R 프로그래밍 #7

2019. 04. 17

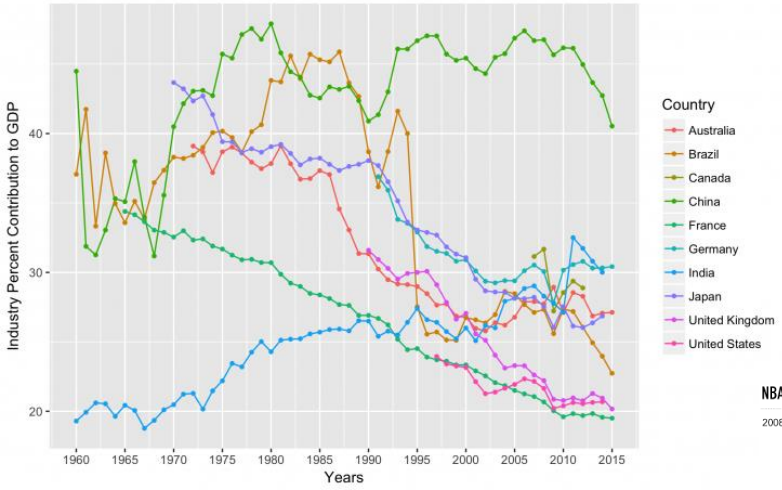
한국생명공학연구원
김하성

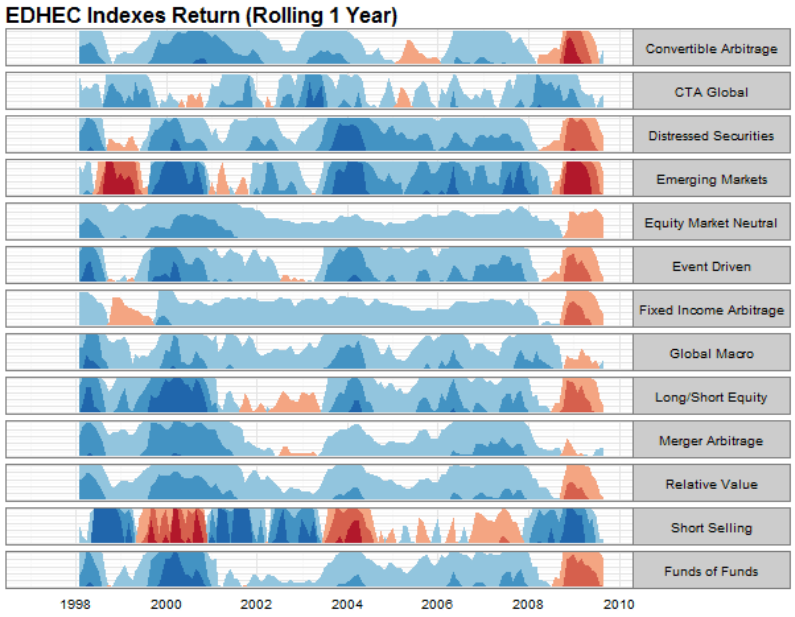
In today's lecture

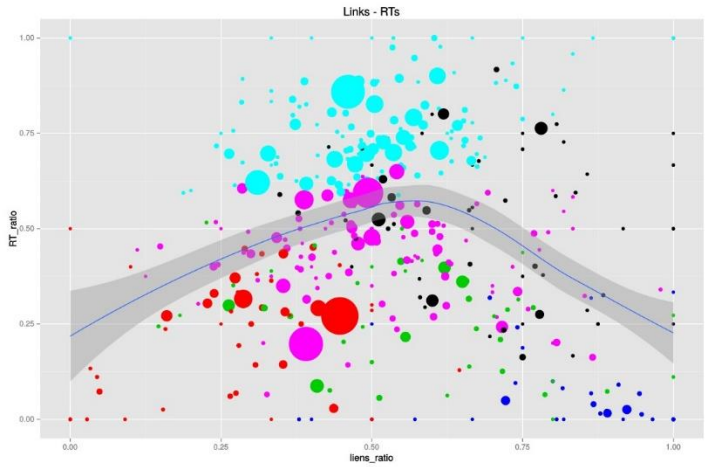
- To understand how to draw plots
  - barplot, line graph
  - + operator for layer

