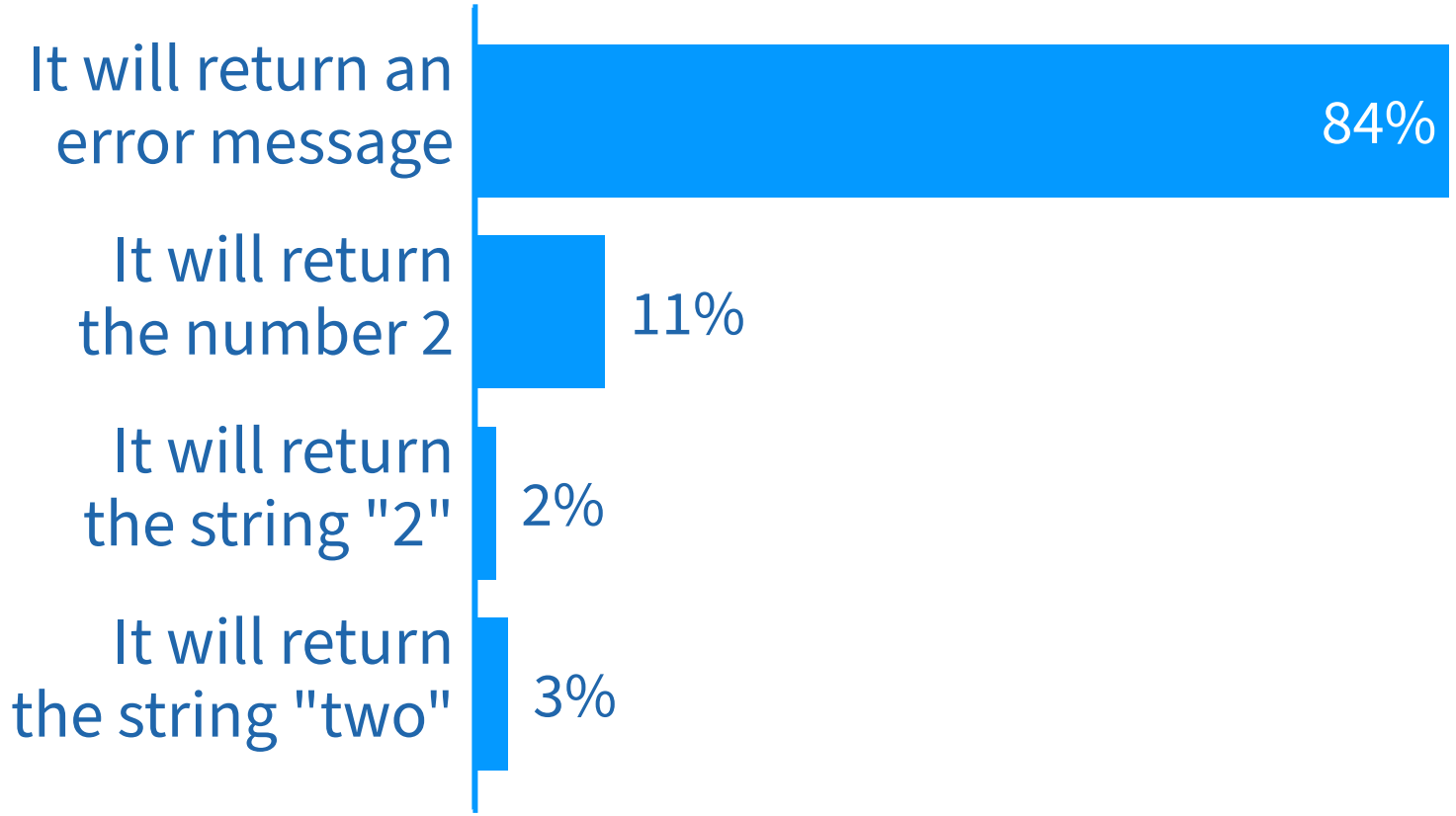
```
## [1] "character"
```

What will happen here?

```
1 + "one"
```

> Answer at `PollEv.com/zahidasghar349`.

What will happen here?



Powered by Poll Everywhere

What will happen here?

```
1 + "one"
```

```
## Error in 1 + "one": non-numeric argument to binary operator
```

Most functions (included arithmetic operators) will only work on objects of a certain class.

A function that only operates on character vectors:

```
greetings <- c("hello", "hola", "ni hao", "heisann")
toupper(greetings)
```

```
## [1] "HELLO"   "HOLA"    "NI HAO"  "HEISANN"
```

Common classes of vectors: `factor`

`factor` vectors contain categorical data where we expect there to be multiple observations at the same level. You will find them often in pre-loaded data set but you can construct them using `factor()`.

```
greetings <- factor(c("hello", "hello", "ni hao"))
greetings
```

