Introduction to R software

- Built-in Commands, and Missing Data Handling

Built in commands

- Commands readily available in R to compute mathematical functions.
  - Use built in commands in computing various quantities.
- How to handle Missing data
- Simulate rolling of die in R

Maximum

```
> max(4.2, 3.7, -2.3, 4)
[1] 4.2
```

```
> max( c(4.2, 3.7, -2.3, 4) )
[1] 4.2
```

Minimum

```
> min(4.2, 3.7, -2.3, 4)
[1] -2.3
```

```
> min( c(4.2, 3.7, -2.3, 4) )
[1] -2.3
```

Overview of other Functions

`abs()`	Absolute value
`sqrt()`	Square root
`round()`, `floor()`, `ceiling()`	Rounding, up and down
`sum()`, `prod()`	Sum and product
`log()`, `log10()`, `log2()`	Logarithms
`exp()`	Exponential function
`sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`	Trigonometric functions
`sinh()`, `cosh()`, `tanh()`, `asinh()`, `acosh()`, `atanh()`	Hyperbolic functions

Examples

```
> sqrt(4)
[1] 2

> sqrt(c(4,9,16,25))
[1] 2 3 4 5
```

Examples

```
> sum(c(2,3,4,5))
[1] 14
```

```
> prod(c(2,3,4,5))      # 2 * 3 * 4 * 5
[1] 120
```

Examples

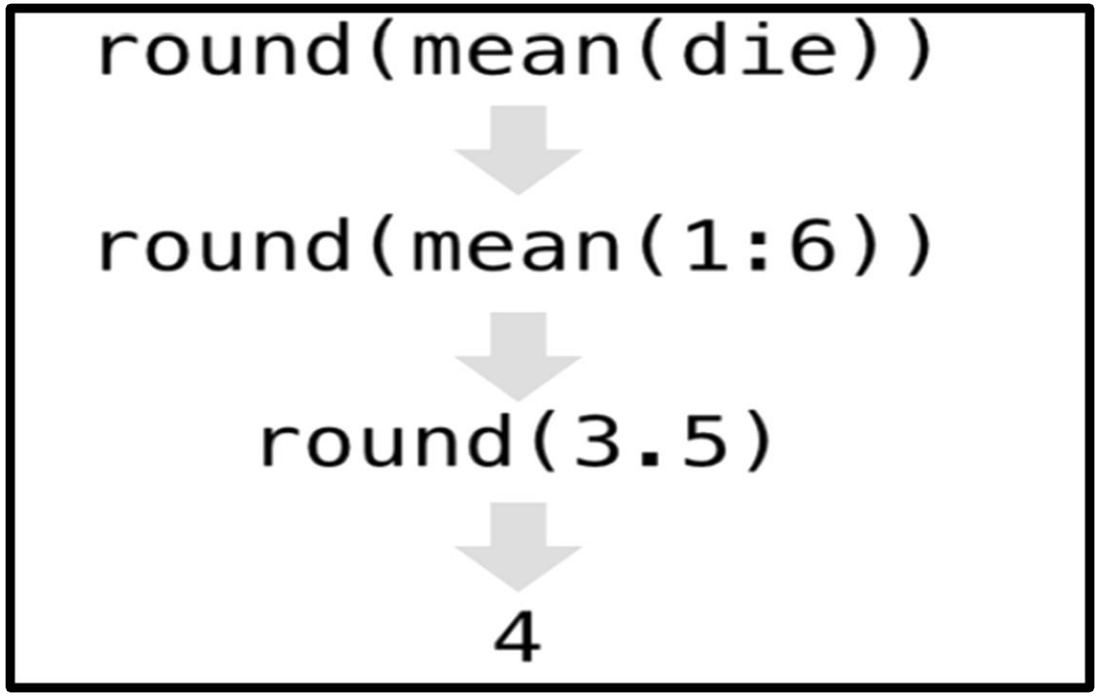
```
> round(3.1415)        # default 0 digits after decimal point
[1] 3

> round(3.1415, digits = 2)        # set the number of digits
[1] 3.14

> factorial(3)
[1] 6
```

Linking of functions

```
> die <- c(1, 2, 3, 4, 5, 6)
> die
[1] 1 2 3 4 5 6
> mean(die)
[1] 3.5
> round(mean(die))
[1] 4
```



When you link functions together, R will resolve them from the innermost operation to the outermost. Here R first looks up die, then calculates the mean of one through six, then rounds the mean.

