



## Day29; 20221017

	날짜
	유형
	태그

GitHub - u8yes/R

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

u8yes/R



<https://github.com/u8yes/R>

A 1 Contributor    0 Issues    ⭐ 1 Star    0 Forks

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/786f8af4-7140-480e-b6f6-529195798de8/06\\_%EB%8D%B0%EC%9D%B4%ED%84%BC\\_%EC%A1%B0%EC%9E%91.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/786f8af4-7140-480e-b6f6-529195798de8/06_%EB%8D%B0%EC%9D%B4%ED%84%BC_%EC%A1%B0%EC%9E%91.pdf)

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/70b52014-5c99-42d0-902a-df5a1a7a3ba4/chap06\\_Data\\_Handling.r](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/70b52014-5c99-42d0-902a-df5a1a7a3ba4/chap06_Data_Handling.r)

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0e3157bb-6f07-4371-86c6-a251efdb8a59/07\\_EDA%C9%980\\_Data\\_%EC%A0%95%EC%A0%9C.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0e3157bb-6f07-4371-86c6-a251efdb8a59/07_EDA%C9%980_Data_%EC%A0%95%EC%A0%9C.pdf)

수집 → 저장 → (전)처리 → 분석

Data - 연속 변수, 이산 변수

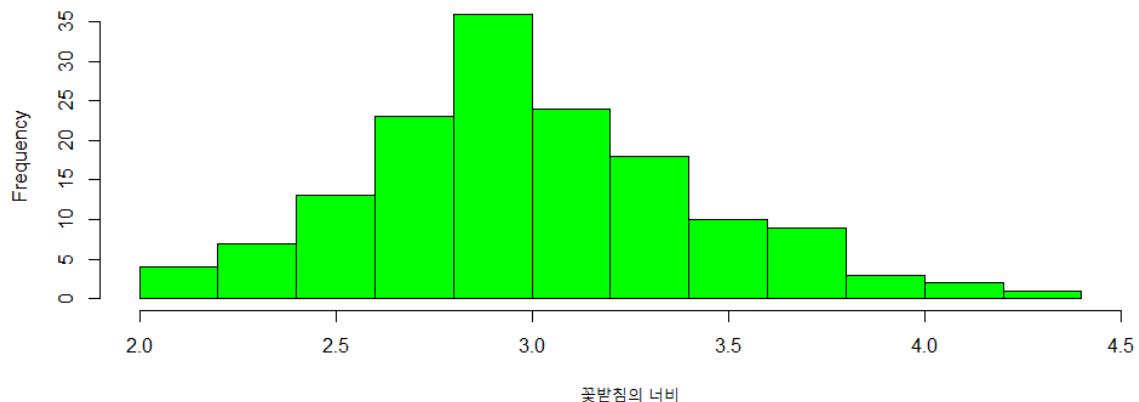
이산 변수 - 정수값(모든 정수값을 의미하지는 않음), (예: 통화 단위)

연속 변수 - 실수값(예: 길이, 키, 몸무게, 시간데이터)

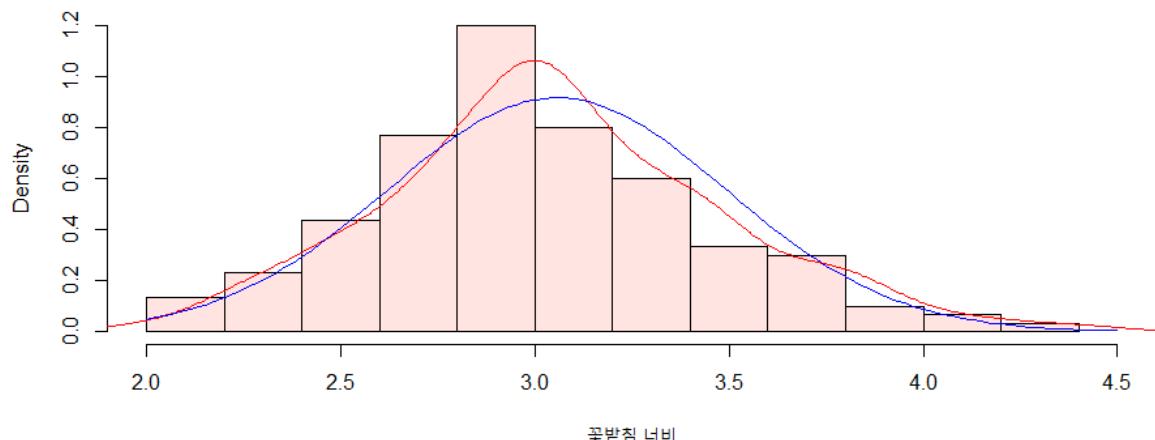
-연속 변수

barplot, dotchart, 박스 플롯, 히스토그램

iris 꽃받침 너비 histogram



iris 꽃받침 너비 histogram



```
# 2) 히스토그램 시각화
#     - 측정값의 범위(구간)를 그래프의 x축으로 놓고, 범위에 속하는 측정값의 빈도수를 y축으로 나타낸 그래프 형태.

data("iris") # iris 데이터 셋 가져오기
head(iris)
table(iris$Species)
#setosa versicolor virginica
# 50      50      50

names(iris)
# "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"

summary(iris$Sepal.Width) # 기술 통계량

hist(iris$Sepal.Width, xlab = "꽃받침의 너비",
     col="green", xlim=c(2.0, 4.5),
     main="iris 꽃받침 너비 histogram") # xlim(2 ~ 4.5까지 지정)
```

```

summary(iris$Sepal.Length)

hist(iris$Sepal.Length, xlab = "꽃받침의 길이",
      col="mistyrose", xlim=c(4.0, 8.0),
      main="iris 꽃받침 너비 histogram")

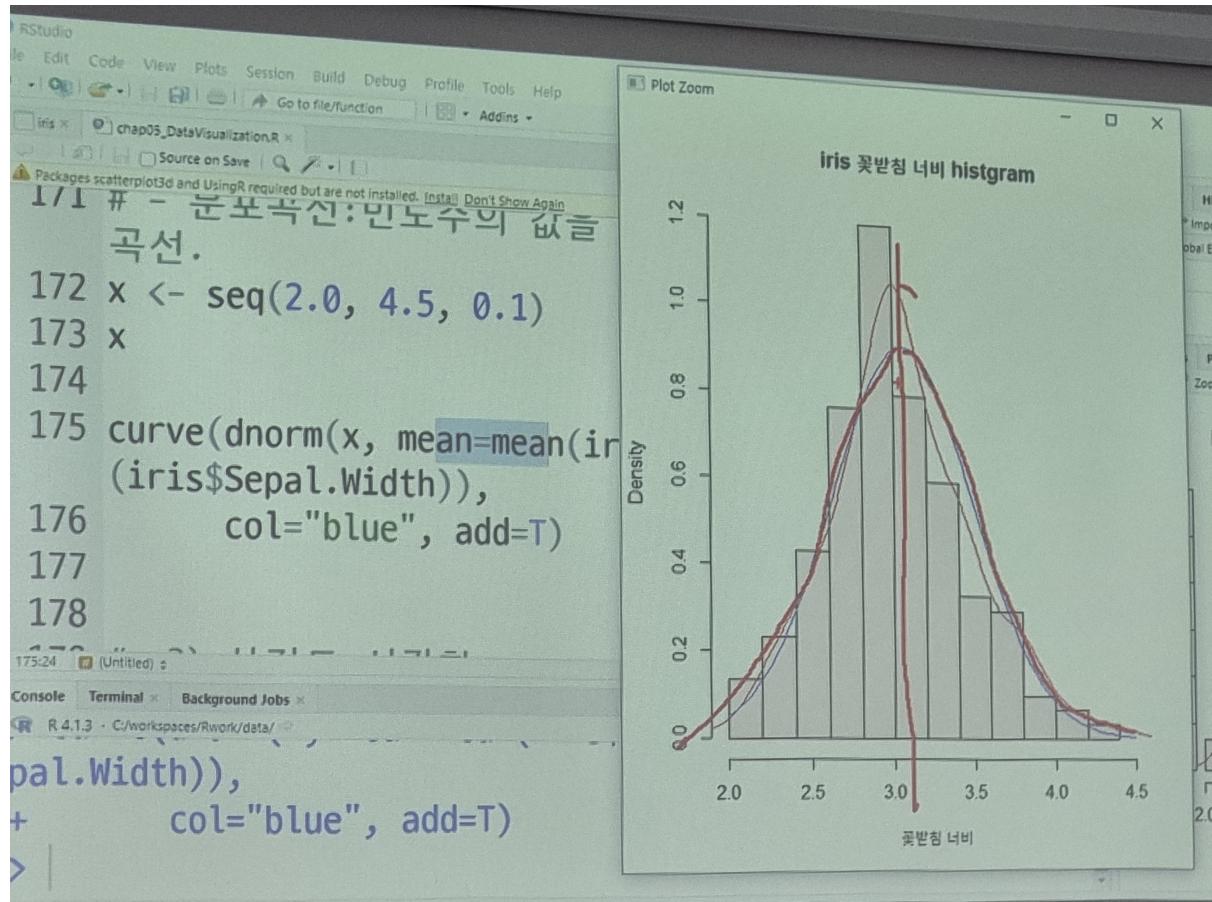
# 확률 밀도로 히스토그램 그리기 - 연속형변수의 확률.
hist(iris$Sepal.Width, xlab = "꽃받침 너비",
      col="mistyrose", xlim=c(2.0, 4.5),
      main="iris 꽃받침 너비 histogram", freq=F) # y축을 빈도수가 아닌 밀도로 계산을 해서 y축을 설정

# 밀도를 기준으로 line을 그려준다.
lines(density(iris$Sepal.Width), col="red")

# 정규분포곡선 추가
# - 분포곡선: 빈도수의 값을 선으로 연결하여 얻어진 곡선.
x <- seq(2.0, 4.5, 0.1) # 0.1단위로 2~4.5까지 연속적인 데이터셋을 반환
x

curve(dnorm(x, mean=mean(iris$Sepal.Width), sd=sd(iris$Sepal.Width)),
      col="blue", add=T) # width값이 파란색(blue)

```



```

160
161 # 확률 밀도로 히스토그램 그리기 - 연속형변수의 확률.
162 hist(iris$Sepal.Width, xlab = "꽃받침 너비",
163       col="mistyrose", xlim=c(2.0, 4.5),
164       main="iris 꽃받침 너비 histogram", freq=F) # y축을 빈도수가 아닌 밀도로 계산을
165       해서 y축을 설정
166 hist
167 ◆ hist      {graphics}
168 ◆ hist.default {graphics}
169 ◆ history     {utils}
170 P HistData::
171
172 # - 분포곡선:빈도수의 값을 선으로 연결하여 얹어진 곡선.
173 x <- seq(2.0, 4.5, 0.1) # 0.1단위로 2~ 4.5까지 연속적인 데이터셋을 반환
174 x
175
176 curve(dnorm(x, mean=mean(iris$Sepal.Width), sd=sd(iris$Sepal.Width)),
177         col="blue", add=T) # width값이 파란색(blue)
178

