R 공부 시작

- 내 [reference card](#reference)

- r reference card : <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>

- r [data visualization](#datavisualization)

Cheat sheet : <https://www.rstudio.com/wp-content/uploads/2015/04/ggplot2-cheatsheet.pdf>

- Minerva R statistical modeling 진도표

|  |
| --- |
| - [Basic R](#minerva) usage |

- Statistical analysis of network data with R

|  |
| --- |
| - [Start Here](#network)  - Basic codes to start visualize network data |

New Skill

|  |
| --- |
| # igraph에서 graph 그릴 때, partitioning하는 방법!  par(mfrow = c(2,2) # 2x2로 나누는 방법  par(mfrow = c(1,2) # 1x2로 나누는 방법  sort(unique(V(fblog)$PolParty)) ## sort & unique  numeric(100)으로 하면  100개 항의 vector가 만들어지는 것 같다.  - dpylr :  x vecter, y vector 중복 제거하고 하나의 column만들기  nodes <- full\_join(x, y, by=’column name’)  - tibble 패키지의 add\_column( ) |

- Rstudio 설치 : <https://www.rstudio.com/>

- setting : tools-global options-appearance에서 화면을 검은색, 혹은 navy색으로 하면 눈의 피로를 덜어준다.

- 새로운 script를 만드려면 : R script를 클릭해야 한다. 종이 버튼. ctrl+shift+n

- 새로운 프로그램 :

“Hello World!”

하고 save 버튼을 눌러서 이름을 정하면 끝!

- 구동하려면 Run 버튼을 눌러야 한다.(Ctrl+Enter)

- R studio의 장점 :

- console에서 작업을 할 수 있으나, 저장을 할 수는 없다.

- console에서 작업을 하면 어떻게 작동하는지 알 수 있기 때문에 매우 편하다!!

- Rstudio 단축키:

<https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts>

전체 실행

control + alt + R

shift + tab 하면

matrix(data, nrow = rows, ncol = cols)

안의 data 뭘 넣어야 하는지 나온다.

- 마우스로 함수 위에 얹어놓으면 설명이 나온다. ㅋㅋ

- 모든 수업 자료 : <https://www.superdatascience.com/rcourse/>

- Help를 사용하자!

?vector

help(‘vector’)

help.search(‘vector’)

**0. 첫수업** - motivation

- R을 통해서 무엇을 할 수 있는가 :

- R 코딩

|  |
| --- |
| - csv file 로딩  mydata <- read.csv(file.choose())  - ggplot2 다운로드  install.packages(“ggplot2”)  🡪 package tab에서 ggplot2에 체크를 하면 사용할 수 있다.  - 표를 그리자  ggplot(data=mydata, aes(x=carat, y=price)) +  geom\_point()    ggplot(data=mydata,  aes(x=carat, y=price, colour=clarity)) +  geom\_point()    🡪 clarity가 있고 범주도 있고.  ggplot(data=mydata[mydata$carat<2.5,],  aes(x=carat, y=price, colour=clarity)) +  geom\_point(alpha=0.1) # 투명도 0.1  🡪 2.5 오른쪽으로 제거하면 다음과 같이 그려진다.    ggplot(data=mydata[mydata$carat<2.5,],  aes(x=carat, y=price, colour=clarity)) +  geom\_point(alpha=0.1) +  geom\_smooth()  🡪 smooth까지 하면 평균이 나온다. |

- R의 기초

**0. Data type**

|  |
| --- |
| is. # Checking datatype  as. # convert |

typeof(x) # type를 알 수 있다.

line 단위로 적용하기 때문에 control + enter해야 한다.

마우스로 끌어서 영역을 지정하면 동시에 execute가능(control + enter)

- 또는 class( )로 하면 알 수 있다. Class는 크게 character, numeric, logical이 있다.

- as.numeric

**1. variable**

numeric에는 integer와 double이 있다

**(1) integer**

x <- 2L # integer로 할 때는 **L**로 표시. (차이점)

**(2) double**

y <- 2.5 # 표시를 안하면 디폴트는 double. 연산을 할 것으로 가정하고 double로 놓은 것이다.

z <- 2 # 소수점이 없어도 double

**(3) complex**

z <- 3 + 2i

**(4) logical**

q1 <- T

q2 <- FALSE # 모두 다 대문자여야 한다.

**(5) Random variable**

- random하게 번호 뽑기

rnorm(1) 🡪 random하게 1개를 뽑는다는 거다. 평균 0, std = 1가 디폴트

- uniform distribution으로 뽑기 : **runif**

matrix(runif(20, 0, 100), nrow=4)

🡨 20개 뽑고, min = 0, max= 100

**2. 연산**

**(1) C <- A + Bva**

- print하려면

C

**(2) 나누기**

var1 <- 2.5

var2 <- 4

result <- var1/var2

**(2-2) 나머지 (modulo)**

%%

x %% 2 ==0

**(3) 제곱근**

answer <- sqrt(var2)

**(4) character variable과 묶는법**

greeting <- "Hello"

name <- "Bob"

message <- **paste**(greeting, name)

message

대박!!!

paste 사이의 space를 없애려면

paste(greeting, name, sep = “”)

하면 된다.

**(5) logical operators (comparison operators)**

4<5

4 == 5

4 != 5

4 <= 5

space를 조심해라!

v <- 1 (1로 assign하라)

v < -2 (-2보다 작다)

- Vector를 사용하면 각 원소 별로 비교해준다.

v <- c(1,2,3,4,5)

> v <2

[1] TRUE FALSE FALSE FALSE FALSE

- 벡터랑 벡터를 비교할 수도 있다!!

> v <- c(1,2,3,4,5)

> v2 <- c(10, 11, 12, 13, 14)

> v < v2

[1] TRUE TRUE TRUE TRUE TRUE

- logical을 변수로 저장할 수 있다.

result <- v < 5

typeof(result)

>> “logical”

- logical

tf <- c(TRUE,FALSE)

tt <- c(TRUE,TRUE)

ft <- c(FALSE, TRUE)

tt & tf

TRUE FALSE

첫번째 항만 비교하는 거면 && ㅣㅣ

ft && tt

FALSE

**(6) not operator**

result2 <- !TRUE

result2

**(7) OR : | (shift \)**

result | result2

>> TRUE (TRUE or FALSE니까)

**(8) AND : &**

result & result2

>> FALSE

**(9) isTRUE(result)**

>> TRUE

**3. 제어구조**

**(1) while loop** : { } 라는 점에서 python이랑 좀 다르다. :도 없다.

- 형태 :

while(TRUE){

print("Hello")

}

- counter 활용

counter <- 0

while(counter<12){

print(counter)

counter <- counter+1

}

**(2) For loop**

- 형태

for(i **in** 1:5){ # vector형태다!!!

print("Hello R")

}

🡪 1:5 는 vector이다. 1 2 3 4 5

x <- rnorm(5)

x

for(i in x){

print(i)

}

**(3) if**

- random하게 번호 뽑기

rnorm(1) 🡪 random하게 1개를 뽑는다는 거다. 평균 0, std = 1가 디폴트

x <- rnorm(1)

if(x > 1){

answer <- "Greater than 1"

}

- 변수를 assign을 없애는 방법

rm(answer)

- Nested if-else statement : if else 안에 또 if else

🡪 elif 없나?

|  |
| --- |
| x <- rnorm(1)  rm(answer)  if(x > 1){  answer <- "Greater than 1"  } else{    if(x>= -1){  answer <- "Between -1 and 1"  } else{  answer <- "Less than 1"  } |

- chaining statement : else if

|  |
| --- |
| x <- rnorm(1)  **rm**(answer)  if(x > 1){  answer <- "Greater than 1"  } else if(x>= -1){  answer <- "Between -1 and 1"  } else{  answer <- "Less than 1"  } |

**(4) Law of Large numbers (Homework)**

|  |
| --- |
| myfunction <- function(n){  #n <- 100  counter <- 0  x=rnorm(n)  # print (x)  for(i in x){  if(-1 < i & i < 1){ # -1<i <1이 안된다!  counter <- counter +1  }  }  return (counter/n)  }  myfunction(100)  myfunction(200)  myfunction(300) |

🡪 중요! : -1 < i < 1이라는 건 없다!

-1 < i & x < 1이어야 한다는 것!

**4. Vector**

- 중요 사실 :

(1) R에서 모든 것은 vector다.

(2) 중요한 것은 한 벡터에는 한 종류의 type만 들어간다. character vector, number vector.

🡪 character랑 double이 만나면 character가 이긴다. 다 character로 바뀜.

MyVector <- c(12, 456, 34.5, 23, 55, "34hello")

typeof(MyVector) 🡪 character

(3) 한 개의 숫자도 vector로 저장된다!!!! 대박!!

**4.1. vector만드는 방법 : c !!!!**

MyFirstVector <- c(3, 45, 56, 732)

|  |
| --- |
| x <- **c**(1, 123, 5, 3, 4) #combine  y <- **seq**(203, 250, 11) #sequence  z <- **rep**("hi!", 3) #replicate  sample(1:10) 🡺 sample을 만들어준다. random하게 1-10까지 랜덤 배열  **append**(v1, v2) |

**4.2. Vector안의 항이 뭔지 확인해보자.**

- is.numeric( )

- is.integer( )

is.numeric(MyFirstVector)

>> TRUE

is.integer(MyFirstVector)

>> FALSE 🡪 그 이유는 디폴트로 double이기 때문이다. 숫자 뒤에 L을 적으면 맞다.

is.double(MyFirstVector)

>> TRUE

V3 <- c('a', 'b23', 'hello')

is.character(V3)

>> TRUE

V3 <- c('a', 'b23', 'hello', 7)

>> 저절로 7이 “7”로 바뀐다.

is.character(V3)

>> TRUE

**4.3. 규칙있는 벡터 만들기**

|  |
| --- |
| x <- c(1, 123, 5, 3, 4) #combine  append(v, v2) # c 로도 할 수 있다!  y <- seq(203, 250, 11) #sequence  z <- rep("hi!", 3) #replicate  **sort** (v) # sort  **rev** (v) # reverse  sample(vector, n)  **lapply** (vector, function) |

# vector 안에 NA를 넣으면 아무것도 아닌 게 된다.

**(1) seq**

seq( ) #sequence - like ':'

seq(1,15)

>> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

🡪 1:15랑 같다.

seq(1, 15, 3) 🡺 seq를 사용하면 step까지 할 수 있다.

- 만약 1, 15 사이의 숫자가 아니면 안 나온다.

- lengthout

> seq(from = 10, to =20, length.out = 100)

[1] 10.00000 10.10101 10.20202 10.30303 10.40404 10.50505 10.60606 10.70707 10.80808 10.90909 11.01010 11.11111 11.21212

[92] 19.19192 19.29293 19.39394 19.49495 19.59596 19.69697 19.79798 19.89899 20.00000

🡪 10부터 20까지 100개가 되도록 한다.

- alongwith

> x <- c("a","b","c")

> x

[1] "a" "b" "c"

> seq(10, 20, along.with = x)

[1] 10 15 20

length를 직접 말하는 게 아니라, example vector를 넣으면 된다.

**(2) rep**

- rep(3, 100) 🡺 3가 100개 나온다.

[1] 3 3 3 3 3 3 3 3 3 3 3 3

[13] 3 3 3 3 3 3 3 3 3 3 3 3

[25] 3 3 3 3 3 3 3 3 3 3 3 3

[37] 3 3 3 3 3 3 3 3 3 3 3 3

[49] 3 3 3 3 3 3 3 3 3 3 3 3

[61] 3 3 3 3 3 3 3 3 3 3 3 3

[73] 3 3 3 3 3 3 3 3 3 3 3 3

[85] 3 3 3 3 3 3 3 3 3 3 3 3

[97] 3 3 3 3

🡪 [ ] 속에 있는 것은 index이다.

- character vector도 만들 수 있다.

rep("a", 5)

[1] "a" "a" "a" "a" "a"

- 벡터를 반복시킬 수도 있다.

x <- c(80, 20)

rep(x, 10)

>>

[1] 80 20 80 20 80 20 80 20

[9] 80 20 80 20 80 20 80 20

[17] 80 20 80 20

- each

> rep(5:6, **times** =10)

5, 6, 5, 6을 계속!

> rep(5:6, **each**=10)

5, 5, .. 5, 6, 6, ... 6

**(3) sort (v)**

sort (v, decreasing = TRUE)

> cv <- c('b','d','a')

> sort(cv)

[1] "a" "b" "d"

- capital도 소문자랑 같은 취급해준다.

sort(unique(V(fblog)$PolParty)) ## unique도 사용할 수 있다.

**(4) rev( v )**

순서를 반대로!

**(5) append( v1, v2 )**

🡪 벡터도 할 수 있고, character(1개 짜리 벡터)도 할 수 있다! List, data frame도 가능하다.

c 랑 똑같다. 다만

append(x, values, **after = length(x)**) # 특정 index 뒤로 오게 할 수 있다.

**(6) sample**

> sample(1:100, 1)

[1] 65

> sample(1:100, 3)

[1] 56 39 6

addrand <- function(x){

ran <- sample(1:100,1)

return(x + ran)

}

**(7) lapply** # list apply

lapply(v, addrand) # go to every single element and add random number

- 주의 : addrand( ) 로 하는게 아니다.

[[1]]

[1] 9

[[2]]

[1] 83

[[3]]

[1] 92

[[4]]

[1] 7

[[5]]

[1] 34

- sapply !! # simple 🡪 이게 더 간단하다.

> sapply(v, addrand)

[1] 97 84 92 67 65

- 만약에 argument가 여러 개이면 지정해줘야한다.

add\_choice <- function(num, choice){

return(num + choice)

}

sapply(v, add\_choice, choice = 100)

ex)

k.nbhds <- graph.neighborhood(karate, order=1)

sapply(k.nbhds, vcount)

**4.4. Vector Indexing / referencing**

**4.4.1. 1개 index**

python이랑 다르게 1부터 시작이다.

w <- c("a", "b", "c", "d", "e")

w[1]

w[2]

w[3]

>> “c”

**4.4.2. 원하지 않는 것 빼기**

- 원하지 않는 것만 빼서 벡터로 만들 수도 있다!

w[-1]

>> [1] "b" "c" "d" "e"

w[-3]

>> [1] "a" "b" "d" "e"

**4.4.3. slicing (원하는 조각 가져오기)**

🡪 벡터를 안에 넣으면 원하는 것만 잡아올 수 있다.

v[ c(1, 3) ]

> w[c(1,3,5)]

[1] "a" "c" "e"

> w[c(-2,-4)]

[1] "a" "c" "e"

**TIP : 순서도 자기 맘대로 가져올 수 있다.**

> v <- c(1, 2, 3)

> names(v) <- c('a','b','c')

> v [c('c','b','a')]

c b a

3 2 1

> w[1:3]

[1] "a" "b" "c"

> w[3:5]

[1] "c" "d" "e"

> w[-3:-5]

[1] "a" "b"

**4.4.4. 원하는 조건을 자동화하기 (Using logical condition statements)**

**🡺 index자리에 Boolean value를 vector형태로 넣을 수 있다는 게 제일 중요하다. (특히 조건을 바로 넣을 수 있다는 것!!) 🡪 data frame 배울 때 매우 중요하다**

- logical masking(filter) 사용:

> v[v>2]

c d

3 4

🡪 v > 2를 하면

a b c d

FALSE FALSE TRUE TRUE

이렇게 나온다. 그 중에서 TRUE가 되는 값을 뽑아주는 것이다.

🡪 심화 : 부등호 식 자체를 filter라는 변수 안에 넣어놓을 수도 있다. Logical comparison을

> filter <- v>2

> filter

a b c d

FALSE FALSE TRUE TRUE

> v[filter]

c d

3 4

대박!! 이렇게 하면 index도 named로 뽑을 수 있다.

|  |
| --- |
| 벡터 단위로 생각하자!!!  TRUE FALSE의 벡터를 만들자!!!!!!!  max.price <- stock.prices == max(stock.prices)  max.price  stock.prices[max.price]  아니면  stock.prices [ stock.prices == max(stock.prices) ] |

lygometry : studying what you don’t know!

- index를 찾으려면 (output이 index)

which(x == vector)

match(c(4,8), vector)

**5. Vectorized operations**

**5.1. 원리와 중요성**

- 벡터간의 덧셈

: 만약 항 수가 같으면 + 연산 하나로 각각 항이 더해진다.

- Recycling of vectors : 그러나 만약에 항 3개짜리 vector랑 10개짜리 vector를 더하게 되면 항 3개 짜리가 replication 되어서 3개 + 3개 + 3개 + 1개로 10개 짜리가 되어서 더해진다. 만약에 6개짜리 + 10개짜리이면 6개 + 4개만 반복된 것이 10개짜리에 더해진다. 그리고 warning을 한다.

v <- c(1,3,5,6,7)

w <- c(1,2,3,4,5,6,7,8,9,0)

v+w

>> [1] 2 5 8 10 12 7 10 13 15 7

- function : vector가 input과 output이 되는 게 굉장히 powerful하다.

- Vectorized operation이 빠른 이유 :

# Vectorized approach

c <- a \* b

c

🡪 벡타 안의 variable이 모두 같은 type라는 것을 이미 알고 있기 때문에 그것을 확인하는 시간이 save되기 때문이 더 빠르다. C에게 delegation을 하기 때문이다.

# devectorized approach

d <- rep(NA, N)

for(i in 1:N){

d[i] <- a[i] \* b[i]

}

🡪 이게 훨씬 오래 걸린다.

**5.2. Function 🡺 믹서기이다 ㅋㅋㅋ**

**5.2.1.** **My reference card**

- 지금까지 배운 모든 function (Arsenal)

- r reference card라고 치면 다 나온다!

