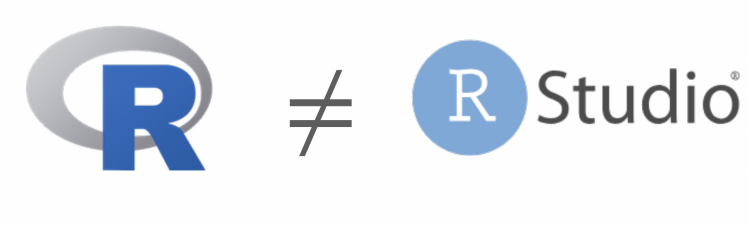
Cheatsheet

2022-11-02

What is R?

- R is a free, open-source statistical programming language
- Many programming languages exist: E.g. java script, C, C++, Python, etc.
- R is widely used by scientists worldwide
- You can use R to do everything: calculating simple summary statistics, performing complex simulations, creating gorgeous plots

What is Rstudio



Rstudio is an IDE - integrated development environment that allows you to write and run R code, visualize figures produced in R, and many other neat things

- Allows one to combine R code, analyses, plots, and written text into elegant documents all in one place using *Rmarkdown*

R as a calculator

```
2+2 #addition
```

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

```
534-430 #subtraction
```

```
## [1] 104
```

```
128421847*3 #multiplication
```

```
## [1] 385265541
```

```
12819482/17 #division
```

```
## [1] 754087.2
```

R as a calculator

```
4 ^ 3 #exponent
```

```
## [1] 64
```

```
4 ** 3 #exponent (also works)
```

```
## [1] 64
```

```
4 %% 3 #modulo: the remainder of a division
```

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

R for logical operations

```
4 == 3 #equality
```

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

```
4 != 3 #non-equality
```

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

```
4 < 3 #logical, lower than
```

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

```
4 >= 3 #logical, greater than or equal to
```

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

```
TRUE == 1 #this may surprise you!
```

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

Assigning values to R objects

```
x <- 4 # assign 4 to R object x

x #print object x
```

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

```
y = 4 # assign 4 to R object y

y #print object y
```

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

```
x == y #logical, equality
```

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

Data types in R

```
x <- 12 # assign number 12 to x

class(x) #check class or data type
```

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

```
x1 <- "12" # assign string 12 to x

class(x1) #check class or data type
```

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

```
y <- FALSE # assign boolean FALSE to y

class(y) #check class or data type
```

```
## [1] "logical"
```

```
y1 <- "FALSE" # assign string "FALSE" to y

class(y1) #check class or data type
```

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

Data Structures

Vectors

All elements must be of the same type.

```
x <- c(12,13,1,5765,12) # concatenate numbers and assign to x

is.vector(x) #logical
```

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

```
class(x) #check class or data type
```

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

```
print(x)
```

```
## [1]   12   13    1 5765   12
```

Data Structures

Vectors

All elements must be of the same type.

```
x1 <- c("Pumpkin",13,1,5765,12) # concatenate numbers and assign to x

is.vector(x1) #logical
```

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

```
class(x1) #check class or data type
```

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

```
print(x1)
```

```
## [1] "Pumpkin" "13"      "1"       "5765"    "12"
```

Data Structures

Lists

Elements can be of different types

```r
x1 <- c("Pumpkin",13,1,5765,12) # concatenate numbers and assign to x

is.vector(x1) #logical
```

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

```r
class(x1) #check class or data type
```

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

```r
print(x1)
```

```
## [1] "Pumpkin" "13"      "1"       "5765"    "12"
```

Data Structures

Matrix

All elements must be of the same type. Two dimensiosn

```r
mat<-matrix(rnorm(n=100, mean=0, sd=1),nrow=2) #sample 100 numbers from normal distribution
#with mean 0 and sd 1
#use them to make a matrix of two rows (50 cols by extension)
dim(mat)
```

```
## [1]  2 50
```

General functions:

`group_by()`: Conduct operations separately by values of a column (or columns).

`summarise()`: Reduces our data from the many observations for each variable to just the summaries we ask for. Summaries will be one row long if we have not group_by() anything, or the number of groups if we have.

`sum()`: Adding up all values in a vector.

`diff()`: Subtract sequential entries in a vector.

`sqrt()`: Find the square root of all entries in a vector.

`unique()`: Reduce a vector to only its unique values.

`pull()`: Extract a column from a tibble as a vector.

`round()`: Round values in a vector to the specified number of digits.

Summarizing location (center):

`n()`: The size of a sample.

`mean()`: The mean of a variable in our sample.

`median()`: The median of a variable in our sample.

Summarizing width (spread):

`max()`: The largest value in a vector.

`min()`: The smallest value in a vector.

`range()`: Find the smallest and larges values in a vector. Combine with diff() to find the difference between the largest and smallest value.

`quantile()`: Find values in a give quantile. Use `diff(quantile(... , probs = c(0.25, 0.75)))` to find the interquartile range.

`IQR()`: Find the difference between the third and first quartile (aka the interquartile range).

`var()`: Find the sample variance of vector.

`sd()`: Find the sample standard deviation of vector.

Sampling in R

`sample(x = , size = , replace = , prob = )`: Generate a sample of size size, from a vector `x`, with `(replace = TRUE)` or without `(replacement = FALSE)` replacement. By default the size is the length of `x`, sampling occurs without replacement and probabilities are equal. Change these defaults by specifying a value for the argument. For example, to have unequal sampling probabilities, include a vector of length `x`, in which the i^th entry describes the relative probability of sampling the i^th value in `x`.

`sample_n(tbl = , size = , replace = , weight = )`: Generate a sample of size `size`, from a `tibble` `tbl`, with `(replace = TRUE)` or without `(replacement = FALSE)` replacement. All arguments are the same as in `sample()` except `weight` replaces `prob`, and `tbl` replaces `x`. `sample_n()` is a function in the `dplyr` package, which is loaded with `tidyverse`.

Basics

```
## <center><img src="https://whitlockschluter3e.zoology.ubc.ca/RLabs/formula_summaries/Intro%20to%20R%2
```