

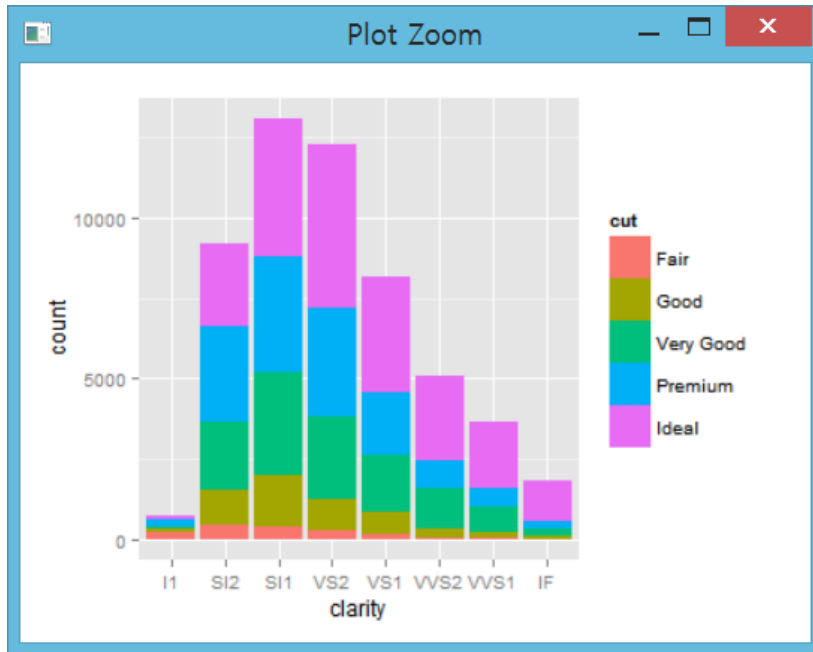
고급 시각화 분석

lattice 패키지

➤ Lattice Plotting System 특징

- ✓ 다차원 데이터를 사용할 경우, 한 번에 여러 개의 plot 생성 가능
- ✓ 높은 밀도의 plot를 효과적으로 그려준다.
- ✓ 직교형태의 그래픽(Trellis graphic) 생성
- ✓ lattice package 제공 함수
 - histogram(), densityplot(), barchart(), dotplot(), xyplot(), equal.count(), cloud()

lattice 패키지



lattice 패키지

➤ lattice package 및 데이터 셋 설치

```
install.packages("lattice")
```

```
library(lattice)
```

```
install.packages("mlmRev")
```

```
library(mlmRev)
```

```
data(Chem97)
```

```
str(Chem97) # data.frame': 31022 obs. of 8 variables:
```

```
head(Chem97,30) # 앞쪽 30개 레코드
```

```
Chem97
```

```
##### Chem97 데이터 셋 설명 #####
```

```
# - mlmRev 패키지에서 제공
```

```
# - 1997년 영국 2,280개 학교 31,022명을 대상으로 A레벨(대학시험) 화학점수
```

```
# - lea 변수 : Local Education Authority(지방교육청:1~131)
```

```
# - score 변수 : A레벨 화학점수(0,2,4,6,8,10)
```

```
# - gender 변수 : 성별
```

```
# - age : 18.5세 기준 월수(범위: -6~+5)
```

```
# - gcsescore 변수 : 고등학교 재학중에 치르는 큰 시험(GCSE : 중등교수학능력정 인증시험)
```

```
# - GCSE : General Certificate of Secondary Education)
```

```
#####
```

lattice 패키지

1. histogram(~x축, dataframe)

```
histogram(~gcsescore, data=Chem97)
```

```
# gcsescore변수를 대상으로 백분율 적용 히스토그램
```

```
# score를 조건변수로 지정(score 단위 적용)
```

```
histogram(~gcsescore | score, data=Chem97)
```

```
histogram(~gcsescore | factor(score), data=Chem97)
```

```
# score를 요인으로 적용
```

```
#####
```

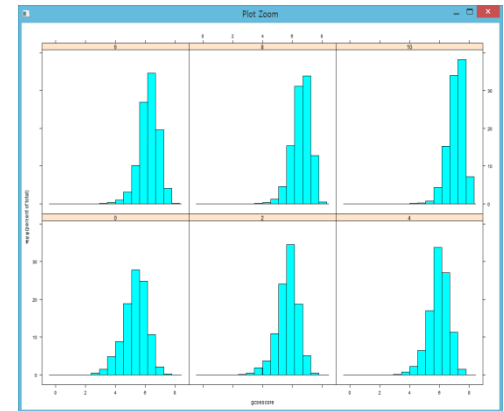
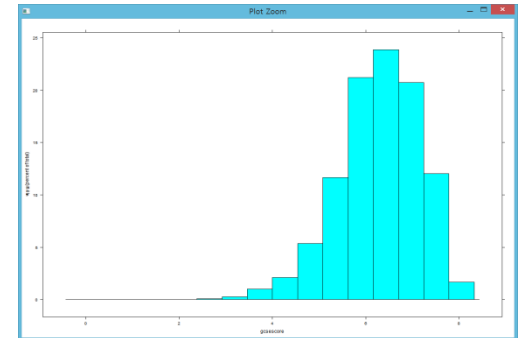
```
# 요인(factor) 수준은 그래픽 출력에 영향을 미친다.
```

```
# 예를들면, score를 x축에 대입할 경우 score가 출력되지만,
```

```
# factor로 변환된 score를 x축에 대입하면, 요인수준이
```