- documentation을 알고 싶으면 ?rnorm

|  |
| --- |
| **<rstudio 단축키>**  control + enter  ctrl + alt + r # 전체 실행  ctrl + shift + n # 새 스크립트  alt - <-  **<Logical operator>**  result2 <- !TRUE  result & result2  result | result2  isTRUE(result)  any(a %% 2:(a-1) == 0) # this is matrix inside any( )  **<print하는 방법>**  print(paste0("Current working dir: ", wd))  cat("Current working dir: ", wd)  sprintf("Current working dir: %s", wd)  🡪 sprintf("The best month was %s, reaching a profit margin of %s", a, max\_value)  **<Character 가지고 놀기>**  paste( )  **<숫자 가지고 놀기>**  - 벡터 통계량  sum(v1)  prod(v1) # 전체곱  sd(v)  var(v)  abs(v) # 절대값  max(vector)  min(vector)  mean <- mean(profit\_margin)  round(123.456,digits=2)  [1] 123.46  df$performance <- round(df$performance,2)  rnorm( )  **sqrt( )**  행렬도 된다는 거  **<class, type 확인하기/변경>**  typeof( ) = class( )  is.numeric( )  is.integer( )  is.double( )  is.character( )  is.vector()  is.matrix()  is.dataframe( )  as.list(v)  as.matrix(v)  **<Function 가지고 놀기>**  **lapply** # list apply  lapply(v, addrand) # go to every single element and add random number  - 주의 : addrand( ) 로 하는게 아니다.  - sapply !! # simple 🡪 이게 더 간단하다.  > sapply(v, addrand)  [1] 97 84 92 67 65  **<벡터 가지고 놀기>**  cv <- vector( ) # 빈 vector 만들기  x <- c(1, 123, 5, 3, 4) #combine  y <- seq(203, 250, 11) #sequence  🡪 seq(from = 10, to =20, length.out = 100)  🡪 seq(10, 20, along.with = x)  z <- rep("hi!", 3) #replicate  > rep(5:6, times =10)  5, 6, 5, 6을 계속!  > rep(5:6, each=10)  5, 5, .. 5, 6, 6, ... 6  sort( )  rev (v) # reverse  sample(1:10) 🡺 sample을 만들어준다. random하게 1-10까지 랜덤 배열  **append**(vector, append할 거)  %in% # 벡터 안에 특정한 수가 있는가  **rnorm(n)**  - default를 바꾸고 싶으면  rnorm(5, mean =2, sd =10)  🡪 sequence가 필요없다.  rnorm(5, sd =10, mean =2)  rnorm(5, 2, 10)  uniform distribution으로 뽑기 : **runif**  matrix(runif(20, 0, 100), nrow=4)  🡨 20개 뽑고, min = 0, max= 100  w[-1]  >> [1] "b" "c" "d" "e"  names(v) <- c('a','b','c')  > v[v>2]  c d  3 4  max.price <- stock.prices == max(stock.prices)  stock.prices[max.price]  stock.prices [ stock.prices == max(stock.prices) ]  **<Matrix가지고 놀기>**  matrix(vector, nrow = ,ncol = ,byrow = T/F )  🡪 뱀처럼 1열 위에서 아래로, 2열 위에서 아래로, 3열 위에서 아래로 순서대로  **rbind**(row vector1, vector2, vector3 )  **cbind**(column1, column vector2, column vector3)  colnames(Salary) <- Seasons  rownames(Salary) <- Players  factor.ani <- **factor**(animal)  fact.temp <- factor ( temps, ordered = True, levels = c(‘cold’, ‘med’, ‘hot’)  mat2 **%\*%** mat2  **colSums**(stock.matrix)  **rowSums**(stock.matrix)  **rowMeans**(stock.matrix)  **t**(x) # transpose  **diag**(x) # diagonal  - Ax=b solve A %\*% x = b  **solve**(a, b)  **solve**(a) # inverse  A[1,]  mat [1:3, ]  x[c(1,5)]  Games[1, , **drop=F**]  **<Data frame가지고 놀기>**  (1) 빈 data frame 만들기 : data.frame( )  > empty <- **data.frame( )**  (2-1) 벡터들을 붙여서 만들기  > days <- c('Mon', 'Tue', 'Wed', 'Thu', 'Fri')  > temp <- c(22, 23, 24, 25, 26)  > rain <- c(T, T, F, F, T)  > df <- **data.frame**(days, temp, rain)  (2-2) Matrix도 바꿀 수 있다.  new\_df <- data.frame(mat)  df1=**as.data.frame**((cbind(x,y)))  (2-3) dataframe 두개를 merge할 수도 있다.(NEW)  merge  merge(x, y, by = intersect(names(x), names(y)),  by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,  sort = TRUE, suffixes = c(".x",".y"),  incomparables = NULL, ...)  total <- merge(batting, salary, c('yearID', 'playerID'))  (3) naming  c1 <- 1:10  c2 <- letters[1:10] # letter는 a~z까지 벡터  df <- data.frame(col.name.1 = c1, col.name.2 = c2)  # 앞에 다가 쓰는게 중요하다. 안 쓰면 c1, c2가 column name이 된다.  - 특정 열 이름 바꾸기  colnames(dataframe)[which(names(dataframe) == "columnName")] <- "newColumnName"  (4) Adding new row  dfnew <- rbind (df, df2)  df2 <- data.frame(col.name.1 = 2000, col.name.2 = 'new')  (5) Adding new columns  df$newcol.copy <- df$newcol  = df[ , ‘newcol.copy2 ] <- df$newcol  df$performance <- round(df$performance,2)  (6) Setting new column names  colnames(df) <- c('1','2','3','4')  colnames(df)[1] <- 'New col name'  - 관찰  head(df, 몇번째 줄) # default is 6번째  tail(df)  str(df)  summary(df)  nrow(df)  ncol(df)  **colnames**(df)  **rownames**(df)  (1) Indexing / referencing  df [ 1 ] 하면 그냥 1번 column이 나온다!!! 신기!!!  df [[5,'col.name.2']] # single cell  df[1,]  df[,1]  df[1:5,c('days','temp')]  **df[ ,'rain'] = df$rain** --> column을 vector 형태로  df['rain'] 🡪 data frame의 구조 유지  df <- data.frame(days,temp,rain)  df[1,] 🡪 row 뽑으면 data frame 유지  🡪 vector를 가져오고 싶다면?  vrow <- **as.numeric**(**as.vector**(df[1,])) # as.numeric을 해야 하는지 의문이다.  df[,1] 🡪 column뽑을 때는 vector로  df[1:5,c('days','temp')]  df[,'rain'] 🡪 df$rain이 더 편하다.  [1] TRUE TRUE FALSE FALSE TRUE  🡪 vector 형태로 가져오게 된다.  **df['rain'] 🡪 이렇게 하면 좀 다름 (data frame의 구조 유지)**  - referencing column : vector로 뽑는 4가지 방법이 있다. 2가지만 기억하기.  **colv1 <- cars$mpg**  **colv2<- cars[ , 'mpg']**  colv3 <-cars[, 1] # uncommon  colv4 <-cars[['mpg']] # uncommon  - column을 data frame으로 뽑는 1가지 방법  cars [‘mpg’] # single set of brackets!!  cars [1]  - 1개의 row만 빼면 벡터가 된다.  Games[1,]  > is.vector(Games[1,])  [1] TRUE  > is.matrix(Games[1,])  [1] FALSE  - 이 현상이 발생하지 않도록 하려면 : drop = F  왼쪽의 row의 name을 drop하지 않도록 해야 한다.  Games[1, , **drop=F**]  - multiple columns 가져오려면?  df[1:3,] # 3개의 row 가져올 수 있다.  df[1:10, ] # 10개의 row 가져올 수 있다.  head(df, 7) # 처음 7개의 row를 가져올 수 있다.  df[-2, ] # everything but second row  mtcars[ , c(1, 2, 3) ]  or mtcars[ , c(‘mpg’, ‘cyl’) ] # vector 형태로  > mtcars [c('mpg', 'cyl')] # data frame을 살리기  앞의 6줄만 : head (mtcars [c('mpg', 'cyl')] )  # Bracket형 1가지 조건  $를 안쓰면 그냥 다 가져온다. ㅋㅋ  🡪 row를 찾는 것이기 때문에 [ , ] 앞에 쓰는 것임  mtcars [ mtcars$mpg > 20 , ]  mtcars [mtcars$mpg >20, 'cyl']  mtcars [mtcars$mpg >20, 2:3]  mtcars[ (mtcars$mpg > 20) & (mtcars$cyl ==6) ,]  # 두가지 조건  mtcars[ mtcars$mpg > 20 & mtcars$cyl ==6 , c('mpg','cyl','hp')]  mean(df[(df$hp>100) & (df$wt >2.5),]$mpg)  # subset  > subset(df, subset = rain == TRUE) # ‘ ‘ 안 해도 된다.  > subset(df, subset = temp > 24)  **subset**(mtcars, mpg > 20 & cyl ==6)  # is.na로 missing data 처리  is.na(mtcars) # output은 data frame  any ( is. na(df) ) # TRUE가 1개는 있는지 : 있으면 TRUE  df [ is.na( df ) ] <- 0  mtcars$mpg [ is.na(mtcars$mpg) ] <- 0  mtcars$mpg [ is.na(mtcars$mpg) ] <- mean(mtcars$mpg)  df <- df[!is.na(df$col), ]  sorted.temp <- **order**(df$temp)  - 순서 거꾸로하려면  > desc.temp <- order( **-**df [‘temp] )  - %in%로 특정 복수 조건의 애들 불러오기(NEW)  subset(total, playerID %in% c('giambja01','damonjo01','saenzol01'))  - dplyr로 더 편하게 하기!!  install.packages('dplyr')  library('dplyr')  - 추가(18. 5. 18)  > full\_join(x column, y column, by=’column name’) ## 중복제거하고 한줄 만들기  > left\_join(dataframe, join할 것) ## 왼쪽에 있는 dataframe을 기준으로 해서 거기에 없는 것만 더 추가하는 것.  ex)  edges <- per\_route %>% left\_join(nodes, by=c('source'='label')) %>% rename()  ## To join by different variables on x and y use a named vector. For example, by = c("a" = "b") will match x.a to y.b.  🡺 per\_route랑 nodes를 합치는데 합치는 기준이 source = label인 것이다.  > add\_column(data.frame, new.column)  다음에 %>%로 select(new.column, everything()) 하면 그 줄에 뒤에 추가되었더라도 첫째줄로 앞으로 오게 된다!!  > %>% group\_by(source, destination)  ## takes an existing tbl and converts it into a grouped tbl where operations are performed "by group". ungroup() removes grouping.  🡪 끝나고 ungroup을 꼭 해줄 것  ex) per\_route <- letters %>% group\_by(source, destination) %>%  summarise(weight = n()) %>% ungroup()  ## n()의 의미 : The number of observations in the current group.  ex)  if (require("nycflights13")) {  carriers <- group\_by(flights, carrier)  summarise(carriers, n())  mutate(carriers, n = n())  filter(carriers, n() < 100)  }  항상 data frame을 먼저 하고 쉼표로 뒤에!!!!  **filter**( ) # subset과 유사  > filter(flights, month ==11, day ==3, carrier=='AA')  **arrange**( ) # filter랑 유사한데, order를 추가적으로!  > arrange(flights, year, month, day, arr\_time)  **slice**( ) # 1~10번째 row까지  > slice(flights, 1:10)  **select**( ) # 원하는 열만 select할 수 있다  > select(flights, carrier, arr\_time, month)  > **distinct**(select (flights, carrier) )  **rename**( ) # 열 명칭 바꾸기  > rename(flights, airline\_carrier = carrier)  **mutate**( ) # 새로운 열을 만들기  > mutate(flights, new\_col = arr\_time - dep\_delay)  transmute( ) # 새로운 열만 가져오기  > transmute(flights, new\_col = arr\_time - dep\_delay)  **summarise**( )  > summarise(flights, avg\_airtime = mean(air\_time, na.rm=TRUE))  **sample\_n**( ) and **sample\_frac**( )  > sample\_n(fligths, 10)  sample\_frac(flights, 0.1)  - pipe operator  df %>% filter(mpg>20) %>% sample\_n(size=5) %>% arrange(desc(mpg))  - tidyr  install.packages('tidyr')  install.packages('data.table')  library(tidyr)  library(data.table)  **gather**()  > gather(df, Quarter, Revenue, Qtr1:Qtr4)  > df %>% gather(Quarter, Revenue, Qtr1:Qtr4)  > stocks.gathered <- stocks %>% gather(stock,price, X:Z)  > stocks.gathered <- stocks %>% gather(stock,price, X,Y,Z)  **spread**()  > stocks.gathered %>% spread(stock,price)  **separate**() ## 와!!!! split이랑 같네 ㅎㅎ  > separate(df, new.col, c('ABC', 'XYZ'))  > separate(data = df, col = new.col, into = c('ABC', 'XYZ'), sep=’-‘)  **unite**()  > unite(df.sep, col = joined, sep = ‘---‘ )  이렇게 하면 된다.  **<List 가지고 놀기>**  # 만들기  my.list <- list(v,m,df)  # naming  my.named.list <- list(sample.vec = v, my.matrix = m, sample.df = df)  # 데이터 꺼내오기  my.named.list[['sample.vec'**]]**  my.named.list$sample.vec  # nested list  double.list <- c(my.named.list, my.named.list)  # structure  str(double.list)  **<Import, export files>**  install.packages('readxl')  library(readxl)  setwd("c:/Users/Owner/Desktop")  excel\_sheets('Sample-Sales-Data.xlsx')  df <- read\_excel('Sample-Sales-Data.xlsx', sheet='Sheet1')  **<Regular expressions>**  grepl ( ‘term’ , text ) # l은 logical이라는 뜻이다.  grep (‘term’, text ) # index location을 잡아준다.  sub("Actor ", "", V(karate)$name) ## Actor를 없애버린다. substitute!!  **<Data visualization>**  3 layers  3. Geometrics  2. Aesthetics : which column you want  1. Data  Next 3 layers  4. Facets  statistics  5. coordinates  Final layer  6. Themes  4. coordinate  pl <- ggplot(mpg, aes(x=displ, y=hwy)) + geom\_point()  # coordinate 조정  pl2 <- pl + **coord\_cartesian**(xlim = c(1, 4), ylim = c(15, 30))  # aspect ratio 고정  pl3 <- pl + **coord\_fixed**(ratio =1/3)  5. faceting  # facet  pl + **facet\_grid**(. ~ cyl)  6. Theme  # package!!!!  install.packages("ggthemes")  library(ggthemes)  # 전체  theme\_set(theme\_minimal())  # 개별적으로 적용  pl + theme\_dark()  **- Histogram -**  install.packages('ggplot2')  install.packages('ggplot2movies')  library(ggplot2)  library(ggplot2movies)  1. Data layer / aesthetics layer  pl <- ggplot(movies, aes(x=rating))  pl + geom\_histogram( )  2. Geometric layer  - Geometrices 꾸미기  pl + geom\_histogram(binwidth = 0.1, color ='cyan1', fill='cyan1', alpha=0.5)  pl2<-pl + geom\_histogram(binwidth = 0.1, aes(**fill=..count..**))    - x, y label  pl3 <- pl2 + xlab('Movie Rating') + ylab('Count')  - title  pl3 + ggtitle("My TITLE")  + theme(plot.title = element\_text(hjust = 0.5))  # 가운데로 오게끔 하는 것.  **- Scatter plots -**  library(ggplot2)  library(data.table)  df <- fread('Economist\_Assignment\_Data.csv', **drop=1**)  # read.csv( ) 로 할 수 있다. drop=1은 사용할 수 없다.  # 대신 이렇게 했다.  df <- data.frame(csv)  df1<- df[,-1]  # Data & Aesthetics (Layer 1, 2)  pl <- ggplot(data = df, aes(x=wt, y=mpg) )  # Geometry (Layer 3)  pl + geom\_point()  pl + geom\_point((aes(size=hp)))  pl + geom\_point((aes(shape=factor(cyl))), size =5)  # hollow circle  geom\_point(shape=1) # Use hollow circles  # 슈퍼 중요 : geometry layer에 aes layer를 넣게 되면 산점도에서 3개의 data feature를 표현할 수 있게 된다는 것이다.  pl + geom\_point((aes(size=hp)))  pl + geom\_point((aes(shape=factor(cyl))), size =5)  # color랑 size를 동시에  pl + geom\_point((aes(shape=factor(cyl),color=factor(cyl))), size =5)  # hex color도 가능  <http://www.color-hex.com/>  pl + geom\_point(color =#94d8a6 , size =5)  # color gradient도 가능  pl2 <-pl + geom\_point(size=4, aes(color=hp))  pl2 + **scale\_color\_gradient**(low="blue", high='red')  # geom\_smooth(colour = ‘red’)  ggplot(dat, aes(x=xvar, y=yvar)) +  geom\_point(shape=1) + # Use hollow circles  geom\_smooth(method=lm, # Add linear regression lin se=FALSE) # Don't add shaded confidence region  # se = FALSE 하면 신뢰구간이 없어진다. standard error(95%)  # method = ‘lm’ 은 linear model을 의미  # loess  Local Polynomial Regression Fitting  #x 축 label  pl5<- pl4 + **scale\_x\_continuous**(name="Corruption Percentions Index, 2011 (10=least corrupt)", limits =c(1,10), breaks=1:10)  pl5  # y축 label  pl6 <- pl5 +**scale\_y\_continuous**(name="Human Development Index, 2011(1=best)", limits = c(0.2,1), breaks=seq(0.1,1,0.1))  # title  pl7 <- pl6 + **ggtitle**("Corruption and Human development") + theme(plot.title = element\_text(hjust = 0.5)) + theme(axis.text.x = element\_text(face = "italic"))  # 각 point labeling  pl3 <- pl2 + **geom\_text**(**aes**(label = Country),  color = "gray20",  data = subset(df, Country **%in%** pointsToLabel),  check\_overlap = T)  🡪 너무 많으니까 수를 줄여줌.  **- Bar plots -**  pl <- ggplot(df, aes(x=class))  pl + geom\_bar(aes(fill=drv))  # position = dodge 🡪 쌓아놓지 않고 옆으로 비껴가게 배열  # legend title 바꾸는 방법  p2 + guides(fill=guide\_legend(title="cyl"))  **- Box plot -**  pl <- ggplot(df, aes(x=factor(cyl), y=mpg))  pl + geom\_boxplot( ) + coord\_flip( )  # geometry layer 추가  pl + geom\_boxplot(aes(fill= factor(cyl)))  **- 2 variable plotting -**  pl <- ggplot(movies, aes( x = year, y=rating))  pl + geom\_bin2d( )  # 색 바꾸기  pl + geom\_bin2d() + scale\_fill\_gradient(high = 'red', low='green')  # bin width 바꾸기  pl<-ggplot(movies, aes(x=year, y=rating))  pl + geom\_bin2d(binwidth=c(1,0.1)) + scale\_fill\_gradient(high = 'red', low='green')    # hexbin을 해보자.  install.package(“hexbin”)  library(hexbin)  pl <- ggplot(movies, aes(x=year, y=rating))  pl2 <- pl + geom\_hex( )  - geom\_density2d()  pl <- ggplot(movies, aes(x=year, y=rating))  pl2 <- pl + geom\_density2d( ) |

- matrix로 만들기

M <- rbind(

revenue.1000,

expenses.1000,

profit.1000,

profit.after.tax.1000,

profit.margin,

good.months,

bad.months,

best.month,

worst.month

)

**(추가1)** regular expression : grep, grepl 그래플 (general regular expression)

grepl ( ‘term’ , text ) # l은 logical이라는 뜻이다.

grep (‘term’, text ) # index location을 잡아준다.

> text <- "Hello there, who are you voting for?"

> grepl('voting',text)

[1] TRUE

> v <- c('a','b','c','d','e')

> grepl('b',v)

[1] FALSE TRUE FALSE FALSE FALSE

> grep('b',v)

[1] 2

**(추가2) Date**

Sys.Date( ) # 오늘 날짜

[1] "2018-04-11" # 이것은 “Date” object이다.

character를 date object로 바꾸려면

c <- "1990-01-01"

my.date <- as.Date(c)

class(my.date)

- 형태가 다른 character를 Date로 바꾸고 싶으면

**format =** 을 사용해보자.

|  |
| --- |
| as.Date("June,01,2002", format = "%B,%d,%Y")  as.Date("Nov-3-90",format="%b-%d-%y")  # %d Day of the month(number)  # %m Month(number)  # %b Month(abbreviated)  # %B Full month name  # %y Year(2 digit)  # %Y Year(4 digit) |

🡪 trouble shooting

#register the old locale to set it back

OL <- Sys.getlocale("LC\_TIME")

#set the new locale

Sys.setlocale("LC\_TIME","C")

as.Date("Nov-3-90",format="%b-%d-%y")

#[1] "2016-03-21"

#Change the locale back to the old value

Sys.setlocale("LC\_TIME", OL)

- as.POSIXct(“11:02:03”, format = “%H:%M:%S”)로 시간까지 할 수 있다.

[1] "2018-04-11 11:02:03 KST"

- 그러나 대부분의 경우에는 strptime이 된다.

> strptime("11:02:03", format="%H:%M:%S")

[1] "2018-04-11 11:02:03 KST"

**5.2.2. Creating Function**

|  |
| --- |
| myplot <- function(data, rows=1:10){  Data <- data[rows,,drop=F]  matplot(t(Data), type="b",pch=15:18, col=c(1:4,6))  legend("bottomleft", inset=0.01, legend=Players[rows,,drop=F], col=c(1:4,6), pch=15:18, horiz = F)  }  myplot(Salary,1:10) |

🡪 default도 설정할 수 있다!

- python처럼

return도 할 수 있다.

🡪 return의 위치를 조심하라!

num\_check <- function(num,v){

for (item in v){

if (item == num){

return(TRUE) # TRUE되면 return 바로 하고 끝난다.

}

}

**return(FALSE) # return TRUE 안 되는 경우에**

}

🡪 내꺼

init <- function(a, x){

b = FALSE

for (i in x){

if (i == a){

b <- !b

}

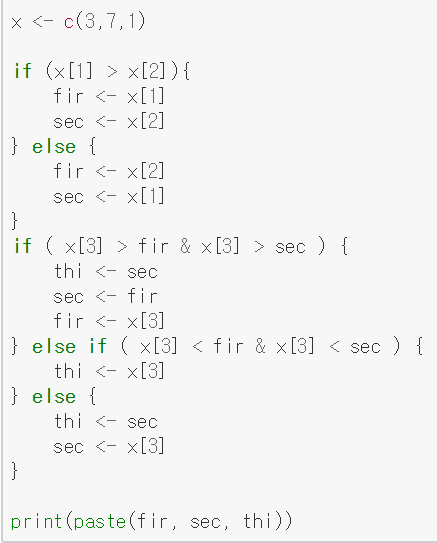
}

return(b)

}

과제에서 배울 점.

- 변수를 미리 지정하는 스킬



- 내 꺼는

order <- function(x){

a <- which(x == max(x))

y <- x[-a]

b <- max(x)

if(y[1]>=y[2]){

b <- c(b, y[1])

b <- c(b, y[2])

} else{

b <- c(b, y[2])

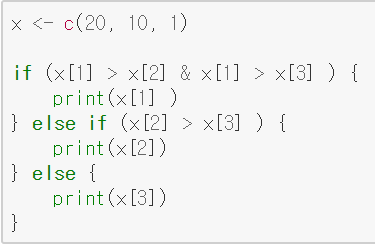
b <- c(b, y[1])

}

print(b)

}

- Max함수 만들 때, 부등호 2개 쓰는 게 핵심



- 내 꺼는

max\_func <- function(x){

if(x[1]>=x[2]){

if(x[1]>=x[3]){

print(x[1])

} else{

if(x[2]>=x[3]){

print(x[2])

} else{

print(x[3])

}

}

} else if(x[2]>x[3]){

print(x[2])

} else if(x[3]>x[2]){

print(x[3])

}

}

- 소수 만들기 (내꺼 짱임)

prime\_check <- function(a){

if (a ==2){

return(TRUE)

}

return(!( 0 **%in%** (a %% 2:(a-1))))

}

🡪 선생님꺼는 any 사용

prime\_check <- function(num) {

if (num == 2) {

return(TRUE)

} else if (**any**(num %% 2:(num-1) == 0)) {

return(FALSE)

} else {

return(TRUE)

}

}

- 3배수 빼고 다 더하기

summer <- function(x,y,z){

vec <- c(x,y,z)

vec\_2<-vec[!(0 == (vec %% 3))]

return(sum(vec\_2))

}

🡪 vectorized operation

**(2) anonymous function # lambda expression**

result <- sapply( v , function(num){num\*2})

# 파이썬에서는

lambda num : num \* 2, num\_list

function(num){

return(num\*2)

} 을 줄인 것이다.

print(result)

**(3) 함수의 빠른 적용!** vector에 다 적용(lapply)

🡪 lambda의 map 기능을 한다.

map(function, num\_list)

**lapply** # list apply

lapply(v, addrand) # go to every single element and add random number

- 주의 : addrand( ) 로 하는게 아니다.

[[1]]

[1] 9

[[2]]

[1] 83

[[3]]

[1] 92

[[4]]

[1] 7

[[5]]

[1] 34

- sapply !! # simple 🡪 이게 더 간단하다.

> sapply(v, addrand)

[1] 97 84 92 67 65

- 만약에 argument가 여러 개이면 지정해줘야한다.

add\_choice <- function(num, choice){

return(num + choice)

}

sapply(v, add\_choice, choice = 100)

**5.3. Packages**

- packages are collections of R functions, data, compiled code in well-defined format.

- Library : the directory where packages are stored

- 우측 하단에 package창이 있는데, 체크된 것은 activate된 것이다.

- install 버튼

- cran이 무엇이냐?

<https://cran.r-project.org/>

- down 받는 방법

**install.packages**("ggplot2")

- package activate하는 방법

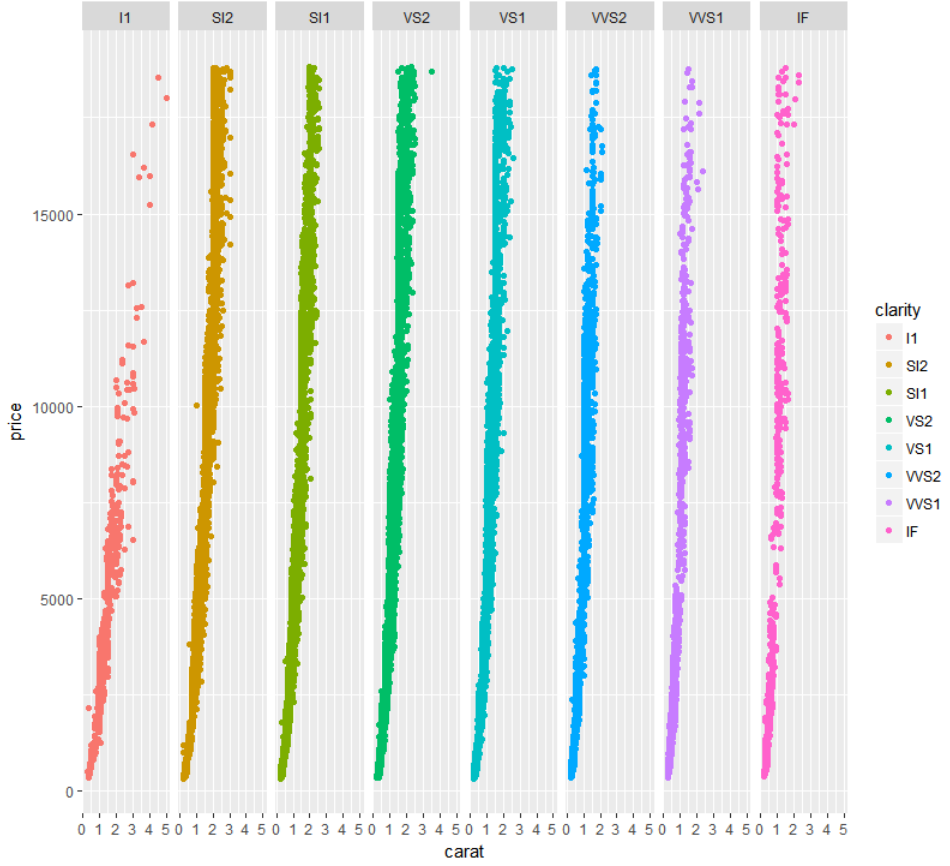
library(ggplot2)

🡪 library라는 함수를 사용해서 activate를 할 수 있다.

ex)

qplot(data=diamonds, carat, price,

colour=clarity, facets =.~clarity)



**5.4. 적용(financial statement)**

New skill

|  |
| --- |
| > month <- c(12,3,6,2,3,7)  > month.abb[month]  [1] "Dec" "Mar" "Jun" "Feb" "Mar" "Jul"  print(paste0("Current working dir: ", wd))  sprintf("Current working dir: %s", wd) |

- full coding (My version)

|  |
| --- |
| #Data  revenue <- c(14574.49, 7606.46, 8611.41, 9175.41, 8058.65, 8105.44, 11496.28, 9766.09, 10305.32, 14379.96, 10713.97, 15433.50)  expenses <- c(12051.82, 5695.07, 12319.20, 12089.72, 8658.57, 840.20, 3285.73, 5821.12, 6976.93, 16618.61, 10054.37, 3803.96)  #Solution  grandize <- function(v){  v <- round(v/1000, digits=0)  print(paste0("$",v,"k"))  }  percentize <- function(v){  print(paste0(round(v,0),"%"))  }  profit <- revenue - expenses  profit  print("profit before taxation")  grandize(profit)  profit\_taxed <- profit \* 0.7  profit\_taxed <- round(profit\_taxed, digits=2)  print("profit after taxation")  grandize(profit\_taxed)  profit\_margin <- 100 \*profit\_taxed / revenue  profit\_margin  print("profit margin in percent")  percentize(profit\_margin)  mean <- sum(profit\_margin)/12  mean  goodmonth <- rep(NA, 12)  for(i in 1:12){  if(profit\_margin[i] > mean){  goodmonth[i] <- "TRUE"  } else{  goodmonth[i] <- "FALSE"  }  }  ㅋㅋㅋㅋㅋㅋ 한 줄이면 끝남  goodmonth <- profit\_margin > mean  for loop을 사용하기 전에 vectorized operation이 될 지를 생각하라!!  goodmonth  badmonth <- rep(NA, 12)  for(i in 1:12){  if(profit\_margin[i] < mean){  badmonth[i] <- "TRUE"  } else{  badmonth[i] <- "FALSE"  }  }  badmonth <- profit\_margin < mean  ㅋㅋㅋ 더 간단히  badmonth <- !goodmonth  ### 대박!!!  badmonth  max\_value <- max(profit\_margin)  min\_value <- min(profit\_margin)  max\_value  a <- which(max\_value == profit\_margin)  b <- match(min\_value, profit\_margin)  sprintf("The best month was %s, reaching a profit margin of %s", month.abb[a], max\_value)  sprintf("The worst month was %s, reaching a profit margin of %s", month.abb[b], min\_value) |

**6. Matrices**

- 데이터를 실행해서 받았으면 새로운 script를 열어서 작업하면 된다.

- Matrix는 dataframe으로 가기 위한 stepping stone이다.

- Matrix도 하나의 type만 들어있을 수 있다. character하나 있으면 다른 숫자가 다 character로 바뀐다.

- dataframe과의 차이점은 무엇인가? data type가 여러 개라는 것?

**6.1. Matrix 만들기**

|  |
| --- |
| matrix(vector, nrow = ,ncol = ,byrow = T/F )  🡪 뱀처럼 1열 위에서 아래로, 2열 위에서 아래로, 3열 위에서 아래로 순서대로  rbind(row vector1, vector2, vector3 )  cbind(column1, column vector2, column vector3) |

- matrix function

x <- c(1,2,3,4,5,6,7,8,9,10)

matrix(x, 2, 5)

[,1] [,2] [,3] [,4] [,5]

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

🡪 nrow로 할 수 있다. row를 몇 개 만들 지를.

- 들어가는 순서를 row부터 차도록 할 수도 있다. : **byrow = T** 🡨 디폴트는 FALSE이다.

x <- 1:20

A <- matrix(x, 4, 5, byrow=T)

A

|  |
| --- |
| <matrix함수로 만들기>  goog <- c(450, 451, 452, 445, 468)  msft <- c(230, 231, 232, 233, 220)  stock <- c(goog, msft)  stock.matrix <- **matrix**(stock, byrow =T, nrow=2)  days <- c('Mon','Tue',"Wed",'Thu','Fri')  st.names <- c("Google", "Microsoft")  colnames(stock.matrix) <- days  rownames(stock.matrix) <- st.names  stock.matrix  >> Mon Tue Wed Thu Fri  Google 450 451 452 445 468  Microsoft 230 231 232 233 220 |

[,1] [,2] [,3] [,4] [,5]

[1,] 1 2 3 4 5

[2,] 6 7 8 9 10

[3,] 11 12 13 14 15

[4,] 16 17 18 19 20

- **rbind**(row vector, vector, vector.... 사례 :

|  |
| --- |
| #Matrix  Salary <- rbind(KobeBryant\_Salary, JoeJohnson\_Salary, LeBronJames\_Salary, CarmeloAnthony\_Salary, DwightHoward\_Salary, ChrisBosh\_Salary, ChrisPaul\_Salary, KevinDurant\_Salary, DerrickRose\_Salary, DwayneWade\_Salary)  rm(KobeBryant\_Salary, JoeJohnson\_Salary, CarmeloAnthony\_Salary, DwightHoward\_Salary, ChrisBosh\_Salary, LeBronJames\_Salary, ChrisPaul\_Salary, DerrickRose\_Salary, DwayneWade\_Salary, KevinDurant\_Salary)  # 기존의 vector를 없앤다.  colnames(Salary) <- Seasons # season은 vector 미리 지정해놓은 것이다.  rownames(Salary) <- Players |

- rbind 사례 :

> stock.matrix

Mon Tue Wed Thu Fri

Google 450 451 452 445 468

Microsoft 230 231 232 233 220

FB <- c(111, 112, 113, 120, 145)

tech.stocks <- **rbind**(stock.matrix, FB)

tech.stocks

Mon Tue Wed Thu Fri

Google 450 451 452 445 468

Microsoft 230 231 232 233 220

FB 111 112 113 120 145

🡪 되게 신기하다!! FB로 row name이 저절로!

- **cbind** 사례

avg <- rowMeans(tech.stocks)

tech.stocks <- cbind(tech.stocks, avg)

tech.stocks

Mon Tue Wed Thu Fri avg

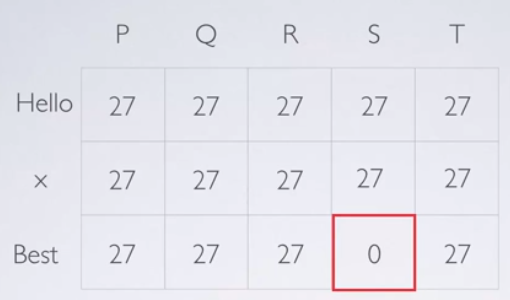
Google 450 451 452 445 468 453.2

Microsoft 230 231 232 233 220 229.2

FB 111 112 113 120 145 120.2

**6.1.2. Naming dimensions**

- naming을 하더라도 원래 column, row 숫자는 그대로 쓰면 된다.



