Introduction to the R Language

Roger Peng, Associate Professor

May 17, 2016

tapply

`tapply` is used to apply a function over subsets of a vector. I don't know why it's called `tapply`.

```
> str(tapply)
function (X, INDEX, FUN = NULL, ..., simplify = TRUE)
```

- `X` is a vector
- `INDEX` is a factor or a list of factors (or else they are coerced to factors)
- `FUN` is a function to be applied
- `...` contains other arguments to be passed `FUN`
- `simplify`, should we simplify the result?

tapply

Take group means.

```
> x <- c(rnorm(10), runif(10), rnorm(10, 1))
> 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
[24] 3 3 3 3 3 3 3
Levels: 1 2 3
> tapply(x, f, mean)
        1         2         3
0.1144464 0.5163468 1.2463678
```

tapply

Take group means without simplification.

```
> tapply(x, f, mean, simplify = FALSE)
$`1`
[1] 0.1144464

$`2`
[1] 0.5163468

$`3`
[1] 1.246368
```

tapply

Find group ranges.

```
> tapply(x, f, range)
$`1`
[1] -1.097309  2.694970

$`2`
[1] 0.09479023 0.79107293

$`3`
[1] 0.4717443 2.5887025
```

split

`split` takes a vector or other objects and splits it into groups determined by a factor or list of factors.

```
> str(split)
function (x, f, drop = FALSE, ...)
```

- `x` is a vector (or list) or data frame
- `f` is a factor (or coerced to one) or a list of factors
- `drop` indicates whether empty factors levels should be dropped

split

```
> x <- c(rnorm(10), runif(10), rnorm(10, 1))
> f <- gl(3, 10)
> split(x, f)
$`1`
 [1] -0.8493038 -0.5699717 -0.8385255 -0.8842019
 [5]  0.2849881  0.9383361 -1.0973089  2.6949703
 [9]  1.5976789 -0.1321970

$`2`
 [1] 0.09479023 0.79107293 0.45857419 0.74849293
 [5] 0.34936491 0.35842084 0.78541705 0.57732081
 [9] 0.46817559 0.53183823

$`3`
 [1] 0.6795651 0.9293171 1.0318103 0.4717443
 [5] 2.5887025 1.5975774 1.3246333 1.4372701
```

split

A common idiom is `split` followed by an `lapply`.

```
> lapply(split(x, f), mean)
$`1`
[1] 0.1144464

$`2`
[1] 0.5163468

$`3`
[1] 1.246368
```

Splitting a Data Frame

```
> 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
```

Splitting a Data Frame

```
> s <- split(airquality, airquality$Month)
> lapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R",
$`5`
   Ozone  Solar.R     Wind
      NA       NA 11.62258
$`6`
    Ozone   Solar.R      Wind
       NA 190.16667  10.26667
$`7`
     Ozone    Solar.R       Wind
        NA 216.483871   8.941935
```

Splitting a Data Frame

```
> sapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R",
                 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

> sapply(s, function(x) colMeans(x[, c("Ozone", "Solar.R",
                                    na.rm = TRUE))
                   5          6          7          8
Ozone       23.61538   29.44444   59.115385   59.96153
Solar.R    181.29630  190.16667  216.483871  171.85714
Wind        11.62258   10.26667    8.941935    8.79354
```

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)
 [1] 1.1 1.1 1.2 1.2 1.3 2.3 2.4 2.4 2.5 2.5
10 Levels: 1.1 2.1 1.2 2.2 1.3 2.3 1.4 ... 2.5
```

Splitting on More than One Level

Interactions can create empty levels.

```
> str(split(x, list(f1, f2)))
List of 10
 $ 1.1: num [1:2] -0.378  0.445
 $ 2.1: num(0)
 $ 1.2: num [1:2] 1.4066 0.0166
 $ 2.2: num(0)
 $ 1.3: num -0.355
 $ 2.3: num 0.315
 $ 1.4: num(0)
 $ 2.4: num [1:2] -0.907  0.723
 $ 1.5: num(0)
 $ 2.5: num [1:2] 0.732 0.360
```

split

Empty levels can be dropped.

```
> str(split(x, list(f1, f2), drop = TRUE))
List of 6
 $ 1.1: num [1:2] -0.378  0.445
 $ 1.2: num [1:2] 1.4066 0.0166
 $ 1.3: num -0.355
 $ 2.3: num 0.315
 $ 2.4: num [1:2] -0.907  0.723
 $ 2.5: num [1:2] 0.732 0.360
```