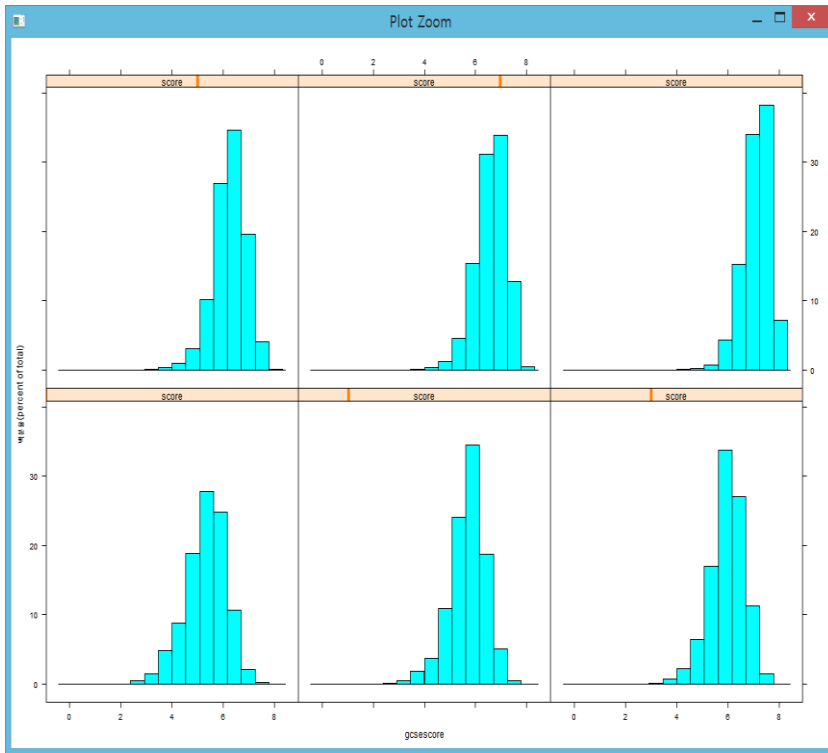
```
# 순서로 적용된다.(0,2,4,6,8,10)
```

```
#####
```

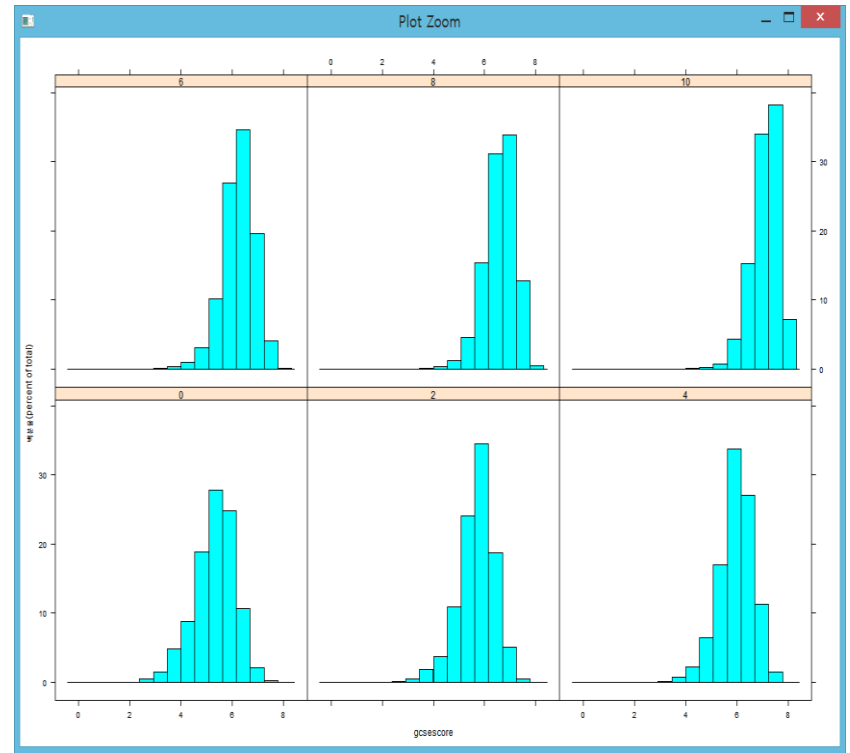


lattice 패키지

score를 조건변수로 지정



score를 요인으로 적용



lattice 패키지

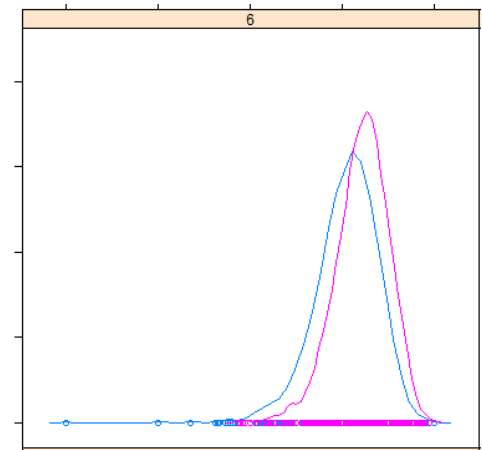
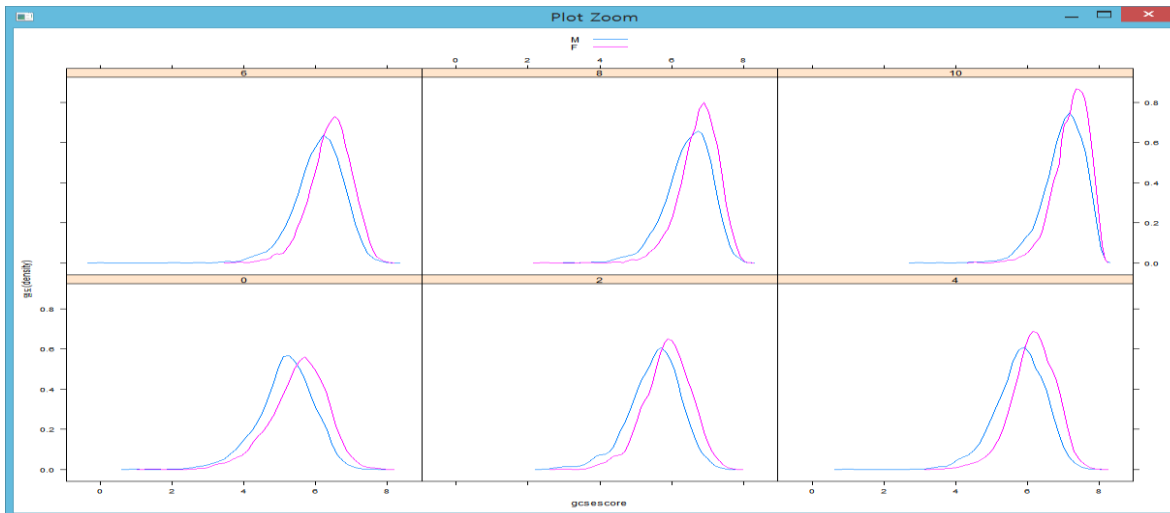
2. densityplot(~x축 | 조건, dataframe, groups=변수)

```
densityplot(~gcsescore | factor(score), data=Chem97,  
            groups = gender, plot.points=F, auto.key = T)
```

```
# 밀도 점 표시 여부 : plot.points=F
```

```
# 범례 표시 : auto.key=T
```

```
# 성별로 GCSE점수를 score 단위로 밀도 플로팅
```



plot.points=T

lattice 패키지

➤ 통계처리를 위한 데이터 형식 변경(matrix -> data.frame/table)