A[3, 4]

A[3, “S”]

A[“Best”, “S”]

A[“Best”, 4]

어떤 거든 간에 가능하다.

- naming 방법

Charlie <- 1:5

Charlie

- Vector naming하기

**names**(Charlie) <- c("a","b","c","d","e")

> Charlie

a b c d e

1 2 3 4 5

- clear name

names(Charlie) <- NULL

- Matrix naming하기

|  |
| --- |
| (1) rownames(Bravo)  temp.vec <-rep(c("a","b","c"), each=3)  Bravo <- matrix(temp.vec, 3, 3)  **rownames**(Bravo) <- c("How","are","you")  (2) colnames(Bravo)  **colnames**(Bravo) <- c("x","y","z") |

**6.1.3. Factor and Categorical matrices**

- factor( )의 목적 : creating dummy variables. Machine learning에 나중에 매우 유용하게 사용될 것이다.

- categorical variable의 종류

(1) nominal categorical variable - No order

ex) dog, cat과 같음, 순서 없이.

animal <- c(‘d’, ‘c’, ‘d’, ‘c’, ‘c’)

factor.ani <- **factor**(animal)

> factor(animal)

[1] d c d c c

Levels: c d

🡪 level을 저절로 나눠주는구나! 컴퓨터가 이해하기 쉽게!

(2) Ordinal categorical variable - has order

temps <- c('cold','med','cold','med','hot','hot','cold')

fact.temp <- factor ( temps, ordered = True, levels = c(‘cold’, ‘med’, ‘hot’)

> fact.temp

[1] cold med cold med hot hot cold

**Levels: hot < med < cold**

🡪 ordered, levels!! 사용!

- factor를 왜 쓰는가?

summary할 때 매우 편하다!!!

- Summary

> summary(temps)

Length Class Mode

7 character character

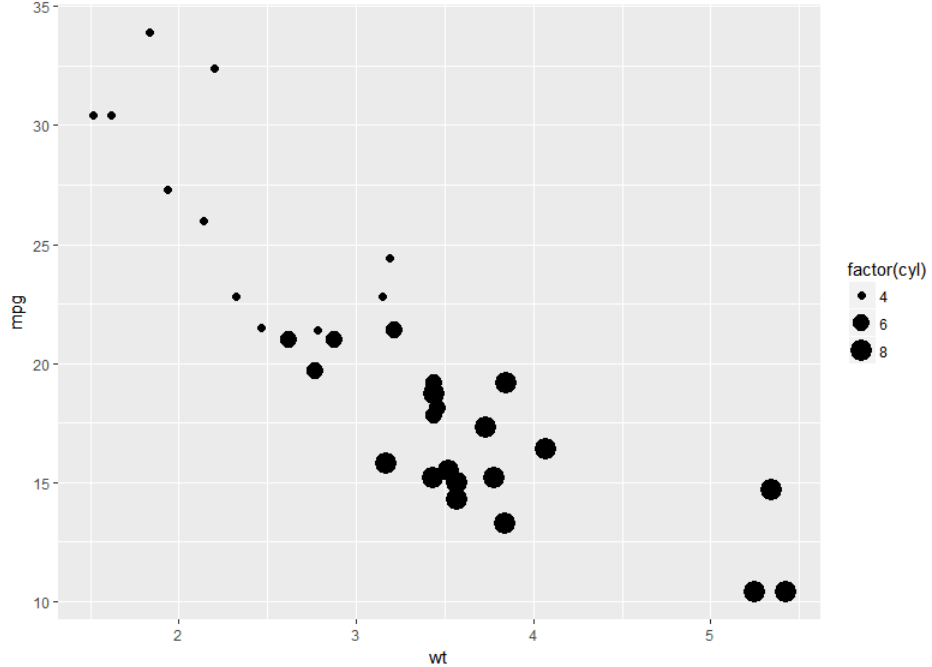
> summary(fact.temp)

hot med cold

2 2 3 🡪 두 개가 보여주는 정보가 확연히 다르다. factor한 거랑 factor 안 한거랑.

- 그래프 그릴 때도 유용 🡨 대박!!!

pl + geom\_point((aes(**size=factor(cyl))**))



**6.2. Matrix arithmetic**

- 다양한 연산

mat <- matrix(1:50, byrow = T, nrow=5)

mat \* 2 (원소 2배)

1/mat

mat /2

mat ^ 2

- Comparison operator

mat > 17

- 행렬과 행렬의 연산

mat + mat # 각 원소끼리

mat / mat

mat ^ mat

mat \* mat # 각 원소끼리

- 행렬 곱 : **%\*%**

mat2 <- matrix(1:9, nrow=3)

mat2 %\*% mat2

- factor( )

**6.3. Matrix operator**

> stock.matrix

Mon Tue Wed Thu Fri

Google 450 451 452 445 468

Microsoft 230 231 232 233 220

>> **colSums**(stock.matrix)

Mon Tue Wed Thu Fri

680 682 684 678 688

>> **rowSums**(stock.matrix)

Google Microsoft

2266 1146

>> **rowMeans**(stock.matrix)

- **t**(x) # transpose

- **diag**(x) # diagonal

- Ax=b solve A %\*% x = b

**solve**(a, b)

- inverse

**solve**(a)

Error in solve.default(mat2) :

Lapack routine dgesv: system is exactly singular: U[3,3] = 0

- **is.matrix**( mat )

> TRUE

**6.4. Matrix indexing**

A[3,4]

- row를 다 뽑으려면

A[1,]

A[2,]

- column을 다 뽑으려면

A[,1]

A[,2] 라고 하면 된다.

- 처음 3개의 row

> mat [1:3, ]

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 1 2 3 4 5 6 7 8 9 10

[2,] 11 12 13 14 15 16 17 18 19 20

[3,] 21 22 23 24 25 26 27 28 29 30

- 마지막 2개의 column

> mat[, 9:10]

[,1] [,2]

[1,] 9 10

[2,] 19 20

[3,] 29 30

[4,] 39 40

[5,] 49 50

- 사이 :

mat [2:3, 5:6]

**6.5. Plotting**

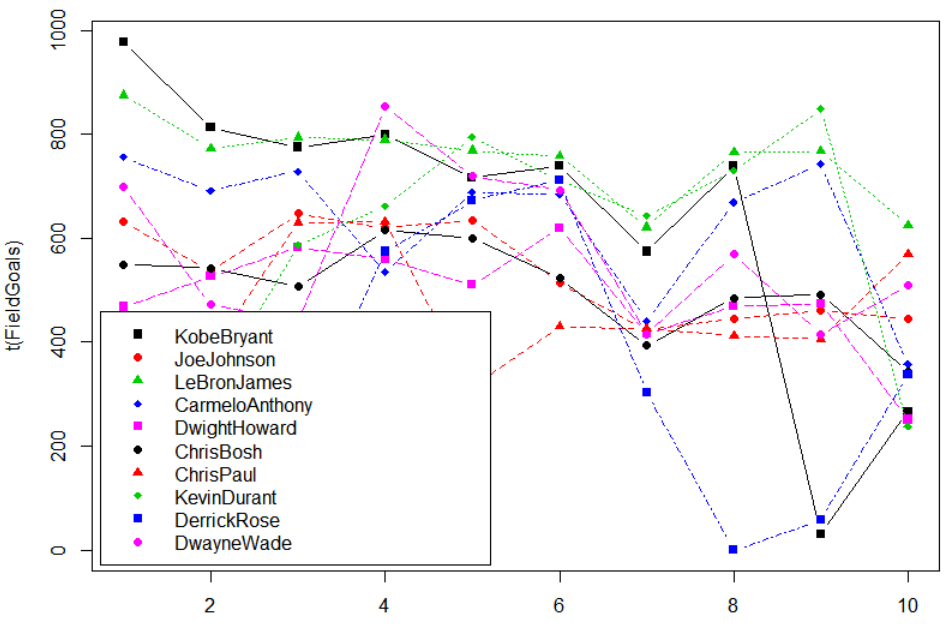
- Transpose

t(FieldGoals)

**matplot**(t(FieldGoals), type = 'b', pch = 15:18, col=c(1:4,6))

legend("bottomleft", inset=0.01, legend = Players, col=c(1:4,6), pch=15:18, horiz=F)

🡪 color랑 point character가 같아야 한다.



- Subsetting

- vector의 subsetting

x <- c("a","b","c","d","e")

x

x[c(1,5)]

x[1]

- matrix의 subsetting

Games[1:3,6:10]

> Games[c(1,10),]

2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

KobeBryant 80 77 82 82 73 82 58 78 6 35

DwayneWade 75 51 51 79 77 76 49 69 54 62

- 1개의 row만 빼면 벡터가 된다.

Games[1,]

> is.vector(Games[1,])

[1] TRUE

> is.matrix(Games[1,])

[1] FALSE

- 이 현상이 발생하지 않도록 하려면 : drop = F

왼쪽의 row의 name을 drop하지 않도록 해야 한다.

Games[1, , **drop=F**]

> Games[1, , drop=F]

2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

KobeBryant 80 77 82 82 73 82 58 78 6 35

is.matrix(Games[1, , drop=F])

>> [1] TRUE

- 이제 3명의 선수만 plotting을 하고 싶다.

Data <- MinutesPlayed[1:3,]

matplot(t(Data), type="b",pch=15:18, col=c(1:4,6))

legend("bottomleft", inset=0.01, legend=Players[1:3], col=c(1:4,6), pch=15:18, horiz = F)

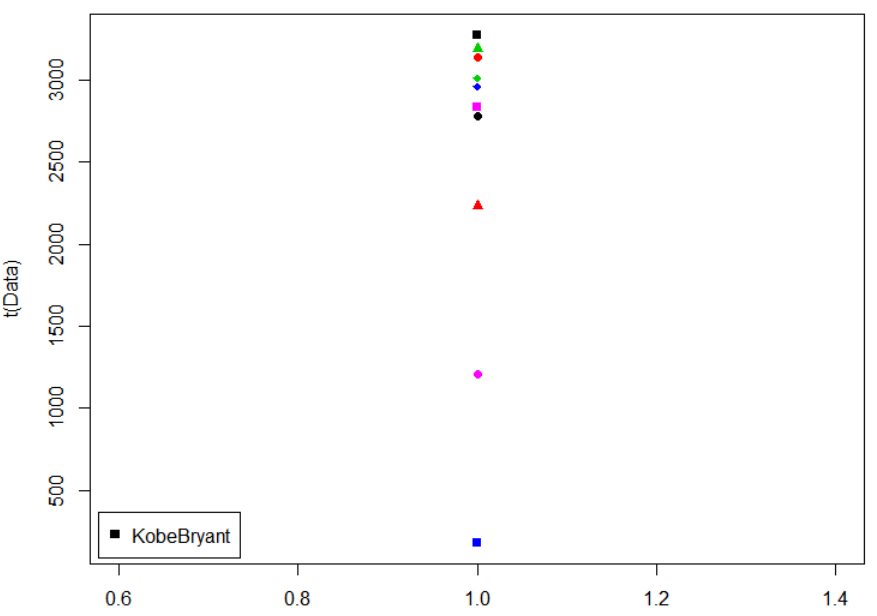
- 이제 1명의 선수만 plotting하고 싶다.

이렇게 하면 잘 못 된다.

Data <- MinutesPlayed[1,]

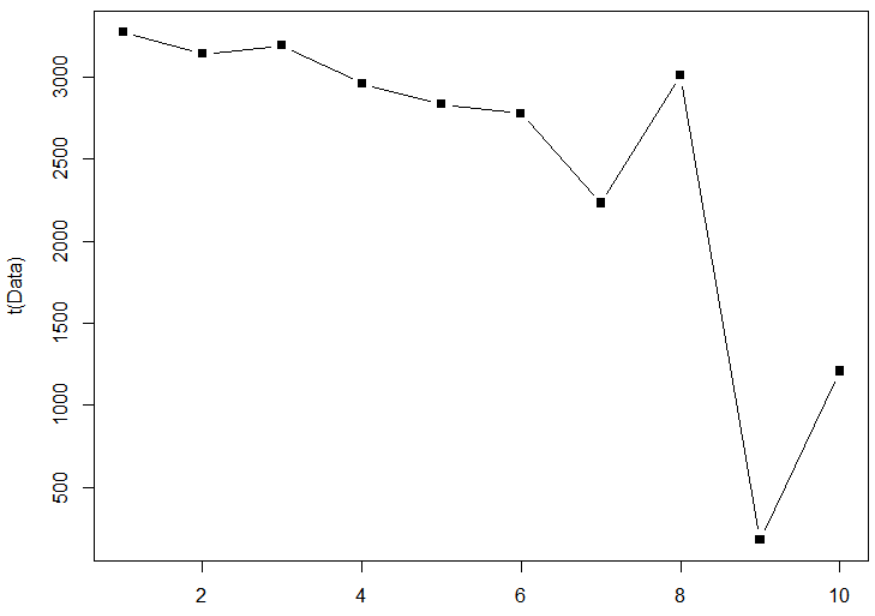
matplot(t(Data), type="b",pch=15:18, col=c(1:4,6))

legend("bottomleft", inset=0.01, legend=Players[1], col=c(1:4,6), pch=15:18, horiz = F)



- 잘 한 경우 : drop = F을 해줘야 matrix가 유지!!

|  |
| --- |
| Data <- MinutesPlayed[1,,drop=F]  matplot(t(Data), type="b",pch=15:18, col=c(1:4,6))  legend("bottomleft", inset=0.01, legend=Players[1,,drop=F], col=c(1:4,6), pch=15:18, horiz = F) |



**7. Data frames**

- 필요한 이유 : matrix의 한계 때문이다. Matrix는 한가지의 data type만 넣을 수 있다. Data frame은 mix data type도 다룰 수 있다.

- 용어 : python/pandas의 DataFrame과 다르다. data frame이다.

- 예시 :

data( ) # 모든 data set을 볼 수 있다.

state.x77

women

**7.1. 큰 데이터 set 관찰법**

|  |
| --- |
| head(df, 몇번째 줄) # default is 6번째  tail(df)  str(df)  summary(df)  nrow(df)  ncol(df)  colnames(df)  rownames(df) |

> states <- state.x77

head(states) # 위에서 6번째줄까지

tail(states) # 아래에서 6번째까지

- 구조를 보는 법 : str( Dataframe )

> str(states)

num [1:50, 1:8] 3615 365 2212 2110 21198 ...

- attr(\*, "dimnames")=List of 2

..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...

..$ : chr [1:8] "Population" "Income" "Illiteracy" "Life Exp" ...

- 여러 통계량 보는 법

summary (states)

Frost Area

Min. : 0.00 Min. : 1049

1st Qu.: 66.25 1st Qu.: 36985

Median :114.50 Median : 54277

Mean :104.46 Mean : 70736

3rd Qu.:139.75 3rd Qu.: 81163

Max. :188.00 Max. :566432

|  |
| --- |
| nrow(df)  ncol(df)  colnames(df)  rownames(df) |

- row가 몇 개냐? (index name 제외)

> nrow(df)

[1] 10

> ncol(df)

2

- colname 이 뭐냐

> colnames(df) or rownames(df)

[1] "col.name.1" "col.name.2"

**7.2. data frame 만드는 방법 🡺 data.frame(vector, vector, vector)**

|  |
| --- |
| (1) 빈 data frame 만들기 : data.frame( )  > empty <- **data.frame( )**  (2-1) 벡터들을 붙여서 만들기  > days <- c('Mon', 'Tue', 'Wed', 'Thu', 'Fri')  > temp <- c(22, 23, 24, 25, 26)  > rain <- c(T, T, F, F, T)  > df <- **data.frame**(days, temp, rain)  (2-2) Matrix도 바꿀 수 있다.  new\_df <- data.frame(mat)  (2-3) dataframe 두개를 merge할 수도 있다.(NEW)  merge  merge(x, y, by = intersect(names(x), names(y)),  by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,  sort = TRUE, suffixes = c(".x",".y"),  incomparables = NULL, ...)  total <- merge(batting, salary, c('yearID', 'playerID'))  🡺 dplyr의 경우에는 left\_join도 할 수 있다!!  (3) naming  - column name을 만들 수도 있다.  df <- data.frame(col.name.1 = c1, col.name.2=c2)  c1 <- 1:10  c2 <- letters[1:10] # letter는 a~z까지 벡터  df <- data.frame(col.name.1 = c1, col.name.2 = c2)  # 앞에 다가 쓰는게 중요하다. 안 쓰면 c1, c2가 column name이 된다.  (4) Adding new row  dfnew <- rbind (df, df2)  (5) Adding new columns  df$newcol.copy <- df$newcol  = df[ , ‘newcol.copy2 ] <- df$newcol  df$performance <- round(df$performance,2)  (6) Setting new column names  colnames(df) <- c('1','2','3','4')  colnames(df)[1] <- 'New col name' |

> df

col.name.1 col.name.2

1 1 a

2 2 b

3 3 c

4 4 d

5 5 e

6 6 f

7 7 g

8 8 h

9 9 i

10 10 j

- 예시

Name <- c("Sam","Frank","Amy")

Age <- c(22,25,26)

Weight <- c(150,165,120)

Sex <- c("M", "M", "F")

df <- data.frame (**row.names** = Name, Age, Weight, Sex)

df

**(4) adding new row**

df2 <- data.frame(col.name.1 = 2000, col.name.2 = 'new')

df2

dfnew <- rbind (df, df2)

dfnew

**(5) adding new columns** 🡺 df **$** 새로운 column을 사용하면 쉽게 column을 추가할 수 있다.

df$newcol <- 2 \* df$col.name.1

- 장점 :

- cbind보다 편하다.

- 복사할 수 있다.

df$newcol.copy <- df$newcol

= df[ , ‘newcol.copy2 ] <- df$newcol

- column값 교체/가공

df$performance <- round(df$performance,2)

**(6) Setting column names**

- column name을 다 바꾸고 싶으면

colnames(df) <- c('1','2','3','4')

- 1개의 colomn name만 바꾸고 싶으면

colnames(df)[1] <- 'New col name'

왜냐하면 colname 자체가 vector니까 이렇게 가능하다.

**7.3. Data frame selection and indexing**

|  |
| --- |
| (1) Indexing / referencing  df [ 1 ] 하면 그냥 1번 column이 나온다!!! 신기!!!  df [[5,'col.name.2']] # single cell  df[1,]  df[,1]  df[1:5,c('days','temp')]  df[ ,'rain'] = df$rain --> column을 vector 형태로  df['rain'] 🡪 data frame의 구조 유지 |

# Some made up weather data

days <- c('mon','tue','wed','thu','fri')

temp <- c(22.2,21,23,24.3,25)

rain <- c(TRUE, TRUE, FALSE, FALSE, TRUE)

> df

days temp rain

1 Mon 22 TRUE

2 Tue 23 TRUE

3 Wed 24 FALSE

4 Thu 25 FALSE

5 Fri 26 TRUE

**(1) Indexing / referencing**

**(1-1) Referencing single cell**

df [ 1 ] 하면 그냥 1번 column이 나온다!!! 신기!!!

🡪 짱이다.

> df [[5,'col.name.2']] # or df [[5,2]]

[1] e

- reassign할 수도 있다.

df[[2, 'col.name.1']] <- 9999

- double bracket을 사용하는 이유 : list에서 extract할 때, type가 달라진다.

🡪 [] extracts a list, [[]] extracts elements within the list

- 실례 :

if( foo[ 'bool' ] ){ print("Hi!") } 🡪 에러가 난다.

This is because the [] method returned a list and a list is not valid object to pass directly into an if() statement. In this case we need to use [[]] because it will return the "bare" object stored in 'bool' which will have the appropriate class:

**(1-2) Referencing row / column**

|  |
| --- |
| df <- data.frame(days,temp,rain)  df[1,] 🡪 row 뽑으면 data frame 유지  🡪 vector를 가져오고 싶다면?  vrow <- as.numeric(as.vector(df[1,]))  df[,1] 🡪 column뽑을 때는 vector로  df[1:5,c('days','temp')]  df[,'rain'] 🡪 df$rain이 더 편하다.  [1] TRUE TRUE FALSE FALSE TRUE  🡪 vector 형태로 가져오게 된다.  df['rain'] 🡪 이렇게 하면 좀 다름 (data frame의 구조 유지)  - referencing column : vector로 뽑는 4가지 방법이 있다. 2가지만 기억하기.  colv1 <- cars$mpg  colv2<- cars[ , 'mpg']  colv3 <-cars[, 1] # uncommon  colv4 <-cars[['mpg']] # uncommon  - column을 data frame으로 뽑는 1가지 방법  cars [‘mpg’] # single set of brackets!!  cars [1]  - multiple columns 가져오려면?  df[1:3,] # 3개의 row 가져올 수 있다.  df[1:10, ] # 10개의 row 가져올 수 있다.  head(df, 7) # 처음 7개의 row를 가져올 수 있다.  df[-2, ] # everything but second row  mtcars[ , c(1, 2, 3) ]  or mtcars[ , c(‘mpg’, ‘cyl’) ] # vector 형태로  > mtcars [c('mpg', 'cyl')] # data frame을 살리기  앞의 6줄만 : head (mtcars [c('mpg', 'cyl')] ) |

> df['rain']

rain

1 TRUE

2 TRUE

3 FALSE

4 FALSE

5 TRUE

**(2) 원하는 조건의 row / column 가져오기**

??????? 질문???

아 왜 row에 다가 넣는지 알 것 같다. row 중에서 이를 만족시키는 row의 index만 가져와라구나.

|  |
| --- |
| # Bracket형 1가지 조건  $를 안 쓰면 그냥 다 가져온다. ㅋㅋ  mtcars [ mtcars$mpg > 20 , ]  mtcars [mtcars$mpg >20, 'cyl']  🡺 만약에 dataframe을 그대로 유지하고 싶으면 drop = F하면 된다.  mtcars [mtcars$mpg >20, 2:3]  mtcars[ (mtcars$mpg > 20) & (mtcars$cyl ==6) ,]  # 두가지 조건  mtcars[ mtcars$mpg > 20 & mtcars$cyl ==6 , c('mpg','cyl','hp')]  mean(df[(df$hp>100) & (df$wt >2.5),]$mpg)  # subset  > subset(df, subset = rain == TRUE) # ‘ ‘ 안 해도 된다.  > subset(df, subset = temp > 24)  **subset**(mtcars, mpg > 20 & cyl ==6) |

**(2-1) Bracket 속의 조건 (Bracket notation)**

mtcars [ mtcars$mpg > 20 , ] # 모든 column을 가져와라라는 의미이다. 쉼표 뒤에 다른 column을 고르는데 조건을 둘 수 도 있다.

df[df['mpg'] >= 20,] # 도 같은 의미

🡪 예를 들어

> mtcars [mtcars$mpg >20, 'cyl']

[1] 6 6 4 6 4 4 4 4 4 4 4 4 4 4

🡺 만약에 dataframe을 그대로 유지하고 싶으면 drop = F하면 된다.

🡪 혹은

> mtcars [mtcars$mpg >20, 2:3]

cyl disp

Mazda RX4 6 160.0

Mazda RX4 Wag 6 160.0

Datsun 710 4 108.0

Hornet 4 Drive 6 258.0

Merc 240D 4 146.7

Merc 230 4 140.8

Fiat 128 4 78.7

Honda Civic 4 75.7

Toyota Corolla 4 71.1

Toyota Corona 4 120.1

Fiat X1-9 4 79.0

Porsche 914-2 4 120.3

Lotus Europa 4 95.1

Volvo 142E 4 121.0

- 주의 사항 : 만약에

mtcars [ mtcars$mpg > 20 ] 만 하게 되면

>> Error in `[.data.frame`(mtcars, mtcars$mpg > 20) :

undefined columns selected

- 두가지 조건을 원하면

> mtcars[ (mtcars$mpg > 20) & (mtcars$cyl ==6) ,]

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

Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

- 거기다가 column 조건도 추가하면

> mtcars[ mtcars$mpg > 20 & mtcars$cyl ==6 , c('mpg','cyl','hp')]

mpg cyl hp

Mazda RX4 21.0 6 110

Mazda RX4 Wag 21.0 6 110

Hornet 4 Drive 21.4 6 110

- What is the average mpg for cars that have more than 100 hp AND a wt value of more than 2.5.

> mean(df[(df$hp>100) & (df$wt >2.5),]$mpg)

아니면

sub <- subset(df, hp>100 & wt>2.5)

mean(sub$mpg)

**(2-2) Subset**

> subset(df, subset = rain == TRUE) # ‘ ‘ 안 해도 된다.

days temp rain

1 Mon 22 TRUE

2 Tue 23 TRUE

> subset(df, subset = temp > 24)

days temp rain

4 Thu 25 FALSE

5 Fri 26 TRUE

🡪 subset = 을 안 써도 된다.

그리고 matrix에서처럼 drop = False도 여기서 쓸 수 있다.

- subset을 사용해서 같은 것을 할 수 있다.

와 대박!!!!!!

> **subset**(mtcars, mpg > 20 & cyl ==6)

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

Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

**7.4. Dealing with missing data**

|  |
| --- |
| is.na(mtcars) # output은 data frame  any ( is. na(df) ) # TRUE가 1개는 있는지 : 있으면 TRUE  df [ is.na( df ) ] <- 0  mtcars$mpg [ is.na(mtcars$mpg) ] <- 0  df <- df[!is.na(df$col), ] |

- missing data가 있는지 확인해보기 : 있으면 TRUE

> is.na(mtcars)

mpg cyl disp hp drat

Mazda RX4 FALSE FALSE FALSE FALSE FALSE

Mazda RX4 Wag FALSE FALSE FALSE FALSE FALSE

Datsun 710 FALSE FALSE FALSE FALSE FALSE

Hornet 4 Drive FALSE FALSE FALSE FALSE FALSE

Hornet Sportabout FALSE FALSE FALSE FALSE FALSE

- 1개라도 TRUE 값이 있냐고 할 때는, any 사용

any ( is. na(df) )

any ( is.na(mtcars$mpg ) ) # mpg column에 있는 지도!

any(a %% 2:(a-1) == 0) # this is matrix inside any( )

- na인 곳을 0으로 대체하고 싶다면

df [ is.na( df ) ] <- 0

와!!!!! 대박!!!!!

mtcars$mpg [ is.na(mtcars$mpg) ] <- 0

!!!! 와!!! 대박대박!!!

|  |
| --- |
| mtcars$mpg [ is.na(mtcars$mpg) ] <- mean (mtcars$mpg) |

cf)

# delete selected missing data rows

df <- df[!is.na(df$col), ]

is.na(df$col)은 na이면 TRUE라고 한다. 그런데 !를 붙이면 거꾸로 된다. TRUE였던 것들이 FALSE가 되기 때문에 na였던 row들은 사라지게 되는 것이다.