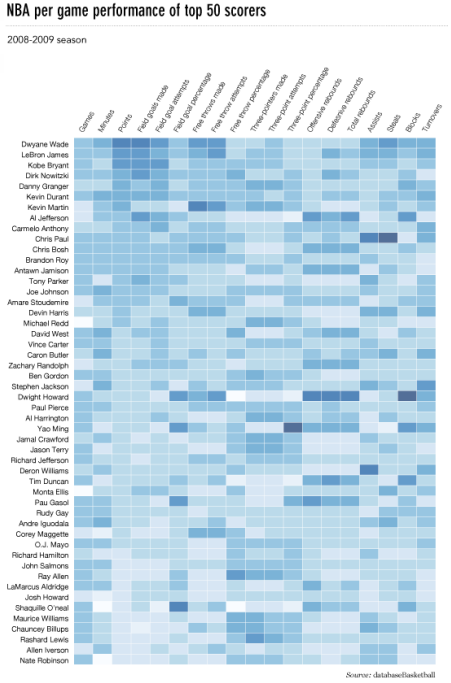
- To understand how to manipulate dataset
  - dplyr
  - %>% (pipe) operator

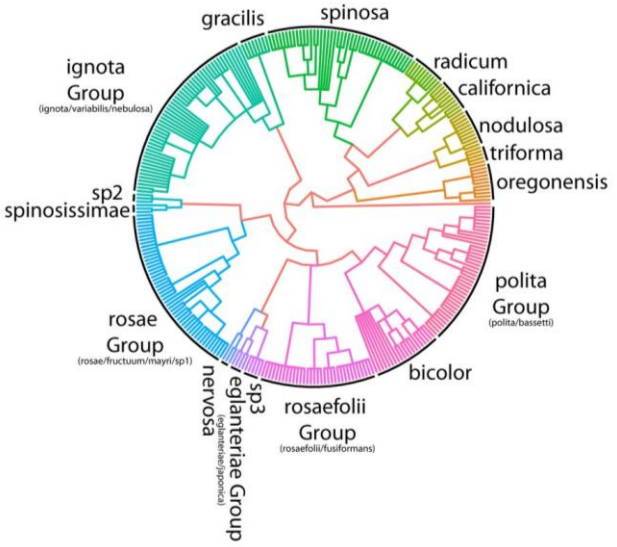
ggplot2


Industry Percent Contribution to GDP
40
30
20
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
Years
Country
Australia
Brazil
Canada
China
France
Germany
India
Japan
United Kingdom
United States


EDHEC Indexes Return (Rolling 1 Year)
Convertible Arbitrage
CTA Global
Distressed Securities
Emerging Markets
Equity Market Neutral
Event Driven
Fixed Income Arbitrage
Global Macro
Long/Short Equity
Merger Arbitrage
Relative Value
Short Selling
Funds of Funds
1998
2000
2002
2004
2006
2008
2010


Links - RTs
RT_ratio
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
liens_ratio


NBA per game performance of top 50 scorers
2008-2009 season
Games
Minutes
Points
Field goals made
Field goal attempts
Field goal percentage
Free throws made
Free throw attempts
Free throw percentage
Three-points made
Three-point attempts
Three-point percentage
Offensive rebounds
Defensive rebounds
Total rebounds
Assists
Steals
Blocks
Turnovers
Dwyane Wade
LeBron James
Kobe Bryant
Dirk Nowitzki
Danny Granger
Kevin Durant
Kevin Martin
Al Jefferson
Carmelo Anthony
Chris Paul
Chris Bosh
Brandon Roy
Antawn Jamison
Tony Parker
Joe Johnson
Amare Stoudemire
Devin Harris
Michael Redd
David West
Vince Carter
Caron Butler
Zachary Randolph
Ben Gordon
Stephen Jackson
Dwight Howard
Paul Pierce
Al Harrington
Yao Ming
Jamal Crawford
Jason Terry
Richard Jefferson
Deron Williams
Tim Duncan
Monta Ellis
Pau Gasol
Rudy Gay
Andre Iguodala
Corey Maggette
O.J. Mayo
Richard Hamilton
John Salmons
Ray Allen
LaMarcus Aldridge
Josh Howard
Shaquille O'neal
Maurice Williams
Chauncey Billups
Rashard Lewis
Allen Iverson
Nate Robinson
Source: databasefbasketball