1) Data set가져오기

```
data(VADeaths)
```

```
##### VADeaths 데이터셋 설명 #####  
# 1941년 미국 버지니아주의 하위계층 사망비율  
# Rural Male(시골출신 남자), Urban Male(도시출신 남자)  
# Rural Female(시골출신 여자), Urban Female(도시출신 여자)  
#####
```

```
VADeaths
```

```
#      Rural Male Rural Female Urban Male Urban Female  
#50-54      11.7         8.7      15.4         8.4  
#55-59      18.1        11.7      24.3        13.6  
#60-64      26.9        20.3      37.0        19.3  
#65-69      41.0        30.9      54.6        35.1  
#70-74      66.0        54.3      71.1        50.0
```

```
# VADeaths 데이터 셋 특성 보기  
class(VADeaths) # matrix  
mode(VADeaths) # numeric
```


lattice 패키지

➤ 통계처리를 위한 데이터 형식 변경

2) 데이터 형식 변경

```
# matrix -> data.frame 변환
df <- as.data.frame(VADeaths)
str(df) # 'data.frame': 5 obs. of 4 variables:
class(df) # data.frame 생성
```

```
# matrix -> table 변환(행열 구조 변환)
dft <- as.data.frame.table(VADeaths)
str(dft) # 'data.frame': 20 obs. of 3 variables:
dft # 1열 기준으로 table 생성
```

```
#  Var1      Var2  Freq
#1 50-54  Rural Male 11.7
#2 55-59  Rural Male 18.1
#   :
#19 65-69 Urban Female 35.1
#20 70-74 Urban Female 50.0
```

lattice 패키지

- matrix -> data.frame

	Rural Male	Rural Female	Urban Male	Urban Female
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

- matrix->data.frame.table

	Var1	Var2	Freq
1	50-54	Rural Male	11.7
2	55-59	Rural Male	18.1
3	60-64	Rural Male	26.9
4	65-69	Rural Male	41.0
5	70-74	Rural Male	66.0
6	50-54	Rural Female	8.7
7	55-59	Rural Female	11.7
8	60-64	Rural Female	20.3
9	65-69	Rural Female	30.9
10	70-74	Rural Female	54.3
11	50-54	Urban Male	15.4
12	55-59	Urban Male	24.3
13	60-64	Urban Male	37.0
14	65-69	Urban Male	54.6
15	70-74	Urban Male	71.1
16	50-54	Urban Female	8.4
17	55-59	Urban Female	13.6
18	60-64	Urban Female	19.3
19	65-69	Urban Female	35.1
20	70-74	Urban Female	50.0

lattice 패키지

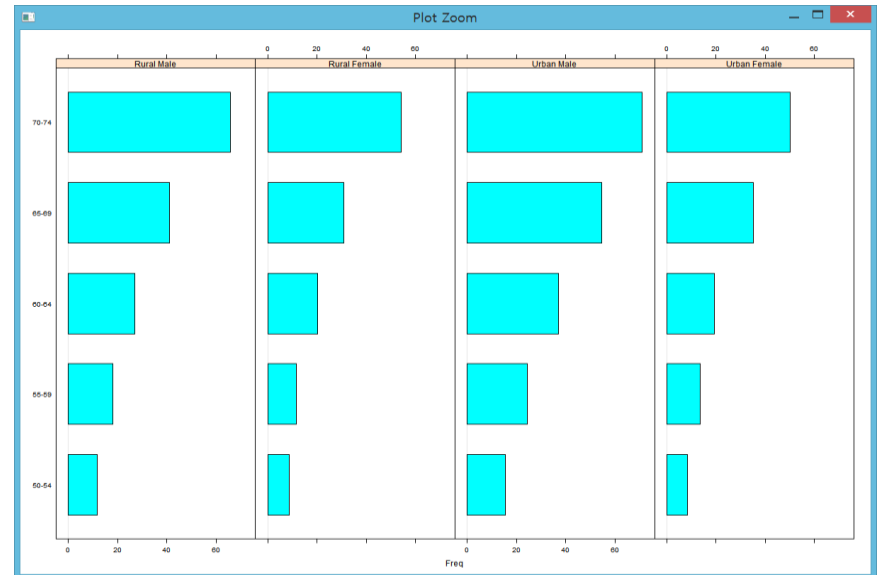
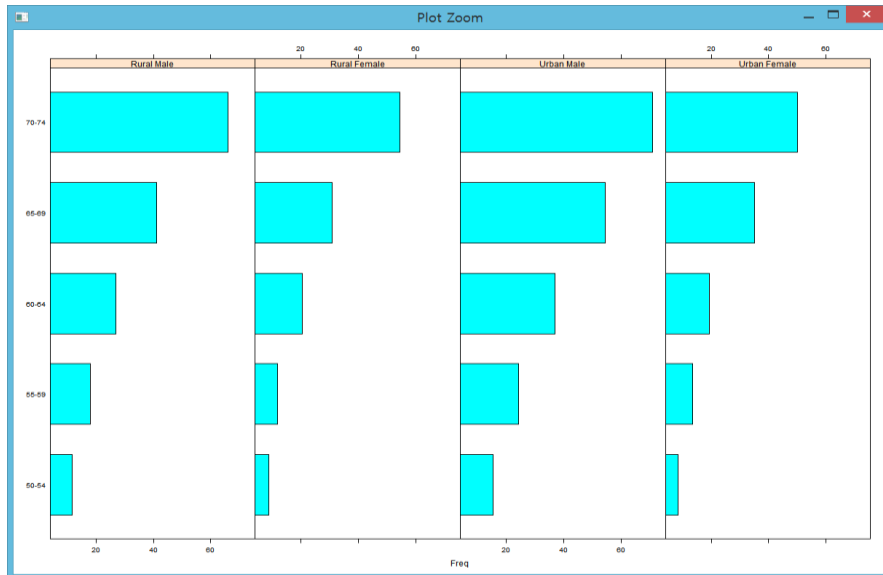
3. barchart(y~x | 조건, dataframe, layout)

```
barchart(Var1 ~ Freq | Var2, data=dft, layout=c(4,1))
```