```

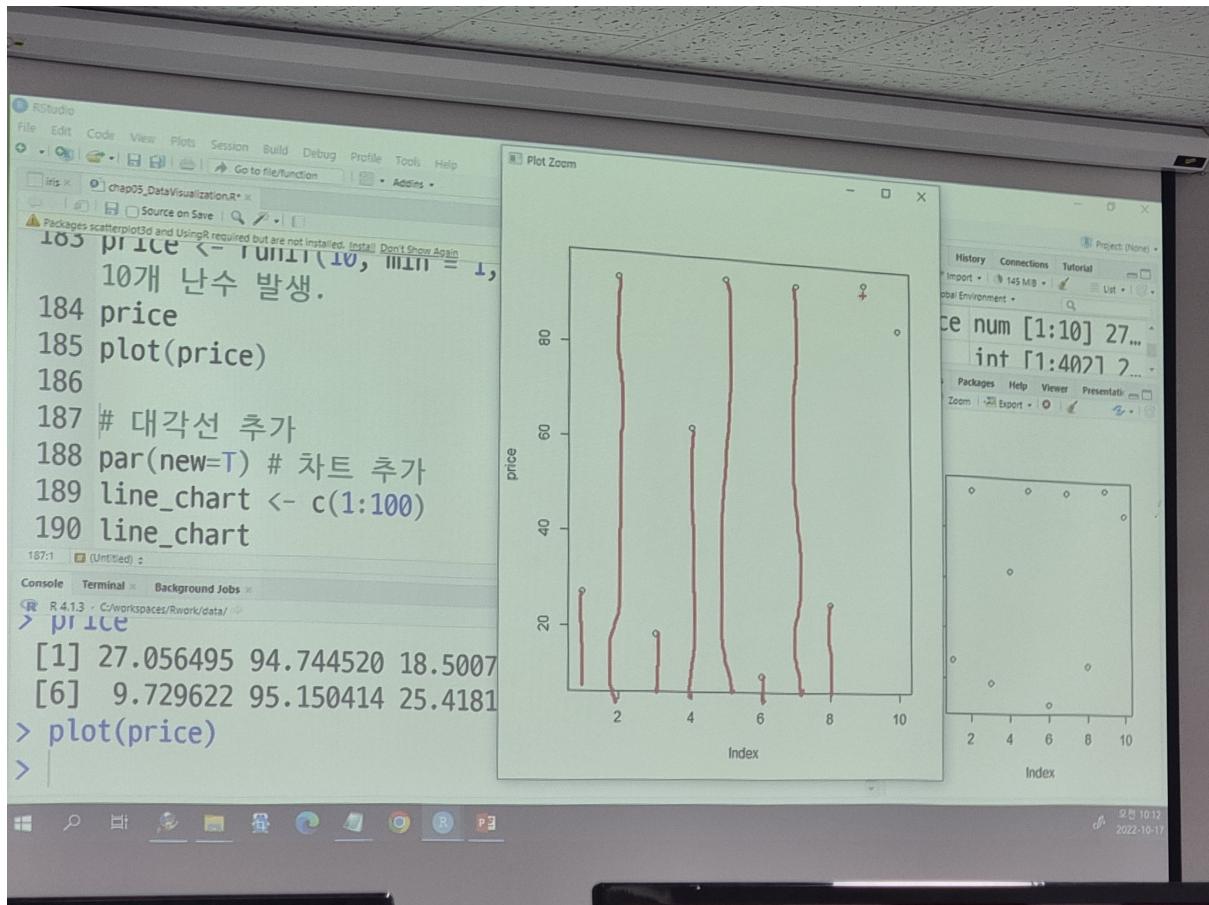
graphics를 R에서 기본적으로 제공

```

# 기본 산점도 시각화
price <- runif(10, min = 1, max = 100) # 1~100 사이의 10개 난수 발생.
price # 난수를 확인 가능
plot(price)

```

각 난수 데이터들

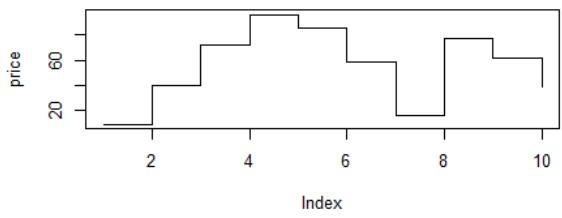
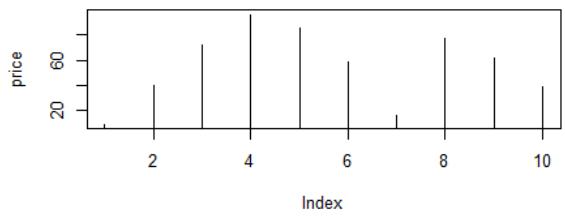
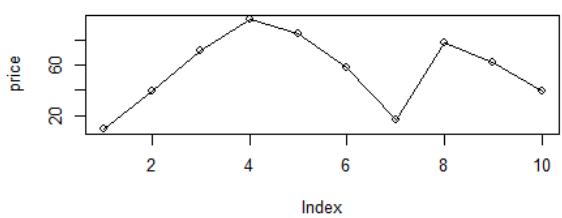
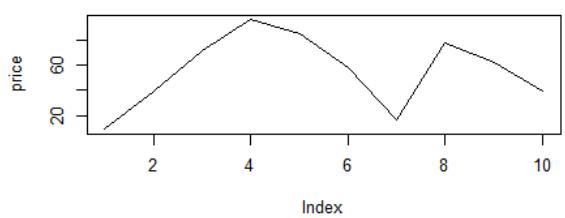


```
# 3) 산점도 시각화
#     - 두 개 이상의 변수들 사이의 분포를 점으로 표시한 차트를 의미.

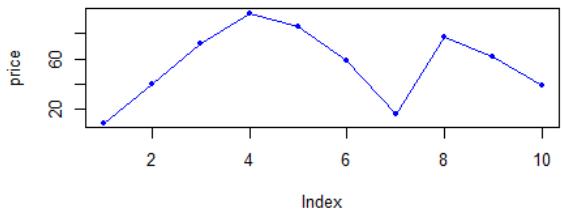
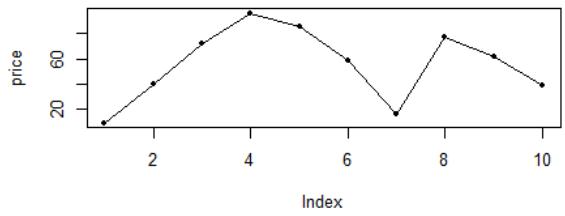
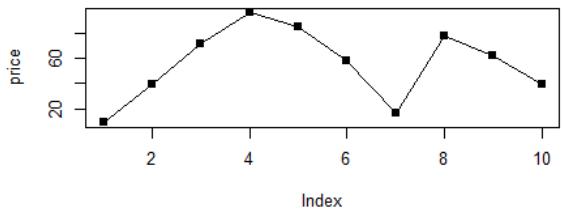
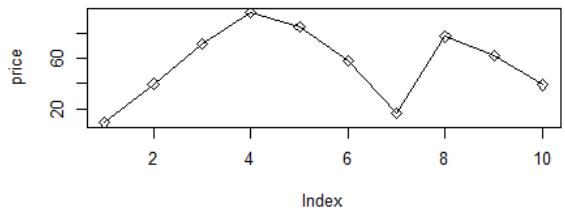
# 기본 산점도 시각화
price <- runif(10, min = 1, max = 100) # 1~100 사이의 10개 난수 발생.
price # 난수를 확인 가능
plot(price)

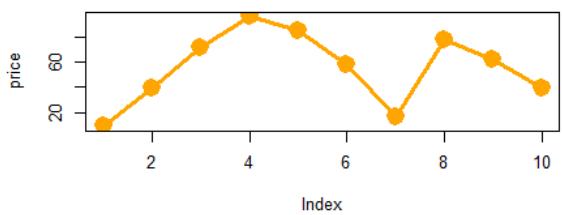
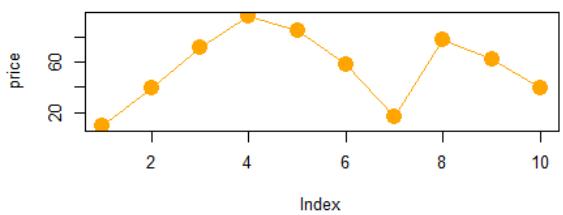
# 대각선 추가
par(new=T) # 차트 추가
line_chart <- c(1:100)
line_chart
plot(line_chart, type = "l", col="red", axes = F, ann = F)

# 텍스트 추가
text(70, 80, "대각선 추가", col="blue")
```

