Apply R

Anatoly Dryga

2/9/2016

Prelim

What are data types:

```
str("1")
```

```
##  chr "1"
```

```
str(1)
```

```
##  num 1
```

```
str(1.0)
```

```
##  num 1
```

```
str(1L)
```

```
##  int 1
```

why this is important?.. think about memory required to store value

If we have **multiple** things:

- of the same type(can be stored/accessed efficiently) hidden from us completely in R(errrr)

```
v <- c(1,2,3)
str(v)
```

```
##  num [1:3] 1 2 3
```

- of different types

```
L <- list(1,"2",3L)
str(L)
```

```
## List of 3
##  $ : num 1
##  $ : chr "2"
##  $ : int 3
```

lapply, sapply

Looping: repeating something over and over.

lapply/sapply - just loops hidden/written differently (this statement can start a **flame** war).

Starting with vector and returning the vector:

```
add100 <- function(x) {x + 100}
sapply(v, add100)
```

```
## [1] 101 102 103
```

(in words applying function add100 for each element of v)

Or using anonymous(if you don't need a name) function:

```
sapply(v, function(x) { x + 100})
```

```
## [1] 101 102 103
```

which is really similar to(but we are not returning vector):

```
for (element in v) print(add100(element))
```

```
## [1] 101
## [1] 102
## [1] 103
```

And now lapply returns list:

```
lapply(v, function(x) { x + 100})
```

```
## [[1]]
## [1] 101
##
## [[2]]
## [1] 102
##
## [[3]]
## [1] 103
```

that fails:

```
lapply(L, function(x) { x + 100})
```

Why? For what data types '+' is defined?

(Beautiful) Hacks

- a dataframe can be used as list of columns:

```
library(lattice)
str(barley)
```

```
## 'data.frame':    120 obs. of  4 variables:
##  $ yield  : num  27 48.9 27.4 39.9 33 ...
##  $ variety: Factor w/ 10 levels "Svansota","No. 462",..: 3 3 3 3 3 3 7 7 7 7 ...
##  $ year   : Factor w/ 2 levels "1932","1931": 2 2 2 2 2 2 2 2 2 2 ...
##  $ site   : Factor w/ 6 levels "Grand Rapids",..: 3 6 4 5 1 2 3 6 4 5 ...
```

```
lapply(barley, function(x) length(unique(x)))
```

```
## $yield
## [1] 114
##
## $variety
## [1] 10
##
## $year
## [1] 2
##
## $site
## [1] 6
```

- every operation is a function call (and we can skip arguments - what a useful feature for bug intoduction)

```
"+"(2,3)
```

```
## [1] 5
```

```
A<-matrix(1:9, 3,3)
B<-matrix(4:15, 4,3)
C<-matrix(8:10, 3,2)
MyList<-list(A,B,C)
MyList
```

```
## [[1]]
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## [[2]]
##      [,1] [,2] [,3]
## [1,]    4    8   12
## [2,]    5    9   13
## [3,]    6   10   14
## [4,]    7   11   15
##
## [[3]]
##      [,1] [,2]
## [1,]    8    8
## [2,]    9    9
## [3,]   10   10
```

```r
lapply(MyList, "[", , 2)
```

```
## [[1]]
## [1] 4 5 6
##
## [[2]]
## [1]  8  9 10 11
##
## [[3]]
## [1]  8  9 10
```

- use indexes and variable from the environment:

```r
sapply(2:3, function(x) { sum(A[, x])})
```

```
## [1] 15 24
```

Arguably cleaner version:

```r
sapply(2:3, function(x, m) { sum(m[, x])}, A)
```

```
## [1] 15 24
```

Why cleaner? globals are pure evil(most of the time)

When to chose lapply vs sapply?

Still to come...