# Lecture 6. Data Frames

R and Data Visualization

BIG2006, Hanyang University, Fall 2022

# Data Frames

On an intuitive level, a *data frame* is like a matrix, with a two-dimensional rows-and-columns structure.

On a technical level, a data frame is a list, with the components of that list being equal-length vectors.

**Note:** It differs from a matrix in that each column may have a different mode (numbers and character strings).

**Note:** R does allow the components to be other types of objects, including other data frames. However, it is rare in practice, we will assume all components of a data frame are vectors.

# Creating Data Frames

```r
kids <- c("Jinho", "younga")
ages <- c(11, 9)
d <- data.frame(kids, ages, stringsAsFactors = FALSE)
d # matrix-like viewpoint
```

```
##     kids ages
## 1  Jinho   11
## 2 younga    9
```

**Note:** If `stringsAsFactors` is not specified, then by default, `stringsAsFactors` = `TRUE`. This means that if we create a data frame from a character vector, R will convert that vector to a *factor*. We'll cover factors in the next lecture.

## Accessing Data Frames

```
d[[1]] # d is a list, access it via component index values
```

```
## [1] "Jinho"  "younga"
```

```
d$kids # access it via component names
```

```
## [1] "Jinho"  "younga"
```

```
d[,1] # a matrix-like fashion
```

```
## [1] "Jinho"  "younga"
```

```
str(d)
```

```
## 'data.frame':    2 obs. of  2 variables:
##  $ kids: chr  "Jinho" "younga"
##  $ ages: num  11 9
```

# Other Matrix-Like Operations

Various matrix operations also apply to data frames.

## Extracting Subdata Frames

```
set.seed(20220912)
examsquiz <- data.frame(Exam.1=round(runif(10,0,5),1),
      Exam.2=round(runif(10,0,5),1),
      Quiz=round(runif(10,0,5),1))
head(examsquiz)
```

```
##   Exam.1 Exam.2 Quiz
## 1    2.2    3.1  3.4
## 2    0.7    4.9  3.9
## 3    1.0    0.7  3.5
## 4    3.3    0.1  2.8
## 5    2.7    2.7  2.3
## 6    2.7    0.7  3.8
```

```r
examsquiz[2:5,]
```

```
##   Exam.1 Exam.2 Quiz
## 2    0.7    4.9  3.9
## 3    1.0    0.7  3.5
## 4    3.3    0.1  2.8
## 5    2.7    2.7  2.3
```

```r
examsquiz[2:5,2] # vector
```

```
## [1] 4.9 0.7 0.1 2.7
```

```r
class(examsquiz[2:5,2])
```

```
## [1] "numeric"
```

- `drop = FALSE`: keep it as a (one-column) data frame

```
examsquiz[2:5,2,drop=FALSE]
```

```
##   Exam.2
## 2    4.9
## 3    0.7
## 4    0.1
## 5    2.7
```

```
class(examsquiz[2:5,2,drop=FALSE])
```

```
## [1] "data.frame"
```

- Filtering: extract the subframe of all students that satisfy the condition.

```
examsquiz[examsquiz$Exam.1 >= 2.8,]
```

```
##    Exam.1 Exam.2 Quiz
## 4     3.3    0.1  2.8
## 10    4.9    1.4  3.5
```

## More on Treatment of NA Values

- `na.rm = TRUE`: telling R to ignore NA values

```
x <- c(2, NA, 4)
mean(x)
```

```
## [1] NA
```

```
mean(x, na.rm=TRUE)
```

```
## [1] 3
```

- `subset()`: row selection without any trouble of specifying `na.rm = TRUE`

```
examsquiz[examsquiz$Exam.1 >= 2.8,]
```

```
##    Exam.1 Exam.2 Quiz
## 4     3.3    0.1  2.8
## 10    4.9    1.4  3.5
```

```
subset(examsquiz, Exam.1 >= 2.8)
```

```
##    Exam.1 Exam.2 Quiz
## 4     3.3    0.1  2.8
## 10    4.9    1.4  3.5
```

- `complete.cases()`: rid the data frame of any observation that has at least one NA values

```
d4 <- data.frame(kids=c("Jinho",NA,"Younga","Daeun"),
                 cities=c("Seoul","Busan","Daejeon",NA))
complete.cases(d4)
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```
d4[complete.cases(d4),]
```

```
##     kids  cities
## 1  Jinho   Seoul
## 3 Younga Daejeon
```