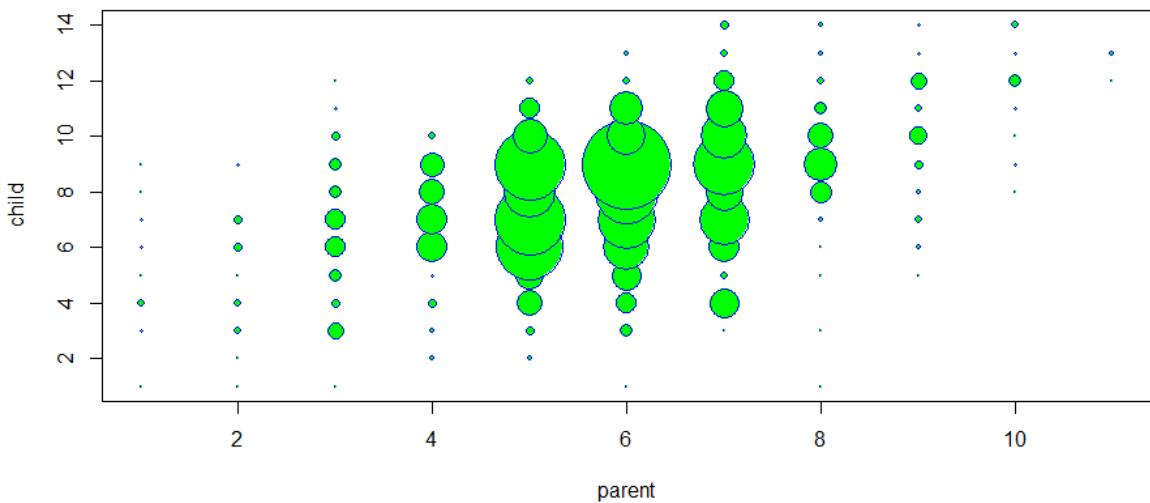


```
# type 속성으로 그리기
par(mfrow=c(2,2)) # 2행 2열 차트 그리기.
plot(price, type = "l") # 유형: 실선
plot(price, type = "o") # 유형: 원형과 실선(원형통과)
plot(price, type = "h") # 유형: 직선
plot(price, type = "s") # 유형: 꺽은선
```





```
# pch 속성으로 그리기
plot(price, type="o", pch=5) # 빈 사각형
plot(price, type="o", pch=15) # 채워진 사각형
plot(price, type="o", pch=20) # 채워진 원형
plot(price, type="o", pch=20, col="blue")
plot(price, type="o", pch=20, col="orange", cex=3.0)
plot(price, type="o", pch=20, col="orange", cex=3.0, lwd=3) # lwd:line width
```



```
# galton 데이터셋 대상 종복 자료 시작화
# galton 데이터셋 가져오기
install.packages("UsingR")
library(UsingR)
data("galton") # 자식과 부모의 키 차이
head(galton) # 빈도수 값을 6번째까지 보여줌
```

```

str(galton) # 'data.frame': 928 obs. of  2 variables: # 928 가정을 대상으로 부모와 자식의 키 차이 데이터 제공
class(galton) # "data.frame"

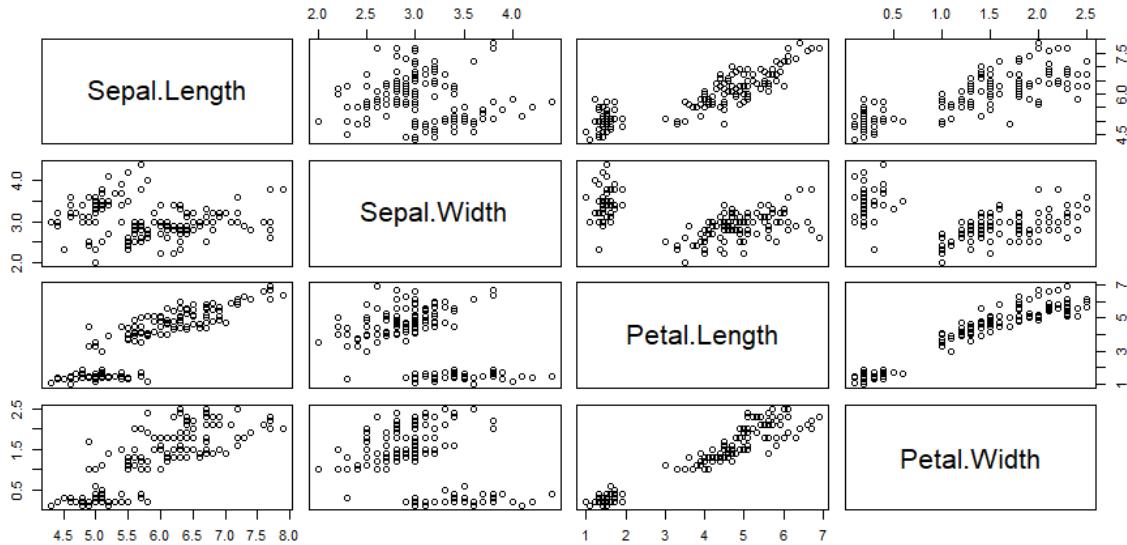
# 데이터프레임으로 변환
galtonData <- as.data.frame(table(galton$child, galton$parent))
head(galtonData)
str(galtonData) # 'data.frame': 154 obs. of  3 variables:

# 컬럼 단위 추출
names(galtonData) <- c("child", "parent", "freq") # 컬럼에 이름 지정.
head(galtonData)

parent <- as.numeric(galtonData$parent)
child <- as.numeric(galtonData$child)

# 점의 크기 확대
plot(parent, child, pch=21, col="blue", bg="green",
      xlab = "parent", ylab = "child",
      cex = 0.2 * galtonData$freq) # 빈도수가 많을수록 0.2가 곱해짐
# plot ch캐릭터(pch)는 21일수록 더 원의 형태로 보여줌

```



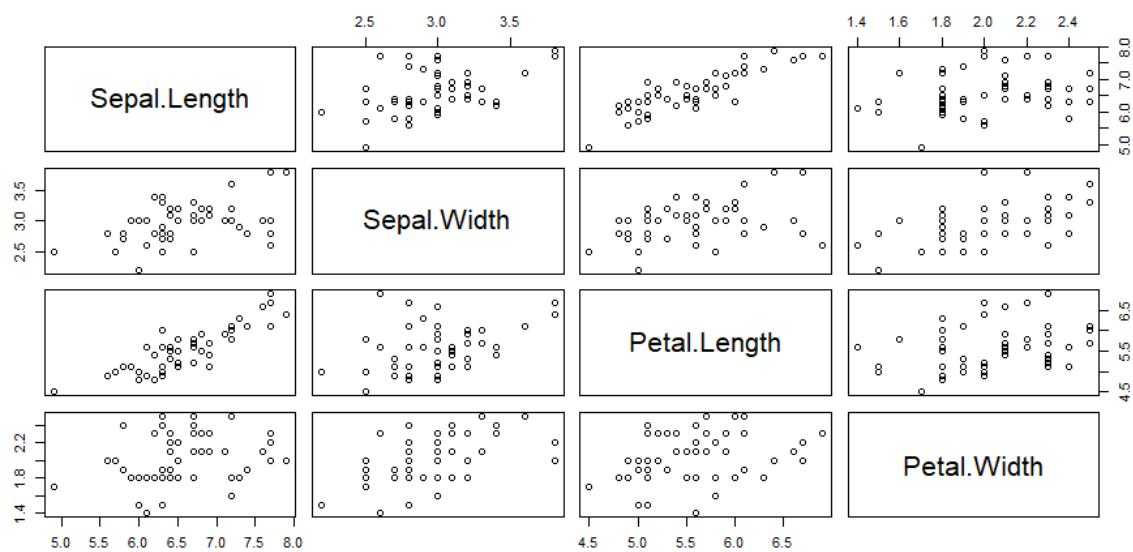
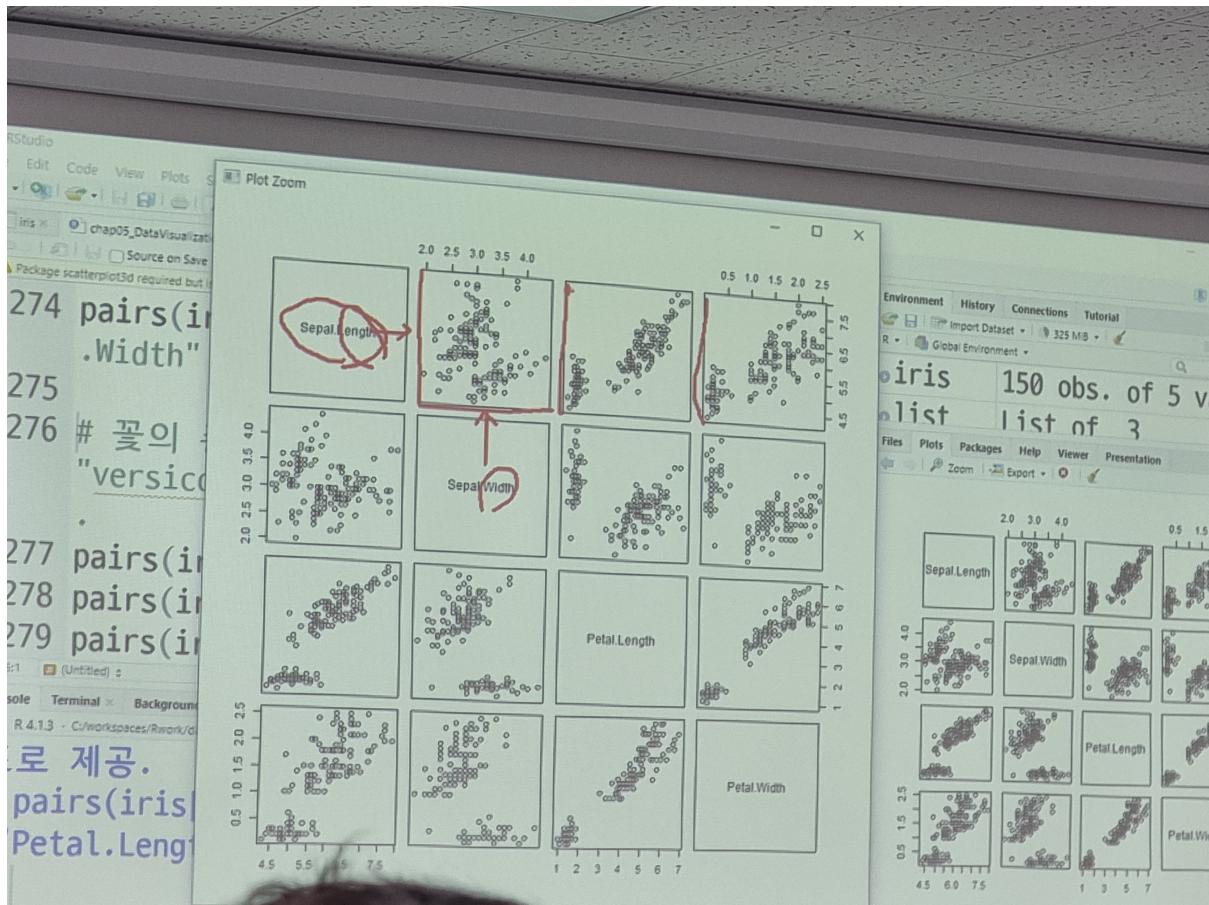
```

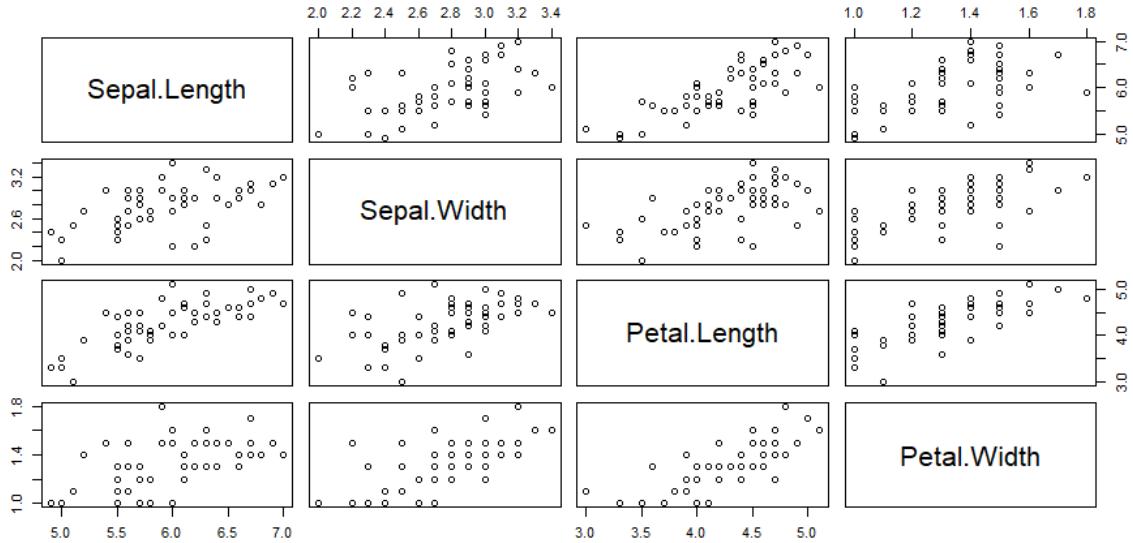
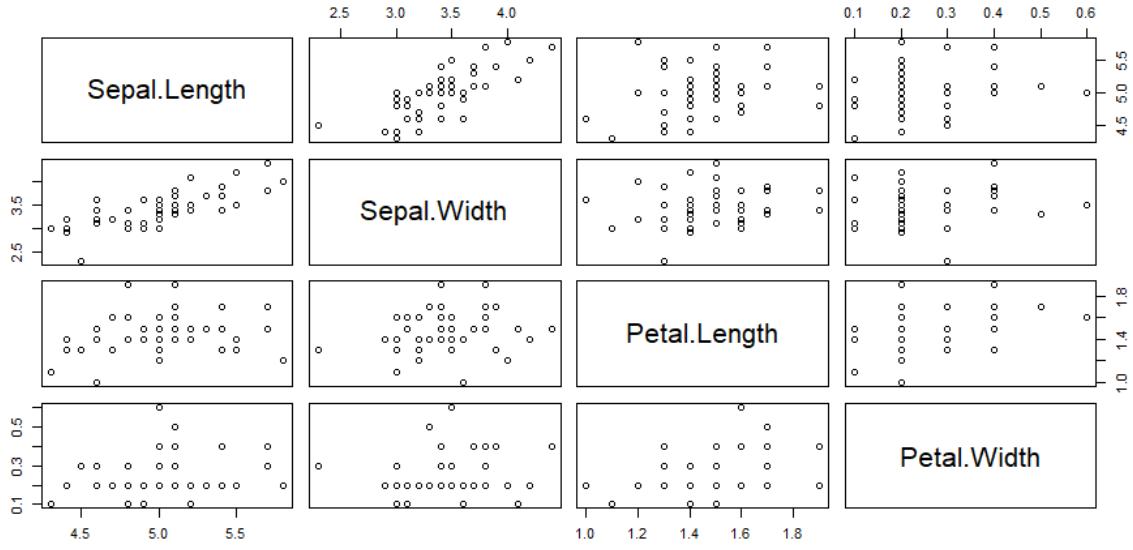
# 5) 변수 간의 비교 시각화
# iris 4개 변수의 상호 비교
attributes(iris)
data(iris)
help(pairs)

# matrix 또는 데이터프레임의 numeric 컬럼을 대상으로 변수들 사이의 비교 결과를 행렬구조의 분산된 그래프로 제공.
pairs(iris[,1:4]) # "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"

# 꽃의 종류가 "virginica"와 "setosa", "versicolor"를 종별 대상으로 4개 변수 상호 비교.
pairs(iris[iris$Species=="virginica", 1:4])
pairs(iris[iris$Species=="setosa", 1:4])
pairs(iris[iris$Species=="versicolor", 1:4])