Var2변수값을 기준으로 가로막대차트 플로팅

```
barchart(Var1 ~ Freq | Var2, data=dft, layout=c(4,1), origin=0)
```

origin=0 : 0부터 시작

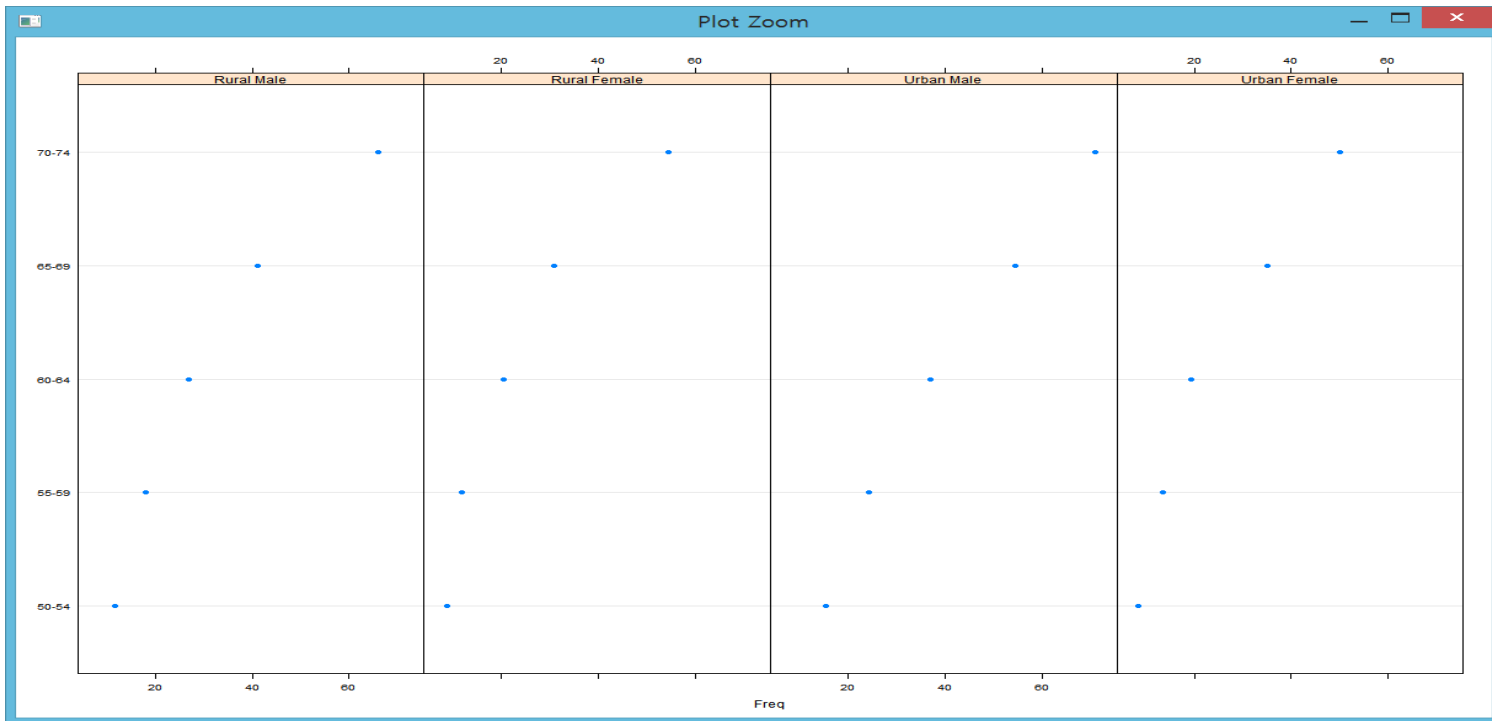


lattice 패키지

4. dotplot(y~x | 조건 , dataframe, layout)

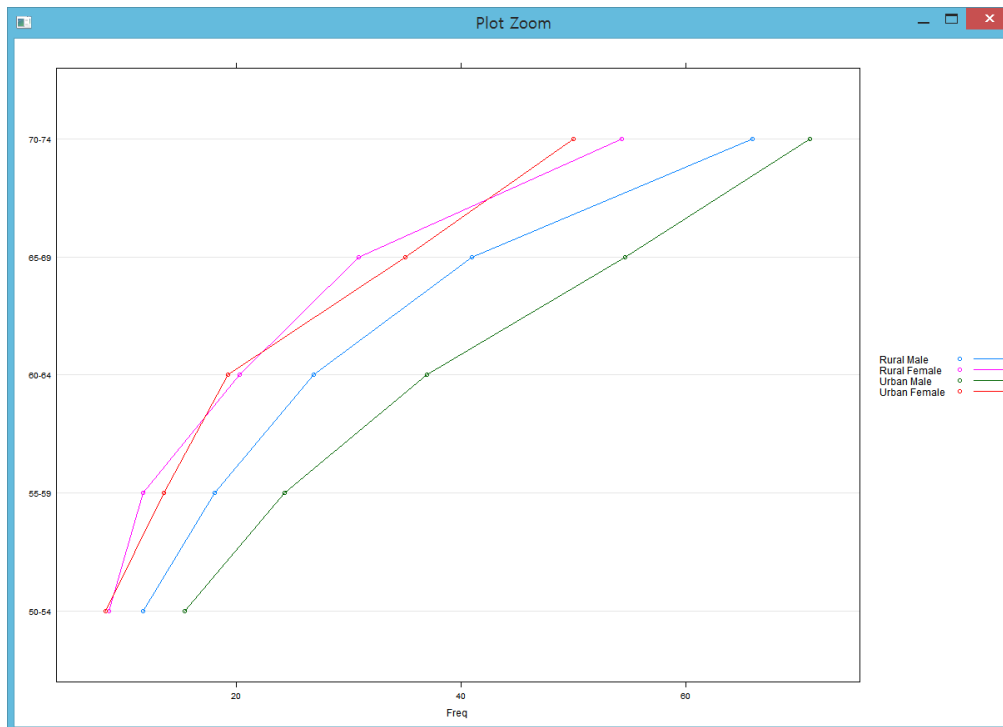
dotplot(Var1 ~ Freq | Var2 , dft)