**7.5. Ordering the data frame** : order( column )

🡺 엑셀에서의 filter 개념이다.

sorted.temp <- **order**(df['temp'])

🡪 df$temp 라고 해도 된다!!!!

sorted.temp <- **order**(df$temp)

> sorted.temp # 작은 거에서 큰 순서대로 index를 vector를 만드는 것이다.

[1] 1 2 3 4 5

- 이것을 활용하는 방법 (내림차순 정렬인가?)

df[sorted.temp, ] 🡺 쉼표를 하는 이유는 row여서 그런게 아닐까 생각된다. 그렇지!! column이었으면 어떻게 됐을까?

>> days temp rain

1 Mon 22 TRUE

2 Tue 23 TRUE

3 Wed 24 FALSE

4 Thu 25 FALSE

- 순서 거꾸로하려면

> desc.temp <- order( **-**df [‘temp] )

> df[desc.temp,]

days temp rain

5 Fri 26 TRUE

4 Thu 25 FALSE

3 Wed 24 FALSE

2 Tue 23 TRUE

1 Mon 22 TRUE

- 만약에 숫자가 아니면

Boolean : FALSE먼저

character : 알파벳 순서

**8. Lists 🡺 organizational tool**

|  |
| --- |
| # 만들기  my.list <- list(v,m,df)  # naming  my.named.list <- list(sample.vec = v, my.matrix = m, sample.df = df)  # 데이터 꺼내오기  my.named.list[['sample.vec']]  my.named.list$sample.vec  # nested list  double.list <- c(my.named.list, my.named.list)  # structure  str(double.list) |

- 목적 : vector, matrix, data frame을 하나의 list 안에 집어넣을 수 있다. 그 다음에 부르기가 편하다.

v <- c(1,2,3)

m <- matrix(1:10, nrow=2)

df <- mtcars

my.list <- list(v,m,df)

my.list

> my.list

[[1]]

[1] 1 2 3

[[2]]

[,1] [,2] [,3] [,4] [,5]

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

[[3]]

mpg cyl disp hp drat wt qsec

Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46

Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02

Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61

- naming

my.named.list <- list(sample.vec = v, my.matrix = m, sample.df = df)

$sample.vec

[1] 1 2 3

$my.matrix

[,1] [,2] [,3] [,4] [,5]

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

$sample.df

mpg cyl disp hp drat wt qsec

Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46

Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02

- 데이터 꺼내오기.

> my.named.list$sample.vec

[1] 1 2 3

> my.named.list[1]

$sample.vec

[1] 1 2 3

> my.named.list['sample.vec']

$sample.vec

[1] 1 2 3

- 그런데 class는 뭐냐?

> class(my.named.list['sample.vec'])

[1] "list"

- [[ ]]를 쓰면 / 혹은 $를 쓰면 그 object을 가져올 수 있다.

> class(my.named.list[['sample.vec']])

[1] "numeric"

- list 안의 list

double.list <- c(my.named.list, my.named.list)

**9. Importing and exporting data**

**(1) CSV file**

파일 쓰기 : write.csv

> **write.csv** (df, file= 'saved\_df.csv')

🡪 index까지 저장이 된다.

> df2

X col.name.1 col.name.2

1 1 1 a

2 2 2 b

3 3 3 c

4 4 4 d

5 5 5 e

6 6 6 f

7 7 7 g

8 8 8 h

9 9 9 i

10 10 10 j

- 파일 읽기

> df2 <- **read.csv**('saved\_df.csv')

ex <- read.csv('my\_example.csv')

🡪

csv <- fread('Economist\_Assignment\_Data.csv')

**(2) Excel files**

|  |
| --- |
| install.packages('**readxl**')  library(readxl)  setwd("c:/Users/Owner/Desktop")  excel\_sheets('Sample-Sales-Data.xlsx')  df <- read\_excel('Sample-Sales-Data.xlsx', **sheet='Sheet1'**) |

- directory 변경

setwd("c:/Users/Owner/Desktop/")

- 전체 sheet을 list로 만들고 싶을 때

entire.workbook <- lapply(excel\_sheets('Sample-Sales-Data.xlsx'), read\_excel, path='c://Users/Owner/Desktop/Sample-Sales-Data.xlsx')

- writing excel file

|  |
| --- |
| install.packages(‘**xlsx**’)  library('xlsx')  write.xlsx(mtcars, "output\_example.xlsx") |

**(3) SQL**

RODBC library 사용

> install.packages(“RODBC”)

library(RODBC)

myconn <- odbcConnect(“Database\_Name”, uid = “User\_ID”, pwd = “password”)

dat <- sqlFetch(myconn, “Table\_Name”)

- RMySQL

- ROracle

- RJDBC

자기의 database에 따라서 다르다.

🡪 구글 검색 r + postgresqi

cran과 연결된 것을 열어서 manual을 본다.

- rbloggers이 매우 좋다.

connection with R 부분을 찾아서 복붙하면 된다.

**(4) Web scraping**

%>%

HTML CSS

를 알고 다시 돌아와라.

**10. Data processing**

10.1. Dplyr : manipulating data

10.2. TIdyr : cleaning data

**10.1.1. Dplyr : manipulating data**

|  |
| --- |
| install.packages('dplyr')  library('dplyr')  항상 data frame을 먼저 하고 쉼표로 뒤에!!!!  **filter**( ) # subset과 유사  > filter(flights, month ==11, day ==3, carrier=='AA')  **arrange**( ) # filter랑 유사한데, order를 추가적으로!  > arrange(flights, year, month, day, arr\_time)  **slice**( ) # 1~10번째 row까지  > slice(flights, 1:10)  **select**( ) # 원하는 열만 select할 수 있다  > select(flights, carrier, arr\_time, month)  > **distinct**(select (flights, carrier) )  **rename**( ) # 열 명칭 바꾸기  > rename(flights, airline\_carrier = carrier)  **mutate**( ) # 새로운 열을 만들기  > **mutate**(flights, new\_col = arr\_time - dep\_delay)  transmute( ) # 새로운 열만 가져오기  > **transmute**(flights, new\_col = arr\_time - dep\_delay)  **summarise**( )  > summarise(flights, avg\_airtime = mean(air\_time, na.rm=TRUE))  **sample\_n**( ) and **sample\_frac**( )  > sample\_n(fligths, 10)  sample\_frac(flights, 0.1)  - pipe operator  df **%>%** filter(mpg>20) %>% sample\_n(size=5) %>% arrange(desc(mpg)) |

(1) filter( )

filter(flights, month ==11, day ==3, carrier=='AA')

여기서는 &를 할 필요가 없다.

🡪 이거랑 subset이랑 매우 유사하다.

> subset(flights, month == 11 & day ==3 & carrier == 'AA')

혹은

flights[flights$month == 11 & flights$day ==3 & flights$carrier == ‘AA’ , ]

(2) arrange

> arrange(flights, year, month, day, arr\_time)

🡪 year 순서대로 하고, 그 안에서 month 순서대로 하고, 그 안에서 day 순서대로.. 등등

- 거꾸로 순서를 하고 싶으면, **desc**( )안에 변수를 넣기

> arrange(flights, year, month, day, desc(arr\_time))

(3) distinct

> distinct(select(flights, carrier))

🡪 원하는 열에 있어서 겹치지 않는 값만!

(4) summarise function # aggregate function을 집어넣는데 쓰인다.

> summarise(flights, avg\_airtime = mean(air\_time, na.rm=TRUE))

avg\_airtime

<dbl>

1 151.

🡪 mean 에서 na.rm가 쓰인 이유는 칸 중에서 NA라고 쓰인 것이 있으므로 remove하기 위해서다.

summarise(flights, total\_airtime = sum(air\_time, na.rm=T))

total\_airtime

<dbl>

1 49326610.

(5) sample\_n(df, n), sample\_frac(df, 0.1)

flight 함수에서 sample을 10개 가져와라. 혹은 10%를 가져와라 할 때 쓸 수 있다.

**10.1.2. pipe operator** %>%

- 필요성 : 만약에 여러가지 dplyr operation을 같이 해야하면, result <- arrange(sample\_n(filter(df, mpg>20), size=5), desc(mpg))

mpg cyl disp hp drat wt qsec vs am gear carb

1 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1

2 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1

3 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2

4 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1

5 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4

🡪 나중에 읽는 것이 너무 어렵다.

- 이렇게 해도 되긴 하다.

a <- filter(df, mpg>20)

b <- sample\_n(a, size = 5)

result <- arrange(b, desc(mpg))

🡪 그러나 data를 너무 많이 저장해야 하는 문제점이 있다.

- 형식

|  |
| --- |
| data %>% op1 %>% op2 %>% po3  🡪 df %>% filter(mpg>20) %>% sample\_n(size=5) %>% arrange(desc(mpg)) |

ex) 순서가 중요하다.

Use pipe operators to get the mean hp value for cars with 6 cylinders.

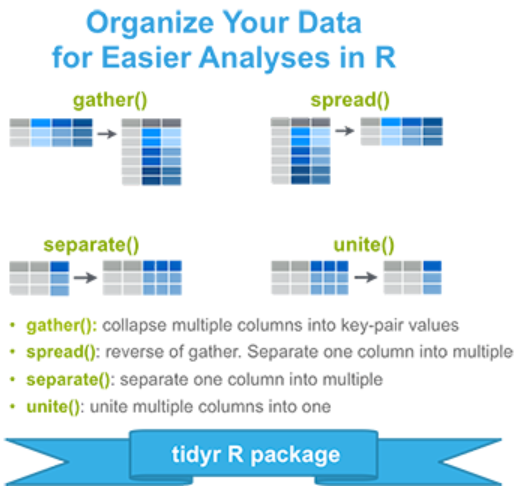
mtcars %>% filter(cyl == 6) %>% summarise(avr=mean(hp)) # avr = 이라고 하면 밑에도 avr로 뜬다.

avr

1 122.2857

뒤로 갈수록 나중에 해야할 것이다.

**10.2. tidyr**



- tidy한 데이터란? row가 observation이고 column이 value값인 경우이다.

install.packages('tidyr')

install.packages('data.table')

🡪 data.table은 data.frame에서 몇 개의 추가기능이 있다.

- data.table 패키지의 장점(vs data.frame) : speed, cleaner syntax

1. much faster and very intuitive by operations

2. You won't accidentally print out a huge data.frame with the need to press Ctrl-C, data.table prevents this sort of accident

3. faster and better file reading with fread

🡺 library(data.table)해야지 쓸 수 있다.

4. the package also provides a number of other utility functions, like %between% or rbindlist that make life better

5. pretty much faster for a lot of basic operations, since a lot of data.frame operations copy the entire thing needlessly

|  |
| --- |
| install.packages('tidyr')  install.packages('data.table')  library(tidyr)  library(data.table) 🡪 fread ㄴㄴ가능케 해줌  **gather**()  > gather(df, Quarter, Revenue, Qtr1:Qtr4)  > df %>% gather(Quarter, Revenue, Qtr1:Qtr4)  > stocks.gathered <- stocks %>% gather(stock,price, X:Z)  > stocks.gathered <- stocks %>% gather(stock,price, X,Y,Z)  **spread**()  > stocks.gathered %>% spread(stock,price)  **separate**() ## 와!!!! split이랑 같네 ㅎㅎ  > separate(df, new.col, c('ABC', 'XYZ'))  > separate(data = df, col = new.col, into = c('ABC', 'XYZ'), sep=’-‘)  **unite**()  > unite(df.sep, col = joined, sep = ‘---‘ )  이렇게 하면 된다. |

**(1) gather**

- 목적 : collapse multiple columns into key-pair values.

- time series 데이터인 경우에 요긴하다.

> gather(df, Quarter, Revenue, Qtr1:Qtr4) 로 합친다.

# Quarter, Revenue column 2개를 만든다. 각각 key와 value가 된다.

comp year Qtr1 Qtr2 Qtr3

1 1 1998 68.70925 78.004806 24.256423

2 1 1999 60.71792 27.266550 55.612019

3 1 2000 34.48091 77.051018 41.735422

4 2 1998 30.37194 5.737046 40.327460

5 2 1999 89.45848 9.399131 7.646704

6 2 2000 88.05125 42.528934 17.351246

7 3 1998 22.00249 56.360101 49.985100

8 3 1999 19.54425 63.383346 27.255326

9 3 2000 39.89005 46.892030 66.299752

위의 data frame이 이렇게 바뀐다.

comp year Quarter Revenue

1 1 1998 Qtr1 68.709254

2 1 1999 Qtr1 60.717925

3 1 2000 Qtr1 34.480906

4 2 1998 Qtr1 30.371941

5 2 1999 Qtr1 89.458475

6 2 2000 Qtr1 88.051245

7 3 1998 Qtr1 22.002488

8 3 1999 Qtr1 19.544250

9 3 2000 Qtr1 39.890048

10 1 1998 Qtr2 78.004806

11 1 1999 Qtr2 27.266550

12 1 2000 Qtr2 77.051018

13 2 1998 Qtr2 5.737046

14 2 1999 Qtr2 9.399131

15 2 2000 Qtr2 42.528934

16 3 1998 Qtr2 56.360101

17 3 1999 Qtr2 63.383346

18 3 2000 Qtr2 46.892030

19 1 1998 Qtr3 24.256423

20 1 1999 Qtr3 55.612019

21 1 2000 Qtr3 41.735422

22 2 1998 Qtr3 40.327460

23 2 1999 Qtr3 7.646704

- time X Y Z

1 2009-01-01 -0.61295744 -1.8183481 4.8008886

2 2009-01-02 0.81020344 0.6002111 3.6628067

3 2009-01-03 0.04919881 1.3301688 -0.2831237

4 2009-01-04 -0.68299972 -1.2666353 5.4649377

5 2009-01-05 -0.94622569 -0.2347791 -0.5153584

6 2009-01-06 -0.10227209 0.7425656 5.6275350

7 2009-01-07 1.14475106 -0.1184269 1.3111036

8 2009-01-08 -2.02033263 -0.5186714 -4.4803051

9 2009-01-09 -1.17570496 0.1230590 8.8044887

10 2009-01-10 1.68464577 -0.2736786 -0.7562996

> stocks

time X Y Z

1 2009-01-01 -0.61295744 -1.8183481 4.8008886

2 2009-01-02 0.81020344 0.6002111 3.6628067

3 2009-01-03 0.04919881 1.3301688 -0.2831237

4 2009-01-04 -0.68299972 -1.2666353 5.4649377

5 2009-01-05 -0.94622569 -0.2347791 -0.5153584

6 2009-01-06 -0.10227209 0.7425656 5.6275350

7 2009-01-07 1.14475106 -0.1184269 1.3111036

8 2009-01-08 -2.02033263 -0.5186714 -4.4803051

9 2009-01-09 -1.17570496 0.1230590 8.8044887

10 2009-01-10 1.68464577 -0.2736786 -0.7562996

stocks.gathered <- stocks %>% gather(stock,price, X:Z)

🡪 X, Y, Z로 쉼표로 할 수도 있다.

time stock price

1 2009-01-01 X -0.61295744

2 2009-01-02 X 0.81020344

3 2009-01-03 X 0.04919881

4 2009-01-04 X -0.68299972

5 2009-01-05 X -0.94622569

6 2009-01-06 X -0.10227209

7 2009-01-07 X 1.14475106

8 2009-01-08 X -2.02033263

9 2009-01-09 X -1.17570496

10 2009-01-10 X 1.68464577

11 2009-01-01 Y -1.81834812

12 2009-01-02 Y 0.60021107

13 2009-01-03 Y 1.33016884

**(2) 반대로 하려면**

stocks.gathered %>% spread(stock,price)

**(3) separate( )**

- 형태 :

separate(data = df, col = new.col, into = c('ABC', 'XYZ', sep = ‘-‘))

🡪 dataframe 넣고, separate하고 싶은 column 지정하고, 뭐랑 뭐로 쪼갤지 보고, 쪼갤 사이의 기호 넣고 하면 된다. alphanumeric character가 아니면 다 그렇게 할 수 있다.

df <- data.frame(new.col=c(NA, "a.x", "b.y","c.z"))

>> new.col

1 <NA>

2 a.x

3 b.y

4 c.z

> separate(df, new.col, c('ABC', 'XYZ'))

ABC XYZ

1 <NA> <NA>

2 a x

3 b y

4 c z

- 3개짜리도 쪼갤 수 있다.

separate(data = df, col = new.col, into = c('ABC', 'XYZ', 'K1'))

(4) 다시 반대로 unite

> unite(df.sep, col = joined, sep = ‘---‘ )

**11. Data visualization (ggplot2)**

|  |
| --- |
| Layer를 더하는 것이 ggplot2이다.  Main idea is adding layers together  3 layers  3. Geometrics  2. Aesthetics : which column you want  1. Data  Next 3 layers  : Facets, statistics, coordinates  🡪 customize 가능  Final layer  Themes |

<추가 layer>

library(ggplot2) # library

pl <- ggplot(data = mtcars, aes(x=mpg, y=hp

pl + geom\_point( ) # data output

- facet ???

pl + facet\_grid(cyl ~.)

- stat # 부드럽게

pl + facet\_grid(cyl ~.) + stat\_smooth( )

- coordinate ???

pl2 <- pl + facet\_grid(cyl ~.) + stat\_smooth( )

pl2 + coord\_caresian(xlim = c(20, 25))

- theme plot

pl2 + coord\_caresian(xlim = c(20, 25)) + theme\_bw( )

**11.1. Histograms**

- cheat sheet

<https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

|  |
| --- |
| install.packages('ggplot2')  install.packages('ggplot2movies')  library(ggplot2)  library(ggplot2movies)  1. Data layer / aesthetics layer  pl <- ggplot(movies, aes(x=rating))  pl + geom\_histogram( )  2. Geometric layer  - Geometrices 꾸미기  pl + geom\_histogram(binwidth = 0.1, color ='cyan1', fill='cyan1', alpha=0.5)  pl2<-pl + geom\_histogram(binwidth = 0.1, aes(**fill=..count..**))    - x, y label  pl3 <- pl2 + xlab('Movie Rating') + ylab('Count')  - title  pl3 + ggtitle("My TITLE")  + theme(plot.title = element\_text(hjust = 0.5))  # 가운데로 오게끔 하는 것. |

- bins : 몇 개인지

- binwidth

- color : 테두리임

- fill = 색깔

- alpha : 0이 투명, 1이 불투명

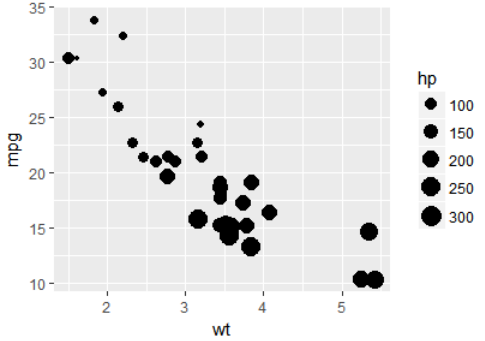
**11.2. scatter plots**

|  |
| --- |
| library(ggplot2)  library(data.table)  df <- fread('Economist\_Assignment\_Data.csv', **drop=1**)  # read.csv( ) 로 할 수 있다. drop=1은 사용할 수 없다.  # 대신 이렇게 했다.  df <- data.frame(csv)  df1<- df[,-1]  # Data & Aesthetics (Layer 1, 2)  pl <- ggplot(data = df, aes(x=wt, y=mpg) )  # Geometry (Layer 3)  pl + geom\_point()  pl + geom\_point((aes(size=hp)))  pl + geom\_point((aes(shape=factor(cyl))), size =5)  # hollow circle  geom\_point(shape=1) # Use hollow circles  # 슈퍼 중요 : geometry layer에 aes layer를 넣게 되면 산점도에서 3개의 data feature를 표현할 수 있게 된다는 것이다.  pl + geom\_point((aes(size=hp)))  pl + geom\_point((aes(shape=factor(cyl))), size =5)  # color랑 size를 동시에  pl + geom\_point((aes(shape=factor(cyl),color=factor(cyl))), size =5)  # hex color도 가능  <http://www.color-hex.com/>  pl + geom\_point(color =#94d8a6 , size =5)  # color gradient도 가능  pl2 <-pl + geom\_point(size=4, aes(color=hp))  pl2 + **scale\_color\_gradient**(low="blue", high='red')  # geom\_smooth(colour = ‘red’)  ggplot(dat, aes(x=xvar, y=yvar)) +  geom\_point(shape=1) + # Use hollow circles  geom\_smooth(method=lm, # Add linear regression lin se=FALSE) # Don't add shaded confidence region  # se = FALSE 하면 신뢰구간이 없어진다. standard error(95%)  # method = ‘lm’ 은 linear model을 의미  # loess  Local Polynomial Regression Fitting  #x 축 label  pl5<- pl4 + **scale\_x\_continuous**(name="Corruption Percentions Index, 2011 (10=least corrupt)", limits =c(1,10), breaks=1:10)  pl5  # y축 label  pl6 <- pl5 +**scale\_y\_continuous**(name="Human Development Index, 2011(1=best)", limits = c(0.2,1), breaks=seq(0.1,1,0.1))  # title  pl7 <- pl6 + **ggtitle**("Corruption and Human development") + theme(plot.title = element\_text(hjust = 0.5)) + theme(axis.text.x = element\_text(face = "italic"))  # 각 point labeling  pl3 <- pl2 + **geom\_text**(**aes**(label = Country),  color = "gray20",  data = subset(df, Country %in% pointsToLabel),  check\_overlap = T)  🡪 너무 많으니까 수를 줄여줌. |

- size : 점의 사이즈

- 중요 : geom\_point( ) 안에다가 aes를 하면 그 데이터셋 내의 다른 feature로 변화를 줄 수 가 있다.

pl + geom\_point((aes(size=hp)))

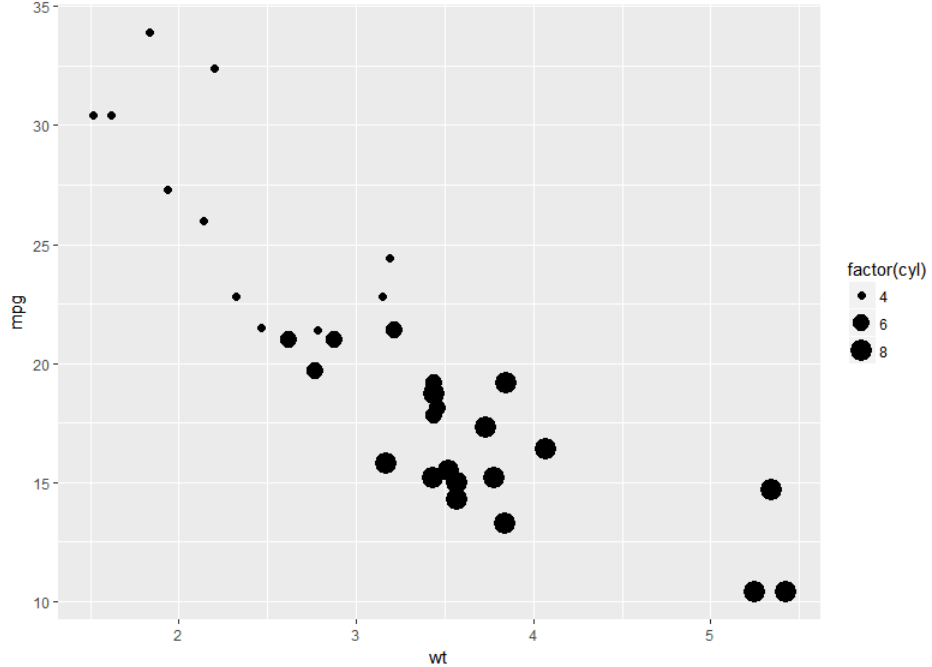


pl + geom\_point((aes(size=cyl)))

🡪 cylinder는 2, 4, 6, 8 짝수 단위이다.

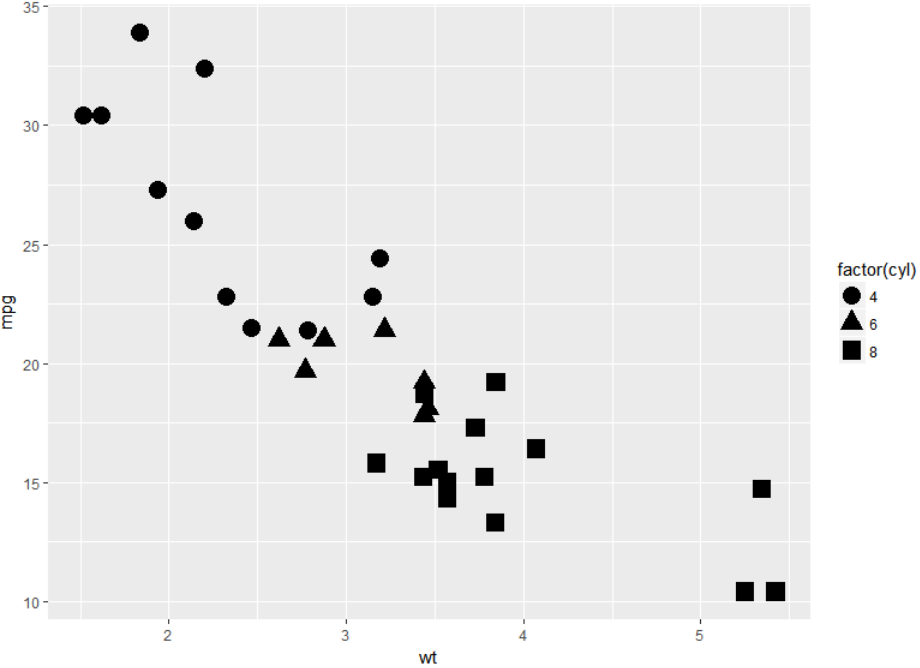
pl + geom\_point((aes(size=factor(cyl))))

이 때 factor를 사용하면 된다. 그러면 categorical type feature가 된다.