```





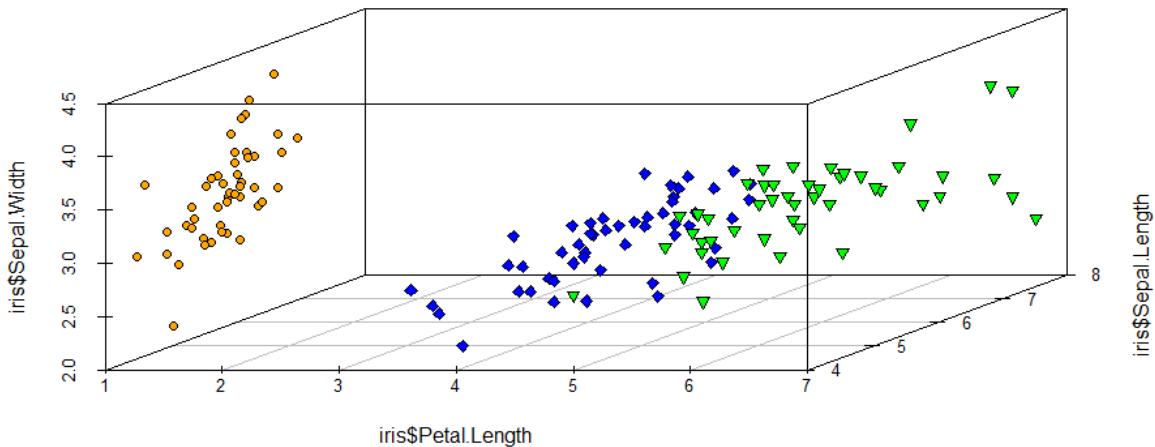
```

# 5) 변수 간의 비교 시각화
# iris 4개 변수의 상호 비교
attributes(iris)
data(iris)
help(pairs)

# matrix 또는 데이터프레임의 numeric 컬럼을 대상으로 변수들 사이의 비교 결과를 행렬구조의 분산된 그래프로 제공.
pairs(iris[,1:4]) # "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"

# 꽃의 종류가 "virginica"와 "setosa", "versicolor"를 종별 대상으로 4개 변수 상호 비교.
pairs(iris$Species=="virginica", 1:4)
pairs(iris$Species=="setosa", 1:4)
pairs(iris$Species=="versicolor", 1:4)

```



```

# 3차원 산점도 시각화
# 패키지 설치 및 로딩
install.packages("scatterplot3d")
library(scatterplot3d)

# Factor의 levels 보기
levels(iris$Species) # "setosa" "versicolor" "virginica"

# 봉꽃의 종류별 분류
iris_setosa <- iris[iris$Species == 'setosa', ]
iris_versicolor <- iris[iris$Species == 'versicolor', ]
iris_virginica <- iris[iris$Species == 'virginica', ]

# 3차원 틀 생성
# scatterplot3d(밀변, 오른쪽변 컬럼명, 左쪽변 컬럼명, type)

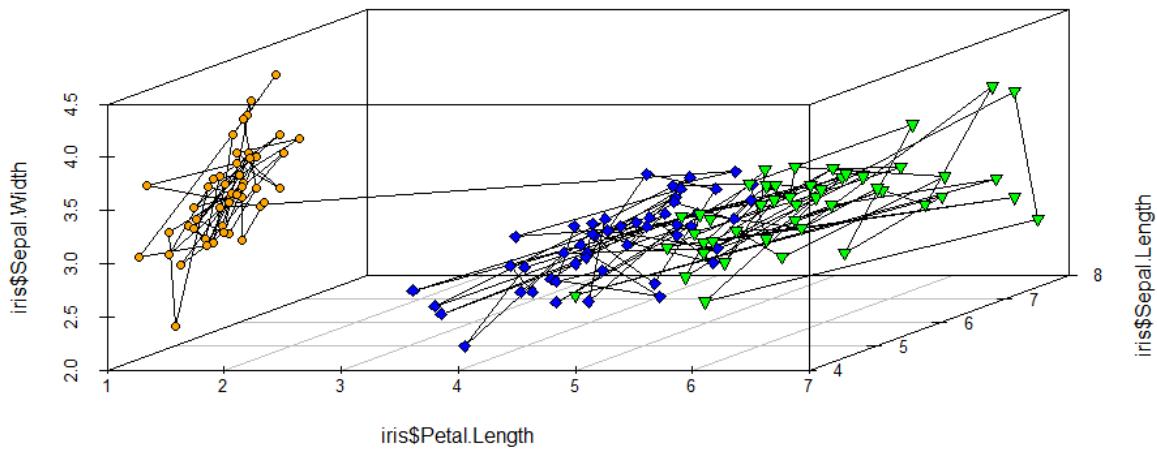
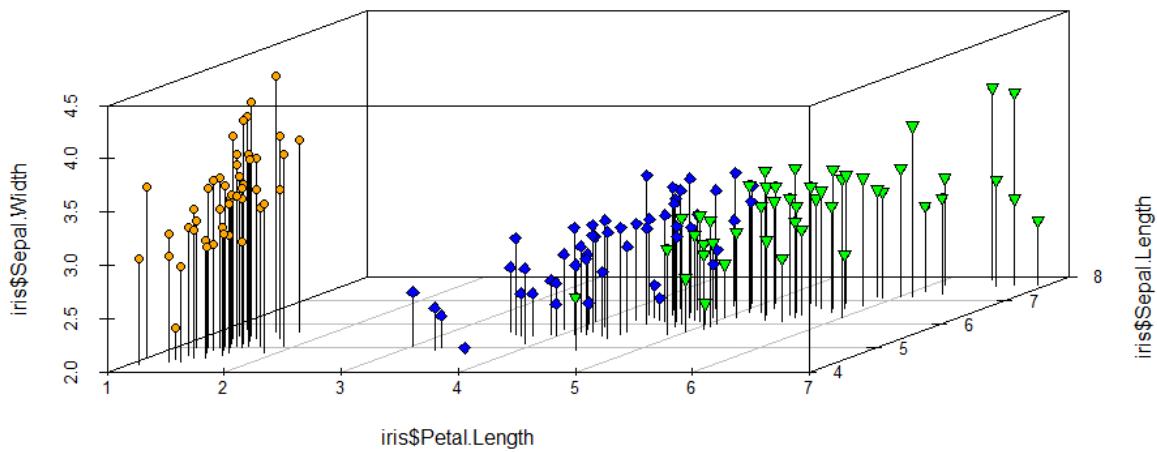
d3 <- scatterplot3d(iris$Petal.Length, iris$Sepal.Length, iris$Sepal.Width, type = 'n') # type : h->수직선, p->점, l->선

# 3차원 산점도 시각화
d3$points3d(iris_setosa$Petal.Length,
            iris_setosa$Sepal.Length,
            iris_setosa$Sepal.Width,
            bg='orange', pch=21) # 타원 산점도

d3$points3d(iris_versicolor$Petal.Length,
            iris_versicolor$Sepal.Length,
            iris_versicolor$Sepal.Width,
            bg='blue', pch=23) # 마름모꼴 산점도

d3$points3d(iris_virginica$Petal.Length,
            iris_virginica$Sepal.Length,
            iris_virginica$Sepal.Width,
            bg='green', pch=25) # 역삼각형 산점도