dotplot(Var1 ~ Freq | Var2 , dft, layout=c(4,1)) # 1행 4열



lattice 패키지

```
# Var2변수를 그룹화하여 점을 연결하여 플로팅
dotplot(Var1 ~ Freq, data=dft, groups=Var2, type="o",
        auto.key=list(space="right", points=T, lines=T))
# 산점도 타입 : type="o" : 원형에 실선 통과
# 범례 : auto.key=list(배치위치, 점 추가, 선 추가)
```



lattice 패키지

5. xyplot(y축~x축, dataframe or list)

➤ Data set 가져오기

```
library(datasets)
```

```
str(airquality) # datasets의 airquality 데이터 활용
```

```
##### airquality 데이터셋 설명 #####
```

```
# datasets 패키지에서 제공
```

```
# 뉴욕시의 대기오염에 관한 데이터셋
```

```
# data.frame ' : 153 obs. of 6 variables:
```

```
# 주요변수 : Ozone(오존), Solar.R(태양열), Wind(바람), Temp(온도), Month(월:5~9),
```

```
#           Day(일:1~31)
```

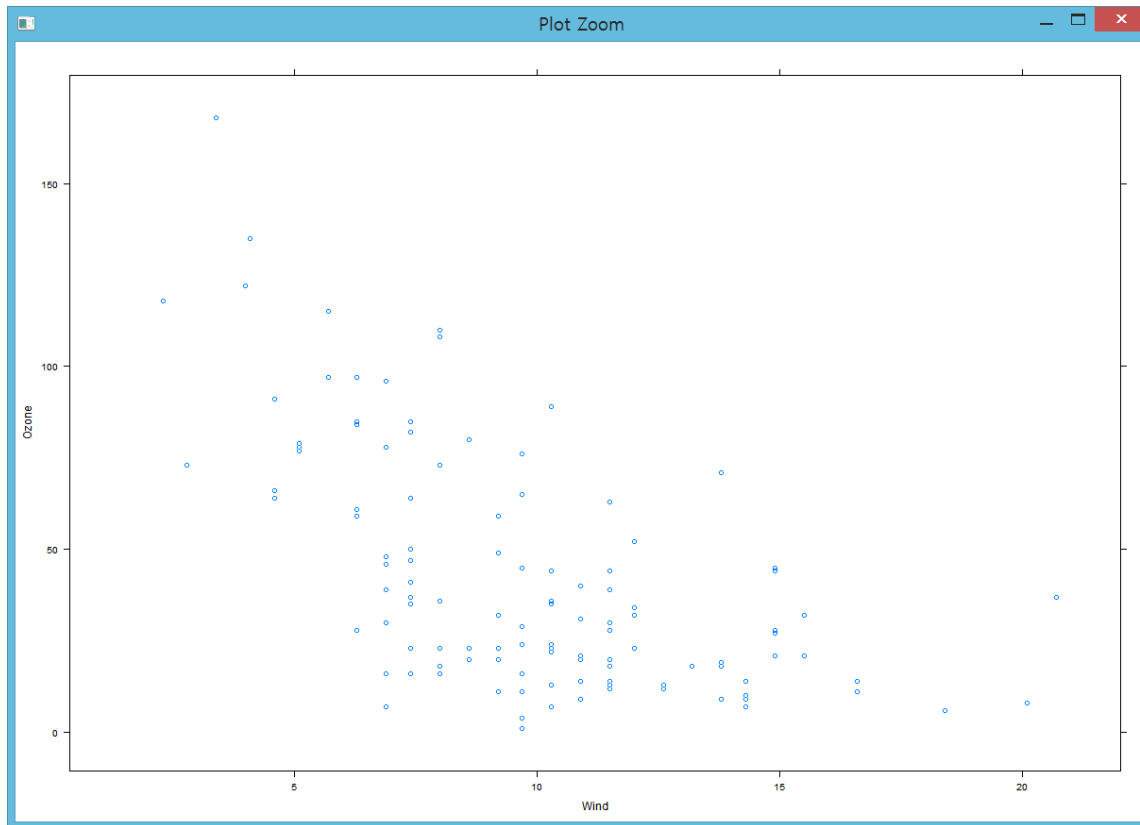
```
#####
```

```
airquality # Ozone Solar.R Wind Temp Month(5~9) Day
```

