# Data Frames

> Created by Woo Hyun (Ray) Hwang

# Data Frame

## Creating data frames

Normally created using `data.frame()`.

`stringAsFactors` default is false from R 4.0.0.

```
kids <- c("Jack", "Jill")
ages <- c(12, 10)
d <- data.frame(kids, ages, stringsAsFactors=FALSE)
d
```

Data frames are n units * p variables, and when we read a csv file, it naturally becomes a data frame. It is a list with vectors of the same length.

```
class(d)
is.list(d) #data frame is a form of list
```

```
#Reading the exams.csv given
exams <- read.csv("exams.csv", header=T)
str(exams)
class(exams)
dim(exams)
colnames(exams) #check the names of columns
is.list(exams)
length(exams) #check the length (number of columns) of the data
names(exams) #just like list format
```

## Indexing

Index by rows and columns. We can obtain a partial data frame.

```
exams[exams$course.id==10, ] #row of which course.id = 10
exams[c(10, 20), ] #10th and 20th row
exams[c(10, 20), 2] #2nd col of 10th & 20th row, gives a vector
class(exams[c(10, 20), 2]) #integer (vector)
exams[c(10, 20), 2, drop=F] #gives data frame
class(exams[c(10, 20), 2, drop=F]) #data frame
exams[exams$course.id==10, "final"] #10th row of colname final
exams[exams$final >=38, ] #rows that have final score over or equal to 38
subset(exams, final > 38) #subset does not count NAs
```

The columns of a data frame are all vectors, so functions can be used on them.

```
median(exams[,2]) #median of the second column
median(exams$mid) #median of the 'mid' column (which is the 2nd col)
```

## NA

```
median(exams$final) #NA - there is a missing value in final
which(is.na(exams$final)) #24th student did not take final
median(exams$final, na.rm=TRUE) #remove NAs and calculate median
exams$final[is.na(exams$final)] <- 0 #change NAs to 0
exams[21:26, ]
median(exams$final)
```

## Graph- Data visualization

Using `plot(x, y, data=DF)` or `plot(y ~ x, data=DF)`.

```
with(exams, plot(final ~ mid, pch=20, xlim=c(0, 45), ylim=c(0, 45),
                 main="Exam Scores")) #pch : option for point shape
with(exams, cor(mid, final))
```

## Another data frame

```
book2 <- read.csv("book2.csv", header=T)
str(book2) #this data is sorted by student.name
summary(book2[, c(2, 3)]) #there are NAs in book2$project
book2$project[is.na(book2$project)] <- 0
summary(book2[, 3])
```

## Merging data frames

If the two data frames have the same order, then `cbind()`.

```
class.record <- cbind(exams, book2)
```

If the orders are different, we need a common key unit identifier. This is done by the `merge()` function.

```
#exams & book2 have course.id in common. This would be the key to merging.
class.record <- merge(exams, book2)
str(class.record)
head(class.record)
```

We can use the order of the key of the 2nd data frame to merge.

```
i.order <- order(book2$course.id)
book2.sort <- book2[i.order, ]
head(book2.sort, 5)
class.record.1 <- cbind(exams, book2.sort[, 4]) #merge the 4th row of book2.sort
head(class.record.1, 5)
```

To use `merge(x, y, by.x, by.y)` or `merge(x, y, by)`

```
kids <- c("Jack", "Jill", "Jillian", "John")
states <- c("CA", "MA", "MA", "HI")
d1 <- data.frame(kids, states)
d1
ages <- c(10, 7, 12)
kids <- c("Jill", "Lillian", "Jack")
d2 <- data.frame(ages, kids)
d2
d <- merge(d1, d2) #if the two have at least one common identifier
d
ages <- c(12,10,7)
pals <- c("Jack", "Jill", "Lillian")
d3 <- data.frame(ages, pals)
d3
merge(d1, d3, by.x="kids", by.y="pals") #same var. diff. names, then use 'by'
```

### `apply()`

`apply(x, 1, f)` applies function f to the rows of the df.

```
#total sum of each row : cols 2, 3, 5, 6 are taken to calculation
apply(class.record[, c(2, 3, 5, 6)], 1, sum)
class.record$total <- apply(class.record[, c(2, 3, 5, 6)], 1, sum) #new col
hist(class.record$total, xlab="score", nclass=10, main="Total scores")
  #nclass default is 10, shows the width
```