```



## chap05\_DataVisualization.R

```
# chap05_DataVisualization

#####
# chapter05. 데이터 시각화
#####

# 이산변수와 연속변수 시각화

# 1. 이산변수(discrete quantitative data) 시각화
#   - 정수 단위로 나누어 측정할 수 있는 변수.

# 1) 막대차트 시각화 - barplot() 함수
#   (1) 세로 막대차트

# 막대 차트 데이터 생성
```

```

chart_data <- c(305, 450, 320, 460, 330, 480, 380, 520) # 정수 데이터
names(chart_data) <- c("2019 1분기", "2020 1분기", "2019 2분기", "2020 2분기", "2019 3분기", "2020 3분기", "2019 4분기", "2020 4분기")

str(chart_data)
chart_data

# 세로 막대 차트
help("barplot")
help(barplot)
barplot(chart_data, ylim = c(0, 600), col = rainbow(8), main = "2019년도 vs 2020년도 분기별 매출현황 비교")

barplot(chart_data, ylim = c(0, 600), col = rainbow(8),
       ylab = "매출액(단위:억원)", xlab = "년도별 분기현황",
       main = "2019년도 vs 2020년도 분기별 매출현황 비교")
# barplot(vector, ylim(0부터 600까지만), col=rainbow(자동 8개의 색상으로), main = 타이틀을 붙어준다. )

# (2) 가로 막대 차트
barplot(chart_data, xlim = c(0, 600),
        ylab = "년도별 분기현황", xlab = "매출액(단위:억원)",
        col = rainbow(8),
        main = "2019년도 vs 2020년도 분기별 매출현황 비교",
        horiz = T)

barplot(chart_data, xlim = c(0, 600),
        ylab = "년도별 분기현황", xlab = "매출액(단위:억원)",
        col = rainbow(8),
        main = "2019년도 vs 2020년도 분기별 매출현황 비교",
        horiz = T, space=1.5) # space = 크기의 간격

barplot(chart_data, xlim = c(0, 600),
        ylab = "년도별 분기현황", xlab = "매출액(단위:억원)",
        col = rainbow(8),
        main = "2019년도 vs 2020년도 분기별 매출현황 비교",
        horiz = T, space=1.5, cex.names=0.8) # character expand 문자 확장 = cex

# red와 blue 색상 4회 반복
barplot(chart_data, xlim = c(0, 600),
        ylab = "년도별 분기현황", xlab = "매출액(단위:억원)",
        main = "2019년도 vs 2020년도 분기별 매출현황 비교",
        horiz = T, space=1.5, cex.names=0.8,
        col=rep(c(2, 4), 4)) # repeat 함수 2:빨간색 4:파란색을 4번 반복하라.
# col=rep(c(2, 4), ) : 검은색(1), 빨간색(2), 초록색(3), 파란색(4), 하늘색(5), 자주색(6), 노란색(7)

barplot(chart_data, xlim = c(0, 600),
        ylab = "년도별 분기현황", xlab = "매출액(단위:억원)",
        main = "2019년도 vs 2020년도 분기별 매출현황 비교",
        horiz = T, space=1.5, cex.names=0.8,
        col=rep(c("green", "yellow"), 4)) # repeat 4번 반복

# 누적 막대 차트
data("VADeaths")
VADeaths

str(VADEaths)
mode(VADEaths) # "numeric"
class(VADEaths) # "matrix" "array" 2차원 배열

# 개별 차트와 누적 차트 그리기

# 누적 차트
par(mfrow=c(1,2)) # 1행 2열 그래프 보기

barplot(VADEaths, col=rainbow(5),
        main = "미국 버지니아주 하위계층 사망 비율")

legend(3.8, 200,
       c("50-54세", "55-59세", "60-64세", "65-69세", "70-74세"),
       cex = 0.8, fill = rainbow(5)) # 3.8 위치

# 개별 차트
barplot(VADEaths, col=rainbow(5),
        main = "미국 버지니아주 하위계층 사망 비율",
        beside=T) # 개별적으로 나누어서 시각화

legend(19, 71,
       c("50-54세", "55-59세", "60-64세", "65-69세", "70-74세"),
       cex = 0.8, fill = rainbow(5))

# 2) 점 차트 시각화 - dotchart()
help("dotchart")

par(mfrow=c(1,1)) # 1행1열 그래프 보기

dotchart(chart_data, color = c("black", "red"),
         xlab = "매출액(단위:억원)",
         cex=1.2,

```

```

main="분기별 판매현황 점 차트 시각화",
labels=names(chart_data))

dotchart(chart_data, color = c("black", "red"),
         xlab = "매출액(단위:억원)",
         cex=1.2,
         main="분기별 판매현황 점 차트 시각화",
         labels=names(chart_data),
         lcolor="blue", pch=2:3)
# pch(plotting character): 원(1), 삼각형(2), +(3)
# cex(character expansion): 레이블과 점의 크기 확대 역할.

# 3) 원형 차트 시각화 - pie() 함수
help(pie)

pie(chart_data, labels = names(chart_data),
     border = 'blue', col=rainbow(8), cex=4)
title("2019~2020년도 분기별 매출현황")

# 2. 연속변수(Continuous quantitative data) 시각화 # Day25; 20221011
# - 시간, 길이 등과 같은 연속성을 가진 변수.

# 1) 상자 그래프 시각화 : 요약정보를 시각화하는데 효과적. 특히 데이터의 분포 정도와 이상치 발견을 목적으로 하는 경우 유용.
help(boxplot)
par(mfrow=c(1,2))
boxplot(VADeaths) # 상자그래프 시각화.
boxplot(VADeaths, range=0)
# range=0:최소값과 최대값 사이를 점선으로 연결하는 역할.

abline(h = 37, lty=3, col="red") # 기준선 추가(lty=3 : 점선)

summary(VADeaths) # median이 2사분위

# 2) 히스토그램 시각화
# - 측정값의 범위(구간)를 그래프의 x축으로 놓고, 범위에 속하는 측정값의 빈도수를 y축으로 나타낸 그래프 형태.

data("iris") # iris 데이터 셋 가져오기
head(iris)
table(iris$Species)
#setosa versicolor virginica
# 50 50 50

names(iris)
# "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"

summary(iris$Sepal.Width) # 기술 통계량

hist(iris$Sepal.Width, xlab = "꽃받침의 너비",
     col="green", xlim=c(2.0, 4.5),
     main="iris 꽃받침 너비 histogram") # xlim(2 ~ 4.5까지 지정)

summary(iris$Sepal.Length)

hist(iris$Sepal.Length, xlab = "꽃받침의 길이",
     col="mistyrose", xlim=c(4.0, 8.0),
     main="iris 꽃받침 너비 histogram")

# 확률 밀도로 히스토그램 그리기 - 연속형변수의 확률.
hist(iris$Sepal.Width, xlab = "꽃받침 너비",
     col="mistyrose", xlim=c(2.0, 4.5),
     main="iris 꽃받침 너비 histogram", freq=F) # y축을 빈도수가 아닌 밀도로 계산을 해서 y축을 설정

# 밀도를 기준으로 line을 그려준다.
lines(density(iris$Sepal.Width), col="red")

# 정규분포곡선 추가
# - 분포곡선: 빈도수의 값을 선으로 연결하여 얻어진 곡선.
x <- seq(2.0, 4.5, 0.1) # 0.1단위로 2~ 4.5까지 연속적인 데이터셋을 반환
x

curve(dnorm(x, mean=mean(iris$Sepal.Width), sd=sd(iris$Sepal.Width)),
      col="blue", add=T) # width값이 파란색(blue)

# 3) 산점도 시각화
# - 두 개 이상의 변수들 사이의 분포를 점으로 표시한 차트를 의미.

# 기본 산점도 시각화
price <- runif(10, min = 1, max = 100) # 1~100 사이의 10개 난수 발생.
price # 난수를 확인 가능
plot(price)

# 대각선 추가
par(new=T) # 차트 추가
line_chart <- c(1:100)
line_chart

```

```

plot(line_chart, type = "l", col="red", axes = F, ann = F)

# 텍스트 추가
text(70, 80, "대각선 추가", col="blue")

# type 속성으로 그리기
par(mfrow=c(2,2)) # 2행 2열 차트 그리기.
plot(price, type = "l") # 유형:실선
plot(price, type = "o") # 유형:원형과 실선(원형통과)
plot(price, type = "h") # 유형:직선
plot(price, type = "s") # 유형:꺽은선

# pch 속성으로 그리기
plot(price, type="o", pch=5) # 빈 사각형
plot(price, type="o", pch=15) # 채워진 사각형
plot(price, type="o", pch=20) # 채워진 원형
plot(price, type="o", pch=20, col="blue")
plot(price, type="o", pch=20, col="orange", cex=3.0) #
plot(price, type="o", pch=20, col="orange", cex=3.0, lwd=3) # lwd:line width

# 4) 중첩 자료 시각화
# 중복된 자료의 수 만큼 점의 크기 확대하기
par(mfrow=c(1, 1)) # 1행 1열

# 두 개의 벡터 객체
x <- c(1, 2, 3, 4, 2, 4)
y <- rep(2, 6)
x; y

# 교차테이블 작성
table(x); table(y)
table(x, y)

# 산점도 시각화
plot(x, y)

# 데이터프레임 생성
xy.df <- as.data.frame(table(x, y))
xy.df

# 좌표에 중복된 수 만큼 점 확대
plot(x, y, pch=15, col="blue",
      xlab = "x 벡터 원소",
      ylab = "y 벡터 원소",
      cex = 0.8 * xy.df$Freq) # cex 문자 캐릭터 expansion

# galton 데이터 셋 대상 중복 자료 시각화
# galton 데이터 셋 가져오기
install.packages("UsingR")
library(UsingR)
data("galton") # 자식과 부모의 키 차이
head(galton) # 빈도수 값은 6번째까지 보여줌
str(galton) # 'data.frame': 928 obs. of  2 variables: # 928 가정을 대상으로 부모와 자식의 키 차이 데이터 제공
class(galton) # "data.frame"

# 데이터프레임으로 변환
galtonData <- as.data.frame(table(galton$child, galton$parent))
head(galtonData)
str(galtonData) # 'data.frame': 154 obs. of  3 variables:

# 컬럼 단위 추출
names(galtonData) <- c("child", "parent", "freq") # 컬럼에 이름 지정.
head(galtonData)

parent <- as.numeric(galtonData$parent)
child <- as.numeric(galtonData$child)

# 점의 크기 확대
plot(parent, child, pch=21, col="blue", bg="green",
      xlab = "parent", ylab = "child",
      cex = 0.2 * galtonData$freq) # 빈도수가 많을수록 0.2가 곱해짐
# plot ch캐릭터(pch)는 21일수록 더 원의 형태로 보여줌

# 5) 변수 간의 비교 시각화
# iris 4개 변수의 상호 비교
attributes(iris)
data(iris)
help(pairs)

# matrix 또는 데이터프레임의 numeric 컬럼을 대상으로 변수들 사이의 비교 결과를 행렬구조의 분산된 그래프로 제공.
pairs(iris[,1:4]) # "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"

# 꽃의 종류가 "virginica"와 "setosa", "versicolor"를 종별 대상으로 4개 변수 상호 비교.
pairs(iris[iris$Species=="virginica", 1:4])