lattice 패키지

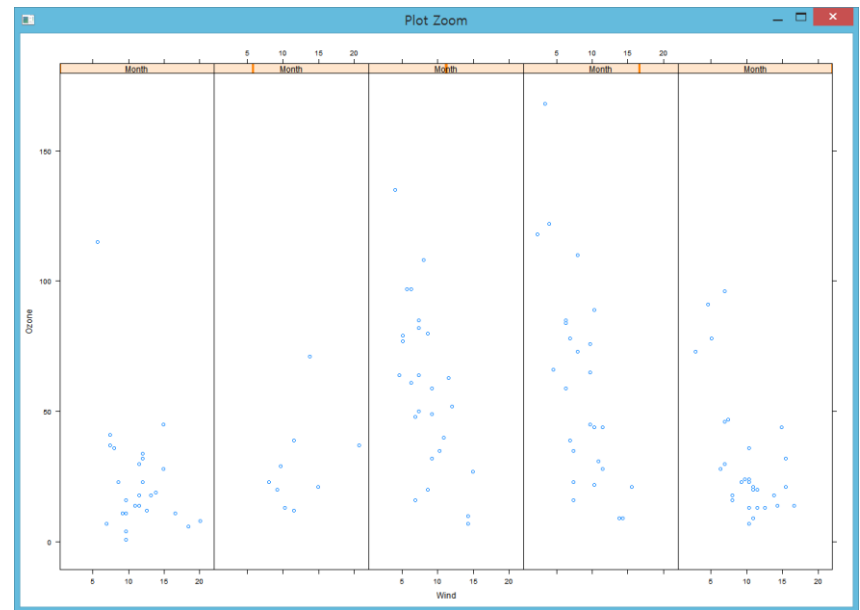
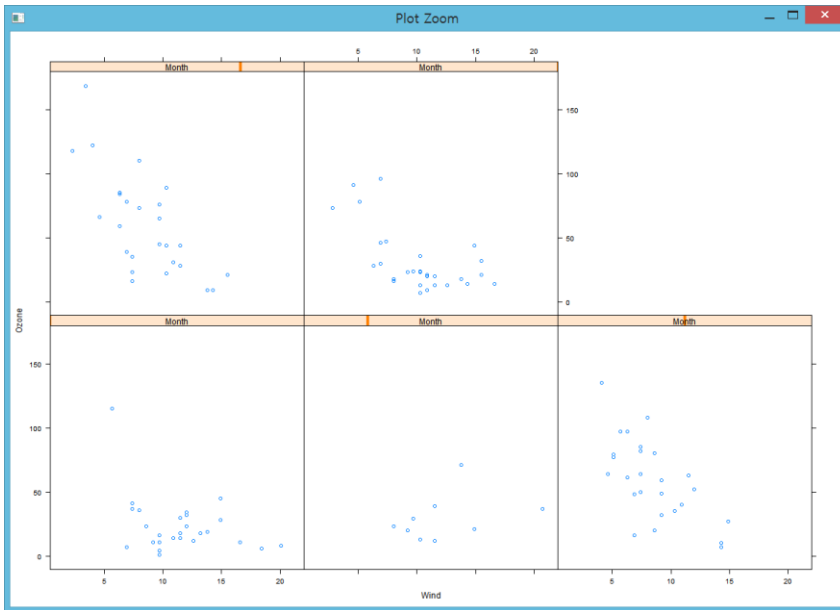
5. `xyplot(y축~x축, dataframe or list)`

`xyplot(Ozone ~ Wind, data=airquality)` # `airquality`의 `Ozone(y)`, `Wind(x)`



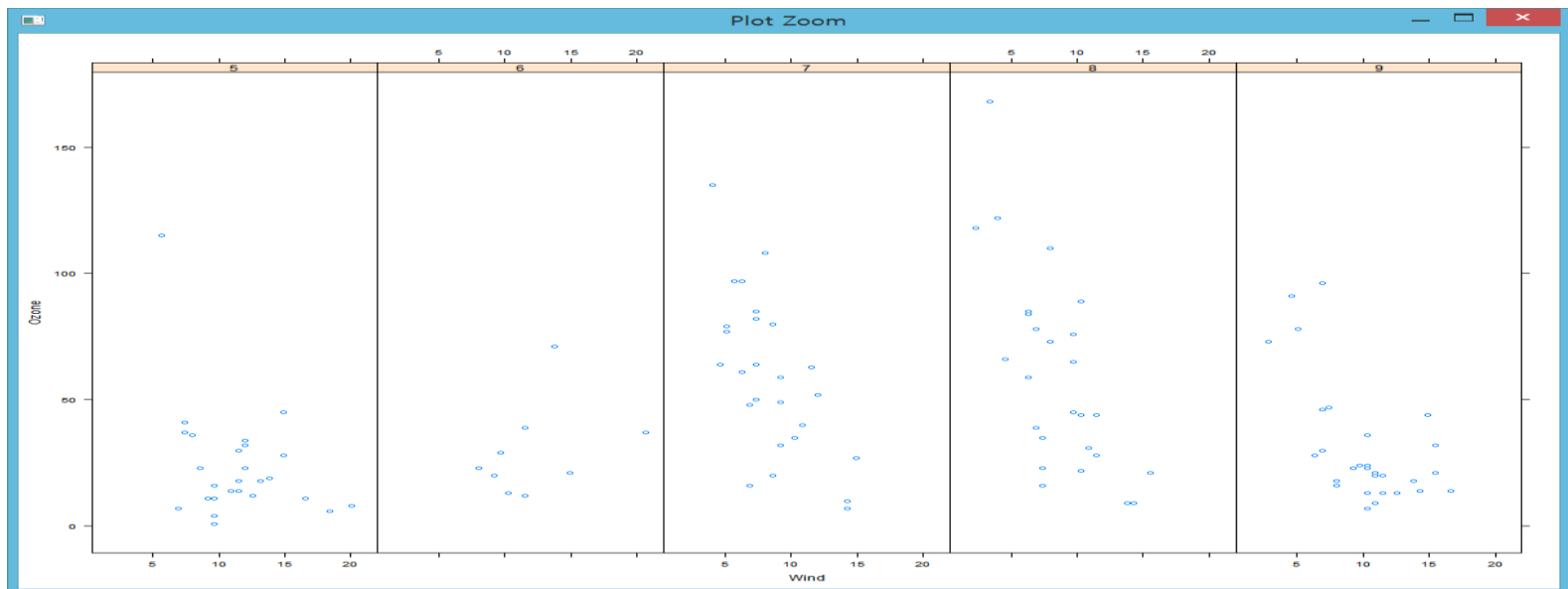
lattice 패키지

```
# airquality 데이터셋의 Month(5~9) 변수 기준(월별) 플로팅  
xyplot(Ozone ~ Wind | Month, data=airquality) # 2행3컬럼  
# default -> layout=c(3,2)  
xyplot(Ozone ~ Wind | Month, data=airquality, layout=c(5,1))  
# 5컬럼으로 플로팅 - 컬럼 제목 : Month
```



lattice 패키지

```
# Month를 int 타입에서 factor 데이터 타입으로 변경
convert <- transform(airquality, Month=factor(Month))
str(convert) # Month 변수의 Factor값 확인
# $ Month : Factor w/ 5 levels "5","6","7","8"
# Month의 factor 타입을 data에 적용
xyplot(Ozone ~ Wind | Month, data=convert, layout=c(5,1))
# 컬럼 제목 : Month 값으로 출력
```



lattice 패키지

➤ Data set 가져오기

```
head(quakes)
```

```
str(quakes) # 'data.frame': 1000 obs. of 5 variables:  
# lat, long, depth, mag, stations
```

```
##### quakes 데이터셋 설명 #####  
  
# R에서 제공하는 기존 데이터셋  
  
# 1964년 이후 피지(태평양) 근처에 발생한 지진 사건  
  
# 주요 변수 : lat:위도,long:경도,depth:깊이(km), mag:리히터규모,stations  
  
#####
```

