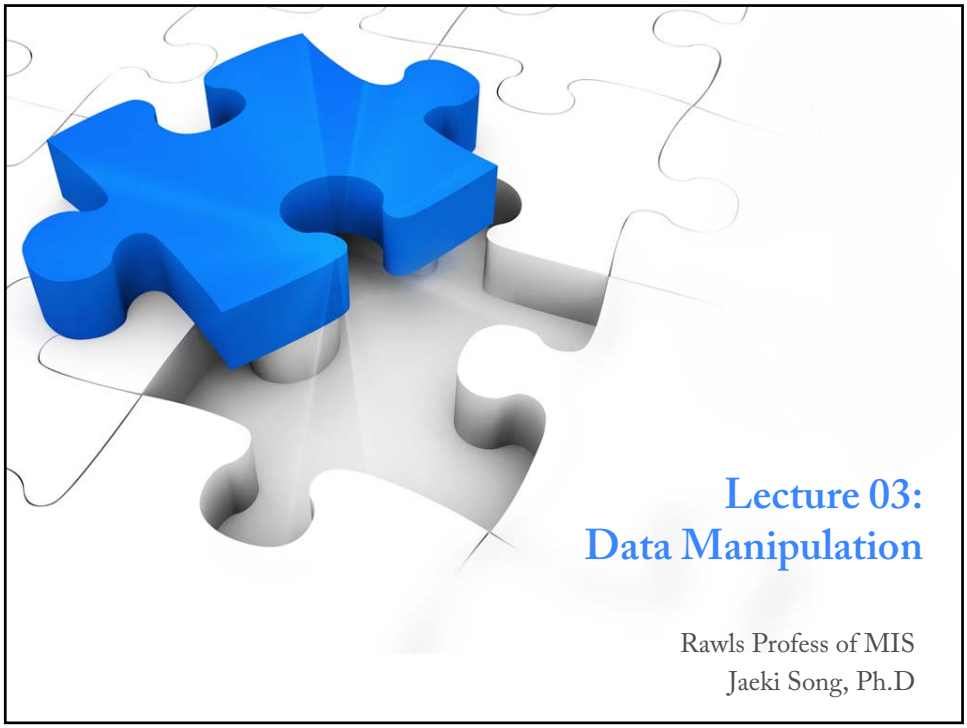


Lecture 03: Data Manipulation

Rawls Profess of MIS
Jaeki Song, Ph.D

Reading Data

- Reading data into R
  - reading CSVs
    - read.csv function

```
> x <-read.csv("C:/Temp/Scripting/Lc03.csv")
> x
  id    name scroe
1  1 Mr. Foo    95
2  2 Ms. Bar    97
3  3 Mr. Baz    92
```

```
> x$name = as.character(x$name)
> str(x)
'data.frame':   3 obs. of  3 variables:
 $ id   : int  1 2 3
 $ name : chr  "Mr. Foo" "Ms. Bar" "Mr. Baz"
 $ scroe: int  95 97 92
```

Data Reshaping

- Manipulating data takes a great deals of effort
  - the data can be rearranged
    - from column oriented to row oriented
  - *rbind* and *cbind*
    - tow datasets with either identical column or the same number or rows
    - combining a few vectors with cbind and then stack them using rbind

Data Reshaping

- rbind

```
> rbind(c(1,2,3), c(4,5,6))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

- cbind

```
> cbind(c(1,2,3), c(4,5,6))
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Group Manipulation

- apply family
  - apply, lapply, sapply, tapply
  - the first member
    - most restrictive
  - must be on a matrix
    - all of the elements must be of the same type
    - 1 meaning to operate over the rows and 2 is meaning to operate over the columns

```
> d <- matrix (1:9, ncol =3)
> d
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> apply (d, 1, sum)
[1] 12 15 18
```

Group Manipulation

- lapply
  - works by applying a function to each element of a *list* and returning the results as a list
    - lapply (X, function)

```
results<-lapply(1:3, function (x) {x*2})
```

- sapply()
  - return vector or matrix

```
> lapply(iris[,1:4], mean)
```

```
sapply(iris [, 1:4], mean)
```

Group Manipulation

- sqlpd package
  - Provides an easy way to perform SQL selects on R data frames
    - contains a single function sqldf whose help file contains more information and examples.
    - reads the indicated file into an sql database creating the database if it does not already exist.
    - applies the sql statement returning the result as a data frame. If the database did not exist prior to this statement it is removed
    - e.g.
      - install.packages("sqldf")
      - library(sqldf)

Group Manipulation

```
> sqldf("select distinct Species from iris")
     Species
1     setosa
2 versicolor
3  virginica
```

```
> sqldf('select avg("Sepal.Length") from iris where "Species" ="setosa"')
  avg("Sepal.Length")
1               5.006
```

```
> sqldf('select species, avg("sepal.length") from iris group by species')
     Species avg("sepal.length")
1     setosa               5.006
2 versicolor               5.936
3  virginica               6.588
```

Group Manipulation

- plyr package
  - epitomizes the "split-apply-combine" method of data manipulation
  - consists of five letters
    - _ _ ply
    - Function input type output type

Function	input type	output type
ddply	data.frame	data.frame
llply	list	list
ldply	list	data.frame

