

R Group

Young W. Lim

2018-02-09 Fri

- 1 Introduction
 - References
 - R Group
 - apply family
 - aggregate
 - plyr
 - data.table

"R for Everyone - Advanced Analytics and Graphic" J. P. Lander

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

- apply family
- aggregate
- plyr
- data.table

```
M <- matrix(1:9, nrow=3)
```

```
apply(M, 1, sum)    # row sum
```

```
apply(M, 2, sum)    # col sum
```

```
rowSums(M)
```

```
colSums(M)
```

apply with NA

```
M[2, 1] <- NA
```

```
apply(M, 1, sum)
```

```
apply(M, 1, sum, na.rm=TRUE)
```

```
rowSum(M)
```

```
rowSum(M, na.rm=TRUE)
```

lapply and sapply

```
L <- list(A=matrix(1:9, 3), B=1:5, C=matrix(1:4,2), D=2)
lapply(L, sum) # returns $A $B $C $D
```

```
sapply(L, sum) # returns a vector A B C D
```

```
N <- c("John", "Baker", "Paul")
```

```
lapply(N, nchar)
```

```
L1 <- list(A=matrix(1:16, 4), B=matrix(1:16, 2), C=1:5)
L2 <- list(A=matrix(1:16, 4), B=matrix(1:16, 8), C=15:1)
```

```
mapply(identical, L1, L2)
```

```
func <- function(x, y)
{
  NROW(x) + NROW(y)
}
```

```
mapply(func, L1, L2)
```


other apply function

`tapply`

`rapply`

`eapply`

`vapply`

`by`

```
require(qqplot2)
```

```
data(diamonds)
```

```
head(diamonds)
```

```
carat, cut, color, clarity, depth, table, price, x, y, z
```

```
cut: Fair, Good, Very Good, Premium, Ideal
```

```
aggregate(price ~ cut, diamonds, mean)
```

```
aggregate(price ~ cut + color, diamonds, mean) # ~ (cut+color
```

```
aggregate(cbind(price, carat) ~ cut, diamonds, mean)
```

```
aggregate(cbind(price, carat) ~ cut + color, diamonds, mean)
```

ddply (1)

```
require(plyr)
head(baseball)
```

```
id, year, stint, team, lq, q, ab, r, h, X2b, X3b, hr, rbi, sb
so, ibb, hbp, sh, sf, gidp
```

```
baseball$sf[baseball$year < 1954] <- 0
any(is.na(baseball$sf))
baseball$hbp[is.na(baseball$hbp)] <- 0
any(is.na(baseball$hbp))
baseball <- baseball[baseball$ab >= 50, ]
baseball$OBP <- with(baseball, (h+bb+hbp)/(ab+bb+hbp+sf))
tail(baseball)
```

ddply (2)

```
obp <- function(data)
{
  c(OBP = with(data, sum(h+bb+hbp)/sum(ab+bb+hbp+sf)))
}
```

```
careerOBP <- ddply(baseball, .variables = "id", .fun = obp)
careerOBP <- careerOBP[order(careerOBP$OBP, decreasing=TRUE),]
head(careerOBP, 10)
```

```
L <- list(A=matrix(1:9, 3), B=1:5, C=matrix(1:4, 2), D=2)
lapply(L, sum)
llply(L, sum)
identical(lapply(L, sum), llply(L, sum))
sapply(L, sum)
lply(L, sum)
```

plyr helper function

```
aggregate(price ~ cut, diamonds, each(mean, median))
```

```
cut, price.mean, price.median
```

```
system.time(dplyr(baseball, "id", nrow))
```

```
iBaseball <- idata.frame(baseball)
```

```
system.time(dplyr(iBaseball, "id", nrow))
```

```
require(data.table)
DF <- data.frame(A=1:10,
  B=letters[ 1:10],
  C=LETTERS[11:20],
  D=rep(c("One", "Two", "Three", length.out=10))
(1,2,3,4,5,6,7,8,9,10) x (A,B,C,D)
```

```
DT <- data.table(A=1:10,
  B=letters[ 1:10],
  C=LETTERS[11:20],
  D=rep(c("One", "Two", "Three", length.out=10))
(1,2,3,4,5,6,7,8,9,10) : x (A,B,C,D)
```

```
class(DF$B)      # factor
class(DT$B)      # character
```

data.table from data.frame

```
diamondsDT <- data.table(diamonds)
diamodnsDT
DT[1:2, ]
DT[DT$A >= 7, ]
DT[, list(A,c)]
DT[, B]
DT[, list(B)]
DT[, "S", with=FALSE]
DT[, c("A", "C"), with=FALSE]
```



```
tables()  
setkey(DT, D)  
DT  
key(DT)  
tables()  
DT["One",]  
DT[c("One", "Two"), ]  
  
setkey(diamondsDT, cut, color)  
diamondsDT[J("Ideal", "E"), ]
```

```
aggregate(price ~ cut, diamonds, mean)
diamondsDT[, mean(price), by=cut]
diamondsDT[, list(price = mean(price)), by=cut]
diamondsDT[, mean(color), by=list(cut, color)]
diamondsDT[, list(price = mean(price), carat = mean(carat)),
             by=list(cut, color)]
diamondsDT[, list(price = mean(price), carat = mean(carat)),
             caratSum = sum(carat)), by=cut]
diamondsDT[, list(price = mean(price), carat = mean(carat)),
             by = list(cut, color)]
```