## Using the rbind() and cbind() Functions and Alternatives

- In using rbind() to add a row, the added row is typically in the form of another data frame or list.

```
d
```

```
##     kids ages
## 1  Jinho   11
## 2 younga    9
```

```
rbind(d, list("Daeun", 15))
```

```
##     kids ages
## 1  Jinho   11
## 2 younga    9
## 3  Daeun   15
```

- We can also create new columns from old ones.

```
eq <- cbind(examsquiz, examsquiz$Exam.2-examsquiz$Exam.1)
class(eq)
```

```
## [1] "data.frame"
```

```
head(eq)
```

```
##   Exam.1 Exam.2 Quiz examsquiz$Exam.2 - examsquiz$Exam.1
## 1    2.2    3.1  3.4                                 0.9
## 2    0.7    4.9  3.9                                 4.2
## 3    1.0    0.7  3.5                                -0.3
## 4    3.3    0.1  2.8                                -3.2
## 5    2.7    2.7  2.3                                 0.0
## 6    2.7    0.7  3.8                                -2.0
```

- A long column name can be changed via `names()`.
- On the other hand, how about the following?

```
examsquiz$ExamDiff <- examsquiz$Exam.2-examsquiz$Exam.1
head(examsquiz)
```

```
##   Exam.1 Exam.2 Quiz ExamDiff
## 1    2.2    3.1  3.4      0.9
## 2    0.7    4.9  3.9      4.2
## 3    1.0    0.7  3.5     -0.3
## 4    3.3    0.1  2.8     -3.2
## 5    2.7    2.7  2.3      0.0
## 6    2.7    0.7  3.8     -2.0
```

## Applying `apply()`

- We can use `apply()` on data frames, if the columns are all of the same type.

```
apply(examsquiz, 1, max)
```

```
##  [1] 3.4 4.9 3.5 3.3 2.7 3.8 3.3 3.8 4.5 4.9
```

# Merging Data Frames

In R, two data frames can be combined using the `merge()` function.

The simplest form is as follows:

```
merge(x,y)
```

**Note:** The above merges data frame `x` and `y`. It assumes that the two data frames have one or more columns with names in common.

```r
d1 <- data.frame(kids=c("Jinho","Younga","Daeun"),
                 cities=c("Seoul","Busan","Daejeon"))
d2 <- data.frame(ages=c(11,8,15),
                 kids=c("Jinho","Yuna","Daeun"))
merge(d1,d2)
```

```
##    kids  cities ages
## 1 Daeun Daejeon   15
## 2 Jinho   Seoul   11
```

**Note:** Here, the two data frames have the variable `kids` in common. R found the rows in which this variable had the same value of `kids` in both data frame, and then created a data frame with corresponding rows and with columns takes from data frames.

- `merge()` has named arguments by `by.x` and `by.y`, which handle cases in which variables have similar information but different names in the two data frames.

```
d3 <- data.frame(ages=c(11,15,8),
                 pals=c("Jinho","Daeun","Yuna"))
merge(d1,d3,by.x="kids",by.y="pals")
```

```
##    kids  cities ages
## 1 Daeun Daejeon   15
## 2 Jinho   Seoul   11
```

**Note:** Although `kids` and `pals` are in different data frames, it was meant to store the same information, and thus the merge made sense.

- Duplicate matches will appear in full in the result, possibly in undesirable ways.

```
d2a <- rbind(d2, list(19, "Daeun"))
d2a
```

```
##   ages  kids
## 1   11 Jinho
## 2    8  Yuna
## 3   15 Daeun
## 4   19 Daeun
```

```
merge(d1,d2a)
```

```
##    kids  cities ages
## 1 Daeun Daejeon   15
## 2 Daeun Daejeon   19
## 3 Jinho   Seoul   11
```

# Applying Functions to Data Frames

```
d
```

```
##     kids ages
## 1  Jinho   11
## 2 younga    9
```

```
dl <- lapply(d,sort)
dl # a list consisting two sorted vectors
```

```
## $kids
## [1] "Jinho"  "younga"
##
## $ages
## [1]  9 11
```

- `dl` is just a list, not a data frame. We could coerce it to a data frame, like this:

```
as.data.frame(dl)
```

```
##     kids ages
## 1  Jinho    9
## 2 younga   11
```

# Reference


- Matloff, N. The Art of R Programming: A Tour of Statistical Software Design. No Starch Press. Chapter 5.