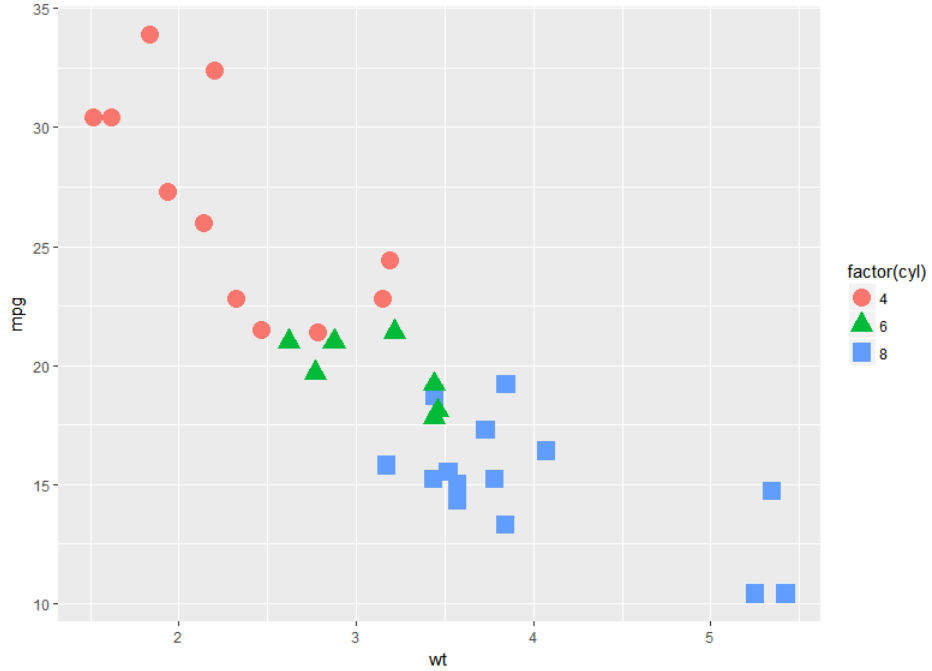
- 대박 : categorical은 shape로 구분하는게 좋지.

pl + geom\_point((aes(shape=factor(cyl))), size =5)



- shape랑 color를 둘 다 aes 안에 넣어보자.

pl + geom\_point((aes(shape=factor(cyl),color=factor(cyl))), size =5)

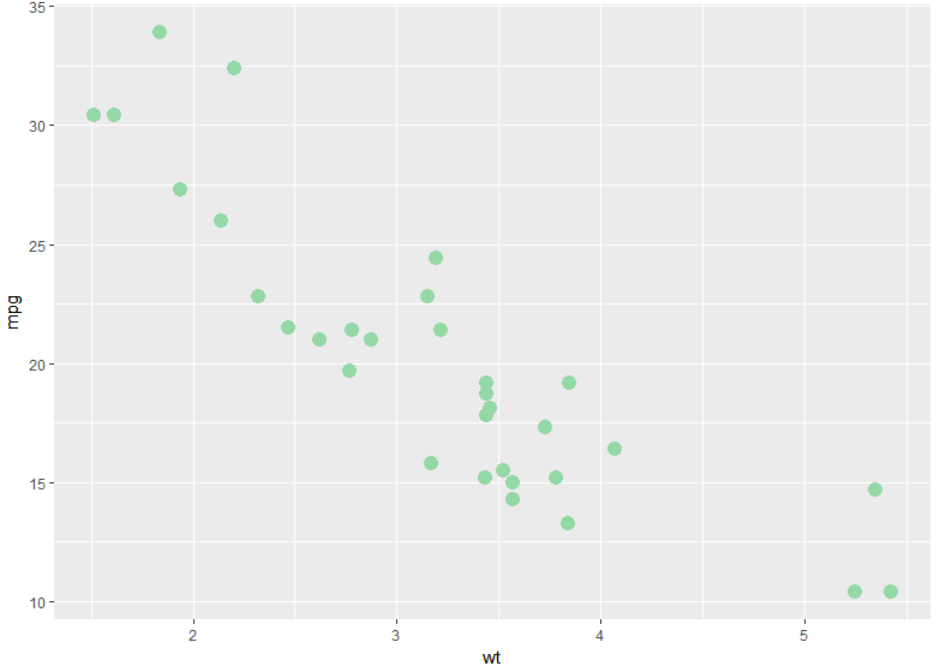


- color : hex color codes

<http://www.color-hex.com/>

color = #94d8a6 복붙하면 된다.

pl + geom\_point(color =#94d8a6 , size =5)

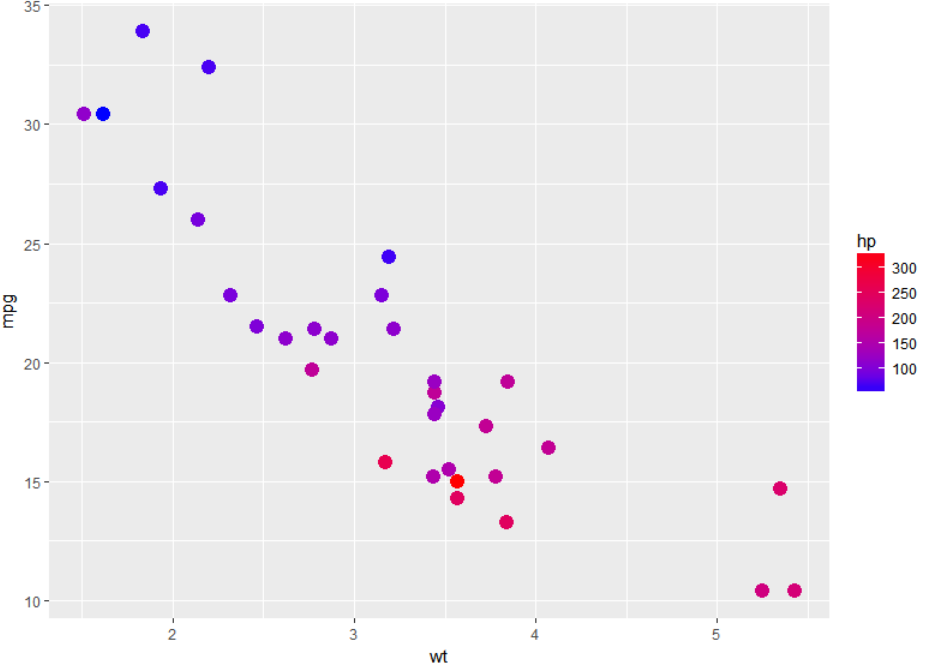


- color gradient을 사용해보자.

pl2 <-pl + geom\_point(size=4, aes(color=hp))

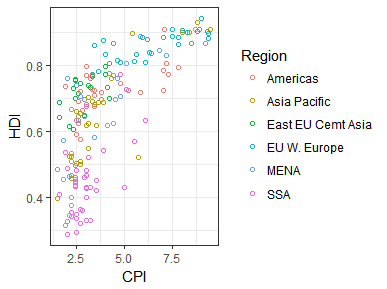
pl2 + **scale\_color\_gradient**(low="blue", high='red')

low와 high를 지정할 수 있다!!!!!!! 대박!!!



- 동그라미 안의 채움을 없애고 싶을 때

pl <- ggplot(df1, aes(x=CPI, y=HDI)) + geom\_point(aes(color=Region), fill=NA, shape=1)



- geom\_smooth( )

pl2 <- pl + geom\_smooth(aes(group=1))

🡪 group 1에 따라서 폭이 달라지는듯. group 1이 뭐지???

ggplot(dat, aes(x=xvar, y=yvar)) +

geom\_point(shape=1) + # Use hollow circles

geom\_smooth(method=lm, # Add linear regression lin se=FALSE) # Don't add shaded confidence region

# se = FALSE 하면 신뢰구간이 없어진다. standard error(95%)

# method = ‘lm’ 은 linear model을 의미

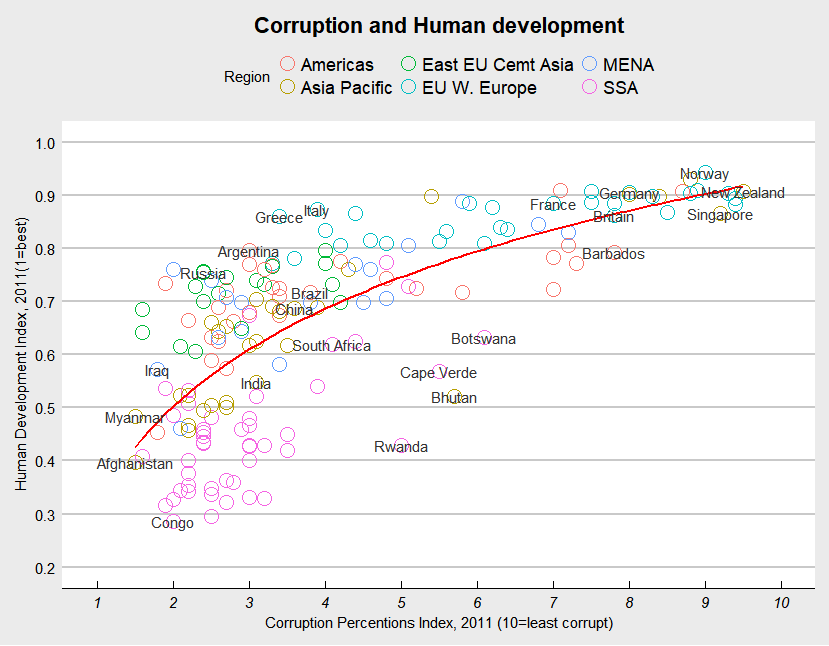
# loess

Local Polynomial Regression Fitting

- x축 scale

pl5<- pl4 + scale\_x\_continuous(name="Corruption Percentions Index, 2011 (10=least corrupt)", limits =c(1,10), breaks=1:10)

- Economist project

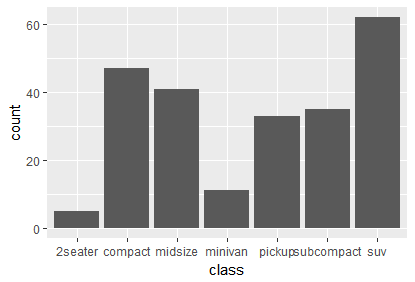


|  |
| --- |
| library(ggplot2)  library(data.table)  setwd("c:/Users/Owner/Desktop")  csv <- fread('Economist\_Assignment\_Data.csv')  df <- data.frame(csv)  df1<- df[,-1]  # df <- fread('Economist\_Assignment\_Data.csv', **drop=1**)  # read.csv( ) 로 할 수 있다. drop=1은 사용할 수 없다.  pl <- ggplot(df1, aes(x=CPI, y=HDI)) + geom\_point(aes(color=Region), fill=NA, shape=1, size = 5)  # ggplot( )안에 aes에서 color를 넣을 수도 있다.  pl <- ggplot(df1, aes(x=CPI, y=HDI, color=Region)) + geom\_point(fill=NA, shape=1, size = 5)  pl2 <- pl + **geom\_smooth**(aes(**group=1)**, method='lm', formula = y~log(x), se=F, color='red')  # I don't know what those methods are  # linear regression에서 더 배운다고 함.  pointsToLabel <- c("Russia", "Venezuela", "Iraq", "Myanmar", "Sudan", "Afghanistan", "Congo", "Greece", "Argentina", "Brazil", "India", "Italy", "China", "South Africa", "Spane", "Botswana", "Cape Verde", "Bhutan", "Rwanda", "France", "United States", "Germany", "Britain", "Barbados", "Norway", "Japan", "New Zealand", "Singapore")  pl3 <- pl2 + **geom\_text**(**aes**(label = Country), color = "gray20", data = subset(df, Country %in% pointsToLabel),check\_overlap = T)  pl3  pl4 <- pl3 + theme\_economist\_white()  pl4  #x 축 label  pl5<- pl4 + **scale\_x\_continuous**(name="Corruption Percentions Index, 2011 (10=least corrupt)", limits =c(1,10), breaks=1:10)  pl5  # y축 label  pl6 <- pl5 +**scale\_y\_continuous**(name="Human Development Index, 2011(1=best)", limits = c(0.2,1), breaks=seq(0.1,1,0.1))  # title  pl7 <- pl6 + **ggtitle**("Corruption and Human development") + theme(plot.title = element\_text(hjust = 0.5)) + theme(axis.text.x = element\_text(face = "italic"))  pl7 |

**11.3. Barplots**

- 변수 1개인데 categorical data + count

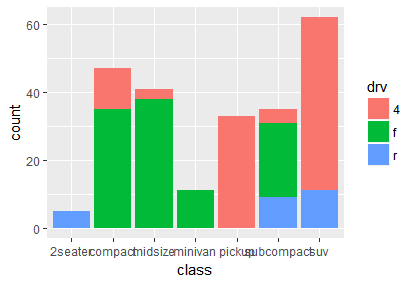
- 히스토그램과의 차이점은 x축에 categorical data가 온다는 것이다. 히스토그램은 사이의 간격이 없었지.



|  |
| --- |
| pl <- ggplot(df, aes(x=class))  pl + geom\_bar(aes(fill=drv))  # position = dodge 🡪 쌓아놓지 않고 옆으로 비껴가게 배열  # legend title 바꾸는 방법  p2 + guides(fill=guide\_legend(title="cyl")) |

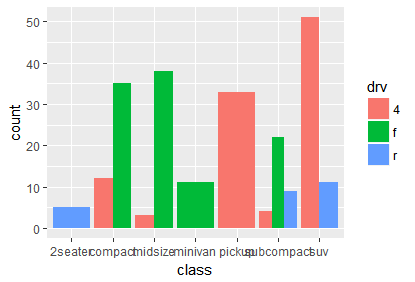
- 여기다가 drv 정보를 넣으려면 어떻게 해야할까?

pl + geom\_bar(aes(fill=drv))



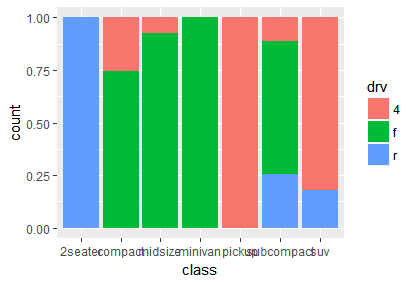
- position = dodge

pl + geom\_bar(aes(fill=drv), position = 'dodge')



- position = fill

pl + geom\_bar(aes(fill=drv), position = 'fill')

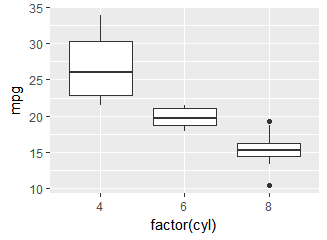


**11.4. Box plots**

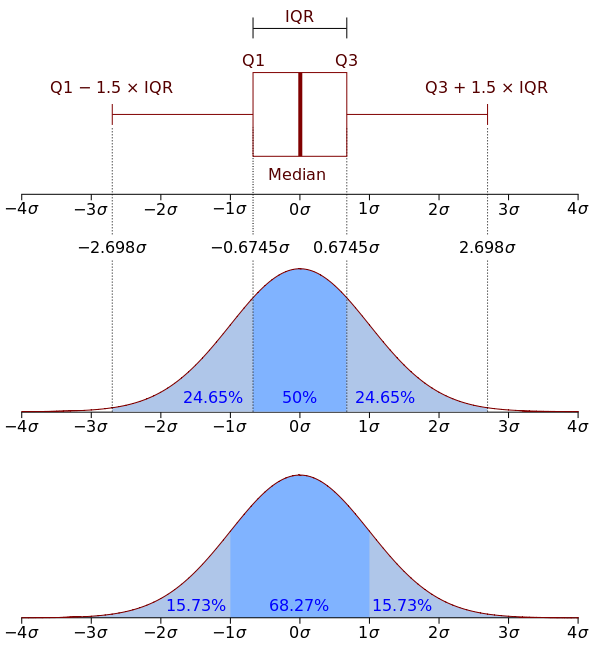
|  |
| --- |
| pl <- ggplot(df, aes(x=factor(cyl), y=mpg))  pl + geom\_boxplot( ) + coord\_flip( )  # geometry layer 추가  pl + geom\_boxplot(aes(fill= factor(cyl))) |

# 만약에 factor를 안 넣으면 에러가 뜬다. categorical data이기 때문에 factor( )를 해줘야 한다.

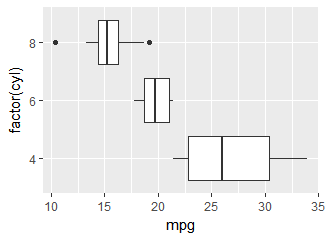
pl <- ggplot(df, aes(x=factor(cyl), y=mpg))



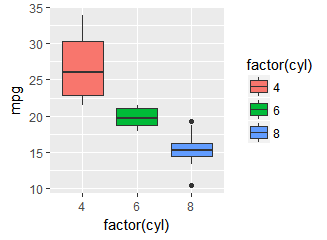
- wisker는 1.5 IQR을 Q1에서 빼거나, Q3에서 더한 것을 의미한다.



- flip하면 옆으로 뒤 바뀐다.

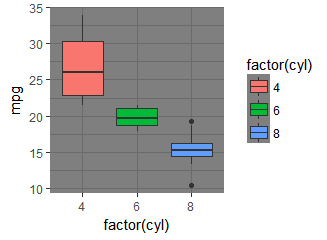


- pl + geom\_boxplot(aes(fill= factor(cyl)))



- theme

pl + geom\_boxplot(aes(fill = factor(cyl)))+ theme\_dark()

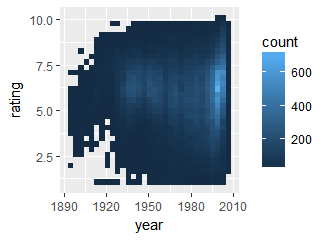


**11.5. 2 variable plotting**

|  |
| --- |
| pl <- ggplot(movies, aes( x = year, y=rating))  pl + geom\_bin2d( )  # 색 바꾸기  pl + geom\_bin2d() + scale\_fill\_gradient(high = 'red', low='green')  # bin width 바꾸기  pl<-ggplot(movies, aes(x=year, y=rating))  pl + geom\_bin2d(binwidth=c(1,0.1)) + scale\_fill\_gradient(high = 'red', low='green')    # hexbin을 해보자.  install.package(“hexbin”)  library(hexbin)  pl <- ggplot(movies, aes(x=year, y=rating))  pl2 <- pl + geom\_hex( )  - geom\_density2d()  pl <- ggplot(movies, aes(x=year, y=rating))  pl2 <- pl + geom\_density2d( ) |

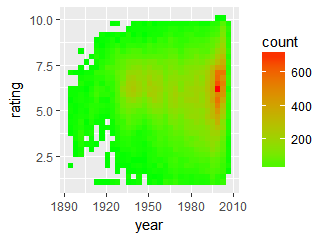
pl <- ggplot(movies, aes( x = year, y=rating))

pl + geom\_bin2d( )



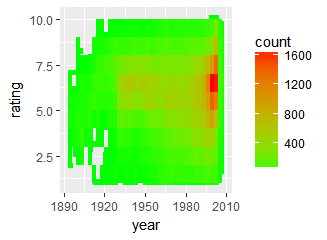
- 색깔 바꾸기

pl + geom\_bin2d() + scale\_fill\_gradient(high = 'red', low='green')



- bin width를 바꾸기

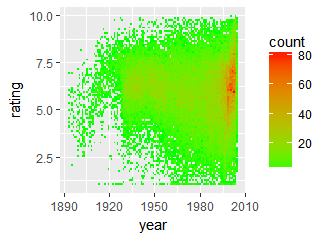
pl + geom\_bin2d() + scale\_fill\_gradient(high = 'red', low='green') + geom\_bin2d(binwidth=c(3,1))



- binwidth 최적화

pl<-ggplot(movies, aes(x=year, y=rating))

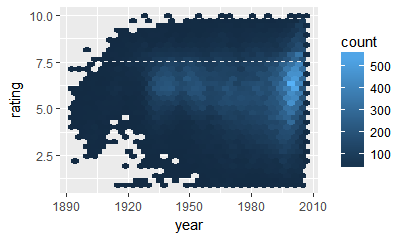
pl + geom\_bin2d(binwidth=c(1,0.1)) + scale\_fill\_gradient(high = 'red', low='green')



- hexbin을 시도해보자.

pl <- ggplot(movies, aes(x=year, y=rating))

pl2 <- pl + geom\_hex()



- 색깔도 바꿔보자.

pl <- ggplot(movies, aes(x=year, y=rating))

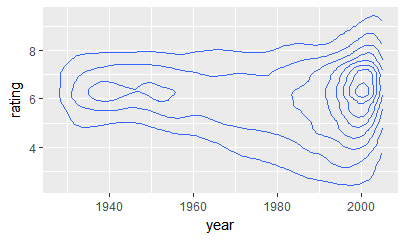
p2 <- pl + geom\_hex() + scale\_fill\_gradient(high = 'red', low='blue')

p2

- geom\_density2d()

pl <- ggplot(movies, aes(x=year, y=rating))

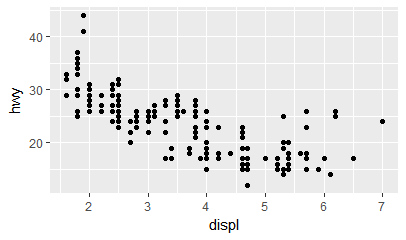
pl2 <- pl + geom\_density2d( )



**11.6. coordinate and faceting**

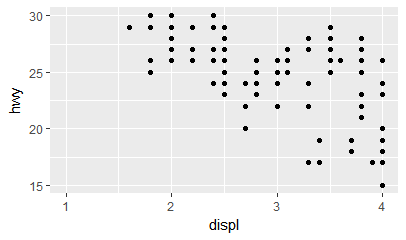
|  |
| --- |
| pl <- ggplot(mpg, aes(x=displ, y=hwy)) + geom\_point()  # coordinate 조정  pl2 <- pl + **coord\_cartesian**(xlim = c(1, 4), ylim = c(15, 30))  # aspect ratio 고정  pl3 <- pl + **coord\_fixed**(ratio =1/3)  # facet  pl + **facet\_grid**(. ~ cyl) |

pl <- ggplot(mpg, aes(x=displ, y=hwy)) + geom\_point()



pl <- ggplot(mpg, aes(x=displ, y=hwy)) + geom\_point()

pl2 <- pl + coord\_cartesian(xlim = c(1, 4), ylim = c(15, 30))



- 아이에 aspect ratio를 고정할 수도 있다.

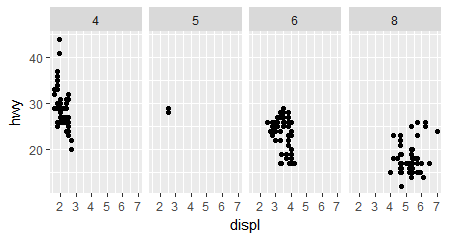
pl2 <- pl + coord\_fixed(ratio = 1/3)

pl3 <- pl + coord\_fixed(ratio =1/3)

- facet

pl <- ggplot(mpg, aes(x=displ, y=hwy)) + geom\_point()

pl + facet\_grid(. ~ cyl)

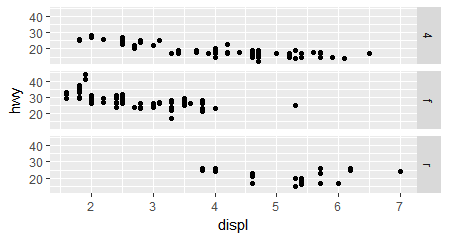


facet\_grid ( y축 ~ x 축 )

. 은 모든 것을 말한다.

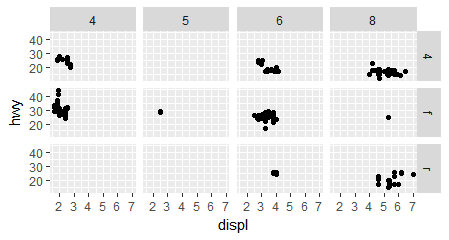
- 만약에 drv을 y축 방향으로 facet(분할)하고 싶으면

pl + facet\_grid(drv ~ . )로 하면 된다.



- 만약에 두 측면으로 다 facet하고 싶으면

pl + facet\_grid(drv ~ cyl)



**11.7. Themes**

|  |
| --- |
| # package!!!!  install.packages("ggthemes")  library(ggthemes)  # 전체  theme\_set(theme\_minimal())  # 개별적으로 적용  pl + theme\_dark() |

- Theme을 전체 다 바꾸고 싶을때는

theme\_set(theme\_~~~~))

**11.8. Plotly**

install.packages("plotly")

- interactive visualizations : 모든 것을 다 만들 수 있다!

- 사이트 : <https://plot.ly/ggplot2/>

- Cheat sheet 같은 것

<http://ropensci.github.io/plotly-test-table/tables/0e3d5ca144d27d8416318824c1b6ec1421a51045/index.html>

|  |
| --- |
| pl <- ggplot(mtcars, aes(mpg, wt)) + geom\_point()  gpl <- ggplotly(pl) |

**12. Capstone project(Money Ball)**

- 문제 : 2002년!! Statistics such as **stolen bases**, **runs batted in**, and **batting average**, typically used to gauge players, are relics of a 19th-century view of the game and the statistics available at that time.

- 해결 : 새로운 척도!!

1. Batting average (타율)

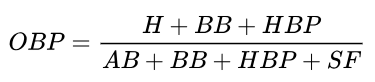
batting average (BA) is defined by the number of hits divided by at bats.

