############################################################################################

# 데이터 조작( plyr, dplyr, reshape2 등)

# 데이터 분석 업무에서 raw data를 얻은 다음 머신러닝 모델링을 위해서 또는 시각화를 위해서

# 이 raw data를 적절한 형태로 변형

# 데이터 변환, 필터링, 전처리 작업이 필요하다

# 이런 작업(데이터 조작)에 특화된 package들이 존재한다.

# plyr => pliers + R (플라이알)

# : 패키지를 구현한 언어 R, 처리속도가 느리다, 데이터량이 많아지면 속도가 많이 늦어짐

# dplyr => data frame + Pliers + R (디플라이알)

# vector나 data frame에 적용할 수 있는 기본 함수

# 실습할 데이터가 필요하다

# iris(아이리스)는 내장되어있다

iris

View(iris)

# iris : 붓꽃의 종류와 크기에 대해 측정한 데이터

# 통계학자 피셔라는 사람이 측정해서 제공한 데이터

# 컬럼 설명을 보자

ls(iris)

# Species : 종(3가지)

# Sepal.Length : 꽃받침의 길이

# Sepal.width : 꽃받침의 너비

# Petal.Length : 꽃잎의 길이

# Petal.Width : 꽃잎의 너비

#기본함수

# 1. head() : 데이터 셋의 앞에서부터 6개의 데이터를 추출한다.

# 데이터 프레임이 아닌경우에도 적용된다.

head(iris,n=1) # head는 n의 기본값이 6으로 설정되어있다.

# 2. tail() : 데이터 셋의 뒤에서부터 6개의 데이터를 추출한다.

# 데이터 프레임이 아닌경우에도 적용된다.

tail(iris,n=3)

# 3. View() : view창에 데이터를 출력,

# 필터 기능을 이용하면 데이터의 분포를 그래프로 볼 수 있다

View(iris)

# 4. dim() : 몇 차원인가, 각 차원의 개수는 몇개인가

dim(iris)

# 결과 : 150행 5열

# 선형자료구조(vector, list)에서는 결과가 NULL이된다. => 2차원 이상에서만 사용

var1 = c(1,2,3,4,5)

dim(var1)

# 5. nrow() : data frame의 행의 개수( 행, row, 측정치, observation )

nrow(iris)

# 6. ncol() : data frame의 열의 개수( 열, column, 변수 )

ncol(iris)

# 7. str() : data frame에 대한 일반적인 정보를 추출

str(iris)

# 8. summary() : data frame의 요약 통계량을 보여준다

summary(iris)

# Min, Max, 사분위, 평균(mean), 중간값(median)

# 9. ls() : data frame의 column 명을 vector로 추출한다(정렬 : 오름차순)

ls(iris)

# 10. rev() : column 순서를 역순으로만든다

rev(iris)

# 11. length() : 길이를 구하는 함수,

# data frame에 length를 구하면 column의 개수를 알려준다

# matrix에 length를 구하면 요소의 개수를 알려준다

length(iris) # 결과 : 5

var1 = matrix(1:12, ncol=3)

var1

length(var1)

#######################################################################################3

#######################################################################################3

#plyr package => dplyr 개량형 package

install.packages("plyr")

require(plyr) #library(plyr)

# 1. key값을 이용해서 두 개의 data frame을 병합하자( 세로방향, 열방향으로 결합 )

# data frame을 만들어보자

x = data.frame(id=c(1,2,3,4,5),

height=c(150,190,170,188,167))

y = data.frame(id=c(1,2,3,6),

weight=c(50,100,80,78))

join(x,y,by="id",type="inner")

# x와 y 데이터를 결합한다, id라는 키값을 이용해서, type은 inner로

# key를 1개 이용해서 결합하는걸 해보았ㄷㅏ !

# key를 2개 이상 이용해서 결합하려고하면?

x = data.frame(id=c(1,2,3,4,5),

gender=c("M","F","M","F","M"),

height=c(150,190,170,188,167))

y = data.frame(id=c(1,2,3,6),

gender=c("F","F","M","F"),

weight=c(50,100,80,78))

join(x,y,by=c("id","gender"),type="inner")

join(x,y,by=c("id","gender"),type="left")

join(x,y,by=c("id","gender"),type="right")

join(x,y,by=c("id","gender"),type="full")

