Vectorization in R

Peilin Chen

11/3/2017

Vectorization

simplest way: adding two vectors together

```
x<-1:4
x
```

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

```
y<-3:6
y
```

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

```
z<-x+y
z
```

```
## [1]  4  6  8 10
```

Usefully functions for vectorization

1.sapply(vec, f)

2.lapply(vec,f)

3.apply(vec,f)

4.apply(matrix,1/2,f)

5.tapply(vector,grouping,f)

6.by(dataframe,grouping,f)

```
M<-matrix(seq(1,20),4,5)
M
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
```

apply min to rows

```
apply(M,1,min)
```

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

```
#if we use loop
for (i in 1:4){
  print(min(M[i,]))
}
```

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

apply min to columns

```
apply(M,2,min)
```

```
## [1]  1  5  9 13 17
```

lapply(list,function):

when you apply a function to each element in a list and return a list back

```
ls<-list(a=1:5,b=2:5,c=10:20)
ls
```

```
## $a
## [1] 1 2 3 4 5
##
## $b
## [1] 2 3 4 5
##
## $c
##  [1] 10 11 12 13 14 15 16 17 18 19 20
```

get the length of the nested list within ls

```
lapply(ls,length)
```

```
## $a
## [1] 5
##
## $b
## [1] 4
##
## $c
## [1] 11
```

get the sum of each nested list

```
lapply(ls,sum)
```

```
## $a
## [1] 15
##
## $b
## [1] 14
##
## $c
## [1] 165
```

sapply(list,function)=> vector

apply the function to each element of a list and return a vector

```
sapply(ls,length)
```

```
##  a  b  c
##  5  4 11
```

```
sapply(ls,sum)
```

```
##   a   b   c
##  15  14 165
```

vapply(list,function,FUN.VALUE)

vapply - When you want to use sapply but perhaps need to squeeze some more speed out of your code.

```
vapply(ls,length,FUN.VALUE = 0L)
```

```
##  a  b  c
##  5  4 11
```

mapply(function,list1,list2)=>elementwise

multivariate version of sapply. mapply applies FUN to the first elements of each … argument, the second elements, the third elements, and so on. Arguments are recycled if necessary. for example I want to get the product of two list elementwisely 1,2,3,4
4,3,2,1,

```
mapply("*",1:4,4:1)
```

```
## [1] 4 6 6 4
```

tapply(vector,grouping,f)

For when you want to apply a function to subsets of a vector and the subsets are defined by some other vector, usually a factor.

```
x <- 1:20
y<-factor(rep(letters[1:5],each=4))
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
y
```

```
##  [1] a a a a b b b b c c c c d d d d e e e e
## Levels: a b c d e
```

To get the summation group by letters.

```
tapply(x,y,sum)
```

```
##  a  b  c  d  e
## 10 26 42 58 74
```

by (dataframe,grouping,)

```
#iris is a dataset about fish
by(iris, iris$Species,summary)
```

```
## iris$Species: setosa
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100  
##  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200  
##  Median :5.000   Median :3.400   Median :1.500   Median :0.200  
##  Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246  
##  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300  
##  Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600  
##        Species  
##  setosa    :50  
##  versicolor: 0  
##  virginica : 0  
##                 
##                 
##                 
## -------------------------------------------------------- 
## iris$Species: versicolor
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width   
##  Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000  
##  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200  
##  Median :5.900   Median :2.800   Median :4.35   Median :1.300  
##  Mean   :5.936   Mean   :2.770   Mean   :4.26   Mean   :1.326  
##  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500  
##  Max.   :7.000   Max.   :3.400   Max.   :5.10   Max.   :1.800  
##        Species  
##  setosa    : 0  
##  versicolor:50  
##  virginica : 0  
##                 
##                 
##                 
## -------------------------------------------------------- 
## iris$Species: virginica
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.900   Min.   :2.200   Min.   :4.500   Min.   :1.400
##  1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100   1st Qu.:1.800
##  Median :6.500   Median :3.000   Median :5.550   Median :2.000
##  Mean   :6.588   Mean   :2.974   Mean   :5.552   Mean   :2.026
##  3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875   3rd Qu.:2.300
##  Max.   :7.900   Max.   :3.800   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    : 0
##  versicolor: 0
##  virginica :50
##
##
##
```

aggregate can be seen as a different way of using tapply()

```r
aggregate(iris$Sepal.Length, list(iris$Species), mean)
```

```
##      Group.1     x
## 1     setosa 5.006
## 2 versicolor 5.936
## 3  virginica 6.588
```

```r
tapply(iris$Sepal.Length , iris$Species , mean)
```

```
##     setosa versicolor  virginica
##      5.006      5.936      6.588
```