AVG = H / AB

2. On-base percentage (출루율)

- 정의 : on-base percentage (OBP; sometimes referred to as on-base average/OBA, as the statistic is rarely presented as a true percentage) is a statistic generally measuring how frequently a batter reaches base.[1]

- 계산 :



H Hits

BB Base on Balls

HBP Hit by pitch

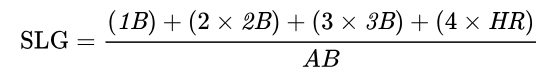
AB At Bats

SF Sacrifice flies

3. Slugging percentage (장타율)

- 정의 : slugging percentage (SLG) is a measure of the batting productivity of a hitter. It is calculated as total bases divided by at bats,

- 계산 :



- 코드

|  |
| --- |
| library(ggplot2)  setwd('C:\\Users\\Owner\\Desktop\\R 공부시작(18. 3. 23~)\\jose\\R-Course-HTML-Notes\\R-for-Data-Science-and-Machine-Learning\\Training Exercises\\Capstone and Data Viz Projects\\Capstone Project')  getwd()  #install.packages('dplyr')  #install.packages('tidyr')  library(dplyr)  library(tidyr)  library(data.table)  batting <- read.csv('batting.csv')  salary <- read.csv('Salaries.csv')  #str(batting)  #summary(batting)  #head(batting)  head(batting$`2B`)  batting$'2B'  colnames(batting)  batting <- mutate(batting, BA = H/AB)  batting <- mutate(batting, OBP = (H+BB+HBP)/(AB+BB+HBP+SF))  batting <- mutate(batting, SLG = (H+X2B+2\*X3B+3\*HR)/AB)  colnames(batting)  # merge the salary data with the batting data  str(salary)  summary(salary)  summary(batting)  # remove the batting data that occured before 1985.  batting <- subset(batting, yearID >= 1985)  colnames(batting)  colnames(salary)  ?merge  total <- merge(batting, salary, c('yearID', 'playerID'))  summary(total)  colnames(total)  # grab 2001 data of 3 players  lost\_players <- subset(total, playerID %in% c('giambja01','damonjo01','saenzol01')) %>% filter(yearID==2001)  # AB sum of them -> 1469  AB\_sum <- sum(lost\_players$AB)  # OBP mean of them -> 1.091606  OBP\_mean <- sum(lost\_players$OBP)  # only salary  sal\_lost <- subset(total, playerID %in% c('giambja01','damonjo01','saenzol01')) %>% filter(yearID==2001) %>% select(playerID, salary)  sum(sal\_lost$salary)    # 3 conditions  # 1. The total combined salary of the three players can not exceed 15 million dollars.  # 2. Their combined number of At Bats (AB) needs to be equal to or greater than the lost players.  # 3. Their mean OBP had to equal to or greater than the mean OBP of the lost players  # get the 2001 data of total dataframe  total\_2001 <- subset(total, yearID=='2001')  nrow(total\_2001)  total\_2001\_15mil <- filter(total\_2001, salary < 15000000)  summary(total\_2001\_15mil)  nrow(total\_2001\_15mil)  arrange(total\_2001\_15mil, salary, AB, OBP)  # only show salary, AB, OBP  combo <- select(total\_2001\_15mil, playerID, salary, AB, OBP)  combo <- filter(combo, salary < 15000000)  summary(combo)  # order them  arrange(combo, desc(OBP), desc(AB), salary)  # visualization  library(ggplot2)  library(ggthemes)  # Histogram of salary  pl <- ggplot(combo, aes(x=salary))  pl2 <- pl + geom\_histogram(bins=80, aes(fill=..count..))  # labeling  pl2 <- pl2 + labs(x = "Salary", y = "Number of athletes") + scale\_x\_continuous(breaks=seq(200000,14000000,1000000))  pl3 <- pl2 + theme\_bw(base\_size = 12, base\_family = "Helvetica")  pl4 <- pl3 + ggtitle("Salary distribution") + theme(plot.title = element\_text(hjust = 0.5))  # coordinates  pl4 + theme(axis.text.x = element\_text(angle = 90, hjust = 1, size=10))    - 산점도  # scatter plot  sc <- ggplot(combo, aes(x=AB, y=OBP))  sc2 <- sc + geom\_point(alpha=0.5,aes(color = salary)) +scale\_color\_gradient(low='blue', high='red')  sc2 + ggtitle("AB vs OBP plot") + theme\_bw() + theme(plot.title = element\_text(hjust = 0.5))    # zoom  # zoom  zoom <- sc2 +scale\_x\_continuous(limits=c(500,700)) + scale\_y\_continuous(limits=c(0.25,0.5))  zoom + theme\_bw()  OB  - ggplotly로 하면 엄청 편하다!!1  zoom <- sc2 +scale\_x\_continuous(limits=c(500,700)) + scale\_y\_continuous(limits=c(0.25,0.5))  zoom1 <- zoom + theme\_bw() + **geom\_text(aes(label = playerID), size=1)**  gzoom1 <- ggplotly(zoom1)  gzoom1    berkmla01  gonzalu01  heltoto01  마우스를 올려놓으면 정보가 나온다!!  선택!!  굿잡!! |

|  |
| --- |
| **Minerva R statistical modeling** |

- probability distribution, hypothesis testing, regression modelling, multivariate analysis

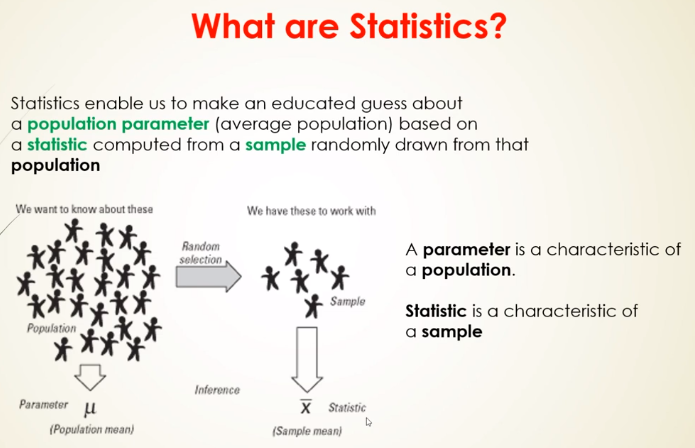
- 어떤 모델을 써야하는지 결정하는 것이 매우 중요하다.

- 통계학의 기초

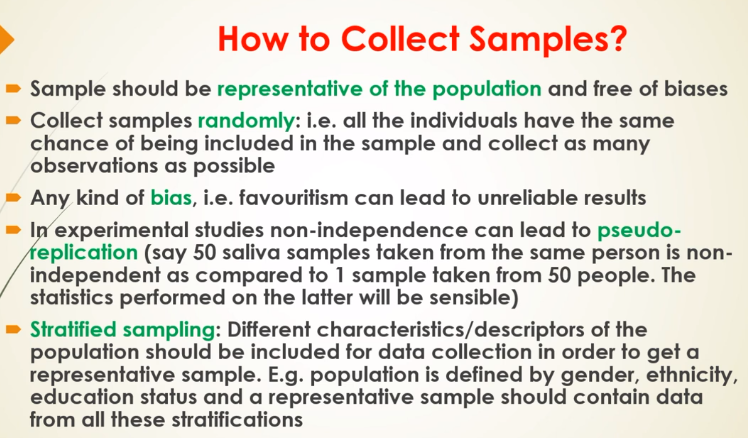
- 전수조사가 불가능하니까 sample을 추출한다.

- Representative sample이 중요하다.

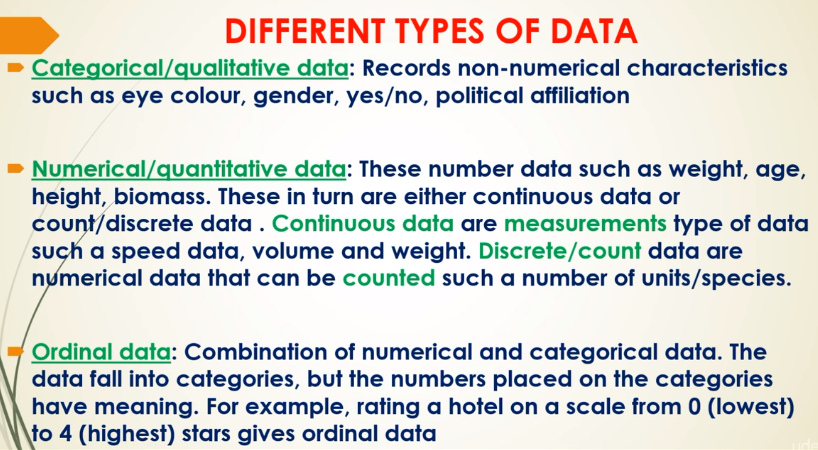
- 통계량으로 population을 표현한다.



- Sampling의 주의점 :



- Data의 종류



🡪 식물의 mass를 하면 어떨까? 어떻게 하면 벼의 growth를 측정할 수 있을까?

|  |
| --- |
| **R utilization** |

R에서 중요한 기능

**1. R markdown 기능 / knit 과 html publish**

- Knit document를 하면 html로 자신의 document를 볼 수 있다!!

- full proper document로 만들 수 있다!!!

: file - new file - **markdown** 하면

- 밑에 지저분한 거 지우고

- insert - R을 누르면 R code chunk가 들어간다.

```{r}

setwd('C:\\Users\\Owner\\Desktop\\R 공부시작(18. 3. 23~)\\Minerva')

```

이렇게 하면 바로 여기로 save가 된다.

🡪 knit을 클릭한 다음에 html로 저장을 하면 편하다!

knit을 할 때에는 안에 setwd, install.packages 내용을 모두 빼야한다.

**2. Importing data**

1) csv 🡪 읽어들이면 default로 dataframe이 된다. 그리고 sep = ‘,’가 디폴트이다.

2) txt

3) excel

4) website : 와 RCurl 개 편하다!!

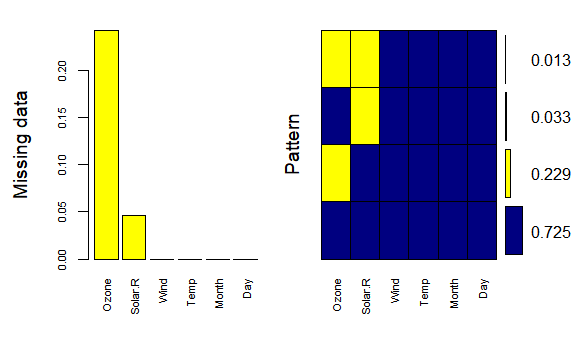
|  |
| --- |
| 1) csv  # header =T가 디폴트이다.  resp1 = read.csv('Resp1.csv', header=T)  head(resp1)  str(resp1)  # sep = ‘ ; ‘ 로 해야 읽히는 경우  winer1 = read.csv("winequality-red.csv", header=T, sep=';')  head(winer1)  2) txt  # txt도 마찬가지  resp2 = read.table('Resp2.txt', header=T)  head(resp2)  3) excel  # excel을 하는 경우  library(readxl)  dfb <- read\_excel('boston1.xls')  head(dfb)  🡺 클릭으로도 가능하다!! import dataset에서 From excel로 하면 바로 됨!  4) website 🡪 RCurl사용  #Using RCurl to read in csv data hosted online on github and other #sites  install.packages('RCurl')  library(RCurl)  data1= read.csv(text=getURL("https://raw.githubusercontent.com/sciruela/Happiness-Salaries/master/data.csv"))  head(data1)  # 와 대박이다!!    만약에 data 중에서 위에 막 설명 같은게 있으면 skip = 7 하면 된다.  data2=read.csv(text=getURL("https://raw.githubusercontent.com/opetchey/RREEBES/master/Beninca\_etal\_2008\_Nature/data/nutrients\_original.csv"), **skip=7**, header=T) |

**3. Indexing and subseting by conditions**

|  |
| --- |
| (1) Subseting columns  # subset 2 column  x = iris[, c("Sepal.Length", "Sepal.Width")]  or  vars <- c("Sepal.Length", "Sepal.Width")  x <- iris[vars]  # subset 1 column  # without turning it into vector  x = iris[, "Sepal.Length", drop=F]  # subset  df\_setosa <- subset(iris, iris$Species =='setosa')  df\_setosa  # filter  library(dplyr)  filter(iris, Species == 'setosa')  (2) Excluding columns  대박!!!!  vars <- **names(iris) %in% c('Species')**  nd <- iris[ **! vars**]  head(nd)  names(iris) %in% c(‘Species’)를 하면 이렇게 나온다.  [1] FALSE FALSE FALSE FALSE  [5] TRUE  즉 이거를 뒤집어서(!) 넣으면 되는 것이다.  # column number로 여러 개 없애는 법  nd <- iris[c(-3,-4)]  nd  🡪 왜 쉼표가 필요없는 걸까? 쉼표없으면 column만 집어줌. dataframe 형태를 유지하면서 |

**4. Data cleaning, removing missing values**

|  |
| --- |
| 1) 가장 simple한 방법  # remove rows containing NAs.  aq <- na.omit(airquality)  aq2 = airquality[complete.cases(airquality),]  2) 0으로 바꾸는 방법  aqty = airquality  aqty[is.na(aqty)] <- 0  3) mean으로 바꾸는 방법  meanOzone <- mean(airquality$Ozone, **na.rm**=T)  🡪 na.rm : mean을 계산하기 전에 NA를 없애주는 기법  그 다음에 meanOzone을 넣어준다.  qqty.fix <- ifelse(is.na(airquality$Ozone), meanOzone , airquality$Ozone)  ifelse의 매우 어려운 기법이다.  ifelse (test, yes, no)  True면 meanOzone을 출력하고, False면 airquality$Ozone을 넣어라는 뜻이다.  4) NA의 패턴을 알고 나서 replace하는 방법  - 2개의 패키지를 다운받아야 한다 : mice & vim  5) Predictive mean mapping  library(mice)  aqty2 = airquality  md.pattern(aqty2)  Wind Temp Month Day Solar.R Ozone  111 1 1 1 1 1 1 0  35 1 1 1 1 1 0 1  5 1 1 1 1 0 1 1  2 1 1 1 1 0 0 2  0 0 0 0 7 37 44  🡪 111개 none NA  library(VIM)  mp <- aggr(aqty2, col=c('navyblue', 'yellow'),  numbers = TRUE, sortVars = TRUE,  labels = names(aqty2), cex.axis=.7,  gap=3, ylab=c("Missing data", "Pattern")  )  # impute  # 500 iterations of predictive mean mapping for imputing  # 5 data sets, 50 iterations of predictive mean mapping for imputing  im\_aqty <- mice(aqty2, m=5, maxit = 50, method = 'pmm', seed=500)  summary(im\_aqty)  head(im\_aqty$imp$Ozone)  completedData <- complete(im\_aqty, 1)  # data set 1으로 change  head(completedData) |



🡪 NA가 yellow로! 72%는 NA가 없고, 22%는 Ozone에서 NA value가 있다.

!!!! 와!!! 대박대박!!!

|  |
| --- |
| mtcars$mpg [ is.na(mtcars$mpg) ] <- mean (mtcars$mpg) |

- Exploratory Data Analysis (EDA)

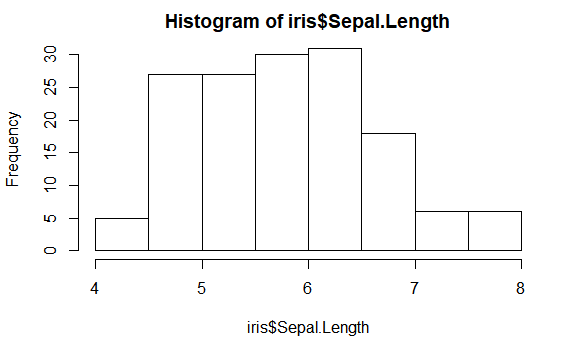
기본 함수 사용

(1) 변수 1개 사용

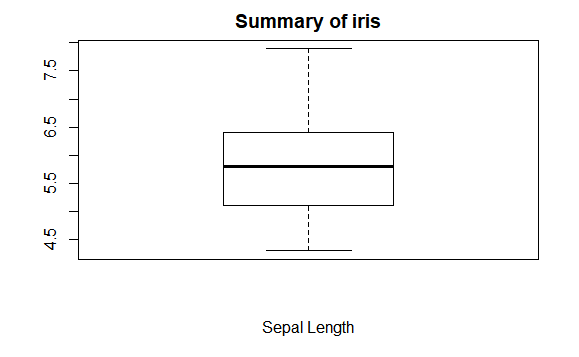
# (1) can explore the distribution of 1 variable

# (2) can explore the relationship between 2 variables.

hist(iris$Sepal.Length)



boxplot(iris$Sepal.Length, main="Summary of iris", xlab= "Sepal Length")



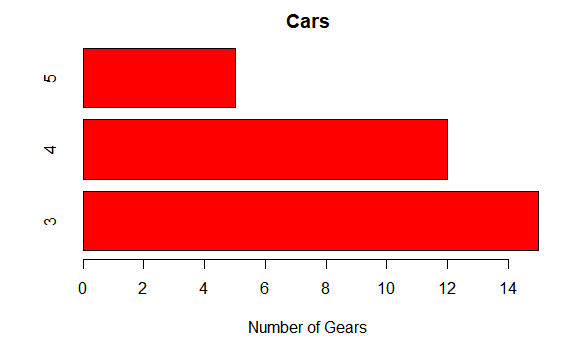
#plotting categorical or count variables

counts <- table(mtcars$gear)

counts

barplot(counts, main="Cars", xlab='Number of Gears')

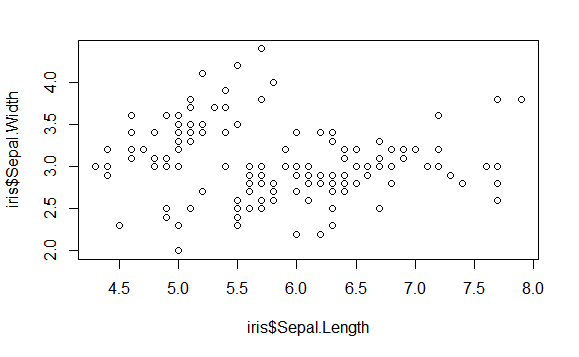
barplot(counts, main='Cars', xlab='Number of Gears', horiz=TRUE, col='red')



(2) 변수 2개 분석

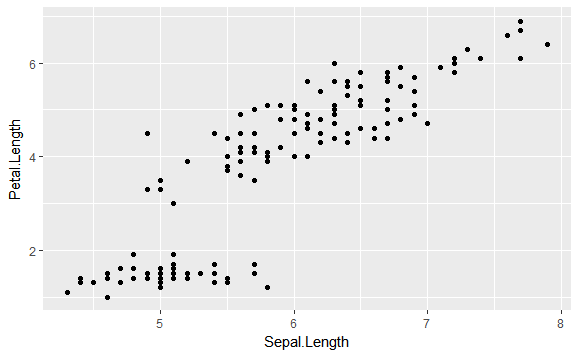
#relationship bw 2 quantitative variables

plot(iris$Sepal.Length, iris$Sepal.Width)

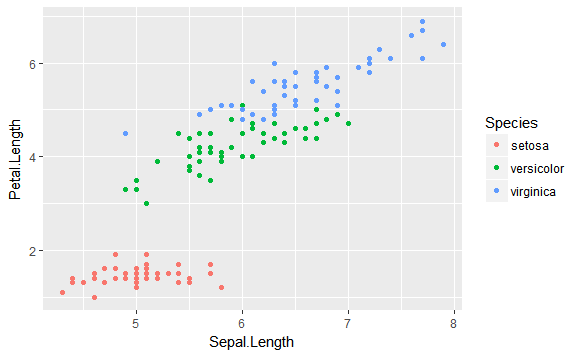


(2-2) qplot으로 진화!

library(ggplot2)

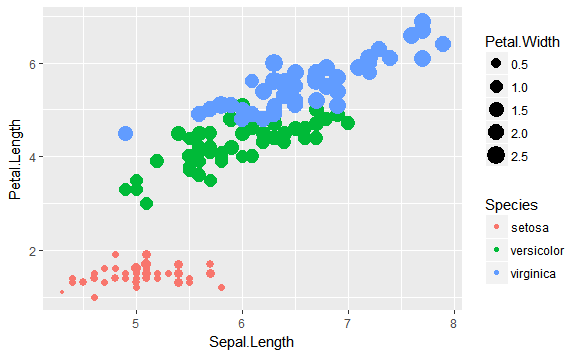


qplot(Sepal.Length, Petal.Length, data=iris, **color=Species**)



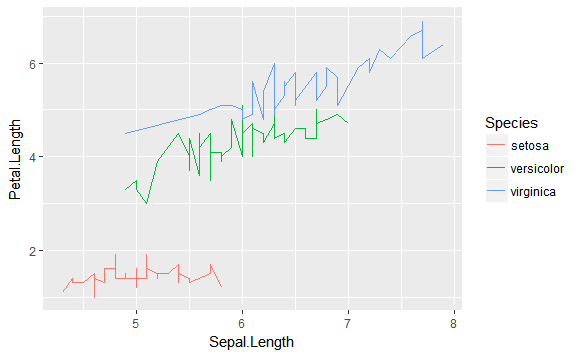
- petal width 정보까지

qplot(Sepal.Length, Petal.Length, data=iris, color=Species, size=Petal.Width)



- Geom = ‘line’

qplot(Sepal.Length, Petal.Length, data=iris, geom= "line", color=Species)

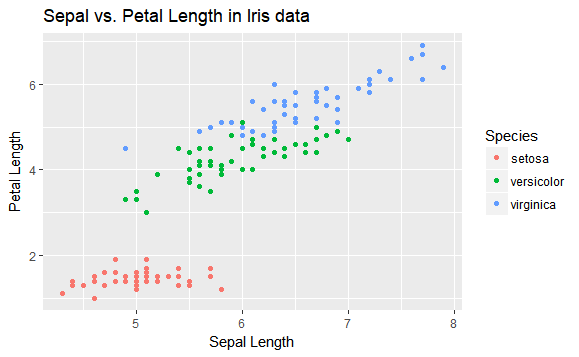


- 표 이름까지 붙이기

qplot(Sepal.Length, Petal.Length, data=iris, color=Species,

xlab = "Sepal Length", ylab = "Petal Length",

main = "Sepal vs. Petal Length in Iris data")



- ggplot 복습

ggplot(data=iris, aes(Sepal.Length, Sepal.Width)) + geom\_point(aes(color = Species))

# 이렇게 위치를 바꿔도 된다.

ggplot(data=iris, aes(Sepal.Length, Sepal.Width. color = Species)) + geom\_point( )

# color나 shape를 할 때는 factor로 해야 한다.

ggplot(mtcars, aes(x=mpg, y=wt, color=factor(gear))) + geom+point( )

만약에 3개의 species에서 각각 보고 싶으면

facet를 하면 된다.

ggplot(iris, aes(Sepal.Length, Sepal.Width)) + geom\_point( ) + facet\_grid(. ~ Species) + geom\_smooth(method = lm)



|  |
| --- |
| Statistical Analysis of Network data with R (SAND) |

- Three format

1. adjacency list 2. edge list 3. adjacency matrices

- 중요한 코드 나열

|  |
| --- |
| - separate network into adjacency matrix and attribute  data(lazega)  A <- get.adjacency(lazega)  v.attributes <- get.data.frame(lazega, what='vertices')  - adjacency matrix를 다시 network로 만들기  as.network(as.matrix(A), directed=FALSE)  summary(Network) 🡪 vertices, edges 다 보여준다.  degree(Graph) |

**2.2. Creating Network Graphs**

# 수동으로 만들 때

g <- graph.formula( ~~~~~ )

# Vertex labeling

V(g)$name <- c(“Sam”, “Mary”, “Tom”)

# union

g <- graph.union(h1, h2)

2.3. Vertex, Edge, and Graph attributes

🡺 $ operator

V(dg)$name <- c(“Sam”, “Mary”, “Tom”)

V(dg)$gender <- c(‘M’, ‘F’, ‘M’)

V(g)$color <- ‘red’

# graph name

g$name <- “Toy Graph”

**2.3.2. Using data frames**

g.lazega <- graph.data.frame(elist.lazega, directed = ‘False’, vertices = v.attr.lazega)

# getting information

vcount(g.lazega)

ecount(g.lazega)

list.vertex.attributes(g.lazega)

**2.4.1. Basic graph concepts**

is.simple(g)

E(mg)$weight <- 1

wg2 <- simplify(mg)

is.simple(wg2)

>> TRUE

# neighbors

neighbors(g, 5)

degree(g)

degree(dg, mode = ‘in’)

degree(dg, mode = ‘out’)

clusters(g)

**네트워크 분석의 간단한 개념들**

simple graph, multigraph

|  |
| --- |
| simple graph가 아닌 multigraph를 simple graph로 바꾸는 방법!  simplify!!!!  즉 vertex와 vertex 사이의 edge가 1개 인 것을 simple graph라고 하고 연결이 다수인 것을 multi graph라고 한다.  > E(fblog.c)$weight <- 1  > fblog.c <- simplify(fblog.c) # multi edge -> weight |

order 🡪 number of vertices

size 🡪 number of indices

directed(digraph), undirected graph

adjacent, neighbors

incident

degree

walk, length

trail, circuit, cycle

distance(shortest 🡪 geodesic distance / longest 🡪 diameter)

- reachable, connected, component,

- weakly connected : if its underlying graph is connected

- strongly connected : if every vertex v is reachable from every other vertex u by a directed walk

component 만드는 방법

|  |
| --- |
| comps <- decompose.graph(yeast)  table(sapply(comps, vcount)) |

- clique, regular graph, tree(directed acyclic graph(DAG))

- directed tree, root

- bipartite graph

**2.4.2. Special types of graphs**

g.full <- graph.full(7)

g.ring <- graph.ring(7)

g.tree <- graph.tree(7, children=2, mode = ‘undirected’)

par(mfrow = c(2, 2))

plot(g.full)

plot(g.ring)

plot(g.tree)

plot(g.star)

Special corner. Minerva’s Network guide

1. The simplest network

|  |
| --- |
| library(igraph)  edges=rbind(c('Dave','Jenny'), c('Pete','Jenny'),c('John','Jenny'),c('Dave','Peter'),c('Dave','John'),c('Peter','Sam'))  head(edges)  g <- graph.edgelist(edges, directed=F)  plot(g, vertex.size=1, vertex.label.dist=0.5) |

