Session 2

Antoine Mayerowitz

17/09/2018

Chapter 1 (continued)

Matrices

Definition

```
X = matrix(nrow= 2, ncol=5)
X = matrix(0, nrow= 2, ncol=5)
X = matrix(1:10, nrow= 2, ncol=5)
print(X) # Let's look at X (! Note that we just redefined X three times.)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

Subsetting

```
X[1, ] # The first row
```

```
## [1] 1 3 5 7 9
```

```
X[ ,1] # First column
```

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

```
X[2,3] # 2nd row, 3rd column
```

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

Operations

```
X = matrix(1:4, 2, 2)
Y = matrix(4:1, 2, 2)
X * Y # Element wise multiplication
```

```
##      [,1] [,2]
## [1,]    4    6
## [2,]    6    4
```

```
X %*% Y # Matrix multiplication
```

```
##      [,1] [,2]
## [1,]   13    5
## [2,]   20    8
```

Task 3

- Create a vector containing 1,2,3,4,5 called v.
- Create the (2,5) matrix m with data (1:10)
- Perform matrix multiplication of m with v. Use the command %*%. What dimension does the output have?
- Why does the command v %*% m not work?

```r
v = 1:5
print(v)
```

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

```r
m = matrix(1:10, nrow = 2, ncol = 5)
print(m)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

```r
dim(m %*% v)
```

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

Lists

Definition

```r
myList = list(
  name = "John",
  age= 32,
  matrix = matrix(1:4, 2, 2)
)
```

Subsetting

```r
myList$name
```

```
## [1] "John"
```

```r
myList[1]
```

```
## $name
## [1] "John"
```

```r
myList[[1]]
```

```
## [1] "John"
```

Task 4

- Copy and paste the above code for ex_list into your R session. Remember that list can hold any kind of R object. Like…another list! So, create a new list new_list that has two fields: a first field called “this” with string content “is awesome”, and a second field called “ex_list” that contains ex_list.

- Accessing members is like in a plain list, just with several layers now. Get the element c from `ex_list` in `new_list`!
- Compose a new string out of the first element in new_list, the element under label this. Use the function paste to print the string 'R is awesome' to your screen.

```
ex_list = list(
  a = c(1, 2, 3, 4),
  b = TRUE,
  c = "Hello!",
  d = function(arg = 42) {print("Hello World!")},
  e = diag(5)
)

new_list = list(
  this = "is awesome",
  ex_list = ex_list
)

new_list$ex_list$c
```

```
## [1] "Hello!"
```

```
paste("R", new_list$this)
```

```
## [1] "R is awesome"
```

DataFrame

Definition

```
data = data.frame(
  x = 1:10,
  y = sample(c("a", "b", "c"), size= 10 , replace= TRUE)
)
```

Some methods

```
nrow(data) # Number of rows
```

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

```
ncol(data) # Number of columns
```

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

```
names(data) # Variables names
```

```
## [1] "x" "y"
```

```
head(data) # Prints a few rows
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 a
## 4 4 a
## 5 5 b
## 6 6 b
```

Example (mtcars)

```
head(mtcars, n=3) # Shows the first 3 rows
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
mtcars[,"mpg"] # Subset only the `mpg` column
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

```
# Subset
mtcars[mtcars$mpg > 20, c("mpg", "cyl")] # Subset lines where mpg > 20 and keep only the columns 'mpg'
```

```
##                 mpg cyl
## Mazda RX4      21.0   6
## Mazda RX4 Wag  21.0   6
## Datsun 710     22.8   4
## Hornet 4 Drive 21.4   6
## Merc 240D      24.4   4
## Merc 230       22.8   4
## Fiat 128       32.4   4
## Honda Civic    30.4   4
## Toyota Corolla 33.9   4
## Toyota Corona  21.5   4
## Fiat X1-9      27.3   4
## Porsche 914-2  26.0   4
## Lotus Europa   30.4   4
## Volvo 142E     21.4   4
```

```
subset(mtcars, subset = mpg > 20, select = c("mpg", "cyl")) # Another way to write it
```

