Session 2 - Dataframes

Jongbin Jung

January 9-10, 2016

Dependencies

- Latest version ($\geq$ `3.1.2`) of `R` (*free* from `https://www.r-project.org/`)
- Latest version of `Rstudio` (also *free* from `https://www.rstudio.com/`)
- A bunch of *free* packages

```
install.packages('nycflights13')  # sample data frame
install.packages('dplyr')
install.packages('tidyr')
```

Data Frames: Introduction

- Data frames are the primary representation of data in `R`
- You can think of a data frame as a two-dimensional *table* of data
- It helps your sanity to always think of data frames as a table where

  ```
  Each column represents a variable/feature
  Each row represents an observation/instance
  ```

- Conceptually, a data frame is also a collection of vectors, i.e., each column is a vector that belongs to the (parent) data frame
- The fastest path to achieving `R`-ninja status is to get familiar with data frames

Data Frames: First Impression

- Let's load an existing data frame to take a look at

```
# install data package (only need to do once)
install.packages('nycflights13')
```

```
# load data package to workspace
library('nycflights13')
```

- The `nycflights13` package contains a single data frame named `flights`
- Contains data (16 variables) on all 336,776 flights that departed NYC (i.e. JFK, LGA, or EWR) in 2013
- See documentation for details on what the 16 variables are

```
?flights
```

Data Frames: First Impression (cont'd)

```
head(flights)  # take a peek at the data frame
```

```
## Source: local data frame [6 x 16]
##
##    year month   day dep_time dep_delay arr_time
##   (int) (int) (int)    (int)     (dbl)    (int)
## 1  2013     1     1      517         2      830
## 2  2013     1     1      533         4      850
## 3  2013     1     1      542         2      923
## 4  2013     1     1      544        -1     1004
## 5  2013     1     1      554        -6      812
## 6  2013     1     1      554        -4      740
## Variables not shown: arr_delay (dbl), carrier
##   (chr), tailnum (chr), flight (int), origin
##   (chr), dest (chr), air_time (dbl), distance
##   (dbl), hour (dbl), minute (dbl)
```

Some Question

- What questions could you ask (and answer) with this data?
  - how many flights were there each day?
  - what was the mean departure delay for flights every month/day?
  - what is the proportion of annual departures from each of the three airports?
  - what else?
- By the end of this session, we'll have the tools to answer most (if not all) of the questions you can come up with!

Data Frame Basics

Simple Example

- Use `data.frame()` function to create a data frame
- Arguments of `data.frame()` are vectors (of equal length) that constitute each column (variable)
- For example, let's create a data frame of the following table:

Age	Personality	Income
24	Good	2000
22	Bad	5800
23	Good	4200
25	Bad	1500
22	Good	6000

Simple Example (cont'd)

- We'll save the data frame to an object (I'll call mine `data`)

```
data <- data.frame(  # start the data.frame()
    age = c(24, 22, 23, 25, 22),
    personality = c('g', 'b', 'g', 'b', 'g'),
    income = c(2000, 5800, 4200, 1500, 6000)
)  # finish the data.frame() function
```

- Note that the new lines are just a matter of coding style, i.e., it makes the code easier to read
- The same data frame can be created in a single line:

```
data <- data.frame(age = c(24, 22, 23, 25, 22),
personality = c('g', 'b', 'g', 'b', 'g'), income
= c(2000, 5800, 4200, 1500, 6000))
```

Simple Example (cont'd)

- Let's take a look at our new data frame

```
data
```

```
##   age personality income
## 1  24           g   2000
## 2  22           b   5800
## 3  23           g   4200
## 4  25           b   1500
## 5  22           g   6000
```

Indexing: The $ Operator

- The $ operator lets you reference elements of an object (e.g., column vectors of a data frame) in R

```
data$age
```

```
## [1] 24 22 23 25 22
```

```
data$personality
```

```
## [1] g b g b g
## Levels: b g
```

- Similar to a . operation in other programming languages (but note that . has no special meaning in R!)

Munging Data with `dplyr`

Tidy Data with `tidyr`