Lecture 8

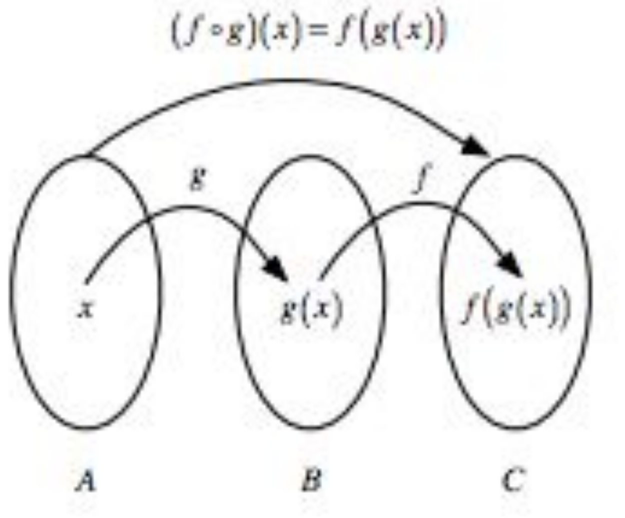
- Pipe Operator %>%
- Basic Structure
- Examples
- Additional **dplyr** Commands

The Pipe Operator: %>%

- Pipe operator commonly used with **dplyr** package to make R code easier to read.
- Enables one to pass the object on left hand side as first argument of function on the right hand side.

```
  x %>% f(y)
#is the same as
  f(x, y)

  x %>%  f(y) %>%  g(z)
#is the same as
  g(f(x, y),z)
```



The Pipe Operator: %>%

- Provides mechanism to pass the object on left hand side as first argument of function on the right hand side.

```
 x %>% f(y)
#is the same as
     f(x, y)

    x %>%  f(y) %>%  g(z)
#is the same as
    g(f(x, y),z)
```

- Use %>% to emphasize a sequence of actions, rather than the object that the actions are being performed on.
- Avoid using %>% when:
  - you need to manipulate more than one object at a time. Reserve pipes for a sequence of steps applied to one primary object.
  - there are meaningful intermediate objects that could be given informative names.

Basic Structure

- Syntax

```
df %>%
        do this operation %>%
        then do this operation %>%
        then do this operation ...
```

- Print output to console

```
Dataset %>%
        Select rows or columns to manipulate %>%
        Arrange or group the data %>%
        Calculate statistics or new variables of interest
```

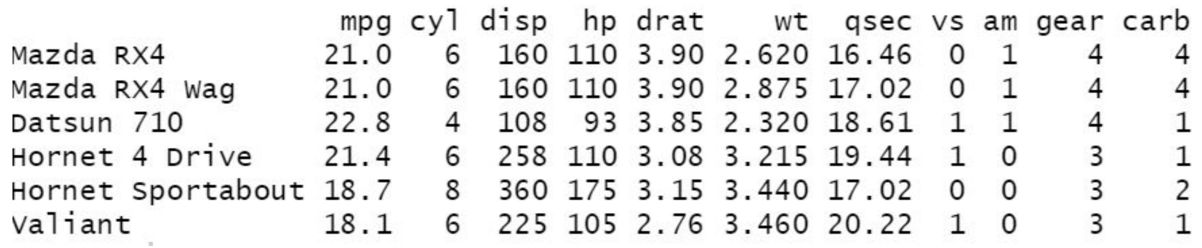
- Create new R object

```
my_summary  <-  Dataset %>%
                     Select rows or columns to manipulate %>%
                      Arrange or group the data %>%
                      Calculate statistics or new variables of
```

Example 1

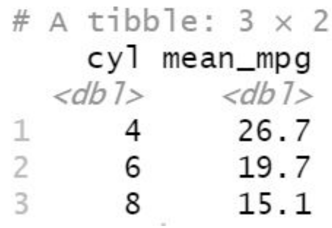
- Use pipe (%>%) operator to summarize one variable in the built-in mtcars dataset

```
library(dplyr)
data("mtcars")
head(mtcars)   #view first six rows of mtcars dataset
```



```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
#summarize mean mpg grouped by cyl
  mtcars %>%
      group_by(cyl) %>%
      summarise(mean_mpg = mean(mpg))
```



```
# A tibble: 3 x 2
    cyl mean_mpg
  <dbl>    <dbl>
1     4     26.7
2     6     19.7
3     8     15.1
```