gracilis
spinosa
radicum
californica
nodulosa
triforma
oregonensis
polita
Group
(polita/bassetti)
bicolor
rosaefolii
Group
(rosaefolii/fusiformans)
sp3
eglanteriae Group
(eglanteriae/japonica)
nervosa
rosae
Group
(rosae/fructuum/mayri/sp1)
sp2
spinosissimae
ignota
Group
(ignota/variabilis/nebulosa)

ggplot grammar

- Components
  - data frame (ggplot)
  - aesthetic factors such as color, size, etc (aes)
  - geometric factors such as point, line, bar, etc (geoms)
  - statistical factors (stats)
  - theme or scale to be used in aes
- How to draw
  - Determine what graph do you want
  - Use ggplot to indicate dataset and its aesthetic factors
  - Add layers to indicate geometric factors and appropriate statistics
  - Add layers for scale/theme

barplot

X axis	Height of bar represents	Common name
Continuous	Count	Histogram
Discrete	Count	Bar graph
Continuous	Value	Bar graph
Discrete	Value	Bar graph

One variable

Two variables

```
setwd("C:\\Rprog\\07")

x <- rnorm(100)
hist(x, br=10)

x <- sample(1:3, 100, replace = T)
barplot(table(x))

x <- rnorm(10)
y <- rnorm(10)
plot(x, y, type="h")

x <- 1:3
y <- table(sample(x, 100, replace = T))
barplot(y)
```

barplot - ggplot

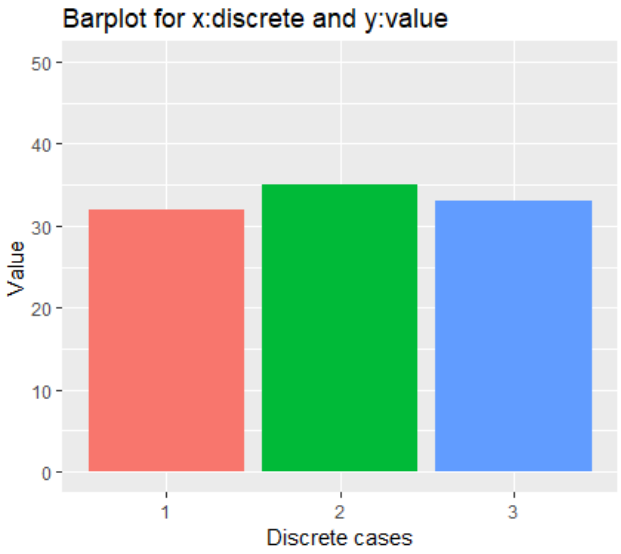
X axis	Height of bar represents	Common name
Continuous	Count	Histogram
Discrete	Count	Bar graph
Continuous	Value	Bar graph
Discrete	Value	Bar graph

```
##
dat <- data.frame(x=rnorm(100))
ggplot(dat, aes(x=x)) +
  geom_bar(stat="bin")

x <- sample(1:3, 100, replace = T)
dat <- data.frame(x=factor(x))
ggplot(dat, aes(x=x)) +
  geom_bar(stat="count")
```