#dplyr에서는 join() => left\_join(),right\_join, ,,, 함수 자체로 구별됨

# 2. 범주형 변수를 이용해서 그룹별 통계량 구하기

str(iris)

# $를 이용하여 data frame의 column을 명시한다

unique(iris$Species)

# unique : iris의 species column에 어떤 값이 있는가

# str, view는 데이터 수가 적을 때 이용, unique는 데이터량이 많을 때 값을 알아내기 위해 사용

# iris의 종별 꽃잎 길이의 평균을 구하세요

# tapply(대상 column, 범주형 column, 적용할 함수) => 범주형 column에 따라 정렬

# tapply는 한번에 한개의 통계만 구할 수 있다.

tapply(iris$Petal.Length,

iris$Species,

FUN=mean)

# iris의 종별 꽃잎 길이의 평균과 표준편차를 구하세요

# 변수간의 얼마만큼의 차이들이 존재하는가

# 표준편차가 크면 양극화가 심하다.

# ddply() : 한번에 여러개의 통계치를 구할 수 있다

# ddply(data frame, column명, 기본통계량, 함수(), 함수())

df <-ddply(iris,

.(Species),

summarise,

avg=mean(Petal.Length),

sd=sd(Petal.Length))

df

#결과는 data frame형태로 나온다.

## plyr에서는 join()과 통계값을 구하는 함수(2개)만 알아두자!

# dplyr을 많이 사용하면서 plyr는 많이 사용하지 않는다

##########################################################################################################

# dplyr은 c++로 구현되어있어서 속도가 빠르다.

# dplyr은 코딩시 chainig을 사용할 수 있다.

var1 <- c(1,2,3,4,5)

var2 <- var1 \* 2

var3 <- var2 + 5

# var1 <- c(1,2,3,4,5) \* 2 + 5

# 중간 변수를 안만들고 코딩하는 방법 : chaining

install.packages("dplyr")

library(dplyr)

# dplyr의 주요함수들

# 1. tbl\_df : 현재 console 크기에 맞추어서 data frame을 추출하라는 함수

tbl\_df(iris)

######################iris 말고 다른 파일로 해보기##########################################################

library(xlsx)

# sheet index : 첫번째 sheet에서 파일을 불러온다

excel <- read.xlsx(file.choose(),

sheetIndex = 1,

encoding = "UTF-8")

#기본 요소들 확인해보기

View(excel)

str(excel)

class(excel)

ls(excel)

tbl\_df(excel)

# 2. rename() : data frame의 column명을 바꿀 수 있다.

# rename(data frame,

# 바꿀컬럼명1=이전컬럼명1,

# 바꿀컬럼명2=이전컬럼명2, ,,,)

# column명을 수정한 새로운 data frame이 리턴된다 !

# 기존의 data frame이 바뀌는것이 아니다 @@@@!!!!!!!

# 제공된 excel을 읽어들여서 data frame을 생성한 후 column명을 보니

# AMT17 : 2017 이용금액

# Y17\_CNT : 2017 이용횟수

# column명을 통일 및 정리해주자

df <- rename(excel,

CNT17 = Y17\_CNT,

CNT16 = Y16\_CNT)

df

######################################################################################

# 3. 하나의 data frame에서 하나 이상의 조건을 이용해서 데이터를 추출하려면 어떻게 해야하나

# filter(data frame, 조건1, 조건2, ,,,)

filter(excel,

SEX == "M",

AREA == "서울")

filter(excel,

SEX == "M" & AREA == "서울")

# 모든 사람들 중에 지역이 서울 혹은 경기인 남성들 중 40살 이상의 사람들의 정보?

# or 는 역슬래쉬@@!!

filter(excel,

SEX == "M" & AGE >= 40,

AREA == "서울" | AREA == "경기")

# 모든 사람들 중에 지역이 서울 혹은 경기 혹은 제주인 남성들 중 40살 이상의 사람들의 정보?

# 하나의 column에서 조건이 많아질 때는 %in% 연산자를 사용할 수 있다.

filter(excel,

SEX == "M" & AGE >= 40,

AREA %in% c("서울","경기","지역"))

