l/s/t/a/mapply Family&Split

Ni

lapply returns a list

```
x=list(a=1:5,b=rnorm(10))

lapply(x, mean)
```

```
## $a
## [1] 3
##
## $b
## [1] -0.0424803
```

```
y=1:4
lapply(y, runif)
```

```
## [[1]]
## [1] 0.6647746
##
## [[2]]
## [1] 0.1824791 0.9165147
##
## [[3]]
## [1] 0.84131335 0.44172721 0.01096049
##
## [[4]]
## [1] 0.4060033 0.9216578 0.7254111 0.7569498
```

```
x=list(a=matrix(1:4,2,2),b=matrix(1:6,3,2))
x
```

```
## $a
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## $b
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```r
# An anonymous function for extracting the first column of each matrix
lapply(x, function(elt)elt[,1])
```

```
## $a
## [1] 1 2
##
## $b
## [1] 1 2 3
```

sapply returns a simplier result

```r
x=list(a=1:5,b=rnorm(10))
sapply(x, mean)
```

```
##          a          b
##  3.0000000 -0.1133435
```

```r
mean(x) # this will not work
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning NA
```

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

apply

```r
x=matrix(rnorm(8),4,2)
rowMeans=apply(x,1,mean) # rowMeans(x)
colMeans=apply(x,2,mean) # colMeans(x)
rowSums=apply(x, 1, sum) # rowSums(x)
colSums(x)
```

```
## [1] -0.3482293  0.9123978
```

```r
#Quantiles of the rows of a matrix
x=matrix(rnorm(20),4,5)
apply(x,1,quantile,probs=c(0.25,0.75))
```

```
##            [,1]        [,2]       [,3]     [,4]
## 25% -0.08061132 -0.04949804 -0.2486162 0.551980
## 75%  0.87935605  0.52436600  0.2138679 1.643882
```

```r
#Average matrix in an array
a=array(rnorm(2*2*10),c(2,2,10))
# The average of 2*2 matrix
apply(a,c(1,2),mean)
```

```
##            [,1]      [,2]
## [1,] 0.03285610 0.2973078
## [2,] 0.07405581 0.2602751
```

```
rowMeans(a,dim=2)
```

```
##            [,1]      [,2]
## [1,] 0.03285610 0.2973078
## [2,] 0.07405581 0.2602751
```

mapply

can take multiple list arguments, then apply a function to the elements of the multiple lists, in parallel

```
# it is tedious to type
#list(rep(1,4),rep(2,3),rep(3,2),rep(4,1))
# mapply makes it easier
mapply(rep, 1:4,4:1)
```

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

Vectorizing a Function

```
noise<-function(n,mean,sd){
  rnorm(n,mean,sd)
}
noise(5,1,2)
```

```
## [1]  3.622502640 -0.023378092  0.096835918 -0.007032672 -1.450647745
```

```
mapply(noise, 1:5,1:5,2)
```

```
## [[1]]
## [1] 1.581195
##
## [[2]]
```

```
## [1] 3.157373 1.376143
##
## [[3]]
## [1] -0.4643758  1.5556715  5.2546965
##
## [[4]]
## [1] 5.635564 2.586031 1.558517 1.434723
##
## [[5]]
## [1] 2.899039 5.855921 1.217099 4.519872 3.076474
```

tapply

is used to apply a function over subsets of a vector. x is a vector

```
x=c(rnorm(10),runif(10),rnorm(10,1))
# factor variable using gl function
# this factor variable has three levels, each level will be repeated 10 times
f=gl(3,10)
f
```

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

```
tapply(x, f, mean)
```

```
##          1          2          3
## -0.2146962  0.4560754  1.5398706
```

```
tapply(x, f, range)
```

```
## $`1`
## [1] -0.7948068  0.8906891
##
## $`2`
## [1] 0.09831382 0.99284077
##
## $`3`
## [1] -1.624206  3.289064
```

split

x is a vector or list or dataframe f is a factor or a list of factors returns a list

```
x<-c(rnorm(10),runif(10),rnorm(10,1))
f<-gl(3,10)
split(x,f)
```

```
## $`1`
##  [1] -0.7650808  0.6805642  1.2683954 -1.9326008  0.8934081 -0.6567783
##  [7]  0.1812757  1.4211479  0.4465087  0.8541604
##
## $`2`
##  [1] 0.1101029 0.7431721 0.9980995 0.8019006 0.8921998 0.2430184 0.8836633
##  [8] 0.7309660 0.9504799 0.5590776
##
## $`3`
##  [1]  0.3033614  0.1162787  0.9898692  0.6801937 -1.9361737  1.1635918
##  [7]  2.2588481  1.3921774  1.6665639  1.5912170
```

```
lapply(split(x,f), mean)
```

```
## $`1`
## [1] 0.2391001
##
## $`2`
## [1] 0.691268
##
## $`3`
## [1] 0.8225928
```

```
tapply(x, f, mean)
```

```
##         1         2         3
## 0.2391001 0.6912680 0.8225928
```

```
# spliting a dataframe
library(datasets)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
dim(airquality)
```

```
## [1] 153   6
```

```
# calculate the mean of ozon, solar, radiation within each month
# first split the data into separate months
s<-split(airquality,airquality$Month)
# Month is converted into factor variables
#lapply(s, function(x) colMeans(x[,c("Ozone","Solar.R","Wind")]))
# simplified the result
sapply(s, function(x) colMeans(x[,c("Ozone","Solar.R",
                                    "Wind")]))
```

```
##                5         6          7        8        9
## Ozone         NA        NA         NA       NA       NA
## Solar.R       NA 190.16667 216.483871       NA 167.4333
## Wind    11.62258  10.26667   8.941935 8.793548  10.1800
```

```
# remove the Nas
sapply(s, function(x) colMeans(x[,c("Ozone","Solar.R",
                                    "Wind")],na.rm=TRUE))
```

```
##                5         6          7          8         9
## Ozone    23.61538  29.44444  59.115385  59.961538  31.44828
## Solar.R 181.29630 190.16667 216.483871 171.857143 167.43333
## Wind     11.62258  10.26667   8.941935   8.793548  10.18000
```

```
# Splitting on more than one level
x=rnorm(10)
f1=gl(2,5)
f2=gl(5,2)
f1
```

```
##  [1] 1 1 1 1 1 2 2 2 2 2
## Levels: 1 2
```

```
f2
```