```

```

pairs(iris[iris$Species=="setosa", 1:4])
pairs(iris[iris$Species=="versicolor", 1:4])

# 3차원 산점도 시각화
# 페키지 설치 및 로딩
install.packages("scatterplot3d")
library(scatterplot3d)

# Factor의 levels 보기
levels(iris$Species) # "setosa" "versicolor" "virginica"

# 봇꽃의 종류별 분류
iris_setosa <- iris[iris$Species == 'setosa', ]
iris_versicolor <- iris[iris$Species == 'versicolor', ]
iris_virginica <- iris[iris$Species == 'virginica', ]

# 3차원 틀 생성
# scatterplot3d(밀변, 오른쪽변 컬럼명, 왼쪽변 컬럼명, type)

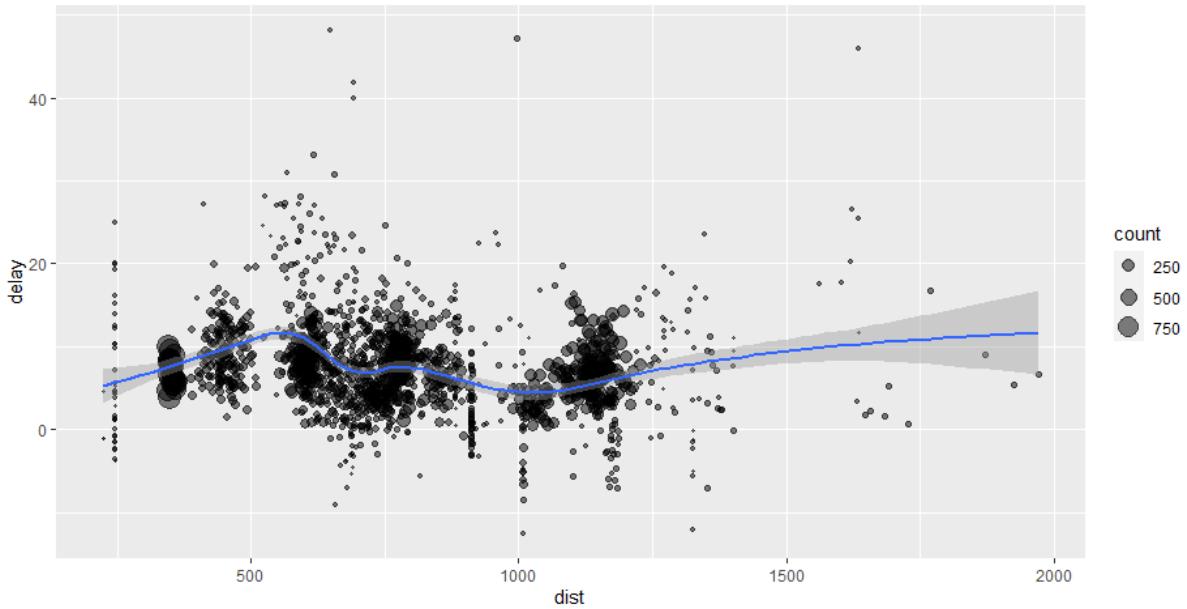
d3 <- scatterplot3d(iris$Petal.Length, iris$Sepal.Length, iris$Sepal.Width, type = 'n') # type : h->수직선, p->점, l->선

# 3차원 산점도 시각화
d3$points3d(iris_setosa$Petal.Length,
            iris_setosa$Sepal.Length,
            iris_setosa$Sepal.Width,
            bg='orange', pch=21) # 타원 산점도

d3$points3d(iris_versicolor$Petal.Length,
            iris_versicolor$Sepal.Length,
            iris_versicolor$Sepal.Width,
            bg='blue', pch=23) # 마름모꼴 산점도

d3$points3d(iris_virginica$Petal.Length,
            iris_virginica$Sepal.Length,
            iris_virginica$Sepal.Width,
            bg='green', pch=25) # 역삼각형 산점도

```



```

install.packages("ggplot2")
library(ggplot2)
ggplot(delay, aes(dist, delay)) +
  geom_point(aes(size=count), alpha=1/2) +
  geom_smooth() +
  scale_size_area()

```

```

# chap06_DataHandling

#####
# chapter06. 데이터 조작
#####

## 1. dplyr 패키지 활용
# - 기존 plyr 패키지는 R 언어로 개발되었으나(인터프리터 방식 언어 - 바로 실행되는 구조, 인간 중심이라서 성능이 떨어짐), dplyr 패키지는 C++언어로 개발되어 처리 속도
# c++ 언어는 Java와 80% 정도가 유사함, 둘 다 C 언어에서 나옴.

install.packages(c("dplyr", "hflights")) # 2개를 설치할 수 있음
library(dplyr)
library(hflights)

hflights
str(hflights) # 'data.frame': 227496 obs. of  21 variables:

# 1.1 콘솔 창의 크기에 맞게 데이터 추출
#   : 콘솔 창 안에서 한 눈으로 파악하기
hflights_df <-tbl_df(hflights) # tbl_df 가독성있게 열로 분류해서 보여줌
hflights_df

# 1.2 조건에 맞는 데이터 필터링

# hflights_df를 대상으로 1월2일 데이터 추출하기.
filter(hflights_df, Month==1 & DayofMonth==2) # AND
# 678 x 21

# 1월 혹은 2월 데이터 추출
filter(hflights_df, Month==1 | Month==2) # OR
# 36,038 x 21

# 1.3 컬럼으로 데이터 정렬
# 년, 월, 출발시간, 도착시간 순으로 오름차순 정렬
arrange(hflights_df, Year, Month, DepTime, ArrTime)

# 년, 월, 출발시간, 도착시간 순으로 내림차순 정렬
arrange(hflights_df, Year, Month, desc(DepTime), ArrTime)

# 1.4 컬럼으로 데이터 검색
# hflights_df에서 년, 월, 출발시간, 도착시간 컬럼 검색하기.
select(hflights_df, Year, Month, DepTime, ArrTime) # 4개의 컬럼(열) 선택.

# 컬럼의 범위 지정하기.
select(hflights_df, Year:ArrTime) # 행은 다 보여줌

# 컬럼의 범위 제외 : Year부터 DayOfWeek 제외
select(hflights_df, -(Year:DayOfWeek)) # '-'는 제외하는 것을 의미

# 1.5 데이터 셋에 컬럼 추가

# 출발 지연 시간과 도착 지연 시간과의 차이를 계산하는 컬럼 추가하기.
mutate(hflights_df, gain = ArrDelay - DepDelay,
       gain_per_hour = gain/(AirTime/60))

# mutate() 함수에 의해서 추가된 컬럼 보기
select(mutate(hflights_df, gain = ArrDelay - DepDelay,
              gain_per_hour = gain/(AirTime/60)),
       Year, Month, ArrDelay, DepDelay, gain, gain_per_hour)

# 1.6 요약 통계치 계산

# 비행시간 평균 계산하기.
summarise(hflights_df, avgAirTime=mean(AirTime, na.rm = T))

# 데이터 셋의 관측치 길이, 출발 지연 시간 평균 구하기
summarise(hflights_df, cnt=n(), delay=mean(DepDelay, na.rm = T))

# 도착시간(ArrTime)의 표준편차와 분산 계산하기
summarise(hflights_df, arrTimeSd=sd(ArrTime, na.rm = T),
           arrTimeVar=var(ArrTime, na.rm = T))

# 1.7 집단변수 대상 그룹화

# 집단변수를 이용하여 그룹화하기
species <- group_by(iris, Species)
str(species)
species

planes <- group_by(hflights_df, TailNum) # TailNum(항공기 일련번호)
delay <- summarise(planes, count=n(), dist=mean(Distance, na.rm = T), delay=mean(ArrDelay, na.rm = T))

