R Basics

Monica Alexander

09/01/2022

R basics

Assign values to variables

The chunk above we used R as a simple calculator (2+2) We can also assign **values** to **variables** with the back arrow i.e. `<-`. For example (execute this chunk to see the outcomes)

```
# assigning the variable x to have a value of 1
x <- 1
# assigning the variable y to have a value of 2
y <- 2
# print these
x
```

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

```
y
```

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

```
# we can add these together too
x+y
```

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

Side note: you may see in other R codes that `=` is also used to assign values to a variable.

Values need not just be numbers:

```
# the c() function allows you to create vectors of numbers (or characters)
z <- c(3,4,3.2,5.1)
z
```

```
## [1] 3.0 4.0 3.2 5.1
```

```
# pull out variable parts of z
z[1]
```

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

```
z[3]
```

```
## [1] 3.2
```

```
# length of z
length(z)
```

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

```
# the 5th element doesn't exist
z[5]
```

```
## [1] NA
```

Character strings:

```
instructor_name <- "Monica Alexander"
first_name <- "Monica"
last_name <- "Alexander"
```

Functions in R are commands that take arguments and do operations to variables/objects. For example, the `paste` function pastes two (or more) strings together:

```
z
```

```
## [1] 3.0 4.0 3.2 5.1
```

```
max(z)
```

```
## [1] 5.1
```

```
mean(z)
```

```
## [1] 3.825
```

```
paste(first_name, last_name)
```

```
## [1] "Monica Alexander"
```

```
paste(first_name, "June", last_name)
```

```
## [1] "Monica June Alexander"
```

Sidenote: R is sensitive to capitalization, both in commands and in variable names. For example using `Paste` you would get an error.

Practice Exercise

Using the codes you learned, write a code that outputs, "your name, program of study".

Getting help

To see what a function does, and to check the arguments, type a "?" and then the function name, for example:

```
?paste
```

Once you execute the code above, you should see that the help file for paste has appeared in the bottom right pane.

Logical statements

It is useful to check to see if variables or objects are less than, equal to or greater than numbers. Below are some examples. Note that:

- Equality is two = signs (not one)
- Each of these statements returns a **logical** value i.e. TRUE or FALSE

```
#equals
x==1
```

```
## [1] TRUE
```

```
x==2
```

```
## [1] FALSE
```

```
# greater than
y>1
```

```
## [1] TRUE
```

```
x>1
```

```
## [1] FALSE
```

```
# greater than or equal to
x>=1
```

```
## [1] TRUE
```

```
# less than
x<9
```

```
## [1] TRUE
```