- 참고 : 열 만드는 방법

library(tidyverse)

edge\_list <- tibble(from=c(1,2,2,3,4), to=c(2,3,4,2,1))

node\_list <- tibble(id = 1:4)

edge\_list

node\_list

2. correspondence network example

|  |
| --- |
| ?current directory  getwd()  library(tidyverse)  library(tidytext)  library(dplyr)  letters <- correspondence  letters  typeof(letters)  ## creating Vertex list  sources <- letters %>% distinct(source) %>% rename(label=source)  destinations <- letters %>% distinct(destination) %>% rename(label=destination)  nodes <- full\_join(sources, destinations, by='label') ## by means the column name  nodes  ?full\_join  ## giving unique IDs for each city  nodes <- add\_column(nodes, id=1:nrow(nodes))  nodes  nodes2 <- add\_column(nodes, id=1:nrow(nodes)) %>% select(id, everything()) ## select makes the id column come first  ## creating edge list  # weight 설정  per\_route <- letters %>% group\_by(source, destination) %>%  summarise(weight = n()) %>% ungroup()  per\_route  edges <- per\_route %>% left\_join(nodes, by=c('source'='label')) %>% rename(from= id)  ## To join by different variables on x and y use a named vector. For example, by = c("a" = "b") will match x.a to y.b.  🡺 merge의 intersection이랑 같은 개념이다.  edges <- edges %>% left\_join(nodes, by=c('destination'='label')) %>% rename(to=id)  edges  edges <- select(edges, from, to, weight)  edges  ## create network  library(network)  ?network  routes\_network <- network(edges, vertex.attr = nodes, matrix.type ='edgelist', ignore.eval=F)  class(routes\_network)  plot(routes\_network, vertex.cex=3)  plot(routes\_network, vertex.cex=2, vertex.col='red', mode='circle')  ## circular network |

- igraph로 다시 해보자.

|  |
| --- |
| library(igraph)  routes\_igraph <- graph\_from\_data\_frame(d=edges, vertices = nodes2, directed=TRUE)  ?graph\_from\_data\_frame  ## condition for d -> changed nodes to nodes2  ## data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes. Since version 0.7 this argument is coerced to a data frame with as.data.frame.  > plot(routes\_igraph, layout=layout\_with\_graphopt, edge.arrow.size=0.2) |

- gggraph로 다시 해보자.

|  |
| --- |
| # NEW ggraph  library(tidygraph)  library(ggraph)  routes\_tidy <- tbl\_graph(nodes=nodes2, edges = edges, directed = T)  ?tbl\_graph  ## also you can make igraph object into tidygraph  routes\_igraph\_tidy <- as\_tbl\_graph(routes\_igraph)  class(routes\_tidy)  ## [1] "tbl\_graph" "igraph"  class(routes\_igraph\_tidy)  ## [1] "tbl\_graph" "igraph"  class(routes\_igraph)  routes\_tidy  ## WOW!! routes with highest weights  ?activate ## activate(.data, what)  routes\_tidy %>%  activate(edges) %>%  arrange(desc(weight))  🡪 순서대로 쫘르륵 나온다!!  # A directed acyclic simple graph with 1 component  #  # Edge Data: 15 x 3 (active)  from to weight  <int> <int> <int>  1 1 10 68  2 2 10 26  3 1 2 5  4 1 11 2  5 2 13 2  6 4 2 2  - 드디어 graph를 만들자.  ## finally!!! geom\_edge\_link()  ggraph(routes\_tidy) + geom\_edge\_link() + geom\_node\_point() +  theme\_graph()    ggraph(routes\_tidy, layout='graphopt') +  geom\_edge\_link(aes(width=weight), alpha=0.8) +  ## 이렇게 edge width를 조정한다.  scale\_edge\_width(range=c(0.2, 2)) +  ## 너무 굵기 때문에 scale을 0.2랑 2사이로 하면 굵기가 적당해진다.  geom\_node\_text(aes(label=label), repel=T) +  labs(edge\_width ="Letters") + ## scale 위에 글자 바꾸기  theme\_graph()  ## repel=TRUE ## overplotting을 방지하는 것이다.  multiple objects can end up sharing the same space, positioned on top of one another. This makes it diffi cult or impossible to see the  individual values, which can undermine analysis.    # linear fashion  ggraph(routes\_igraph, layout='linear')+  geom\_edge\_arc(aes(width=weight), alpha=0.8)+  scale\_edge\_width(range=c(0.2, 2))+  geom\_node\_text(aes(label=label))+  labs(edge\_width="Letters")+  theme\_graph() |

**3. Visualizing Network data**

|  |
| --- |
| library(sand) # 우리 책에서 쓰이는 여러가지 자료  g.l <- graph.lattice(c(5 ,5 ,5))  # 실제로 그리기!!!  igraph.options(vertex.size=3, vertex.label=NA, edge.arrow.size = 0.5) # option 설정  > par(mfrow=c(1,2)) # 구조 배치  > plot(g.l, layout=layout.circle)  > title(“5x5x5 Lattice”)  > plot(aidsblog, layout=layout.circle)  > title(“Blog Network”)  🡪 igraph.options 에 해당하는 내용은 plot( ) 안에 집어넣으면 된다.  1. edge.arrow.size  2. vertex.label  3. vertex.size  4. vertex.col  5. layout =  # 여러가지 layout  (1) Circle  plot(g.l, layout=layout.circle)  (2) Spring-embedder methods  - Fruchterman.reingold  plot(g.l, layout=layout.fruchterman.reingold)  - enhanced version of spring-embedder methods  VxOrd  - DrL method  (3) energy-placement methods  plot(g.l, layout = layout.kamada.kawai)  (4) tree 🡨 cycle이 있는 data는 표시하는데 제한이 있게 된다.  plot(g.l, layout = layout.reingold.tilford(g.tree, circular = T))  plot(g.l, layout = layout.reingold.tilford)  (5) graphopt (인간에게 optimization된 graph)  plot(graph, layout=layout.graphopt  🡺 graphopt exists because of the inability of freely-available graph layout optimizers to produce human-digestable graphs for many of the graphs I have needed |

**3.4. Decorating graph layouts**

(중요) Fungi는 네모, bacteria는 세모로 할 수 있다!!

library(igraphdata)

data(karate)

set.seed(42)

l <- layout.kamada.kawai(karate)

plot(karate, layout=l)

library(igraphdata)

data(karate)

set.seed(42)

l <- layout.kamada.kawai(karate)

네트워크 자료의 기본 구성 보기

|  |
| --- |
| list.vertex.attributes(fblog)  list.edge.attributes(fblog) |

# igraph.options

🡺 plot 함수 안에 넣어도 된다!

Vertex attribute : vertex.label, vertex.size, vertex.color, vertex.shape

(1) vertex.lable하는 방법!!!!

# vertex가 작을 때, 글씨를 옆으로 나오게 하는 방법

V(karate)$label.dist <- ifelse(V(karate)$size >=10, 0, 0.75) # setting off the label to the size of the vertex

(2) vertex.size하는 방법!!

plot(karate, layout = l, vertex.label=V(karate), vertex.size = 10)

or

igraph.options(vertex.size = 10)

(3) Vertex 크기를 바꾸는 방법 (edge개수대로)

V(karate)$size <- 4\*sqrt(graph.strength(karate))

(4) Edge굵기를 바꾸는 방법

# graph.strength(karate)

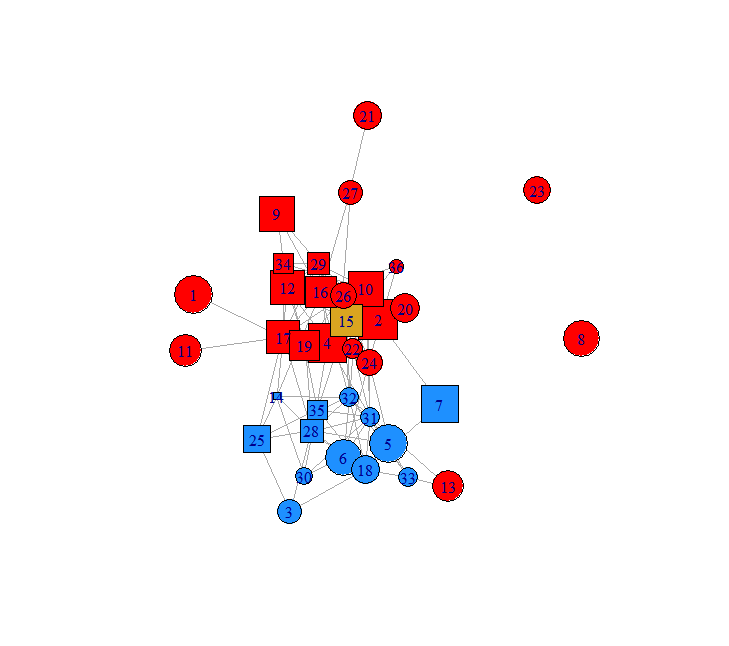
## 이것은 incident edges의 weight를 재준다.

E(karate)$width <- E(karate)$weight ## how was this weight attribute made??

- 시작

|  |
| --- |
| library(igraphdata)  data(karate)  set.seed(42)  l <- layout.kamada.kawai(karate)  # plot undecorated first  igraph.options(vertex.size = 10)  plot(karate,layout =l)    # label  list.vertex.attributes(karate)  karate  V(karate)$label  V(karate)$name  V(karate)$name  sub("Actor ", "", V(karate)$name)  V(karate)$label <- sub("Actor ", '', V(karate)$name)  ## Mr Hi and John A included  V(karate)$shape  list.vertex.attributes(karate)  V(karate)$Faction  # Change shape  V(karate)$shape <- 'circle'  V(karate)[c('Mr Hi', 'John A')]$shape <- 'rectangle'  # change color by faction  V(karate)[Faction==1]$color <- 'red'  V(karate)[Faction==2]$color <- 'dodgerblue'  # Vertex area proportional to vertex strength  # total weight of incident edges  V(karate)$size <- 4\*sqrt(graph.strength(karate))  V(karate)$size <- V(karate)\*0.5  # weight edges by number of common activities  E(karate)$width <- E(karate)$weight ## how was this weight attribute made??  # Color edges by within/between faction  F1 <- V(karate)[Faction==1]  F2 <- V(karate)[Faction ==2]  E(karate)[F1 %--% F1]$color <- 'pink'  E(karate)[F2 %--% F2]$color <- 'lightblue'  E(karate)[F1 %--% F2]$color <- 'yellow'  # offset vertex labels for smaller pints(default = 0)  V(karate)$label.dist <- ifelse(V(karate)$size >=10, 0, 0.75) # setting off the label to the size of the vertex  # plot  plot(karate, layout=l) |

더 컴팩트한 예시



|  |
| --- |
| library(sand)  data(lazega)  # office location by color  colbar <- c('red', 'dodgerblue', 'goldenrod')  upgrade\_graph(lazega)  colbar  # V(lazega)$Office  # > [1] 1 1 2 1 2 2 2 1 1 1 1 1 1 2 3 1 1 2 1 1 1 1 1 1 2 1 1 2 1 2 2 2 2 1 2 1  v.colors <- colbar[V(lazega)$Office]  # type of practice indicated by vertex shape  v.shapes <- c('circle', 'square')[V(lazega)$Practice]  # vertex size proportional to years with firm  v.size <- 3.5 \* sqrt(V(lazega)$Years)  # label vertices according to seniority  v.label <- V(lazega)$Seniority  plot(lazega, layout=layout.fruchterman.reingold(lazega), vertex.color=v.colors, vertex.shape=v.shapes, vertex.size = v.size, vertex.label=v.label) |

**3.5. Visualizing large networks**

- kamada-kawai method

|  |
| --- |
| library(sand)  summary(fblog)  str(fblog)  list.vertex.attributes(fblog)  list.edge.attributes(fblog)  party.names <- sort(unique(V(fblog)$PolParty))  set.seed(42)  l <- layout.kamada.kawai(fblog)  #factorize -> don't quite understand the reason for doing this  party.nums.f <- as.factor(V(fblog)$PolParty)  party.nums <- as.numeric(party.nums.f) # how can a number be a color?  plot(fblog, layout=l, vertex.label=NA, vertex.size=3, vertex.color = party.nums) |

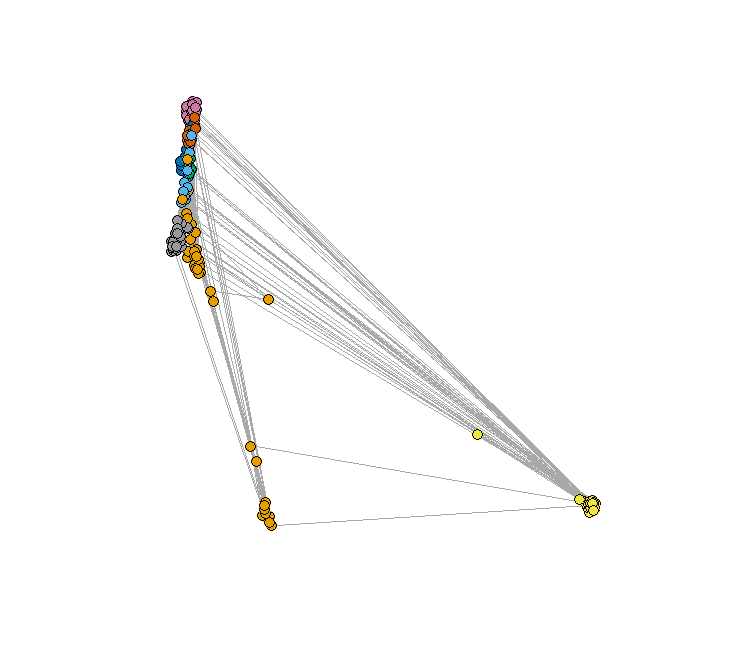
- Drl method

# DrL method

set.seed(42)

l <- layout.drl(fblog)

plot(fblog, layout=l, vertex.size=5, vertex.label=NA, vertex.color=party.nums)



- Graph partitioning method

- 정의 : coarsen a network graph, replacing groups of vertices with single meta-vertices

- 방법 : contract.vertices( )

|  |
| --- |
| # graph partitioning methods  fblog.c <- contract.vertices(fblog, party.nums)  E(fblog.c)$weight <- 1  fblog.c <- simplify(fblog.c) # multi edge -> weight  party.size <- as.vector(table(V(fblog)$PolParty)) ## table -> vector V(fblog)$PolParty only shows the name of the party, so to count them you need table  party.names <- sort(unique(V(fblog)$PolParty))  plot(fblog.c, vertex.size=5\*sqrt(party.size),  vertex.label = party.names,  vertex.color=V(fblog.c), edge.width=sqrt(E(fblog.c)$weight), vertex.label.dist=1.5,  edge.arrow.size=0) |

- egocentric network visualizations

- 정의 : 일부분만 보는 방법

|  |
| --- |
| data(karate)  k.nbhds <- graph.neighborhood(karate, order=1)  sapply(k.nbhds, vcount)  # [1] 17 10 11 7 4 5 5 5 6 3 4 2 3 6 3 3 3 3 3 4 3 3 3 6  # [25] 4 4 3 5 4 5 5 7 13 18  # therefore, vertex 1 and 34 are the largest  k.1 <- k.nbhds[[1]]  k.34 <- k.nbhds[[34]]  par(mfrow=c(1,2))  plot(k.1, vertex.label=NA, vertex.color=c('red', rep('lightblue',16)))  plot(k.34, vertex.label=NA, vertex.color=c(rep('lightblue',17), 'red')) |

- Other tools

(1) Rgraphviz package

- 사용법 : write it to a file in the dot file format that is supported by igraph’s write.grapsh function

(2) Pajek

(3) Cytoscape

(4) Gephi

file format : gexf 🡨 rgexf package로 convert it

**4. Descriptive analysis of Network Graph Characteristics**

|  |
| --- |
| MAP  1. Vertex characteristic  - degree centrality  - closeness centrality  - betweenness centrality  - eigenvalue centrality  2. Edge characteristic : edge betweenness centrality  3. Network cohesion characterizing  - Cliques  - census in dyad, triads  - motif  - density  - transitivity  - reciprocity  - connectivity : giant component, k-vertex-connected, vertex connectivity, vertex-cut  - graph partitioning |

4.2. summaries of vertex and edge characteristics

4.2.1. Vertex characteristic

(1) vertex degrees

- degree distribution

|  |
| --- |
| dev.off()  library(sand)  data(karate)  hist(degree(karate), col='lightblue', xlim=c(0,50), xlab='Vertex Degree', ylab='Frequency', main='') |

- vertex strength

: a useful generalization of degree, simply obtained by summing up the weights of edges incident to a given vertex.

|  |
| --- |
| par(mfrow=c(2,1))  hist(graph.strength(karate), col='pink', xlab="Vertex strength", main="")  E(karate)$weight  # [1] 4 5 3 3 3 3 2 2 2 3 1 3 2 2 2 2 6 3 4 5 1 2 2 2 3 4 5 1 3 2 2 2 3 3 3 2  # [37] 3 5 3 3 3 3 3 4 2 3 3 2 3 4 1 2 1 3 1 2 3 5 4 3 5 4 2 3 2 7 4 2 4 2 2 4  # [73] 2 3 3 4 4 5 |

- log scale로 바꿔보자

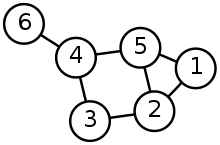
|  |
| --- |
| par(mfrow=c(1,2))  library(igraphdata)  data(yeast)  ecount(yeast)  vcount(yeast)  # to draw histogram you don't need degree.distribution, just raw data needed  d.yeast <- degree(yeast)  hist(d.yeast, col='blue', main='Degree Distribution')  # degree.distribution(yeast)  dd.yeast <- degree.distribution(yeast)  d <- 1:max(d.yeast)-1  max(d.yeast)  # because of log, no 0 allowed  ind <- (dd.yeast != 0)  plot(d[ind], dd.yeast[ind], log='yx', col='blue', xlab="log-Degree", ylab="Log-Intensity", main='Log-Log Degree distribution') |

- to understand the manner in which vertices of different degrees are linked with each other

🡪 average degree fo the neighbors of a given vertex

\* neighborhood의 개념!!!

The neighbourhood of a vertex v in a graph G is the subgraph of G induced by all vertices adjacent to v. For example, the image shows a graph of 6 vertices and 7 edges. Vertex 5 is adjacent to vertices 1, 2, and 4 but it is not adjacent to 3 and 6. The neighbourhood of vertex 5 is the graph with three vertices, 1, 2, and 4, and one edge connecting vertices 1 and 2.



- knn

Average nearest neighbor degree

: calculate the average nearest neighbor degree of the given vertices and the same quantity in the function of vertex degree

# knn 만들면 두가지 값이 출력이 된다.

knn

A numeric vector giving the average nearest neighbor degree for all vertices in vids.

knnk

A numeric vector, its length is the maximum (total) vertex degree in the graph. The first element is the average nearest neighbor degree of vertices with degree one, etc.

|  |
| --- |
| # average degree of the neighbors of a given vertex  a.nn.deg.yeast <- graph.knn(yeast, V(yeast))$knn  dev.off()  plot(d.yeast, a.nn.deg.yeast, log='xy', col='goldenrod', xlab='Log Vertex Degree', ylab="Log Average Neighbor Degree") |

**(2) vertex centrality measures**

**(2-0) vertex degree centrality**

**(2-1) closeness centrality**

- 정의 : vertex is central if it is ‘close’ to many other vertieces

- 식 : (V) = 1 /

\* dist = geodesic distance between the vertices

- normalize를 한다.

**(2-2) betweenness centrality**

- 정의 : summarizing the extent to which a vertex is located between other pairs of vertices. 🡪 vertices that sit on many paths are likely more critical to the communication process

- 식 [Freeman] :

=

\* 분자 : total number of shortest paths between s and t that pass through v.

\* 분모 : total number of shorted paths between s and t.

**(2-3) eigenvector centrality**

- 정의 : ‘status’, ‘prestige’, or ‘rank’ 🡪 the more central neighbors of a vertex are, the more central that vertex itself is.

- 식 [Bonacich] :

(v) =

A= 의 해이다. A는 adjacency matrix를 의미한다.

🡪 optimal choice of is the largest eigenvalue ot A, and hence is the corresponding eigenvector.

\* double colon의 의미

For a package pkg, pkg::name returns the value of the exported variable name in namespace pkg

|  |
| --- |
| library(sand)  data(karate)  A <- get.adjacency(karate, sparse = F)  ## get adjacency matrix  library(network) ## make matrix 🡪 network  g <- network::as.network.matrix(A) ## For a package pkg, pkg::name returns the value of the exported variable name in namespace pkg  dev.off()  par(mfrow=c(2,2))  # degree centrality  library(sna)  sna::gplot.target(g, degree(g), main="Degree",  circ.lab = FALSE, circ.col='skyblue',  userarrows = FALSE,  vertex.col = c('blue', rep('red', 32), 'yellow'),  edge.col = 'darkgray'  )  # Closeness centrality  sna::gplot.target(g, closeness(g), main="Closeness",  circ.lab = FALSE, circ.col='skyblue',  userarrows = FALSE,  vertex.col = c('blue', rep('red', 32), 'yellow'),  edge.col = 'darkgray'  )  # Betweenness centrality  sna::gplot.target(g, betweenness(g), main="Betweenness",  circ.lab = FALSE, circ.col='skyblue',  userarrows = FALSE,  vertex.col = c('blue', rep('red', 32), 'yellow'),  edge.col = 'darkgray'  )  # Eigenvalue centrality  sna::gplot.target(g, evcent(g), main="Eigenvalue",  circ.lab = FALSE, circ.col='skyblue',  userarrows = FALSE,  vertex.col = c('blue', rep('red', 32), 'yellow'),  edge.col = 'darkgray'  )    🡪 target graph는 다른 여러 용도로 쓸 수 있을 것 같다!!!!  - 특징 :  1. betweenness centrality에서 가장 극명하게 차이가 난다. 그 이유는 이 karate network의 구조가 star-like하기 때문이다. |

\* 부가 개념 : hubs and authority의 개념

Directed graph에서는 HITS algorithm이라는 개념이 있다.

- hubs and authorities

- hub란 밖으로 나가는 화살표가 많은 것

- authority란 안으로 들어오는 화살표가 많은가의 기준

🡪 이것이 eigenvector centrality이다!!

|  |
| --- |
| data(aidsblog)  l <- layout.kamada.kawai(aidsblog)  dev.off() # clean the screen  par(mfrow = c(1,2))  # hub  plot(aidsblog, layout = l, main = "Hubs", vertex.color='lightblue', vertex.label = "", vertex.size= 10 \* sqrt(hub.score(aidsblog)$vector))  # authority  plot(aidsblog, layout=l, main='Authorities', vertex.color='lightblue',vertex.label='', vertex.size=10\*sqrt(authority.score(aidsblog)$vector))    # what is hub score?? authority.score?  hub.score(aidsblog)  typeof(hub.score(aidsblog))  ?hub.score ## it gives Value of 1. vector, 2. value.  hub.score(aidsblog)$value |

4.2.3. Characterizing Edges

- 지금까지는 vertex의 특징에 대한 description만 했었는데 edge에 대해서도 이런 말을 할 수 있을까?

(1) Edge betweenness centrality 🡪 vertex의 개념을 그대로 가지고 온 것이다.

|  |
| --- |
| > eb <- edge.betweenness(karate)  > E(karate)[order(eb, decreasing=T)[1:3]]  + 3/78 edges from 4b458a1 (vertex names):  [1] Actor 20--John A Mr Hi --Actor 20 Mr Hi --Actor 32  🡪 이것을 보면 Actor 20이 중요하다는 것을 알 수 있다. |

\* 부가 개념

line.graph 🡪 edge를 vertex로 바꾸고, vertex를 edge로 바꾸면 된다.

- line.graph의 command로 하면 된다.

**4.3. Network Cohesion : Continuing with measures of network cohesion**

- 정의 : extent to which subsets of vertices are cohesive or ‘stuck together’.

- What portion of a measured internet topology would seem to constitue the ‘backbone’? 🡺 how do we define and summarize ‘cohesion’?

- 종류 : (1) local 🡪 triads

(2) global 🡪 giant components

(3) specified explicitly 🡪 cliques

(4) implicitly 🡪 clusters, community

4.3.1. Subgraphs and Censuses

(1) Cliques

- 정의 : complete subgraphs and hence are subsets of vertices that are fully cohesive.

- 부가 : maximal clique 🡪 clique that is not a subset of a larger clique

- weakened notion of clique : k-core decomposition of the network

🡺 k-core of graph : a subgraph of G for which all vertex degrees are at least k, and such that no other subgraph obeying the same condition contains it.(maximal)

🡺 와 여기서 또 다른 core의 개념이 나오는구나. core의 수학적 의미는 좀 다르구나.

The notion of cores is particularly popular in visualization, as it provides a way of decomposing a network into layors, like onion. 🡪 using target plot

|  |
| --- |
| >> cliques(karate)  [[168]]  + 2/34 vertices, named, from 4b458a1:  [1] Actor 2 Actor 8  [[169]]  + 3/34 vertices, named, from 4b458a1:  [1] Mr Hi Actor 2 Actor 8  [[170]]  + 2/34 vertices, named, from 4b458a1:  [1] Mr Hi Actor 8  >> sapply(cliques(karate), length)  [1] 1 1 1 1 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 1 2 1 2 1 2 1 2 2 2 2 3 2  [33] 2 1 2 3 2 3 2 2 2 1 2 2 3 2 1 2 3 2 3 2 1 2 3 2 1 2 3 2 1 2 3 2  [65] 1 2 3 2 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2 3 3 2 3 2 2 1 2 3  [97] 2 3 2 1 2 3 4 3 2 3 2 3 2 1 2 3 2 2 1 2 3 4 3 2 3 2 3 2 2 1 2 3  [129] 4 3 2 3 3 2 3 2 2 1 2 3 4 5 4 3 4 3 2 3 4 3 2 3 2 2 1 2 3 4 5 4  [161] 3 4 3 2 3 4 3 2 3 2  > table(sapply(cliques(karate), length))  1 2 3 4 5  34 78 45 11 2  🡪 clique이 5개 짜리는 2개밖에 없다.  > cliques(karate)[sapply(cliques(karate), length)==5]  [[1]]  + 5/34 vertices, named, from 4b458a1:  [1] Mr Hi Actor 2 Actor 3 Actor 4 Actor 14  [[2]]  + 5/34 vertices, named, from 4b458a1:  [1] Mr Hi Actor 2 Actor 3 Actor 4 Actor 8  - maximal clique  > table(sapply(maximal.cliques(karate), length))  2 3 4 5  11 21 2 2 |

- k-core decomposition of the network

|  |
| --- |
| # k core graph  cores <- graph.coreness(karate) # remember coreness  library(sna)  sna::gplot.target(g, cores, circ.lab=F,  circ.col = 'skyblue', usearrows=F,  vertex.col=cores, edge.col='darkgray')  detach('package:sna')  detach('package:network') |
|  |

- Cencus

- dyads and triads

- dyads 🡪 null(no directed edges, asymmetric(one directed edge), mutual(two directed edges)

- 정의 : a census of the possible states of these two classes of subgraphs, i.e., counting how many times each state is observed in a graph G, can yield insight into the nature of the connectivity in the graph.

|  |
| --- |
| aidsblog <- simplify(aidsblog)  dyad.census(aidsblog)  $mut  [1] 3  $asym  [1] 177  $null  [1] 10405 |

\* 부가 개념 🡪 motif

- motif 정의 : small connected subgraphs of interest are commonly termed motifs.