```



```

View(wide)

# 파일 저장 및 읽기
setwd("D:/heaven_dev/workspaces/R/output")
write.csv(wide, 'wide.csv', row.names = F)

wide_read <- read.csv('wide.csv')
colnames(wide_read) <- c('id', 'day1', 'day2', 'day3', 'day4', 'day5', 'day6', 'day7')
wide_read

# 2.2 넓은 형식을 긴 형식으로 변경

# melt() 함수 이용
long <- melt(wide_read, id='id') # id를 기준으로 긴 형식으로 변경해주라
long

# 컬럼명 수정
colnames(long) <- c("id", "Date", "Buy")
head(long)

# reshape2 패키지의 smiths 데이터 셋 구조 변경하기
data("smiths")
smiths

# wide -> long
long <- melt(smiths, id=1:2)
long

# long -> wide
dcast(long, subject + time ~ ...)

# 2.3 3차원 배열 형식으로 변경

# airquality 데이터 셋 구조 변경
data("airquality") # New York의 대기에 대한 질
View(airquality)
str(airquality) # 'data.frame': 153 obs. of  6 variables:

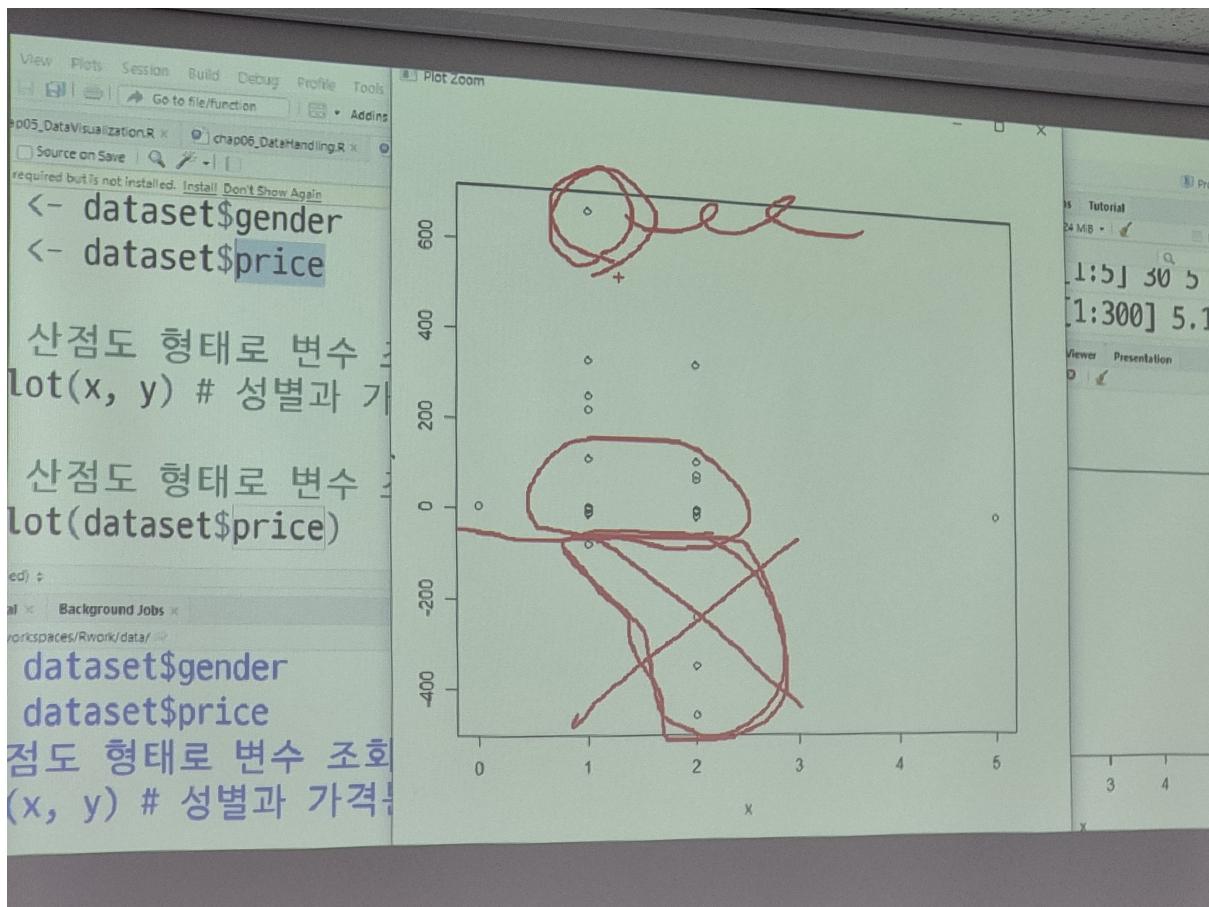
# 컬럼명 대문자 일괄 변경
names(airquality) <- toupper(names(airquality)) # 컬럼명 대문자 변경.
head(airquality)

# 월과 일 컬럼으로 나머지 4개 컬럼을 끌어서 긴 형식 변경
air_melt <- melt(airquality, id=c("MONTH", "DAY"), na.rm = T)
head(air_melt) # MONTH DAY variable value

# 일과 월 컬럼으로 variable 컬럼을 3차원 형식으로 분류
names(air_melt) <- tolower(names(air_melt)) # 컬럼명 소문자 변경
acast <- acast(air_melt, day-month-variable) # 3차원 구조 변경 # day행month열variable면
acast
class(acast) # "array"

# 월 단위 variable(대기관련 컬럼) 컬럼 합계
acast(air_melt, month-variable, sum) # array cast = acast?

```



이상치 - 값이 음수, 성별이 5나 0으로

```

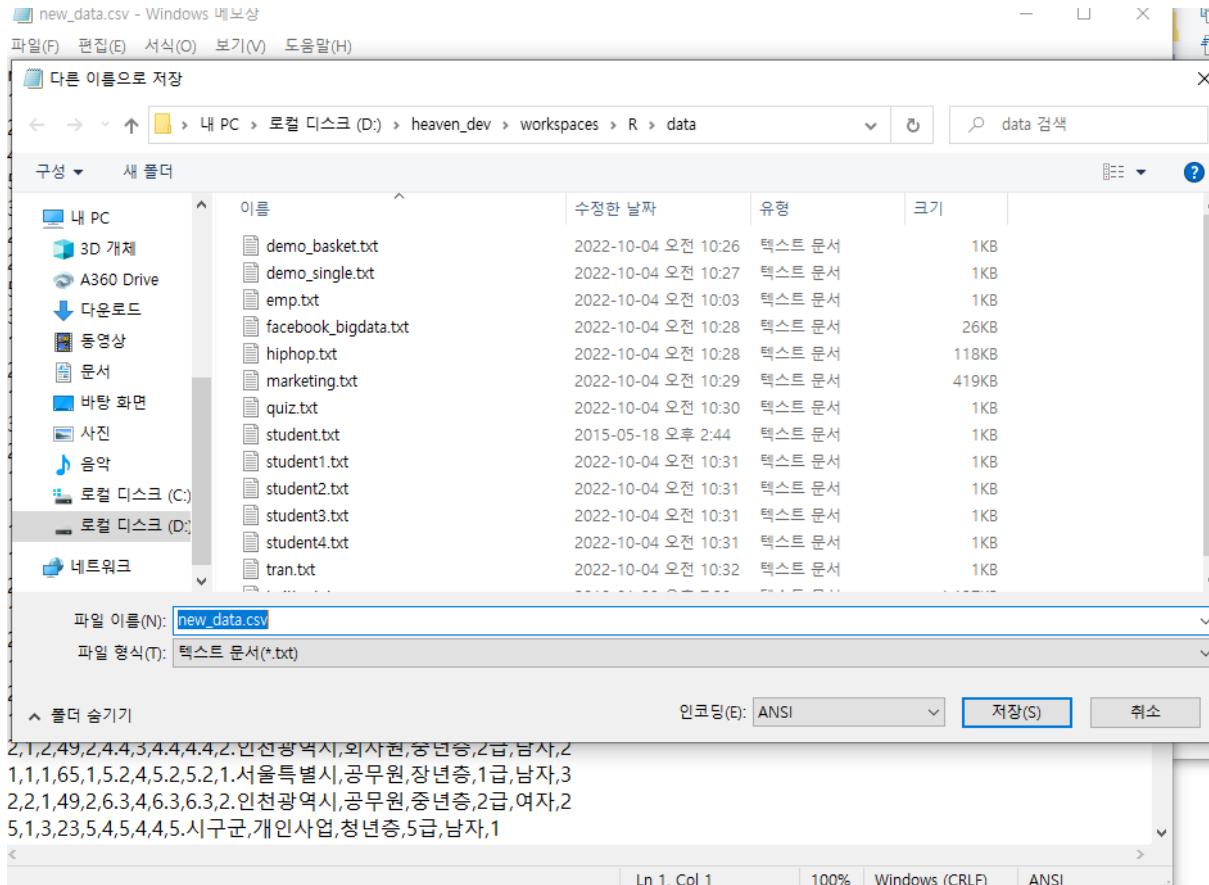
2 | 133
2 |
3 | 0000003344
3 | 55555888999
4 | 000000000000001111111122233334444
4 | 566666777777889999
5 | 0000000000000000111111111222222223333334444
5 | 5555555566667777888899
6 | 0000000000000011111112222222223333333333444444444
6 | 55557777777788889999
7 | 000111122
7 | 777799

```

```

# price 변수의 데이터 정제와 시각화
dataset2 <- subset(dataset, price >= 2 & price <= 8)
length(dataset2$price) # 251
stem(dataset2$price) # 줄기와 잎 도표 보기

```



```

# chap07_EDA&DataPreprocessing

#####
# chapter07. EDA & 데이터 전처리
#####

# - 자료분석에 필요한 데이터를 대상으로 불필요한 데이터를 처리하는 필터링과 전처리 방법에 대해서 알아본다.

# 1. EDA(Exploratory Data Analysis) - 탐색적 자료 분석
#   : 수집한 자료를 다양한 각도에서 관찰하고 이해하는 과정으로 그래프나 통계적 방법을 이용해서 자료를 직관적으로 파악하는 과정.

# 1.1 데이터 셋 보기

# 데이터 가져오기
setwd("D:/heaven_dev/workspaces/R/data")
dataset <- read.csv("dataset.csv", header = T) # 헤더가 있는 경우
View(dataset)

# 1) 데이터 조회
#   - 탐색적 데이터 분석을 위한 데이터 조회

# 전체 데이터 보기
print(dataset) # 콘솔창 출력
View(dataset) # utils package, 뷰어창 출력

# 데이터의 앞부분과 뒷부분 보기
head(dataset, 10)
tail(dataset) # 마지막 6개 데이터

# 1.2 데이터 셋 구조 보기

# 데이터 셋 구조
names(dataset) # 변수명(컬럼) # feature이름 출력
attributes(dataset) # $names / $class / $row.names
str(dataset) # 데이터 구조보기(자료구조/관측치(행)/컬럼명(열) / 자료형)

```

```

# 1.3 데이터 셋 조회
dataset$age # 데이터 셋 접근 방법.
dataset$resident

length(dataset) # 7 : 컬럼의 갯수
length(dataset$age) # 300 : 행(데이터)의 갯수

# 조회 결과 변수 저장
x <- dataset$gender
y <- dataset$price

# 산점도 형태로 변수 조회
plot(x, y) # 성별과 가격분포-극단치 발견

# 산점도 형태로 변수 조회
plot(dataset$price)

# ["컬럼명"] 형식으로 특정 변수 조회
dataset["gender"] # dataset$gender
dataset["price"]

# [색인(index)] 형식으로 변수 조회
dataset[2] # 두번째 컬럼(gender) - 출력형태: 열 중심
dataset[6] # price
dataset[3,] # 3행 전체 # 3 NA 1 2 41 4 4.7 4
dataset[,3] # 3열 전체 # job

# 두 개 이상의 [색인(index)] 형식으로 변수 조회
dataset[c("job", "price")]
dataset[c(2, 6)] # gender / price

dataset[c(1, 2, 3)] # resident/gender/job
dataset[c(1:3)]
dataset[1:3]

dataset[c(2,4:6,3,1)] # gender age position price job resident # 순서를 바꿔서 가져올 수도 있다.
dataset[-c(2)] # dataset[c(1, 3:7)] # gender항목만 빠져있음

# dataset의 특정 행/열을 조회하는 경우
dataset[,c(2:4)]
dataset[c(2:4),]
dataset[-c(1:100),]

# 2. 결측치(NA) 처리
# 2.1 결측치 확인

# summary() 함수 이용
summary(dataset$price)
#   Min. 1st Qu. Median   Mean 3rd Qu.   Max.   NA's
# -457.200  4.425  5.400  8.752  6.300  675.000    30

sum(dataset$price) # NA

# 2.2 결측치 제거

# sum() 함수에서 제공되는 속성 이용
sum(dataset$price, na.rm = T) # 2362.9 # na rm(ove)

# 결측데이터 제거 함수 이용
price2 <- na.omit(dataset$price)
sum(price2)
length(price2) # 270

# 2.3 결측치 대체 # 결측치 - 담겨져있지 않은 데이터

# 결측치를 0으로 대체하기
x <- dataset$price # price vector 생성
head(x)
dataset$price2 <- ifelse(!is.na(x), x, 0) # 0으로 대체
View(dataset)
sum(dataset$price2) # 2362.9

# 결측치를 평균으로 대체하기
x <- dataset$price # price vector 생성
head(x)
dataset$price3 <- ifelse(!is.na(x), x, round(mean(x,na.rm=T),2)) # 평균으로 대체 # 소수점 2째자리까지만 return해줌.
View(dataset)
sum(dataset$price2) # 2362.9

# 3. 이상치(극단치) 처리 # 이상치 - 범위를 벗어나는 데이터
# 3.1 범주형 변수 이상치 처리

# 범주형 변수의 이상치 확인
table(dataset$gender)
# 0 1 2 5 : (범주)