```
##                 mpg cyl
## Mazda RX4      21.0   6
## Mazda RX4 Wag  21.0   6
## Datsun 710     22.8   4
## Hornet 4 Drive 21.4   6
## Merc 240D      24.4   4
## Merc 230       22.8   4
## Fiat 128       32.4   4
## Honda Civic    30.4   4
## Toyota Corolla 33.9   4
## Toyota Corona  21.5   4
## Fiat X1-9      27.3   4
## Porsche 914-2  26.0   4
## Lotus Europa   30.4   4
## Volvo 142E     21.4   4
```

Task 5

- How many observations are there in mtcars?
- How many variables?
- What is the average value of mpg?
- What is the average value of mpg for cars with more than 4 cylinders, i.e. with cyl>4?

```
nrow(mtcars)
```

```
## [1] 32
```

```
ncol(mtcars)
```

```
## [1] 11
```

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```
mean(subset(mtcars, subset= cyl > 4)$mpg)
```

```
## [1] 16.64762
```

```
# Create new var
data = mtcars[, c("mpg", "cyl" )]
data$myCol = data$mpg / data$cyl
head(data)
```

```
##                    mpg cyl    myCol
## Mazda RX4         21.0   6 3.500000
## Mazda RX4 Wag     21.0   6 3.500000
## Datsun 710        22.8   4 5.700000
## Hornet 4 Drive    21.4   6 3.566667
## Hornet Sportabout 18.7   8 2.337500
## Valiant           18.1   6 3.016667
```

Basic Programming

Variables

```
# TYPE OF DATA :
1 # integer
```

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

```
1.02 # numeric
```

```
## [1] 1.02
```

```
"Hello" # String
```

```
## [1] "Hello"
```

```
"a" # character
```

```
## [1] "a"
```

```
1:2 # vector
```

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

```
  # and also matrix, list, dataframe

# We bind those objects to a variable, which is juste a name, a placeholder, a reference, you name it.
```

Control Flow

```
# How to treat objects differently based on some characteristics.
# Below we write a if/else statement that turns x into |x|
x = 2
if (x < 0) {
  x = -x
} else {
  x = x
}
```

Loops

```
# Iterate along a sequence (can be any sequence)
for (i in c("mangos", "bananas", "apples")){
  print(paste("I love",i))
}
```

```
## [1] "I love mangos"
## [1] "I love bananas"
## [1] "I love apples"
```

Functions

Task 6

- Write a for loop that counts down from 10 to 1, printing the value of the iterator to the screen.
- Modify that loop to write "i iterations to go" where i is the iterator
- Modify that loop so that each iteration takes roughly one second. You can achieve that by adding the command Sys.sleep(1) below the line that prints "i iterations to go".
- Finally, let's create a function called ticking_bomb. it takes no arguments, it's body is the loop you wrote in the preceding question. The only think you should add to the body is a line after the loop finishes, printing "BOOOOM!" with print("BOOOOM!"). You can repeatedly redefine the function in the console, and try it out with ticking_bomb().

```
for (i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```
for (i in 10:1){
  print(paste(i, "iterations to go"))
  Sys.sleep(1)
}
```

```
## [1] "10 iterations to go"
## [1] "9 iterations to go"
## [1] "8 iterations to go"
## [1] "7 iterations to go"
## [1] "6 iterations to go"
## [1] "5 iterations to go"
## [1] "4 iterations to go"
## [1] "3 iterations to go"
## [1] "2 iterations to go"
## [1] "1 iterations to go"
```

```
ticking_bomb = function(){
  for (i in 10:1){
    print(paste(i, "iterations to go"))
    Sys.sleep(0.3)
  }
  print('BOOOOM')
}

ticking_bomb() # Don't forget to call the function, and don't forget the parentheses !!
```

```
## [1] "10 iterations to go"
## [1] "9 iterations to go"
## [1] "8 iterations to go"
## [1] "7 iterations to go"
## [1] "6 iterations to go"
## [1] "5 iterations to go"
## [1] "4 iterations to go"
## [1] "3 iterations to go"
## [1] "2 iterations to go"
## [1] "1 iterations to go"
## [1] "BOOOOM"
```