exercise 7-1) Barplot

- Barplot with two variables using ggplot

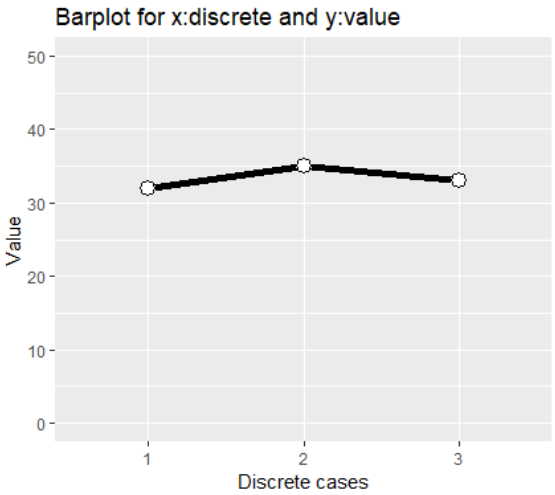
```
x <- rnorm(10)
y <- rnorm(10)
##
##  your code
##

x <- factor(1:3)
y <- tabulate(sample(x, 100, replace=T))
##
##  your code
##

## what if…
x <- 1:3
y <- tabulate(sample(x, 100, replace=T))
##
##  your code
##
```



Line graph

- The data points must be grouped so that it knows which points to connect

```
ggplot(dat, aes(x=x, y=y, fill=x, group=1)) +
  geom_line(stat="identity") +
  guides(fill=FALSE) +
  xlab("Discrete cases") +
  ylab("Value") +
  ylim(c(0,50))+
  ggtitle("Barplot for x:discrete and y:value")
```



Dataset for exercise

Experiment conditions

- **Cell types: 1~4**
- **Drug type: 1 (phenol)**
- **Drug concentrations: 11 points**
- **Replications: 4 times**

Dataset

```
setwd("C:\\Rprog\\07")

source("read_plate.R")

design_file_name <- "exp_design2.xlsx"
data_file_names <- c("20171012-phenol-1.xls",
                     "20171012-phenol-2.xls",
                     "20171227-phenol-1.xls",
                     "20171227-phenol-2.xls")

mydata1 <- multiple_plate_excel_reader2(design_file_name, data_file_names[1], sheet4design=1)
mydata2 <- multiple_plate_excel_reader2(design_file_name, data_file_names[2], sheet4design=2)
mydata3 <- multiple_plate_excel_reader2(design_file_name, data_file_names[3], sheet4design=3)
mydata4 <- multiple_plate_excel_reader2(design_file_name, data_file_names[4], sheet4design=4)

mydata <- rbind(mydata1, mydata2, mydata3, mydata4)
```

```
> head(mydata)
  well_names        OD    GFP sample_names replication drugname concentration
1        G02 0.9042823 124002            1           1   phenol         0e+00
2        F02 0.9368631 127999            1           1   phenol         5e-02
3        E02 0.9228352  44070            1           1   phenol         5e-01
4        D02 0.8994368   4280            1           1   phenol         5e+00
5        C02 0.9145258   3928            1           1   phenol         5e+01
6        B02 0.9241626   3882            1           1   phenol         5e+02
> dim(mydata)
[1] 308   7
```

Dataset

Experiment conditions

Cell types: 1~6

Drug type: 1 (phenol)