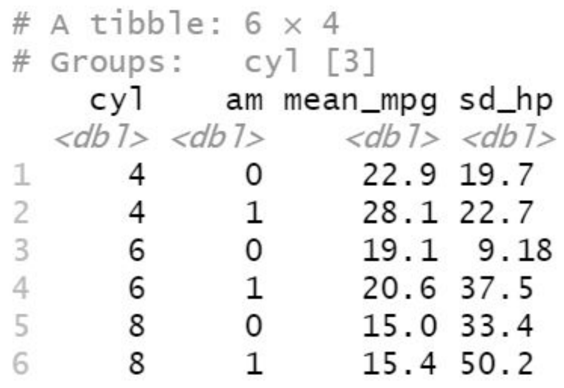
Essentially, the code says:

- Take the **mtcars** data frame.
- Group it by the **cyl** variable.
- Then summarize the mean value of the **mpg** variable.

Example 2

- Use pipe (%>%) operator to group and summarize multiple variables in the built-in mtcars dataset

```
#summarize mean mpg and standard dev of hp grouped by cyl and am
 mtcars %>%
     group_by(cyl, am) %>%
     summarise(mean_mpg = mean(mpg),
           sd_hp = sd(hp))
```



```
# A tibble: 6 x 4
# Groups:   cyl [3]
    cyl    am mean_mpg sd_hp
  <dbl> <dbl>    <dbl> <dbl>
1     4     0     22.9 19.7 
2     4     1     28.1 22.7 
3     6     0     19.1  9.18
4     6     1     20.6 37.5 
5     8     0     15.0 33.4 
6     8     1     15.4 50.2 
```

Essentially, the code says:

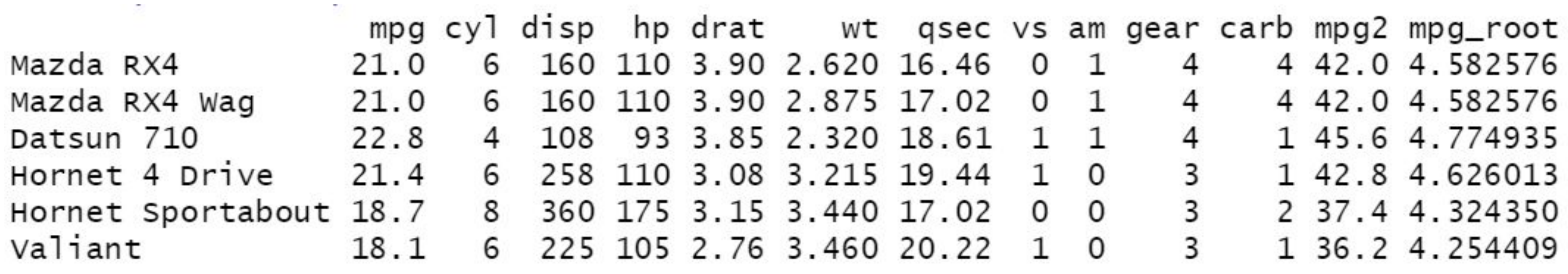
- Take the **mtcars** data frame.
- Group it by the **cyl** and the **am** variables.
- Then summarize the mean value of the **mpg** variable and the standard deviation of the **hp** variable.

Example 3

- Use pipe (%>%) perator along with mutate function from **dplyr** package to create two new variables in the built-in mtcars data frame

```
#add two new variables in mtcars
 new_mtcars <- mtcars %>%
         mutate(mpg2 = mpg*2,
                mpg_root = sqrt(mpg))

#view first six rows of new data frame
head(new_mtcars)
```



```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb mpg2 mpg_root
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 42.0 4.582576
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 42.0 4.582576
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 45.6 4.774935
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 42.8 4.626013
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 37.4 4.324350
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 36.2 4.254409
```

Essentially, the code says:

- Take the **mtcars** data frame.
- Create a new column called **mpg2** and a new column called **mpg_root**.

Additional **dplyr** Commands

- View() used with mutate() to see all the columns of a dataset
- between()
- Helping functions within select()
  - starts_with("abc"): matches names that begin with “abc”.
  - ends_with("xyz"): matches names that end with “xyz”.
  - contains("ijk"): matches names that contain “ijk”
  - matches("(.)\\1"): selects variables that match a regular expression.
  - num_range("x", 1:3): matches x1, x2 and x3.
  - rename(), which is a variant of select() that keeps all the variables that aren’t explicitly mentioned.
  - any_of()
- Cumulative and rolling aggregates
  - **dplyr** provides cumsum(), cumprod(), cummin(), cummax(), and cummean()
- Ranking min_rank(), desc() order()
- Measures of spread
  - sd()   standard measure of spread
  - IQR() interquartile range
  - mad() median absolute deviation