tasks for session 1

Florian Oswald

8/18/2018

task 1

1. Create a vector of five ones, i.e. `[1,1,1,1,1]` `rep(1,5)`
2. Notice that the colon operator `a:b` is just short for *construct a sequence* ***from*** `a` ***to*** `b`. Create a vector the counts down from 10 to 0, i.e. it looks like `10,9,8,7,6,5,4,3,2,1,0`! `10:0`
3. the `rep` function takes additional arguments `times` (as above), and `each`, which tells you how often *each element* should be repeated (as opposed to the entire input vector). Use `rep` to create a vector that looks like this: `1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3` `rep(1:3,times=2,each=3)`

task 2

1. Create a vector filled with 10 numbers drawn from the uniform distribution (hint: use function `runif`) and store them in `x`. `x = runif(10)`
2. Using logical subsetting as above, get all the elements of `x` which are larger than 0.5, and store them in `y`. `y = x[x>0.5]`
3. using the function `which`, store the *indices* of all the elements of `x` which are larger than 0.5 in `iy`. `iy = which(x>0.5)`
4. Check that `y` and `x[iy]` are identical. `identical(y,x[iy])` or `all(y == x[iy])`

Task 3

1. Create a vector containing `1,2,3,4,5` called v. `v = 1:5`
2. Create a (2,5) matrix `m` containing the data `1,2,3,4,5,6,7,8,9,10`. The first row should be `1,2,3,4,5`. `m = matrix(data = 1:10,nrow=2,ncol=5,byrow=T)`
3. Perform matrix multiplication of `m` with `v`. Use the command `%*%`. What dimension does the output have? `dim(m%*% v)`,
4. Why does `v %*% m` not work? non-conformable

Task 4

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

Task 5

1. How many observations are there in `mtcars`? `nrow(mtcars)`
2. How many variables? `ncol(mtcars)`
3. What is the average value of `mpg`? `mean(mtcars$mpg)`
4. What is the average value of `mpg` for cars with more than 4 cylinders, i.e. with `cyl>4`? `mean(subset(mtcars,subset=cyl>4)$mpg)`

Task 6

1. Write a for loop that counts down from 10 to 1, printing the value of the iterator to the screen.

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

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

1. Modify that loop to write "i iterations to go" where `i` is the iterator

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

```
## [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. 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".

```
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"
```