Drug concentrations: 11 points

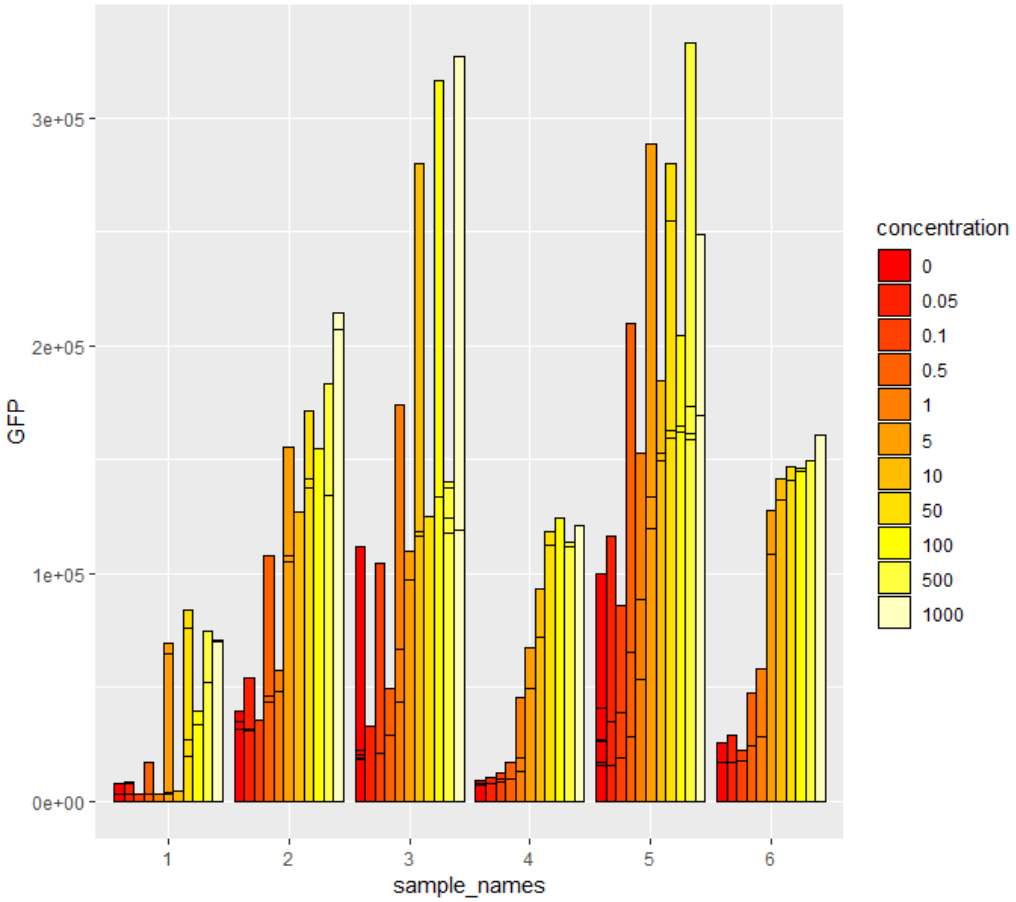
Replications: 4 times

```
> head(mydata)
  well_names        OD    GFP sample_names replication drugname concentration
1        G02 0.9042823 124002            1           1   phenol          0e+00
2        F02 0.9368631 127999            1           1   phenol          5e-02
3        E02 0.9228352  44070            1           1   phenol          5e-01
4        D02 0.8994368   4280            1           1   phenol          5e+00
5        C02 0.9145258   3928            1           1   phenol          5e+01
6        B02 0.9241626   3882            1           1   phenol          5e+02
> dim(mydata)
[1] 308   7
```

```
> str(mydata2)
'data.frame':   308 obs. of  7 variables:
 $ well_names   : chr  "G02" "F02" "E02" "D02" ...
 $ OD           : num  0.904 0.937 0.923 0.899 0.915 ...
 $ GFP          : num  124002 127999 44070 4280 3928 ...
 $ sample_names : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ replication  : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ drugname     : Factor w/ 1 level "phenol": 1 1 1 1 1 1 1 1 1 1 ...
 $ concentration: Factor w/ 11 levels "0","0.05","0.1",..: 1 2 4 6 8 10 1 2 3 3 ...
```

barplot - ggplot

```
library(ggplot2)

ggplot(data=mydata2, aes(x=sample_names, y=GFP, fill=concentration)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  scale_fill_manual(values = heat.colors(11))
```



Introducing dplyr

Hadley Wickham

2014-01-17

Categories: Packages

`dplyr` is a new package which provides a set of tools for efficiently manipulating datasets in R. `dplyr` is the next iteration of `plyr`, focussing on only data frames. `dplyr` is faster, has a more consistent API and should be easier to use. There are three key ideas that underlie `dplyr`:

1. Your time is important, so Romain Francois has written the key pieces in Rcpp to provide blazing fast performance. Performance will only get better over time, especially once we figure out the best way to make the most of multiple processors.

2. Tabular data is tabular data regardless of where it lives, so you should use the same functions to work with it. With `dplyr`, anything you can do to a local data frame you can also do to a remote database table. PostgreSQL, MySQL, SQLite and Google bigquery support is built-in; adding a new backend is a matter of implementing a handful of S3 methods.