Group Manipulation

- ddply( )
  - "The Split-Apply-Combine for Data Analysis" data

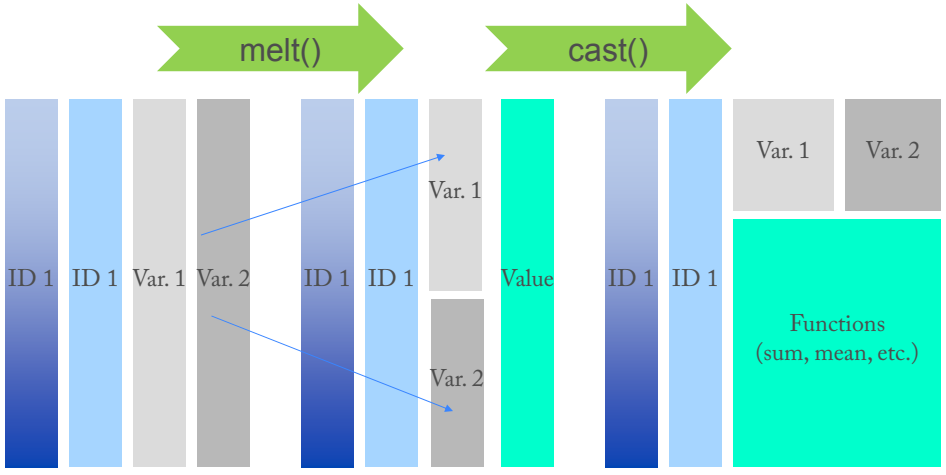
```
ddply(baseball, .(id), function(sub) {mean(sub$g)})
```

```
##           id       V1
## 1  aaronha01 143.39130
## 2  abernte02  40.05882
## 3  adairje01  77.66667
## 4  adamsba01  25.36842
```

Data Reshaping

- reshape
  - melting data
    - going from column orientation to row orientation
  - casting data
    - going from row orientation to column orientation
- Hadley Wickham's *reshape2* package

Data Reshaping



Data Reshaping

- melt( )

```
> head(french_fries)
   time treatment subject rep potato buttery grassy rancid painty
61    1         1       3   1    2.9     0.0    0.0    0.0    5.5
25    1         1       3   2   14.0     0.0    0.0    1.1    0.0
62    1         1      10   1   11.0     6.4    0.0    0.0    0.0
26    1         1      10   2    9.9     5.9    2.9    2.2    0.0
63    1         1      15   1    1.2     0.1    0.0    1.1    5.1
27    1         1      15   2    8.8     3.0    3.6    1.5    2.3
```

```
> m <- melt(id=1:4, french_fries)
> head(m)
  time treatment subject rep variable value
1    1         1       3   1   potato   2.9
2    1         1       3   2   potato  14.0
3    1         1      10   1   potato  11.0
4    1         1      10   2   potato   9.9
5    1         1      15   1   potato   1.2
6    1         1      15   2   potato   8.8
```

Data Reshaping

- cast( )

```
> c <- cast (data=m, treatment~variable, fun=mean)
> head(c)
  treatment   potato buttery    grassy   rancid   painty
1         1 6.887931      NA 0.6491379 4.065517 2.583621
2         2 7.001724      NA 0.6629310 3.624569       NA
3         3       NA      NA        NA       NA       NA
```

Data Reshaping

- dcast()
  - cast the molten data back into the wide format
  - arguments
    - first argument- the data to be used
    - second argument- formula
    - third argument
      - the column (as a character) that holds the values to be populated into the new columns

Data Reshaping

```
> smiths
     subject time age weight height
1 John Smith    1  33     90   1.87
2 Mary Smith    1  NA     NA   1.54
```

```
> m<-melt(id=1:2, smiths)
> m
     subject time variable value
1 John Smith    1      age 33.00
2 Mary Smith    1      age    NA
3 John Smith    1   weight 90.00
4 Mary Smith    1   weight    NA
5 John Smith    1   height  1.87
6 Mary Smith    1   height  1.54
```

```
> x<-dcast(m, subject+time~...)
> x
     subject time age weight height
1 John Smith    1  33     90   1.87
2 Mary Smith    1  NA     NA   1.54
```

Data Table

- Extends and enhances the functionality of data.frames

```
> iris_table <-as.data.table(iris)
> iris_table
     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
  1:          5.1         3.5          1.4         0.2    setosa
  2:          4.9         3.0          1.4         0.2    setosa
  3:          4.7         3.2          1.3         0.2    setosa
  4:          4.6         3.1          1.5         0.2    setosa
  5:          5.0         3.6          1.4         0.2    setosa
 ---
146:          6.7         3.0          5.2         2.3 virginica
147:          6.3         2.5          5.0         1.9 virginica
148:          6.5         3.0          5.2         2.0 virginica
149:          6.2         3.4          5.4         2.3 virginica
150:          5.9         3.0          5.1         1.8 virginica
> iris_table[1:7,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1:          5.1         3.5          1.4         0.2  setosa
2:          4.9         3.0          1.4         0.2  setosa
3:          4.7         3.2          1.3         0.2  setosa
4:          4.6         3.1          1.5         0.2  setosa
5:          5.0         3.6          1.4         0.2  setosa
6:          5.4         3.9          1.7         0.4  setosa
7:          4.6         3.4          1.4         0.3  setosa
```

```
> x<-data.table(x = c(1,2,3), y=c("a", "b", "c"))
> x
   x y
1: 1 a
2: 2 b
3: 3 c
```