The middle thingies

Steen Ladelund
zsly@novonordisk.com

November 26, 2018

Outline

Data types in R
- Numeric types
- Logicals
- Characters in R
- Factors
- Vectors
- Lists
- Exercise - dataframes

Conditional execution: `if() else`
- Syntax
- Example: caching

Conditional assignment: `ifelse(test, yes, no)`
- Syntax
- Exercise

Iteration over vectors and lists
- `lapply`
- Exercise

Bootstrapping in R

Data types in R - numeric types

`integer:`

```
class(1);class(1L)
```

```
[1] "numeric" [1] "integer"
```

`numeric(double):` Floating numbers.

```
cat(pi); class(pi); typeof(pi)
```

```
3.141593[1] "numeric" [1] "double"
```

```
format(pi, sci = T, digits = 10)
```

```
[1] "3.141592654e+00"
```

`complex:` Complex numbers.

```
A <- 1 + 1i; cat(A)
```

```
1+1i
```

```
cat(exp(A))
```

```
1.468694+2.287355i
```

```
c(Re(A), Im(A))
```

```
[1] 1 1
```


Logicals

- Takes values `TRUE` and `FALSE`.

```
TRUE & TRUE
```

```
[1] TRUE
```

```
TRUE & FALSE
```

```
[1] FALSE
```

```
TRUE | FALSE
```

```
[1] TRUE
```

```
xor(TRUE, TRUE)
```

```
[1] FALSE
```

Logicals - coercion

- Coercion to and from logicals is liberal and often silent:
- ```
  if(8) cat('Yes!') else cat('No')
  ```

  Yes!

  ```
  if(0L) cat('Yes!') else cat('No')
  ```

  No

  ```
  if(-3) cat('Yes!') else cat('No')
  ```

  Yes!

  ```
  as.logical('T')
  ```

  [1] TRUE

  ```
  as.logical('false')
  ```

  [1] FALSE

  ```
  as.logical('a')
  ```

  [1] NA

Characters in R

- Contains a string:

  ```
  A <- "ABC"
  ```

- Lots of functions work on character variables in R
  - `substr(x, start, stop)`
  - `grep(pattern, x)`
  - `strsplit(x, split)`

Factors

- Data type with given attainable values (levels).
- ```
  A <- factor(sample(rep(c('F', 'M'), 5)))
  str(A)
  ```

  Factor w/ 2 levels "F","M": 2 2 1 2 1 1 1 2 1 2
- Warning when factor levels are 'numbers':

```
A <- cumsum(1:5)
rbind(A, as.numeric(factor(A)))
```

```
##   [,1] [,2] [,3] [,4] [,5]
## A    1    3    6   10   15
##      1    2    3    4    5
```

```
str(factor(A))
```

```
##  Factor w/ 5 levels "1","3","6","10",..: 1 2 3 4 5
```


Vectors

- Default behavior in R is vector arithmetics, ie. operations are elemtwise on vectors.

  ```
  A <- c(1, 2); B <- c(3, 4); A+B
  ```

  [1] 4 6
- Vectors are *atomic* tuples. All elements has to be of the same type.

```
c(1, 'A')
```

```
##  chr [1:2] "1" "A"
```

Mixed type structures - list and data.frames

- Lists are arbitrary type structs in R:

```
list(1, 'A')
```

[[1]] [1] 1
[[2]] [1] "A"

- Subsetting is a litte different for lists:
  - `X[i]` returns a list, even if only one element is chosen:

```
X <- list('a', 1)
X[1]
```

[[1]] [1] "a"

  - To get the element without being in a list use `[[i]]`:

```
X[[1]]
```

[1] "a"

Exercise: data frames

- Data frames are lists where all elements are of the same length (like in a nice spreadsheet).
- Build a data frame using the `data.frame` function and examine the `class` and `mode` attributes of the result.
- Try out subsetting with both `[` and `[[`.
- For instance use the code below to get started:

```
A <- c(1, 3, 9)
B <- factor(c("M", "F", "M"))
```


Conditional execution - syntax

- `if(cond) expr1 [else expr2]`
-

```
if(T) cat('Oh yeah!') else cat('Oh no!')
```

Oh yeah!

```
if(F) cat('Oh yeah!') else cat('Oh no!')
```

Oh no!

-

```
A <- c(T, F)
if(A) cat('Oh yeah!') else cat('Oh no!')
## Warning in if (A) cat("Oh yeah!") else cat("Oh
no!"):  the condition has length > 1 and only the
first element will be used
```

Oh yeah!

Conditional execution - example: caching

- I often use to cache stuff that takes a long time:
-

```
New <- FALSE ## Set to TRUE for fresh run or complete run.
if(New) {
    a <- {stuff that takes a loooong time}
    save(a, file = "a.RData")
    } else load("a.RData")
```

- Or to execute (silent) controls on data:
-

```
T1 <- 2
T2 <- 1
if(T1 <= T2) stop("Wrong ordering of times")
```

Conditional assignment - syntax

- `ifelse(test, yes, no)`

  test: evaluates to vector of type `logical`
  yes: result when `test` is `TRUE`
  no: result when `test` is `FALSE`

- ```
  A <- sample(1:10)
  cat(A)
  ```

  4 1 10 2 6 5 7 3 8 9

  ```
  B <- ifelse(
      test = A <= 5,
      yes = 1,
      no = 2
  )
  cat(B)
  ```

  1 1 2 1 2 1 2 1 2 2

Conditional assignement - exercise

- Based on `Temp` in the `airquality` data mark days as hotter or colder than the day before. Hint: `dplyr::lag` shifts a variable one 'back' providing the temperature the previous day.

lapply

- `lapply(X, FUN)` iterates over elements and return result in list!
- with named function:

```
str(lapply(pi*(1:4)/4, sin))
```

```
## List of 4
##  $ : num 0.707
##  $ : num 1
##  $ : num 0.707
##  $ : num 1.22e-16
```

- or a *lambda*-like exprssion (un-names functions):

```
lapply(1:4, function(x) x %% 2)
```

[[1]] [1] 1
[[2]] [1] 0
[[3]] [1] 1
[[4]] [1] 0

- And finally

```
sapply(1:4, '%%', e2 = 2)
```

[1] 1 0 1 0

lapply - example

- `data.frame`'s are lists:

```
data(airquality)
lapply(airquality, summary)
```

```
## $Ozone
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00   18.00   31.50   42.13   63.25  168.00      37
##
## $Solar.R
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     7.0   115.8   205.0   185.9   258.8   334.0       7
##
## $Wind
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.700   7.400   9.700   9.958  11.500  20.700
##
## $Temp
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   56.00   72.00   79.00   77.88   85.00   97.00
##
## $Month
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.000   6.000   7.000   6.993   8.000   9.000
##
## $Day
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     8.0    16.0    15.8    23.0    31.0
```
```

lapply exercise

- sapply is a "simple" version of lapply returning a vector intead of a list.
- Use args(sapply) to get the syntax for sapply and use it to create a table of the data-classes in the airquality dataset.

Bootstrapping in R

- Bootstrapping is resampling with replacement
- Allows for assessment of sampling variance, eg. CI of quantiles
- ```
  sample(x = 1:5, size = 2) ## Random subset
  ```

  ```
  [1] 5 3
  ```

  ```
  sample(x = 1:5) ## Permutation
  ```

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

  ```
  sample(x = 1:5, replace = T) ## Resampling
  ```

  ```
  [1] 3 2 4 3 2
  ```
  ```