lattice 패키지

```
xyplot(lat~long, data=quakes, pch=".") # 지진발생 위치(위도와 경로)
```

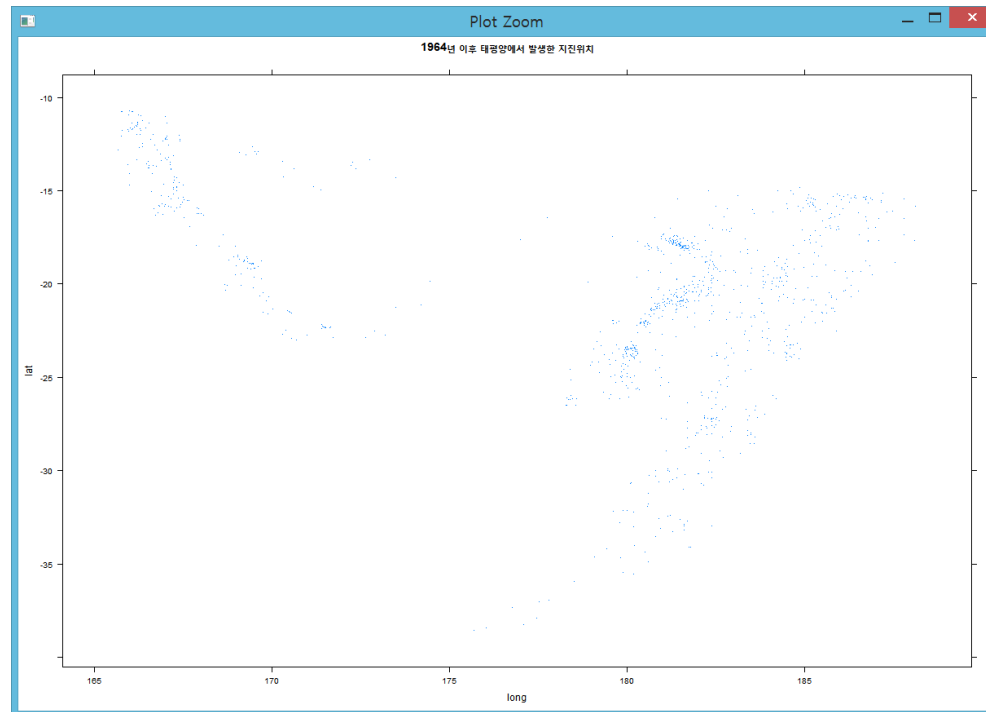
```
# 그래프를 변수에 저장
```

```
tplot<-xyplot(lat~long, data=quakes, pch=".")
```

```
# 그래프에 제목 추가
```

```
tplot2<-update(tplot, main="1964년 이후 태평양에서 발생한 지진위치")
```

```
print(tplot2)
```



lattice 패키지

6. equal.count(변수, number=n, overlap=n)

변수값을 대상으로 지정한 영역 만큼 수량 카운터

```
numgroup<- equal.count(1:100, number=3, overlap=0)
```

1~100 데이터를 대상으로 겹치지 않게 3개 영역으로 구분 영역별 count 구하기

```
numgroup
```

```
#Intervals:
```

```
#   min   max count
```

```
#1  0.5 33.5   33
```

```
#2 33.5 67.5   34
```

```
#3 67.5 100.5  33
```

lattice 패키지

지진의 깊이를 3영역으로 구분하여 카운팅

```
depthgroup<-equal.count(quakes$depth, number=3, overlap=0)
```

```
depthgroup
```

```
#Intervals:
```

```
#  min  max count
```

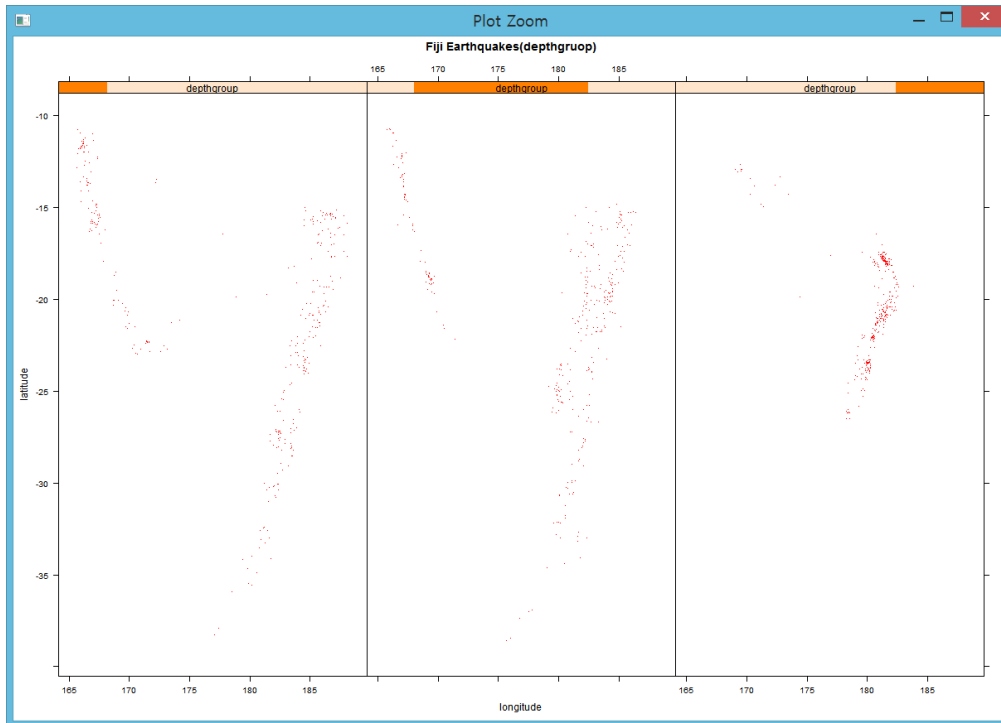
```
#1 39.5 139.5  335
```

```
#2 138.5 498.5  338
```

```
#3 497.5 680.5  334
```