`apply(x, 2, f)` applies function f to the cols of the df.

```
#quantiles by 0.25 of each column : all rows are taken to calculation
apply(class.record[, c(2,3,5,6,7)], 2, quantile, prob=seq(0, 1, 0.25))
output <- apply(class.record[, c(2,3,5,6,7)], 2, quantile, prob=seq(0, 1, 0.25))
class(output) #matrix, array
as.data.frame(output)
```

## Factor vs. Character

`stringsAsFactors=FALSE` is default. For characters to be factorized, `stringAsFactors=TRUE` is needed.

```
set.seed(1)
g <- sample(c("A","B"), 100, replace=T)
x <- rnorm(100, 50, 10)
D <- data.frame(grp=g, score=x, stringsAsFactors=T) #char -> factors
str(D) #check for the factorization
boxplot(score ~ grp, data=D, xlab="score", horizontal = T) #horizontal switch
```

### `complete.cases()`

If a row has no NAs, returns TRUE. If not, returns FALSE.

```
#Randomly create a df with NAs
set.seed(1)
x <- data.frame(math=rnorm(100), engl=rnorm(100))
x[sample(1:100, 20), 1] <- NA #random 20 NAs in col 1 (math)
x[sample(1:100, 20), 2] <- NA #random 20 NAs in col 2 (engl)
sum(complete.cases(x)) #66 rows with no NAs
x.1 <- x[complete.cases(x), ] #indexing only completed cases with no NAs
str(x.1)
sum(is.na(x.1)) #there are no NAs
```

## sort & subset of a data frame

To order a data frame by vector a or by vector (a, b) :

`x[order(a), ]` or `x[order(a, b), ]`, note that it ends with a blank [, ]

```
data(mtcars)
str(mtcars)
mtcars[with(mtcars, order(cyl, disp)), ] #order from smallest to largest
mtcars[with(mtcars, order(cyl, -disp)), ] #order from largest to smallest (disp)
```

To subset, use `subset(x, criteria)`.

```
data(airquality)
str(airquality)
# obtain a subset of airquality such that Temp is above top 5%
subset(airquality, Temp > quantile(Temp, 0.95)) #quantile gives the percentage
```

## MLB analysis of Park Chanho stats and salaries

`{r results=FALSE} library(Lahman) data("Pitching") data("Salaries") str(Pitching) str(Salaries)`

```
#Getting the pitching stats of Park Chanho (id: parkch01)
pitching.1 <- subset(Pitching, playerID == 'parkch01')
str(pitching.1)
salaries.1 <- subset(Salaries, playerID == 'parkch01')
str(salaries.1)
Park <- merge(pitching.1, salaries.1, by = "yearID") #yearID is the key
str(Park)
```

```
#Pick out only the cols we need
Park[, c(1:5, 34)]
#Plotting
with(Park, plot(ERA, salary/1000, xlim=c(3, 12), type="n", main="Park Chanho"))
with(Park, text(ERA, salary/1000, substring(yearID, 3, 4),
                col=c('red', 'blue'), cex=0.8))
  #substring(data, start, end) : extracts text
  #col : ERA is red, salary/1000 is blue
  #cex determines the size of text scaled to the size of point
```

## Four joins

With A and B, full join (A or B) / inner join (A and B) / right join (A) / left join (B). Think of these as calculations of sets.

```
df1 <- data.frame(CustomerId = c(1,2,3,5,7), Product = c(rep("Toaster", 3),
                                                         rep("Radio", 2)))
df2 <- data.frame(CustomerId = c(1,2,4,6), State = c(rep("Alabama", 2),
                                                     rep("Ohio", 2)))
merge(x = df1, y = df2, by = "CustomerId", all = FALSE) #inner join
merge(x = df1, y = df2, by = "CustomerId", all = TRUE) #full join
merge(x = df1, y = df2, by = "CustomerId", all.x = TRUE) #left join
merge(x = df1, y = df2, by = "CustomerId", all.y = TRUE) #right join
```

Using the `dplyr` package, `{r message=FALSE} library(dplyr) inner_join(df1, df2, by="CustomerId") full_join(df1, df2, by="CustomerId") left_join(df1, df2, by="CustomerId") right_join(df1, df2, by="CustomerId")`