Note 6. Data Frame

Introduction to Statistical Programming

Chanmin Kim

Department of Statistics
Sungkyunkwan University

2022 Spring

Creating Data Frames

- Data frame: Format is similar to matrix, but each column may have different mode.
- Data frame is a list.
- `data.frame()`: Creating data frame objects by combining vectors and matrices.
  - `stringsAsFactors = F` (default): When data frames are created from character vectors, character vectors are converted into *factor*.

```
> name <- c('Kim','Park','Lee')
> age <- c(25,23,21)
> male <- c(T,F,T)
> x <- data.frame(name,age,male, stringsAsFactors=T); x
  name age  male
1  Kim  25  TRUE
2 Park  23 FALSE
3  Lee  21  TRUE
```

Creating Data Frames

- `str()`: It shows structure of any R objects.

```
> str(x)
'data.frame': 3 obs. of  3 variables:
 $ name: chr  "Kim" "Park" "Lee"
 $ age : num  25 23 21
 $ male: logi  TRUE FALSE TRUE

> y <- data.frame(name,age,male, stringsAsFactors=T);
> str(y)
'data.frame': 3 obs. of  3 variables:
 $ name: Factor w/ 3 levels "Kim","Lee","Park": 1 3 2
 $ age : num  25 23 21
 $ male: logi  TRUE FALSE TRUE
```

Accessing Data Frames

- Access data frame like either list or matrix objects.

```
> x[[2]]
[1] 25 23 21
> x[2:3]
  age  male
1  25  TRUE
2  23 FALSE
3  21  TRUE
> x$age
[1] 25 23 21

> x[,2]
[1] 25 23 21
> x[,2:3]
  age  male
1  25  TRUE
2  23 FALSE
3  21  TRUE
```

Operations of Data Frames

- Operations of data frames are very similar to matrix objects.
- Extracting a subset from data frame.
  - Indexing: If `drop=F`, data frame with one column ⇒ data frame. Otherwise, it is vector.
  - Filtering.
  - `subset()`: Exclude obs. with `NA`.

```
> x <- data.frame(x1=c(6,3,6,3,8),x2=1:5,x3=7:11); x
  x1 x2 x3
1  6  1  7
2  3  2  8
3  6  3  9
4  3  4 10
5  8  5 11
> # Indexing
> x[1:3,]
  x1 x2 x3
1  6  1  7
2  3  2  8
3  6  3  9
```

Extracting Subsets

```
> class(x[1:3,])
[1] "data.frame"
> class(x[1:3,1])
[1] "numeric"
> class(x[1:3,1,drop=F])
[1] "data.frame"

> # Filtering
> x[x$x1 >= 5,]
  x1 x2 x3
1  6  1  7
3  6  3  9
5  8  5 11
> x[x[,1] >= 5,]
  x1 x2 x3
1  6  1  7
3  6  3  9
5  8  5 11
```

Extracting Subsets

```
> # subset()
> x <- data.frame(x1=c(6,3,NA,3,8),x2=1:5,x3=7:11); x
  x1 x2 x3
1  6  1  7
2  3  2  8
3 NA  3  9
4  3  4 10
5  8  5 11
> x[x$x1 >=5,]
   x1 x2 x3
1   6  1  7
NA NA NA NA
5   8  5 11
> subset(x,x1 >= 5)
  x1 x2 x3
1  6  1  7
5  8  5 11
```

Removing Obs. with NA

- `complete.cases()`:
  - Obs. that have at least one `NA` $\Rightarrow$ `FALSE`.

```
> x <- data.frame(x1=rep('a',5),x2=c(6,3,NA,3,NA),x3=7:11); x
  x1 x2 x3
1  a  6  7
2  a  3  8
3  a NA  9
4  a  3 10
5  a NA 11
> complete.cases(x)
[1]  TRUE  TRUE FALSE  TRUE FALSE
> y <- x[complete.cases(x),]; y
  x1 x2 x3
1  a  6  7
2  a  3  8
4  a  3 10
```

Adding Obs.

- `rbind()`:
  - Objects to be combined should be the same mode for each column.
  - Both data frame & list objects are possible.
  - When all elements of data frame have the same mode (i.e., numeric, character, etc.), matrix or vector objects with that mode can be added (Column names should be matched with data frame).

```
> d1 <- data.frame(name=c('Kim','Choi','Park','Lee'),
+ age=c(22,27,24,32))
> d2 <- data.frame(age=c(23,22),name=c('Yoo','Kang'))
> rbind(d1,d2)
  name age
1  Kim  22
2 Choi  27
3 Park  24
4  Lee  32
5  Yoo  23
6 Kang  22
```

Adding Obs.

```
> d1 <- data.frame(name=c('Kim','Choi','Park','Lee'),
+ age=c(22,27,24,32))
> d2 <- list(age=c(23,22),name=c('Yoo','Kang'))
> rbind(d1,d2)
  name age
1  Kim  22
2 Choi  27
3 Park  24
4  Lee  32
5  Yoo  23
6 Kang  22
```