3. The bottleneck in most data analyses is the time it takes for you to figure out what to do with your data, and dplyr makes this easier by having individual functions that correspond to the most common operations (`group_by`, `summarise`, `mutate`, `filter`, `select` and `arrange`). Each function does one only thing, but does it well.

plyr.had.co.nz

plyr

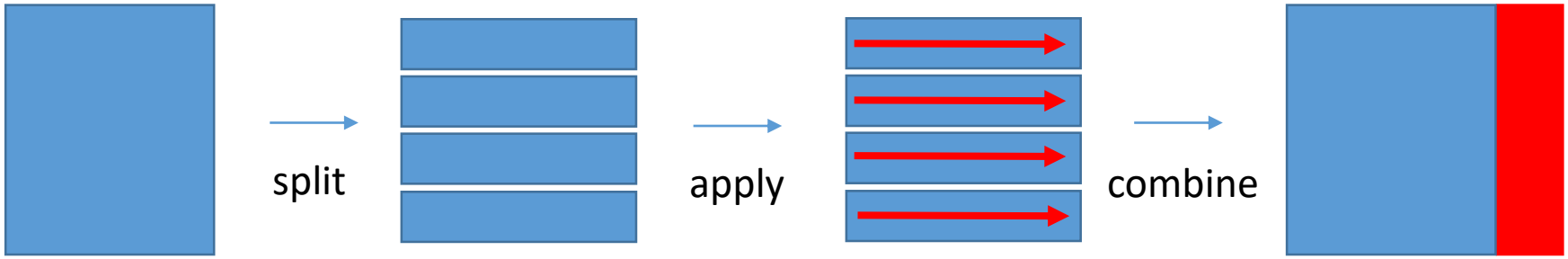
The split-apply-combine strategy for R

plyr is a set of tools for a common set of problems: you need to **split** up a big data structure into homogeneous pieces, **apply** a function to each piece and then **combine** all the results back together. For example, you might want to:

- fit the same model to subsets of a data frame
- quickly calculate summary statistics for each group

News

- Plyr 1.7
- Plyr 1.6
- Plyr 1.5



dplyr

- **The Pipe Operator: %>% (similar with "+" in ggplot)**
- **%>% takes the output of its lhs statement and makes it the input of the rhs (next) statement**

f(x) == x %>% f

- **Short cut in Rstudio: Shift + Ctl + m (Alt+_ for <-)**
- **Placeholder . operator**

```
head(iris)
str(iris)
iris %>% head(10)
iris %>% str

round(pi, digits=6)
6 %>% round(pi, digits=.)
```

exercise 7-2) dplyr

- Change the code by using %>% and . operators

```
1. head(iris)

2. str(iris)

3. x <- 1:100
   mean(x)

4. df <- data.frame(x=c(1:100), y=c(201:300))
   colMeans(df)

5. x <- 1:5
   paste("1", letters[x], sep="")
```