- 형식 : graph.motifs

**4.3.2. Density and related notions of relative frequency**

- 개념의 필요성 : network cohesion의 개념은 pre-specified notion of substructure를 놓고 graph G에 있나 봤다. 그러나, relative frequency는 그 정도(cohesive한 정도)의 spectrum을 보는 것이다.

- 개념의 종류 : density, clustering coefficient(clustering의 정도= transitivity)

(1) density

- 정의 : density of a graph is the frequency of realized edges relative to potential edges.

subgraph H = (VH, EH)

- 식 : den(H) = EH / ( VH(VH-1)/2 )

G가 directed이면 분모의 2로 나누는 거 빼기

|  |
| --- |
| ego.instr <- induced.subgraph(karate, neighborhood(karate, 1, 1)[[1]])  # neighborhood(karate,order=1, 1) ## 1st vertex's neighbor  # neighborhood(karate, 1, 1)[[1]] ## [[1]] removes list type  ego.admin <- induced.subgraph(karate, neighborhood(karate,1, 34)[[1]])  # I don't know what the difference is of subgraph and induced subgraph  ?induced.subgraph  graph.density(karate)  graph.density(ego.instr)  graph.density(ego.admin)  > graph.density(karate)  [1] 0.1390374  > graph.density(ego.instr)  [1] 0.25  > graph.density(ego.admin)  [1] 0.2091503  🡪 오 확실히 ego-centric networks가 more dense하다는 것을 손쉽게 알 수 있구나. |

(2) clustering coefficient = transitivity

- 정의 : transitivity, fraction of transitive triples (Global한 개념)

- 식 :

clT = 3 τ(G) / τ3 (G)

\* τ : number of triangles in a graph

\* τ3 (G) : number of connected triples (i.e., a subgraph of three vertices connected by two edges, also sometimes called 2-star

- local한 analogue

cl(v) = τ(v) / τ3(v)

\* τ : v가 포함된 삼각형의 개수

\* τ3 (v) : the number of connected triangles

- 형식 : transitivity(karate)

|  |
| --- |
| > transitivity(karate)  [1] 0.2556818  > transitivity(karate, 'local', vids=c(1,34))  [1] 0.1500000 0.1102941  🡪 vertex 1이 더 dense하구나!!! |

(3) Reciprocity 🡨 directed graph에 한정해서

- 정의 : the extent to which there is reciprocation among ties in a directed network.

- 측정의 두가지 방법

1) dyads로 🡪 reciprocated(mutual) directed edges divied by the number of dyads with a single, unreciprocated edge

> reciprocity(aidsblog, mode='default')

[1] 0.03278689

2) directed edges 🡪 total number of reciprocated edges divided by the total number of edges.

> reciprocity(aidsblog, mode='ratio')

[1] 0.01666667

**4.3.3. Connectivity, Cuts and Flows**

- 메인 질문 : whether the graph separates into distinct subgraphs

🡪 flow of formation in the graph

- Connectivity 측정 방법 :

\* connected : if every vertex is reachable from every other

(1) Giant component : Often it is the case that one of the connected components in a graph G dominates the others in magnitude.

(2) refined notion of connectivity : 개념

- 개념 : if an arbitrary subset of k vertices(edges) is removed from a graph the remaining subgraph is connected.

- k-vertex-connected : 랜덤하게 k-1개까지의 vertex를 없애도 subgraph가 connected(reached from any point to point)

- vertex connectivity : largest integer k sub that G is k-vertex-connected.

|  |
| --- |
| > vertex.connectivity(giant.comp)  [1] 1  > edge.connectivity(giant.comp)  [1] 1 |

(3) Vertex cut 개념

- 정의 : if the removal of particular set of vertices in a graph disconnects the graph, that set is called a vertex-cut

- 의의 : provide a sense of where a network is vulnerable

- articulation point

|  |
| --- |
| > yeast.cut.vertices <- articulation.points(giant.comp)  > yeast.cut.vertices  + 350/2375 vertices, named, from b89f150:  [1] YDR091C YGR103W YER133W YDL145C YGL237C YPL235W  [7] YHR099W YPR023C YPR018W YML102W YKL113C YDR328C  [13] YDR074W YIL053W YBR126C YPR160W YMR105C YJL138C  [19] YDR416W YLR453C YBL102W YOR036W YBR288C YNR006W  [25] YPL195W YNR069C YLL049W YKR083C YPR182W YLR275W  [31] YKL074C YDR364C YDL098C YBR009C YML110C YMR153W  [37] YCR086W YDL089W YPR028W YML008C YDL100C YIL033C  [43] YIR008C YDR156W YGL070C YER042W YDL161W YOR111W  [49] YHL016C YPL172C YMR114C YDL164C YGL207W YGL153W  [55] YKR059W YOL139C YDR054C YER125W YLR371W YOR265W  + ... omitted several vertices  > length(yeast.cut.vertices)  [1] 350 |

- shortest.pahts,

graph.maxflow

graph.mincut

|  |
| --- |
| data(yeast)  comps <- decompose.graph(yeast)  sapply(comps, vcount)  table(sapply(comps, vcount))  > 2 3 4 5 6 7 2375  > 63 13 5 6 1 3 1  🡪 엄청나게 큰 녀석이 존재한다!!! |

- giant component의 성질 : small world property

(a) the shortest-path distance between pairs of vertices is generally quite small

(b) the clustering is relatively high

|  |
| --- |
| (a) small path 확인하기.  > average.path.length(giant.comp)  [1] 5.09597  > diameter(giant.comp)  [1] 15  🡪 longest path도 별로 길지 않은 편  (b) clustering도 크다.  > transitivity(giant.comp)  [1] 0.4686663 |

**4.4. methods for graph partitioning (community detection)**

- 정의 : segmentation of a set of elements into ‘natural’ subsets.

- 이유 : useful tool for finding, in an unsupervised fashion, subsets of vertices that demonstrate a ‘cohesiveness’. 🡺 계속 cohesiveness을 가늠할 수 있는 지표들이구나.

- community detection 방법(clustering 방법?) : (1) Hierarchical clustering (2) Spectral clustering

4.4.1. Hierarchical clustering

- 방법 :

1) agglomerative(successive coarsening of partitions through the process of merging),

2) divisive(successive refinement of partitions through the process of splitting)

🡪 cost measures

mod = 2

🡪 뭔지 이해를 잘 못했다.(pg 60)

- 형식 : agglomerative hierarchical clustering algorithm

|  |
| --- |
| fastgreedy.community(karate)  IGRAPH clustering fast greedy, groups: 3, mod: 0.43  + groups:  $`1`  [1] "Actor 9" "Actor 10" "Actor 15" "Actor 16" "Actor 19"  [6] "Actor 21" "Actor 23" "Actor 24" "Actor 25" "Actor 26"  [11] "Actor 27" "Actor 28" "Actor 29" "Actor 30" "Actor 31"  [16] "Actor 32" "Actor 33" "John A"    $`2`  [1] "Mr Hi" "Actor 2" "Actor 3" "Actor 4" "Actor 8"  [6] "Actor 12" "Actor 13" "Actor 14" "Actor 18" "Actor 20"  [11] "Actor 22"  + ... omitted several groups/vertices  > kc <- fastgreedy.community(karate)  > length(kc) ## length를 하면 group 개수가 나옴  [1] 3  > sizes(kc)  Community sizes  1 2 3  18 11 5  > membership(kc)  Mr Hi Actor 2 Actor 3 Actor 4 Actor 5 Actor 6 Actor 7  2 2 2 2 3 3 3  Actor 8 Actor 9 Actor 10 Actor 11 Actor 12 Actor 13 Actor 14  2 1 1 3 2 2 2  Actor 15 Actor 16 Actor 17 Actor 18 Actor 19 Actor 20 Actor 21  1 1 3 2 1 2 1  Actor 22 Actor 23 Actor 24 Actor 25 Actor 26 Actor 27 Actor 28  2 1 1 1 1 1 1  Actor 29 Actor 30 Actor 31 Actor 32 Actor 33 John A  1 1 1 1 1 1  대박!!!  > plot(kc, karate)    - dendrogram  library(igraph)  library(ape)  library(sand)  data(karate)  kc <- fastgreedy.community(karate)  dendPlot(kc, mode = 'phylo') ## dendPlot is igraph's function |

**4.4.2. Spectral clustering**

- 개념 : spectral graph theory that associate the connectivity of a graph G with the eigen-analysis of certain matrices.

- 이론 : (Laplacian of a graph G) matrix L = D - A.

the number of components in a graph is directly related to the number of non-zero eigenvalues of the graph Laplacian.

\* spectral graph theory : a graph G will consist of K connected components if and only if lambda1=lambda2=lambda3=lambda4=..=lambda k=0 and lambda(k+1) > 0

- 의미 : 만약에 둘로 잘 쪼개질 것 같다면, 람다1은 0, 람다2는 0과 가깝다?

람다2를 Fiedler vlaue라고 하고, 이에 대응하는 벡터, x2를 Fiedler vector라고 한다.

🡪 iterative하게 둘로 쪼갠다 계속.

- 형식 :

|  |
| --- |
| k.lap <- graph.laplacian(karate)  eig.anal <- eigen(k.lap)  plot(eig.anal$values, col='blue',  ylab='Eigenvalues of Graph Laplacian')    🡪 only one is exactly zero.(connected하기 때문에!? 무슨 말이지?) / second smallest eigenvalue lambda2 is quite close to zero. |

- Fiedler vector

|  |
| --- |
| k.lap <- graph.laplacian(karate)  eig.anal <- eigen(k.lap)  f.vec <- eig.anal$vectors[,33] ## 33th is the fiedler vector  faction  ?get.vertex.attribute  faction <- get.vertex.attribute(karate, 'Faction') ## [1] 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2  dev.off()  f.colors <- as.character(length(faction)) ## why????, you don’t need this. tested below with so vector.  f.colors[faction ==1] <- 'red'  f.colors[faction ==2] <- 'cyan'  plot(f.vec, pch=16, xlab='Actor Number', ylab='Fiedler vector entry',  col = f.colors  ) # pch=16 is black filled circle  abline(0, 0, lwd=2, col='lightgray') # put axis, lwd = 'line width'    🡪  # it doesn't matter what variable you use. You don't need f.color vector of specific length  so <- ''  so[faction ==1] <- 'red'  so[faction ==2] <- 'cyan'  so  #[1] "red" "red" "red" "red" "red" "red" "red" "red" "cyan" "cyan" "red" "red" "red"  #[14] "red" "cyan" "cyan" "red" "red" "cyan" "red" "cyan" "red" "cyan" "cyan" "cyan" "cyan"  #[27] "cyan" "cyan" "cyan" "cyan" "cyan" "cyan" "cyan" "cyan" |

- 형식 : leading.eigenvector.community

**4.4.3. Validation of graph partitioning**

- relavant characteristic이 있으면 이걸로 partitioning 결과랑 비교를 해볼 수도 있겠다.

- 비교를 안해보면 graph partitioning may be viewed as a tool for discovering such subsets in the absence of knowledge of these characteristics.

|  |
| --- |
| # if you know graph's relavant characteristic e.g. class in yeast  data(yeast)  # first make giant component  decomp.graph <- decompose.graph(yeast)  sapply(decomp.graph, vcount)  table(sapply(decomp.graph, vcount))  yeast.gc <- decomp.graph[[1]]  list.vertex.attributes(yeast)  ## [1] "name" "Class" "Description"  # start  func.class <- get.vertex.attribute(yeast.gc, "Class")  table(func.class)  # A B C D E F G M O P R T U  # 51 98 122 238 95 171 96 278 171 248 45 240 483  #spectral graph partitioning  yc <- fastgreedy.community(yeast.gc)  c.m <- membership(yc)  table(c.m, func.class, useNA=c('no')) ## NA는 안써도.  - visualization 한번 해보자  matrix <- table(c.m, func.class, useNA=c('no'))  typeof(matrix)  heat <- as.data.frame(matrix)  library(ggplot2)  ggplot(data=heat, aes(x=func.class, y=c.m, fill=Freq)) + geom\_tile() |

**4.5. finishing with topics of assortativity and mixing**

- 정의 : assortative mixing 🡪 selective linking among vertices, according to a certain characteristics

- assortativity coefficient 개념 : quantify the extent of assortative mixing in a given network

- 예시 : categorical case

- 식 : 생략

- 의미 : 1에 가까울수록 there is perfect assortative mixing(when edges only connect vertices of the same category)

|  |
| --- |
| assortativity.nominal(yeast, (V(yeast)$Class=='P')+1, directed=F)  ## 이거하면 R program 에러난다. 이게 무슨 의미인지도 모르겠음(pg67) |

- 예시2 : continuous

🡪 용도 : one common use of the assortativity coefficient r is in summarizing degree-degree correlation of adjacent vertices.

\* degree, degree correlation이 뭘까?

|  |
| --- |
| > library(igraph)  > data(yeast)  > assortativity.degree(yeast)  [1] 0.4610798 |

**Chapter 5.**

**5. Mathematical models for Network Graphs**

- 네트워크 모델의 구성요소

(1) E : collection(ensemble) of possible graphs

(2) P : Probability distribution of G

(3) : vector of parameters

- 모델의 종류 :

1) defined more from a mathematical perspective

- classical random graph models (5.2.)

- more recent generalizations (5.3.)

- mimic certain observed real-world properties (5.4.)

- several examples

2) a statistical perspective 🡪 6장

**5.2. Classical random graph models**

- 정의 : model specifying a collection E and a uniform probability P over E. 🡪 equal probability on all graphs of a given order and size.

- 다른 표현 : Bernoulli random graph model.

- 형식 : erdos.renyi.game

- 성질 : relatively few vertices on shortest paths between vertex pairs. / low clustering(low transitivity)

- 구현 :

|  |
| --- |
| library(sand)  set.seed(42)  g.er <- **erdos.renyi.game**(100, 0.02) # choice of Nv=100 vertices and a probability of 0.02  plot(g.er, layout=layout.circle, vertex.label=NA)    > table(sapply(decompose.graph(g.er), vcount))  1 2 3 4 71  15 2 2 1 1  🡪 giant component는 있구나!  hist(degree(g.er), col='lightblue', xlab='degree', main='') |

**5.3. Generalized random graph models**

- Erdos-Renyi model : common characteristic is that the size of the graphs G be equal to some fixed Ne.

- Beyond Erdos-Renyi model : collection of all graphs G with a pre-specified degree sequence.

🡪 degree sequence가 fixed된 경우의 graph의 집합도 해당되는구나!!

- 형식 : degree.sequence.game(degs, method=’vl’)

🡺 uniformly sample random graphs with fixed degree sequence

- 예시 :

|  |
| --- |
| # example random graph  degs <- c(2, 2, 2, 2, 3, 3, 3, 3) ## degree sequence  g1 <- degree.sequence.game(degs, method='vl')  g2 <- degree.sequence.game(degs, method='vl')  dev.off()  par(mfrow=c(1,2))  plot(g1, vertex.label=NA)  plot(g2, vertex.label=NA)  # nevertheless, the graphs are not isomorphic!  graph.isomorphic(g1, g2)  # [1] FALSE  c(ecount(g1), ecount(g2))  # [1] 10 10  🡪 degree의 평균 = 2Ne / Nv  Random graph의 조건에서  (1) degree 평균이 같고  (2) Nv(vertex수)가 같기 때문에 Ne(edge수)도 같다. |

- random graph 성질

: all other characteristics are free to vary to the extent allowed by the chosen degree sequence

🡪 degree sequence가 같기만 하면, diameter라든지 transitivity라든지 달라도 된다.

|  |
| --- |
| data(yeast)  degs <- degree(yeast)  fake.yeast <- degree.sequence.game(degs, method='vl')  all(degree(yeast) == degree(fake.yeast))  ?all ## Given a set of logical vectors, are all of the values true?  # [1] TRUE  > diameter(yeast)==diameter(fake.yeast)  [1] FALSE  > transitivity(yeast)==transitivity(fake.yeast)  [1] FALSE |

- degree sequence외의 다른 성질들도 같아야 class E에 넣을 수 있다는 모델

: MCMC method(Markov chain Monte Carlo)

**5.4. Network Graph Models based on mechanisms**

- 의의 : models explicitly designed to mimic certain observed ‘real-world’ properties.

**5.4.1. Small-World Models**

- 의의 : 실제 세계에서는 a) high level of clustering, b) small distances between most nodes.

-

|  |
| --- |
| # vertices 25, neighborhood 5, rewiring probability p=0.05  g.ws <- watts.strogatz.game(1, 25, 5, 0.05)  plot(g.ws, layout=layout.circle, vertex.label=NA) |

- 특징 : small world model을 만드려면 rewiring percentage(p)를 작게하면 된다. rewiring percentage가 낮으면 clustering이 커진다. 즉, transitivity를 해봤을 때 점점 올라간다. diameter로 작아진다.

- p의 증가에 따른 효과 보기

|  |
| --- |
| library(igraph)  # effect of p  steps <- seq(-4, -0.5, 0.5)  len <- length(steps)  cl <- numeric(len)  apl <- numeric(len)  ntrials <- 10  for (i in (1:len)){  cltemp <- numeric(ntrials)  apltemp <- numeric(ntrials)  for (j in (1:ntrials)){  g <- watts.strogatz.game(1, 1000, 10, 10^steps[i]) ## 1, N, neighbor, rewiring percentage  cltemp[j] <- transitivity(g)  apltemp[j] <- average.path.length(g)    }  cl[i] <- mean(cltemp)  apl[i] <- mean(apltemp)  }  plot(steps, cl/max(cl), ylim=c(0,1), ylab='Clustering and Average Path Length', lwd=3, type='l', col='blue', xlab=expression(log[10](p)))  lines(steps, apl/max(apl), lwd=3, col='red') |

**5.4.2. Preferential attachment models**

- 의의 : network changes at any given point in time.

- 식 : G(t)는 Nv(t) = Nv(0) + t vertices

Ne(t) = Ne(0) +tm edges

- 원리 : intuitively we would expect that a number of vertices of comparatively high degree should gradually emerge as t increases. 🡺 preferential attachment

- 모델 : Barabasi and albert model

- 형식 : barabasi.game( )

|  |
| --- |
| set.seed(42)  g.ba <- barabasi.game(100, directed=FALSE)  g.ba  dev.off()  par(mfrow=c(1,2))  plot(g.ba, layout= layout.circle, vertex.label=NA)  hist(degree(g.ba), col='lightblue', xlab='Degree', ylab='Frequency', main='')    > summary(degree(g.ba))  Min. 1st Qu. Median Mean 3rd Qu. Max.  1.00 1.00 1.00 1.98 2.00 9.00 |

**5.5. Assessing significance of Network graph characteristics**

- 의미 : 만약에 Graph G에서 어떠한 구조적 특성을 발견했다고 치자. 그렇다면 그것이 유의미(significant)한가? 다시 말하면, unusual, unexpected한가? 그것의 기준이 바로 uniform draw from G라는 것이다.

**5.5.1. Assessing the number of communities in a network**

- 예시 : fastgreedy.community로 했는데 3개의 community가 나왔다.

- 새로운!!! table 기법!!

🡺 하나의 표로 나타내서 비교할 수 있게 만들었구나!!

> table(index, results)

results

index 4 5 6 7

0 22 57 20 1

1 24 54 21 1

- 방법 :

|  |
| --- |
| library(sand)  data(karate)  nv <- vcount(karate)  ne <- ecount(karate)  degs <- degree(karate)  ntrials <-100  # let's generate classical random graphs  numcomm.rg <- numeric(ntrials)  ?erdos.renyi.game  library(igraph)  # (1) community detection from classic random graphs of same order and size  for (i in (1:ntrials)){  g.rg <- erdos.renyi.game(nv, ne, type='gnm') # The type of the random graph to create, either gnp (G(n,p) graph) or gnm (G(n,m) graph).therefore we put number of vertex and edge.    c.rg <- fastgreedy.community(g.rg)  numcomm.rg[i] <- length(c.rg)    } ## for 100 trials, getting the number of community from classic random graphs  # (2) community detection form classic random graphs of same degree  numcomm.grg <- numeric(ntrials)  for (i in (1:ntrials)){  g.grg <- degree.sequence.game(degs, method = 'vl')  c.grg <- fastgreedy.community(g.grg)  numcomm.grg[i] <- length(c.grg)  }  results <- c(numcomm.rg, numcomm.grg)  index <- c(rep(0, ntrials), rep(1, ntrials))  counts <- table(index, results)/ntrials  barplot(counts, beside=TRUE, col=c('blue', 'red'), xlab='Number of Communities', ylab='Relative frequency', legend=c('Fixed Size', 'Fixed Degree sequence'))    - 결론 : 아하! karate network에서 3 group이었는데, 확실히 classic random model을 해보니까 5정도 있어야 하니... unusual하구나 |

5.5.2. Assessing small world properties

- Small world property 복습

(a) small shortest-path distance

(b) the clustering is relatively high

(a)

(b) directed graph의 clustering을 보기 위해서 새로운 cl(v) 사용(pg 82 참조)

|  |
| --- |
| 일단 clustering coefficient를 만들기 위한 함수를 만든다.  ## NEW: testing small world properties  library(igraphdata)  data(macaque)  summary(macaque)  ## IGRAPH f7130f3 DN-- 45 463 -- ## -> 45 vertices, 463 link  ##+ attr: Citation (g/c), Author (g/c), shape (v/c), name (v/c)  clust.coeff.dir <- function(graph){  A <- as.matrix(get.adjacency(graph))  S <- A + t(A)  deg <-degree(graph, mode=c('total')) ## both in and out  num <- diag(S %\*% S %\*% S) # upper part of the fraction  denom <- diag(A %\*% A)    denom <- 2\*(deg\*(deg - 1) - 2 \* denom)  cl <- mean(num / denom)  return(cl)  }  ## (directional) classical random graphs  ntrials <- 1000  nv <- vcount(macaque)  ne <- ecount(macaque)  cl.rg <- numeric(ntrials)  apl.rg <- numeric(ntrials)  for (i in (1:ntrials)){  g.rg <- erdos.renyi.game(nv, ne, type='gnm', directed =TRUE)  cl.rg[i] <- clust.coeff.dir(g.rg)  apl.rg[i] <- clust.coeff.dir(g.rg)  }  > summary(cl.rg)  Min. 1st Qu. Median Mean 3rd Qu. Max.  0.2160 0.2304 0.2337 0.2340 0.2375 0.2531  > summary(apl.rg)  Min. 1st Qu. Median Mean 3rd Qu. Max.  0.2160 0.2304 0.2337 0.2340 0.2375 0.2531  > clust.coeff.dir(macaque)  [1] 0.5501073  > average.path.length(macaque)  [1] 2.148485  par(mfrow=c(1,2))  hist(cl.rg, col='blue', xlab='clustering coefficient',main='clustering coefficient distribution')  hist(apl.rg, col='pink', xlab='average path length',main='apl distribution')    🡪 classic random model에 비교해봤을 때, clustering이 높고 average path length가 짧으므로 small world property를 가지고 있구나라는 것을 알 수 있다. |

Chapter 6. Statistical model for network graphs

- 배우는 이유 : classic random model come up short, when the purpose is statistical model building.

🡪 Robins and Morris : “A good statistical network graph model needs to be both estimable from data and a reasonable representation of that data to be amenable to examining which competing effects might be the best explanation to the data

- 3가지 종류 : 1) exponential random graph model 🡪 generalized regression model(GLMs)과 유사

2) stochastic block model 🡪 mixture of classic random graph models

3) latent network models 🡪 variant of the common practice of using both observed and unobserved(latent) variables in modeling the outcome.

6.2. Exponential random graph models

- ERGMs(Exponential random graph models)

- ergm package를 위해서 먼저 setting하기

|  |
| --- |
| library(igraph)  library(sand)  data(lazega)  A <- get.adjacency(lazega)  v.attributes <- get.data.frame(lazega, what='vertices')  library(ergm)  lazega.s <- network::as.network(as.matrix(A), directed=FALSE)  network::set.vertex.attribute(lazega.s, "Office", v.attributes$Office)  network::set.vertex.attribute(lazega.s, "Practice", v.attributes$Practice)  network::set.vertex.attribute(lazega.s, "Gender", v.attributes$Gender)  network::set.vertex.attribute(lazega.s, "Seniority", v.attributes$Seniority) |

**6.2.2. Specifying a model**

- 문제 : frequently produce models that fit quite poorly to real data.

- 보완책과 여러 모델 :

|  |
| --- |
| library(igraph)  library(sand)  data(lazega)  A <- get.adjacency(lazega)  v.attributes <- get.data.frame(lazega, what='vertices')  library(ergm)  lazega.s <- network::as.network(as.matrix(A), directed=FALSE)  network::set.vertex.attribute(lazega.s, "Office", v.attributes$Office)  network::set.vertex.attribute(lazega.s, "Practice", v.attributes$Practice)  network::set.vertex.attribute(lazega.s, "Gender", v.attributes$Gender)  network::set.vertex.attribute(lazega.s, "Seniority", v.attributes$Seniority)  my.ergm.bern <- formula(lazega.s ~ edges)  my.ergm.bern  summary.statistics(my.ergm.bern)  # include star counts  my.ergm <- formula(lazega.s ~ edges + kstar(2) + kstar(3) + triangle)  summary.statistics(my.ergm)  my.ergm <- formula(lazega.s ~ edges + gwesp(1, fixed=TRUE))  summary.statistics(my.ergm)  # Also see the main or second-order effects by  lazega.ergm <- formula(lazega.s ~ edges  + gwesp(log(3), fixed=TRUE)  + nodemain('Seniority')  + nodemain('Practice')  + match('Practice')  + match('Gender')  + match('Office')  )  summary(lazega.ergm) |

6.2.3. Model fitting

- 원리 : maximum likelihood estimators(MLEs)

- ergm 🡪 Markov chain Monte Carlo maximum likelihood estimation

6.3. Network Block model

- OTU table to R

<http://geoffreyzahn.com/getting-your-otu-table-into-r/>