# 4.arrange(data frame, column1, column2, ,,, )

# 정렬의 기준은 오름차순 ! =default값

# 만약 내림차순으로 정렬하려면 desc() 함수 사용

# 서울, 남자, 2017년 처리 금액이 400,000원 이상인 사람을 나이가 많은 순으로 출력하자

result <- filter(excel,

SEX == "M" & AREA == "서울",

AMT17 >= 400000)

go <- arrange(result,

desc(AGE))

go

# 위를 chaining을 통해 연결해보자

# chaining 기호는 %>% 이고, 이 때 arrange 안에 data frame을 다시 넣지 않아도 된다

result <- filter(excel,

SEX == "M" & AREA == "서울",

AMT17 >= 400000) %>%

arrange(desc(AGE))

result

# 5. select(data frame, column1, column2, ,,, )

# 내가 추출하고자 하는 column을 지정해서 해당 column만 추출할 수 있다.

# # 서울, 남자, 2017년 처리 금액이 400,000원 이상인 사람을 나이가 많은 순으로 id와 나이, 2017년도 처리 건수를 출력하자

result <- filter(excel,

SEX == "M" & AREA == "서울",

AMT17 >= 400000) %>%

arrange(desc(AGE)) %>%

select(ID,AGE,Y17\_CNT)

result

# ID부터 AGE까지만 출력할 때 select("ID:AGE")로 잡아도 출력된다 # 일일히 안쳐도됨

# SEX 만 제외하고 선택하고 싶을 때 select(-"SEX")

# 6. 새로운 column을 생성할 수 있다.

#dataframe$새로운 컬럼명 = "값"

# 17년도 처리 금액(50만 이상)을 기준으로 등급을 vip 나머지는 normal로 => 새로운 column을 생성할 것

# data frame의 column을 생성하는 기본 기능!

excel$GRADE = ifelse(excel$AMT17 >= 500000,

"VIP",

"NORMAL")

View(excel)

# dplyr은 새로운 column을 생성하기 위해 함수를 제공

# mutate(data frame,

# 컬럼명1 = 수식1,

# 컬럼명2 = 수식2, ,,,)

# 경기, 여자 기준 / 2017년도 처리금액을 이용하여 / 처리금액의 10%를 가산한 값으로 / 새로운 컬럼 AMT17\_REAL 만들고 /

# AMT17\_REAL이 45만원 이상인 경우 / VIP 컬럼을 만들어서 TRUE, FALSE로 나타내자

result <- filter(excel,

SEX == "F" & AREA == "경기")%>%

mutate(AMT17\_REAL = AMT17 \* 1.1)

excel$VIP = ifelse(result$AMT17\_REAL >= 450000,

"TRUE",

"FALSE")

View(result)

## 한방에 처리하고 싶을 때

result <- filter(excel,

SEX == "F" & AREA == "경기")%>%

mutate(AMT17\_REAL = AMT17 \* 1.1,

VIP = ifelse(AMT17\_REAL >=450000,TRUE,FALSE)

)

View(result)

# 7. group\_by() & summaries()

# group\_by(dataframe, 조건 ), summarise(data frame, 조건)

df <- filter(excel,

AREA == "서울" & AGE > 30) %>%

group\_by(SEX)%>%

summarise(sum=sum(AMT17), cnt=n())

# sum은 그룹의 합 cnt 는 각각의 개수

df

# 8. plyr package의 join 함수가 dplyr에서는 각 기능별로 독립적인 함수로 제공된다.

# left\_join(data frame1, data frame2, by= key)

# right\_join()

# inner\_join()

# full\_join()

# 9. bind\_rows(df1,df2) : 두 개의 df의 column명이 같아야 행이 제대로 결합된다.

# column명이 같지 않으면 컬럼을 생성해서 결합이 된다.

df1 <- data.frame(x=c("a","b","c"))

df1

df2 <- data.frame(x=c("d","e","f"))

df2

bind\_rows(df1,df2)

df1 <- data.frame(x=c("1","2","3"))

df1

df2 <- data.frame(y=c("1","2","3"))

df2

bind\_rows(df1,df2)

###################################################################################################