iris dataset

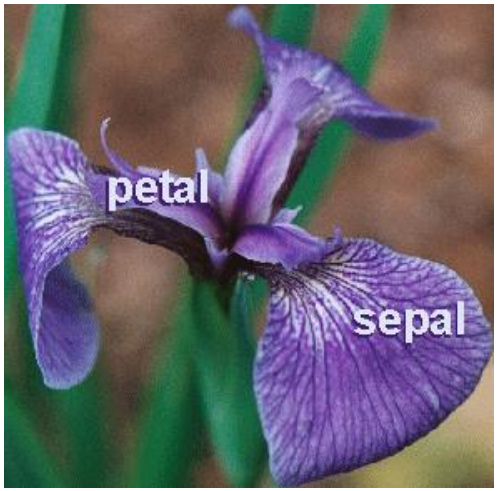
Iris Versicolor

Iris Setosa

Iris Virginica

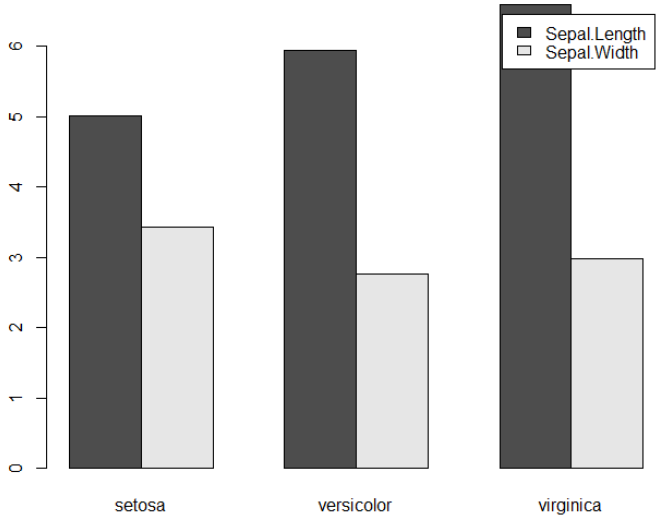
Fisher's/Anderson's iris data set:
measurements (cm) of the sepal length and width and petal length and width (4 features) for 50 flowers from each of 3 species (*Iris setosa*, *versicolor*, and *virginica*)

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```



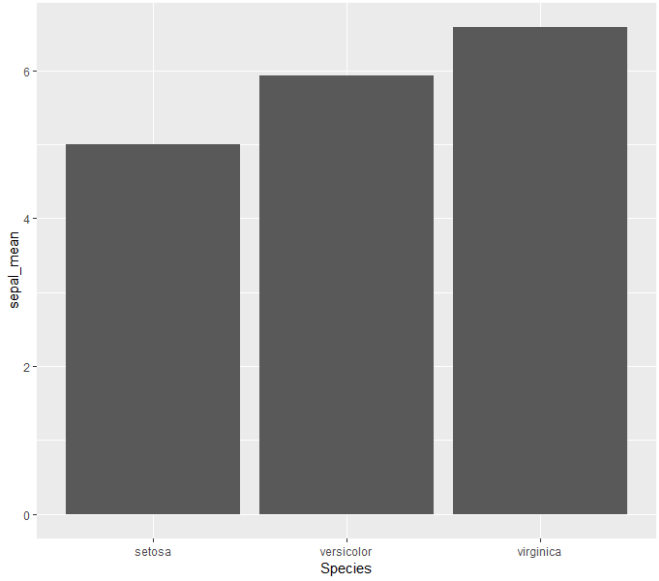
Mean comparison among groups

```
iris_split <- split(iris, iris$Species)
iris_means <- lapply(iris_split, function(x){colMeans(x[,1:4])})
iris_means_df <- data.frame(iris_means)

barplot(iris_means_df)
iris_means_df_sepal <- as.matrix(iris_means_df[1:2,])
barplot(iris_means_df_sepal)
barplot(iris_means_df_sepal, beside = T)
?barplot
barplot(iris_means_df_sepal, beside = T,
legend.text=rownames(iris_means_df_sepal))
```



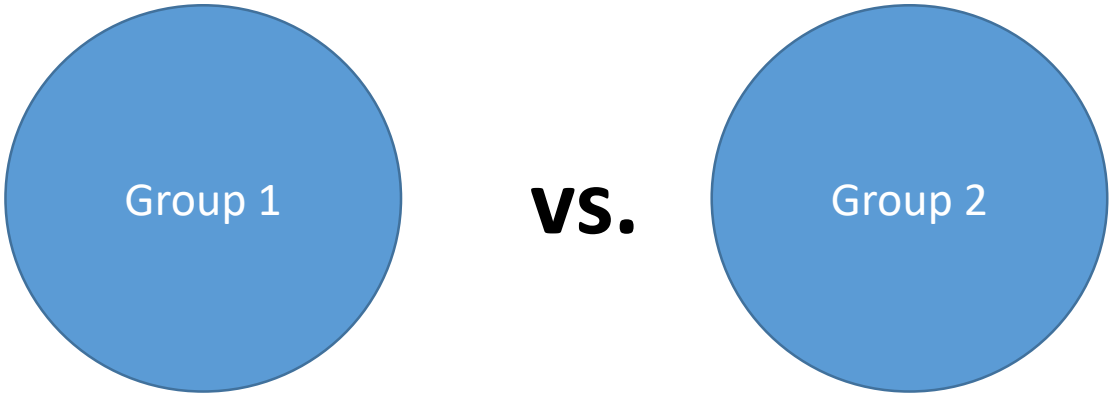
iris dataset with dplyr and ggplot

```
iris_sepal <- iris %>%
  group_by(Species) %>%
  summarise(sepal_mean=mean(Sepal.Length), sepal_width=mean(Sepal.Width))

ggplot(iris_sepal, aes(x=Species, y=sepal_mean)) +
  geom_bar(stat="identity")


iris %>%
  group_by(Species) %>%
  summarise(sepal_mean=mean(Sepal.Length), sepal_width=mean(Sepal.Width)) %>%
  ggplot(aes(x=Species, y=sepal_mean)) +
  geom_bar(stat="identity")
```