Adding Obs.

```
> d1 <- data.frame(name=c('Kim','Choi','Park','Lee'),
+ age=c(22,27,24,32),stringsAsFactors=T)
> d2 <- list(age=c(23,22),name=c('Yoo','Kang'))
> rbind(d1,d2)
> rbind(d1,d2)
  name age
1  Kim  22
2 Choi  27
3 Park  24
4  Lee  32
5 <NA>  23
6 <NA>  22
Warning message:
In `[<-.factor`(`*tmp*`, ri, value = c("Yoo", "Kang")) :
  invalid factor level, NA generated

> d1 <- data.frame(x1=c(1,4,3),x2=1:3)
> d2 <- matrix(0,2,2)
> colnames(d2) <- c('x1','x2')
> d3 <- rbind(d1,d2)
> class(d3)
```

Adding Variables

- `cbind()`: A vector or matrix (with one mode) can be added to data frame as columns.
- `dataframe$varname = vector object`: Creating and assigning a new variable.

```
> d <- data.frame(x1=c(1,4,3),x2=1:3)
> cbind(d,matrix(1,3,2))
  x1 x2 1 2
1  1  1 1 1
2  4  2 1 1
3  3  3 1 1
> cbind(d, d$x1-d$x2)
  x1 x2 d$x1 - d$x2
1  1  1           0
2  4  2           2
3  3  3           0
```

Adding Variables

```
> d$diff <- d$x1 - d$x2; d
  x1 x2 diff
1  1  1    0
2  4  2    2
3  3  3    0

> d[[4]] <- 1:3; d
  x1 x2 diff V4
1  1  1    0  1
2  4  2    2  2
3  3  3    0  3

> d[5:6] <- matrix(1,3,2); d
  x1 x2 diff V4 V5 V6
1  1  1    0  1  1  1
2  4  2    2  2  1  1
3  3  3    0  3  1  1
```

Merging Data Frames

- `merge()`: Combine two data frames according to the values of common variable.

```
> d1 <- data.frame(name=c('Kim','Choi','Park'),age=c(22,27,24))
> d2 <- data.frame(male=c(T,F,T),name=c('Park','Kim','Kang'))

> merge(d1,d2,by='name')
  name age  male
1  Kim  22 FALSE
2 Park  24  TRUE

> merge(d1,d2,by='name',all=T)
  name age  male
1 Choi  27    NA
2  Kim  22 FALSE
3 Park  24  TRUE
4 Kang  NA  TRUE
```

Merging Data Frames

```
> merge(d1,d2,by='name',all.x=T)
  name age  male
1 Choi  27    NA
2  Kim  22 FALSE
3 Park  24  TRUE

> merge(d1,d2,by='name',all.y=T)
  name age  male
1  Kim  22 FALSE
2 Park  24  TRUE
3 Kang  NA  TRUE

> # Different names of ID
> d1 <- data.frame(name=c('Kim','Choi','Park'),age=c(22,27,24))
> d2 <- data.frame(male=c(T,F,T),last=c('Park','Kim','Kang'))
> merge(d1,d2,by.x='name',by.y='last')
  name age  male
1  Kim  22 FALSE
2 Park  24  TRUE
```

Merging Data Frames

```
> # Duplicated ID
> d1 <- data.frame(name=c('Kim','Choi','Park'),age=c(22,27,24))
> d2 <- data.frame(male=c(T,F,T),name=c('Park','Kim','Kim'))
> merge(d1,d2,by='name')
  name age  male
1  Kim  22 FALSE
2  Kim  22  TRUE
3 Park  24  TRUE
```

Applying Functions

- `apply()`: Apply a function to each column or row of data frame ⇒ return a vector or matrix.
- `lapply()`: Apply a function to each column of data frame ⇒ return a list.

```
> d <- data.frame(x1=2:5,x2=6:9,x3=c(5,8,1,5))

> apply(d,1,mean)
[1] 4.333333 6.000000 4.333333 6.333333

> apply(d,2,mean)
  x1   x2   x3
3.50 7.50 4.75
```

Applying Functions

```
> d <- data.frame(name=c('Kim','Park','Choi'),
+ age=c(22,27,24),stringsAsFactors=F)
> x <- apply(d,2,sort); x
     name   age
[1,] "Choi" "22"
[2,] "Kim"  "24"
[3,] "Park" "27"
> x <- lapply(d,sort); x
$name
[1] "Choi" "Kim"  "Park"
$age
[1] 22 24 27
> as.data.frame(x)
  name age
1 Choi  22
2  Kim  24
3 Park  27
# as.data.frame()
> a = matrix(0,2,2); x = as.data.frame(a)
> is.data.frame(x)
[1] TRUE
```

Converting into Data Frames

- `as.data.frame()`: Convert objects into data frames.

```
# as.data.frame()
> a <- matrix(0,2,2)
> x <- as.data.frame(a)
> is.data.frame(x)
[1] TRUE
```