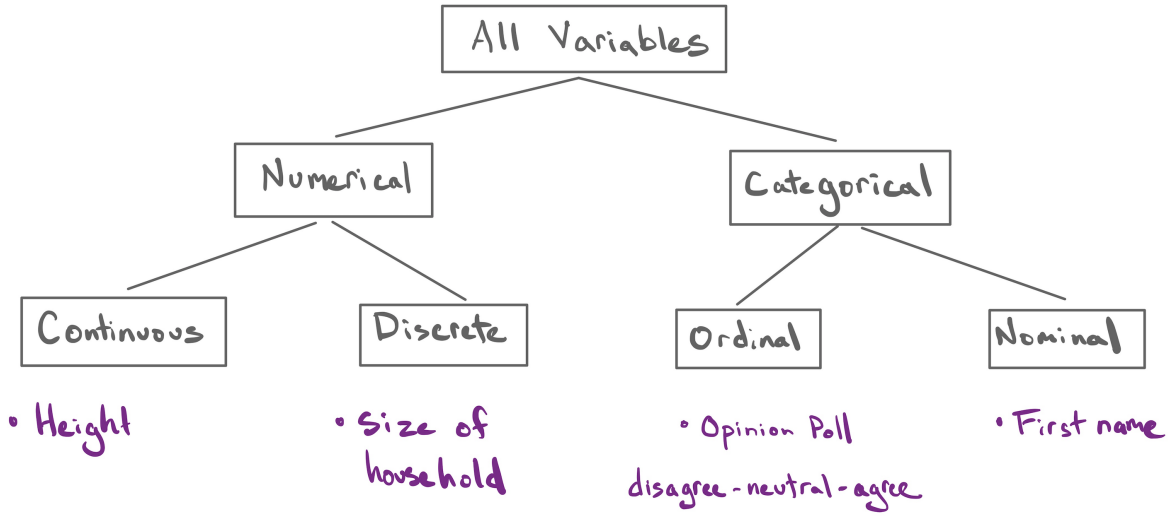
```
## [1] hello  hello  ni hao
## Levels: hello ni hao
```

```
class(greetings)
```

```
## [1] "factor"
```

`factor` is a natural class for *categorical variables*.

The Taxonomy of Data in R

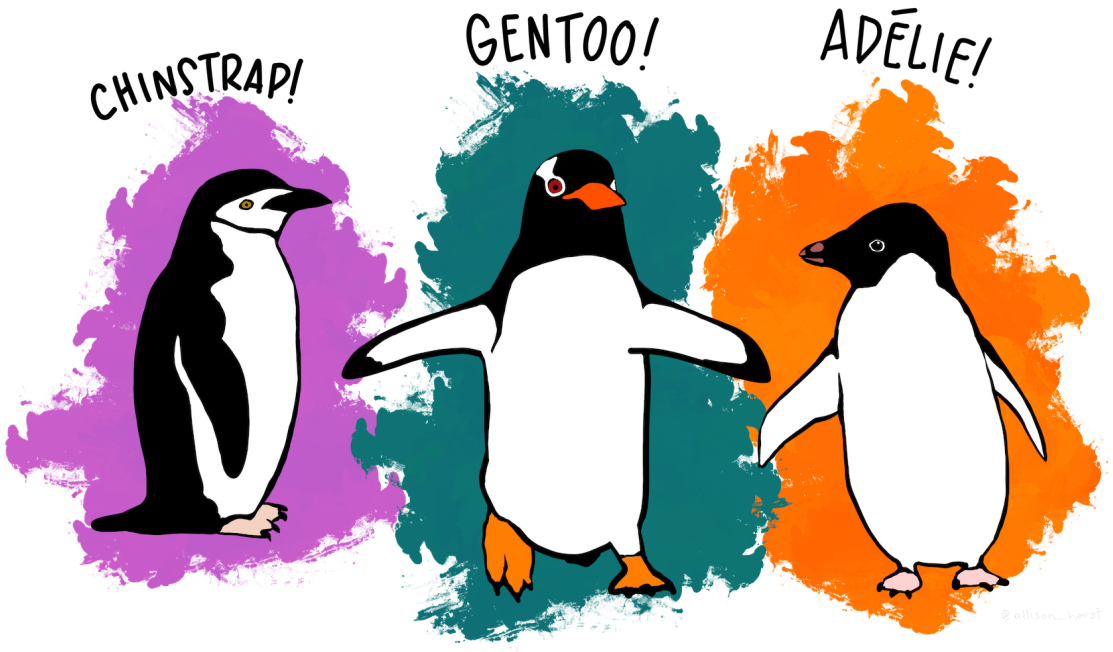


In R you will generally see:

- `numeric` for *continuous* and *discrete* numerical variables
- `factor` and `character` for *nominal* categorical variables
- There is an `ordered` factor for *ordinal* variables

Building Data Frames

Recall from last lecture...



Kristen Gorman set out to record and publish observations made on adult penguins near Palmer Station, Antarctica.

Penguins data frame

bill_depth_mm	bill_length_mm	species
16.6	40.9	Chinstrap
19.5	50.0	Chinstrap
16.1	49.9	Gentoo
17.0	55.9	Gentoo

What is the unit of observation?
What are the types of variables in the Taxonomy of Data?
What classes do those correspond to in R?

> Let's see if we can build this data frame...

Demo

Building a data frame

You can combine vectors into a data frame using the `tibble()` function from the extension library of R that we'll be using called the `tidyverse`.

```
bill_depth_mm <- c(15.0, 17.1, 18.7, 18.9)
bill_length_mm <- c(47.5, 40.2, 39.0, 35.3)
species <- c("Gentoo", "Adelie", "Adelie", "Adelie")
```

```
library(tidyverse)
penguins_df <- tibble(bill_depth_mm, bill_length_mm, species)
```

The penguins data frame

```
penguins_df
```

```
## # A tibble: 4 × 3
##   bill_depth_mm bill_length_mm species
##           <dbl>          <dbl> <chr>
## 1          15             47.5 Gentoo
## 2          17.1           40.2 Adelie
## 3          18.7           39   Adelie
## 4          18.9           35.3 Adelie
```

Looking Ahead

- Lab: R Workshop
- Lab 2 due by Thursday 11:59 PM
- Lab 1 preferrable by tomorrow (before lecture)

References

Images from Hands on Programming with R by Garret Grolemund. Art by Alison Horst.