Examples

To find sum of squares:

```
> x = c(2,3,4,5)

> z = sum(x^2)          # 2^2 + 3^2 + 4^2 + 5^2

> z
[1] 54
```

Examples

To find sum of squares of deviation from mean

```
> x = c(2,3,4,5)
```

$$z = \sum_{i=1}^{n} (x_i - \overline{x})^2 = \sum_{i=1}^{n} x_i^2 - n\overline{x}^2$$

```
> z = sum(x^2)-length(x)*mean(x)^2

> length(x)
[1] 4

> z
[1] 5
```

Missing Values

Missing values

- Missing values are denoted by **NA** or **NaN** for undefined mathematical operations.
- **`is.na()`** is used to test objects if they are NA
- **`is.nan()`** is used to test for NaN
- NA values have a class also, so there are integer NA, character NA, etc.
- ***`A NaN value is also NA but the converse is not true`***

Difference between NA and NaN

- **NA** (“Not Available”) is generally interpreted as a missing value
- **NaN** (“Not a Number”) means **0/0**
- Therefore, **NaN ≠ NA** and there is a need for NaN and NA.
- **`is.na()`** returns **TRUE** for both NA and NaN, however **`is.nan()`** returns **TRUE** for NaN (0/0) and **FALSE** for NA.
- As the elements of an atomic vector must be of the same type there are multiple types of NA values such as NA_integer_, NA_real_, etc.

```
> ## Create a vector with NAs in it
> x <- c(1, 2, NA, 10, 3)
> ## Return a logical vector indicating which elements are NA
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> ## Return a logical vector indicating which elements are NaN
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
```

```
> ## Now create a vector with both NA and NaN values
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

Example : How to work with missing data

```
> x <- c(10,20,NA,40)  # data vector
```

```
> mean(x)
  [1] NA
```

$$\frac{10+20+\mathrm{NA}+40}{4}$$

```
> mean(x, na.rm = TRUE)  # NAs can be removed
  [1] 23.33333
```

$$\frac{10+20+40}{3}=23.33$$

Example : How to work with missing data

The null object, called `NULL`, is returned by some functions and expressions.

Note that `NA` and `NULL` are not the same.

`NA` is a placeholder for something that exists but is missing.

`NULL` stands for something that never existed at all.

Some R functions

Simulation of roll of die

There is an R function that can help “roll” the die. You can simulate a roll of the die with R’s **sample function**.

```
> sample(x = die, size = 1)  # sample takes two arguments: a vector named x and a number
[1] 2                        # named size. sample will return size elements from the vector
> sample(x = die, size = 1)
[1] 4
> sample(x = die, size = 1)
[1] 1
> sample(x = die, size = 1)
[1] 1
> sample(x = die, size = 1)
[1] 5
```

Look up list of arguments in an R function

If you're not sure which names to use with a function, you can look up the function's arguments with **args**.

For example, you can see that the round function takes two arguments, one named x and one named digits:

```
> args(round)
function (x, digits = 0)
NULL
```

Sample with Replacement

If you set size = 2, you can simulate a pair of dice: sample will return two numbers, one for each die.

By default, sample builds a sample without replacement. To achieve die rolls with replacement (independent die rolls) add the argument **replace = TRUE:**

```
> sample(die, size = 2, replace = TRUE)
[1] 5 4
> sample(die, size = 2, replace = TRUE)
[1] 6 3
> sample(die, size = 2, replace = TRUE)
[1] 5 6
```

Function to simulate roll of a pair of dice

```
# function to simulate roll of pair of dice
roll = function() {
  die <- 1:6
  dice <- sample(x = die, size = 2, replace = TRUE)
  sum(dice)
}
```