```

```

# 2 173 124 1 : (빈도수)

pie(table(dataset$gender)) # 파이차트

# subset() 함수를 이용한 데이터 정제하기
dataset <- subset(dataset, gender==1 | gender==2) # 바로 이상치 데이터를 제외할 수가 있다
dataset
length(dataset$gender) # 297
table(dataset$gender)
pie(table(dataset$gender))
#1 2
#173 124

# 3.2 연속형 변수의 이상치 처리.
dataset <- read.csv('dataset.csv', header = T)
View(dataset)
dataset$price # 세부데이터 보기
plot(dataset$price)
summary(dataset$price)

# price 변수의 데이터 정제와 시각화
dataset2 <- subset(dataset, price >= 2 & price <= 8)
length(dataset2$price) # 251
stem(dataset2$price) # 줄기와 잎 도표 보기

# age 변수에서 NA 발견
summary(dataset2$age) # NA's -> 16
boxplot(dataset2$age)

# 4. 코딩 변경

# 4.1 가독성을 위한 코딩 변경
table(dataset2$resident)
View(dataset2)

dataset2$resident2[dataset2$resident == 1] <- '1. 서울특별시'
dataset2$resident2[dataset2$resident == 2] <- '2. 인천광역시'
dataset2$resident2[dataset2$resident == 3] <- '3. 대전광역시'
dataset2$resident2[dataset2$resident == 4] <- '4. 대구광역시'
dataset2$resident2[dataset2$resident == 5] <- '5. 시구군'
dataset2[c("resident","resident2")]
View(dataset2)

# job 컬럼을 대상으로 코딩 변경하기
dataset2$job2[dataset2$job == 1] <- '공무원'
dataset2$job2[dataset2$job == 2] <- '회사원'
dataset2$job2[dataset2$job == 3] <- '개인사업'
dataset2[c("job","job2")]
View(dataset2)

# 4.2 척도 변경을 위한 코딩 변경

# 나이(age) 변수를 청년층, 중년층, 장년층으로 코딩 변경하기.
dataset2$age2[dataset2$age <= 30] <- "청년층"
dataset2$age2[dataset2$age > 30 & dataset2$age <= 55] <- "중년층"
dataset2$age2[dataset2$age > 55] <- "장년층"

# 4.3 역코딩을 위한 코딩 변경

# 만족도(survey)를 긍정순서로 역코딩
survey <- dataset2$survey
csurvey <- 6-survey # 역코딩 # 만족과 불만족 점수를 뒤바꿔서 만족을 강조하고자 6을 빼줌
csurvey

dataset2$survey2 <- csurvey
mean(dataset2$survey2, na.rm = T) # 3.358566 # 5점대가 만점

# 5. 탐색적 분석을 위한 시각화

# 5.1 범주형 vs 범주형
getwd()
setwd("D:/heaven_dev/workspaces/R/data")
new_data <- read.csv("new_data.csv", header = T)
View(new_data)

# 범주형(resident) vs 범주형(gender) 데이터 분포 시각화 # Day30;

## 성별에 따른 거주지역 분포 현황
resident_gender <- table(new_data$resident2, new_data$gender2)
resident_gender

barplot(resident_gender, beside = T, horiz = F,
       col=rainbow(5),
       legend=row.names(resident_gender),
       main="성별에 따른 거주지역 분포 현황")

## 거주지역에 따른 성별 분포 현황

```

```

gender_resident <- table(new_data$gender2, new_data$resident2)
gender_resident

barplot(gender_resident, beside = T, horiz = F,
        col=rainbow(2),
        legend=row.names(gender_resident),
        main="거주지역에 따른 성별 분포 현황")

barplot(gender_resident, beside = T, horiz = T,
        col=rainbow(2),
        legend=row.names(gender_resident),
        main="거주지역에 따른 성별 분포 현황")

barplot(gender_resident, beside = F, horiz = T,
        col=rainbow(2),
        legend=row.names(gender_resident),
        main="거주지역에 따른 성별 분포 현황")

barplot(gender_resident, beside = F, horiz = F,
        col=rainbow(2),
        legend=row.names(gender_resident),
        main="거주지역에 따른 성별 분포 현황")

# 5.2 연속형 vs 범주형

# 나이(age/연속형) vs 직업(job2/범주형) 데이터 분포 시각화
install.packages("lattice") # chap08
library(lattice)

# 직업유형에 따른 나이 분포 현황
?densityplot
densityplot(~ age, data=new_data, groups = job2,
            plot.points=T, auto.key = T)
# plot.points=F: 밀도점 표시 여부(x), auto.key=T: 범례

# 5.3 연속형 vs 범주형 vs 범주형

# price(연속형) vs gender(범주형) vs position(범주형) 데이터 분포 시각화

# (1) 성별에 따른 직급별 구매비용 분포 현황 분석
densityplot(~ price|factor(gender2), data=new_data,
            groups = position2,
            plot.points=T, auto.key = T)

# (2) 직급에 따른 성별 구매비용 분포 현황 분석
densityplot(~ price|factor(position2), data=new_data,
            groups = gender2,
            plot.points=T, auto.key = T)

# 5.4 연속형 vs 연속형 vs 범주형

# price(연속형) vs age(연속형) vs gender2(범주형)
xyplot(price ~ age|factor(gender2), data=new_data)

# 6. 파생변수 생성

# 6.1 더미(dummy) 형식으로 파생변수 생성

# 데이터 파일 가져오기
getwd()
setwd("C:/workspaces/Rwork/src/data")

user_data <- read.csv('user_data.csv', header = T)
View(user_data)
table(user_data$house_type)
# 1 2 3 4
# 32 47 21 300

# 더미변수 생성
# 주택유형(단독주택, 빌라) : 0, 아파트 유형(아파트, 오피스텔) : 1
house_type2 <- ifelse(user_data$house_type==1 | user_data$house_type==2, 0, 1)
house_type2[1:10]
head(house_type2, 10)

# 파생변수 추가
user_data$house_type2 <- house_type2

# 6.2 1:N -> 1:1 관계로 파생변수 생성

# 고객 식별번호(user_id) vs 상품유형(product_type)간의 1:1 파생변수 생성

# 데이터 파일 가져오기
pay_data <- read.csv('pay_data.csv', header = T)
View(pay_data)

```

```

table(pay_data$product_type)
# 1   2   3   4   5
# 55  82  89 104  70

# 고객별 상품유형에 따른 구매금액 합계 파생변수 생성
library(reshape2) # 구조 변경을 위한 패키지 로딩.
product_price <- dcast(pay_data,user_id ~ product_type, sum, na.rm=T)
View(product_price)

# 컬럼명 수정
names(product_price) <- c('user_id', '식료품(1)', '생활용품(2)', '의류(3)', '잡화(4)', '기타(5)')
head(product_price) # 컬럼명 수정 확인.

# 고객별 지불유형에 따른 구매상품 개수 파생변수 생성
pay_price <- dcast(pay_data,user_id ~ pay_method, length)
View(pay_price)

# 6.3 파생변수 합치기

# 고객 정보 테이블에서 파생변수 추가
install.packages("plyr")
library(plyr)
user_pay_data <- join(user_data, product_price, by='user_id')
View(user_pay_data)

# 병합(위에 결과)된 데이터를 대상으로 고객별 지불 유형에 다른 구매상품 개수 병합하기
user_pay_data <- join(user_pay_data, pay_price, by='user_id')
View(user_pay_data)

# 7. 표본 샘플링

# 7.1 정제(cleaning) 데이터 저장하기
View(user_pay_data)

write.csv(user_pay_data, "cleanData.csv", quote = F, row.names = F)

data <- read.csv("cleanData.csv", header = T)
View(data)

# 7.2 표본 샘플링

# 표본 추출하기
nrow(data) # 400, data의 행수 구하기(Number of Rows)
choice1 <- sample(nrow(data), 30) # 30개 무작위 추출
choice1
data[choice1, 1]

# 50~data 길이 사이에서 30개 무작위 추출
choice2 <- sample(50:nrow(data), 30)
choice2

# 다양한 범위를 지정해서 무작위 샘플링
choice3 <- sample(c(10:50, 80:150, 160:190), 30)
choice3

# iris 데이터 셋을 대상으로 7:3 비율로 데이터 셋 생성.
data("iris")
dim(iris) # 150  5

idx <- sample(1:nrow(iris), nrow(iris) * 0.7)
training <- iris[idx, ] # 학습데이터 셋
testing <- iris[-idx, ] # 검정데이터 셋

dim(training) # 105  5
dim(testing) # 45  5

# 7.3 교차 검정 샘플링
# - 평가의 신뢰도를 높이기 위해 동일한 데이터 셋을 N등분하여 N-1개의 학습데이터 모델을 생성하고, 나머지 1개를 검정데이터로 이용하여 모델을 평가하는 방식.

# 데이터 셋을 대상으로 K겹(fold) 교차 검정 데이터 셋 생성.
name <- c('a','b','c','d','e','f')
score <- c(90, 85, 70, 85, 60, 74)
df <- data.frame(Name=name, Score=score)
df

# 교차 검정을 위한 패키지 설치
install.packages("cvTools")
library(cvTools)

cross <- cvFolds(n=6, K=3, R=1, type = "random")
cross

str(cross)

# which를 이용하여 subsets 데이터 참조

```

```
cross$subsets[cross$which == 1, 1] # k=1인 경우
cross$subsets[cross$which == 2, 1] # k=2인 경우
cross$subsets[cross$which == 3, 1] # k=3인 경우

r <- 1 # 1회전
K <- 1:3 # 3겹(fold)
for(k in K){ # 3회전
  datas_idx <- cross$subsets[cross$which==k,r]
  cat('k=', k, '검정데이터\n')
  print(df[datas_idx,])

  cat('훈련데이터\n')
  print(df[-datas_idx,])
}
```