Quiz 6-1) Data structure

- Which object or data structure will you use to compare two groups of datasets?
- How many variables do we need for the comparison in this example?



Data structure for data analysis

x	y
1	1
2	6
5	7
7	8

=

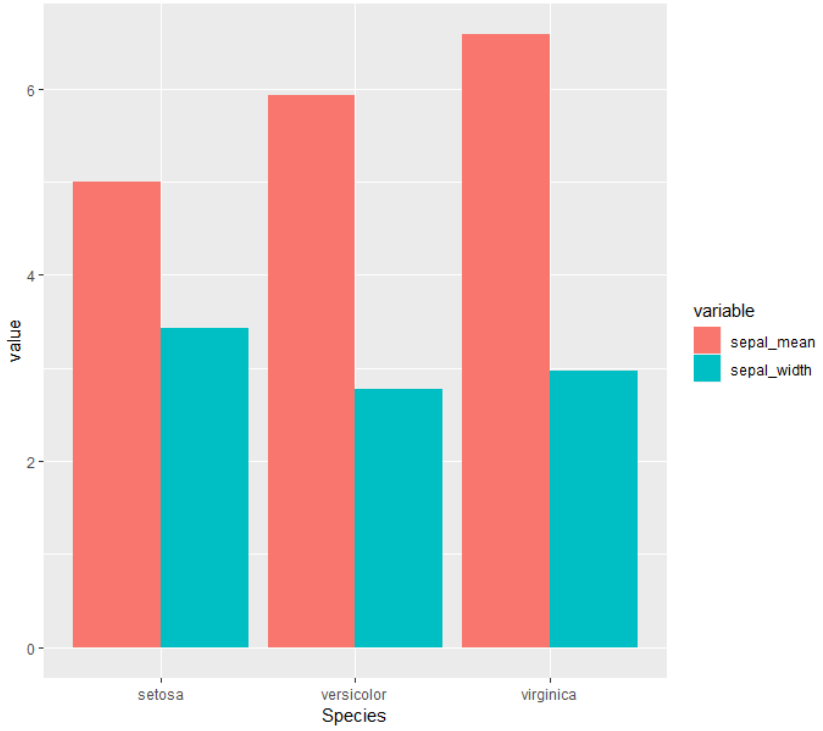
variable	value
x	1
x	2
x	5
x	7
y	1
y	6
y	7
y	8

Melt data

```
df <- data.frame(x=c(1:10), y=c(201:210))   colMeans(df)

x <- c(1:10)
y <- c(201:210)
value <- c(x, y)
variable <- c(rep("x", length(x)), rep("y", length(y)))
df <- data.frame(value, variable)


df %>% group_by(variable) %>% summarise(m=mean(value))
```

iris dataset with dplyr and ggplot

```
iris %>%
  group_by(Species) %>%
  summarise(sepal_mean=mean(Sepal.Length), sepal_width=mean(Sepal.Width)) %>%
  melt %>%
  ggplot(aes(x=Species, y=value, fill=variable)) +
  geom_bar(stat="identity", position="dodge")
```



Next

- R visualization III
  - ggplot2
- Data manipulation II