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

```
interaction(f1,f2) # concatenates the levels of one with the other, will get ten levels
```

```
##  [1] 1.1 1.1 1.2 1.2 1.3 2.3 2.4 2.4 2.5 2.5
## Levels: 1.1 2.1 1.2 2.2 1.3 2.3 1.4 2.4 1.5 2.5
```

```
#interactions can create empty levels
# we can use split(), which will automatically call the interaction function
str(split(x,list(f1,f2)))
```

```
## List of 10
##  $ 1.1: num [1:2] -0.237 0.615
##  $ 2.1: num(0)
##  $ 1.2: num [1:2] -1.63 -1.06
##  $ 2.2: num(0)
##  $ 1.3: num -1.9
##  $ 2.3: num 1.03
##  $ 1.4: num(0)
##  $ 2.4: num [1:2] 1.965 -0.854
##  $ 1.5: num(0)
##  $ 2.5: num [1:2] -1.86 -1.05
```

```r
# there are some empty levels, we can drop them
str(split(x,list(f1,f2),drop = TRUE))
```

```
## List of 6
##  $ 1.1: num [1:2] -0.237 0.615
##  $ 1.2: num [1:2] -1.63 -1.06
##  $ 1.3: num -1.9
##  $ 2.3: num 1.03
##  $ 2.4: num [1:2] 1.965 -0.854
##  $ 2.5: num [1:2] -1.86 -1.05
```

Split is a very handy function for splitting arbutrary objects according to levels of factor and then applying any type of function to those split elements of that list

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

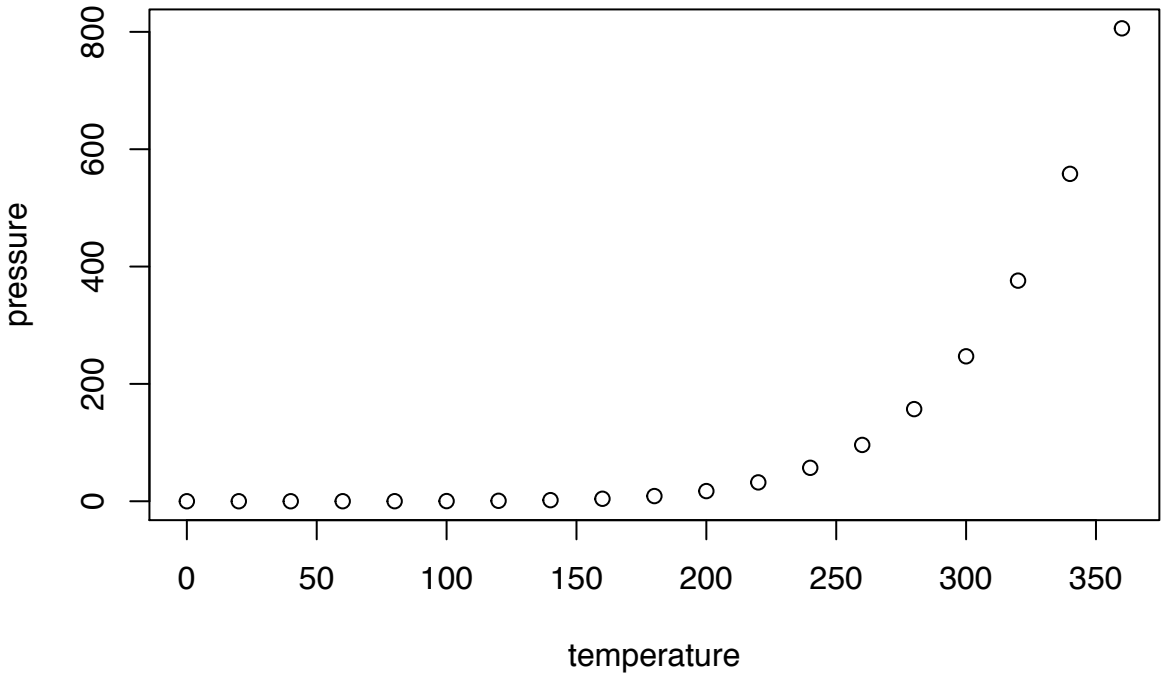
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.