Basic R Notebook: Lecture 1

Ying Xuan

Preliminaries

Returns the current working directory

```
getwd()
```

```
## [1] "/Users/yingxuan/Documents/SUTD/Term 6/TAE/W1"
```

Provides help on a specific function

```
help(getwd)
# OR
?getwd
```

Sets the current working directory

```
setwd("/Users/yingxuan/Documents/SUTD/Term 6/TAE/W1")
```

Lists the files in a directory

```
dir("/Users/yingxuan/Documents/SUTD/Term 6/TAE/W1")
```

```
##  [1] "Exercise"                "Old faithful"
##  [3] "PracticeR.Rmd"           "README.md"
##  [5] "week1lec1_class.nb.html" "week1lec1_class.pdf"
##  [7] "week1lec1_class.Rmd"     "week1lec1_up.pdf"
##  [9] "week1lec1_up.Rmd"        "week1lec2_class.nb.html"
## [11] "week1lec2_class.Rmd"     "week1lec2.Rmd"
## [13] "WHO"
```

Lists objects stored in the R workspace

```
ls()
```

```
## character(0)
```

Assigns a number to a variable

```
x <- 1
x
```

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

(Alternative to) assigning a number to a variable. Mostly '<-' is preferred for assignment.

```
x = 1
x
```

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

Common functions - exponential, inverse, power, addition

```
x <- 1
exp(x)
```

```
## [1] 2.718282
```

```
1/x
```

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

```
x^3
```

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

```
y <- x + 6
y
```

```
## [1] 7
```

Remove a variable from the workspace

```
rm(y)
```

Numbers and vectors

Concatenates (combines) numbers to form a vector

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

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

Accessing specific elements of the vector

```
x <- c(1, 2, 3)
x[3]
```

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

Applying operations to the vector - term by term inverse, concatenate vectors, exponentiation

```r
x <- c(1, 2, 3)
y <- c(x, 1, x)
y
```

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

You can overload the sum operator by recycling the shorter vector

- mathematically adding vectors of different sizes are not permitted
- if one variable is only of one value, it is used as an element wise addition to all elements of the vector variable
- if both variables are vectors of different length, element wise addition still happens, and the shorter vector is recycled for element wise addition
- if both variables are vectors of the same length, addition is done according to vector element sequence

```r
x <- c(1, 2, 3)
y <- c(x, 1, x)
z <- x + y
```

```
## Warning in x + y: longer object length is not a multiple of shorter object
## length
```

```r
z
```

```
## [1] 2 4 6 2 3 5 4
```

Finding the maximum and minimum elements and identifying the location (index) of the first max and all max

```r
x <- c(1, 2, 3)
max(x) # returns value
```

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

```r
which.max(x) # returns index
```

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

```r
# OR
which(x == max(x)) # returns index
```

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

```r
min(x)
```

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

```r
which.min(x)
```

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

Other operations - sum, product, mean, variance, standard deviation

```r
x <- c(1, 2, 3)
sum(x)
```

```
## [1] 6
```

```r
prod(x)
```

```
## [1] 6
```

```r
mean(x)
```

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

```r
sd(x)
```

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

For a vector, it provides a six number summary including min, max, mean, 1st quartile, median and 3rd quartile. This can be used with other objects too.

```r
x <- c(1, 2, 3)
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.5     2.0     2.0     2.5     3.0
```

Maximum entry in a concatenated vector

```r
x <- c(1, 2, 3)
y <- c(4, 5, 7)
max(x, y)
```

```
## [1] 7
```

Parallel maximum returns a vector of length equal to the longest argument

- In each element of this returned vector, it returns the maximum/largest element in that index no matter which individual input vector it is from
- length of vector returned = longest input vector in the argument
- arguments are to be of either the same length, or else an argument will be fractionally recycled (re-used to give a common length across all vectors)

```
?pmax
x <- c(1, -2, 3, 5, 3)
y <- c(x, 1, x)
x
```

```
## [1]  1 -2  3  5  3
```

```
y
```

```
##  [1]  1 -2  3  5  3  1  1 -2  3  5  3
```

```
pmax(x, y)
```

```
## Warning in pmax(x, y): an argument will be fractionally recycled
```

```
##  [1]  1 -2  3  5  3  1  1  3  5  5  3
```

```
# vectors under comparison
# lengthened x vector VS original y vector
# 1 -2 3 5 3 1 -2 3 5 3 1
# 1 -2 3 5 3 1 1 -2 3 5 3
```

Remove all variables from the workspace

```
rm(list=ls())
```

Differences in the assignment using <- and ==. Try exp(a=1:5) and exp(a<-1:5)

```
exp(a <- 1:5)
```

```
## [1]   2.718282   7.389056  20.085537  54.598150 148.413159
```

```
# exp(a = 1:5)
```

Generating vectors using a variety of commands

```
x <- -3:8 # start and end values are invlusive
x
```

```
##  [1] -3 -2 -1  0  1  2  3  4  5  6  7  8
```

```
# seq(from, to, by)
# from and to values are inclusive
seq(-3, 8, 2) # last parameter = interval
```

```
## [1] -3 -1  1  3  5  7
```

```
rep(x, times = 3) # times parameter = number of times to repeat (one full cycle repeated)
```

```
##  [1] -3 -2 -1  0  1  2  3  4  5  6  7  8 -3 -2 -1  0  1  2  3  4  5  6  7  8 -3
## [26] -2 -1  0  1  2  3  4  5  6  7  8
```

```
rep(x, each = 3) # each parameter = number of times each element to appear (together in sequence)
```

```
##  [1] -3 -3 -3 -2 -2 -2 -1 -1 -1  0  0  0  1  1  1  2  2  2  3  3  3  4  4  4  5
## [26]  5  5  6  6  6  7  7  7  8  8  8
```

Returns logical vector based on the check

```
x <- -2:3
x
```

```
## [1] -2 -1  0  1  2  3
```

```
x > 1
```

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

Dealing with missing entries

```
x <- -2:3
x
```

```
## [1] -2 -1  0  1  2  3
```

```
is.na(x)
```

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

```
y <- c(x, NA)
y
```

```
## [1] -2 -1  0  1  2  3 NA
```

```
is.na(y)
```

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

END OF LECTURE 1.