lattice 패키지

```
# depthgroup 변수 기준으로 플로팅  
xyplot(lat ~ long | depthgroup, data=quakes,  
        main="Fiji Earthquakes(depthgruop)",  
        ylab="latitude", xlab="longitude", pch=".", col='red')
```



lattice 패키지

```
# depthgroup 기준 - 3행 2열
xyplot(lat ~ long | magnitude*depthgroup, data=quakes,
       main="Fiji Earthquakes",
       ylab="latitude", xlab="longitude",
       pch="@", col=c("red", "blue"),
       scales=list(x=list(alternating=c(1,1,1))),
       between=list(y=1),
       par.strip.text=list(cex=0.7),
       par.settings=list(axis.text=list(cex=0.7)))

# 추가 옵션
# scales=list(x=list(alternating=c(1,1,1))) : x축 이름 아래쪽 일괄 배치
# between=list(y=1) : y축 사이 여백
# par.strip.text=list(cex=1.2) : 그룹 변수명 텍스트 크기
# par.settings=list(axis.text=list(cex=1.2)) : 축이름 텍스트 크기
```

lattice 패키지

7. cloud() -> 3차원(z ~ y * x) 산점도 그래프 플로팅

```
cloud(depth ~ lat * long , data=quakes,
```

```
      zlim=rev(range(quakes$depth)),
```

```
      xlab="경도", ylab="위도", zlab="깊이")
```

```
# depth ~ lat * long : depth(z축), lat(y축) * long(x축)
```

```
# zlim=rev(range(quakes$depth)) : z축값 범위 지정
```

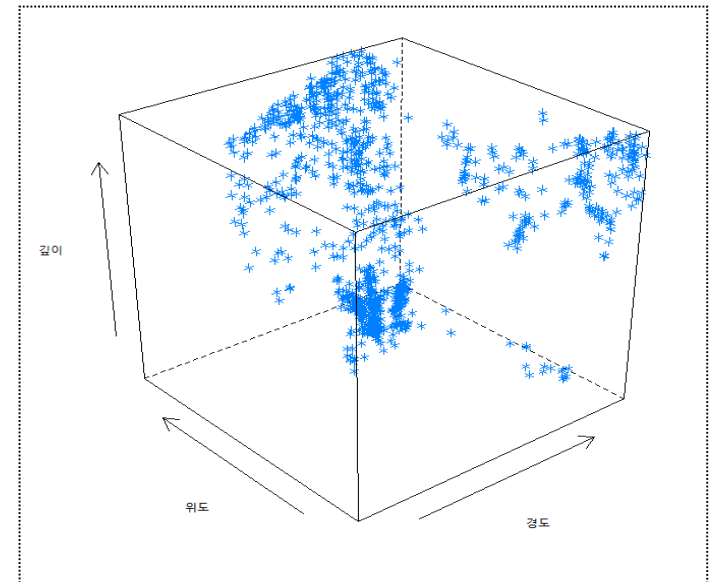
```
# panel.aspect=0.75 : 테두리 사이즈
```

```
# screen=list(z=105,x=-70) : z,x축 회전
```

```
# xlab="Longitude" : x축 이름
```

```
# ylab="Latitude" : y축 이름
```

```
# zlab="Depth" : z축 이름
```



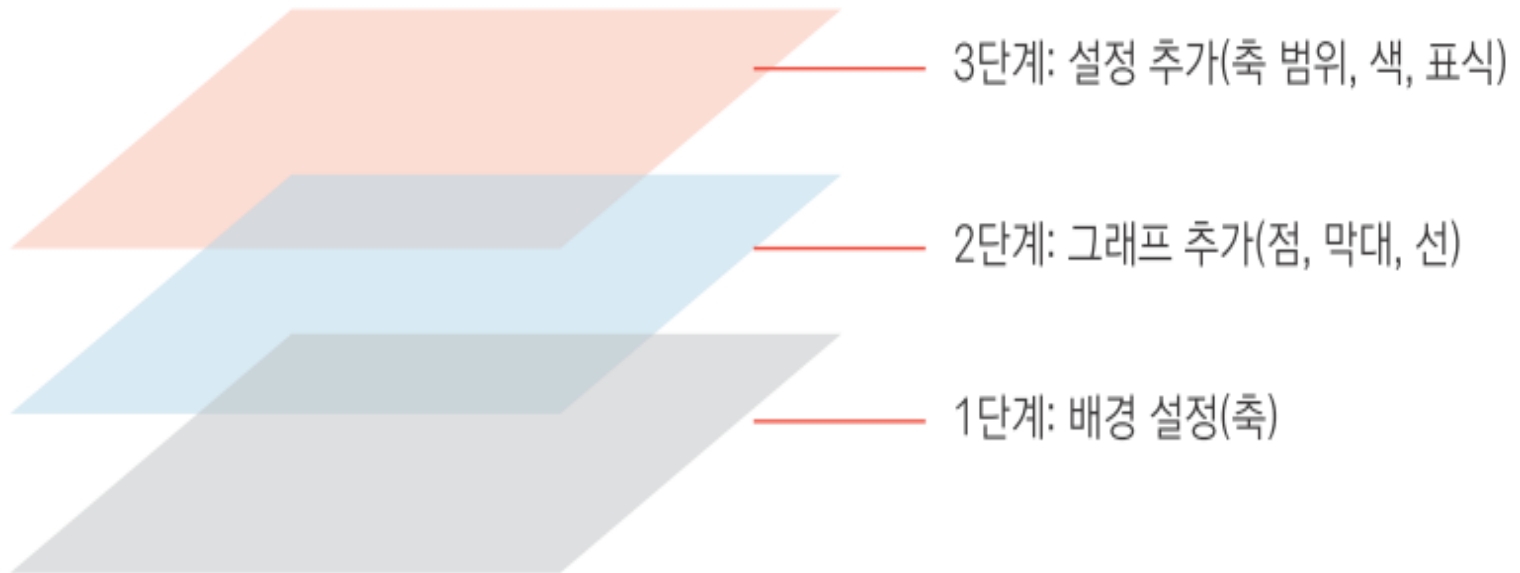
ggplot2 패키지

➤ ggplot2 그래픽 패키지 특징

- ✓ 기하학적 객체들(점,선,막대 등)에 미적 특성(색상, 모양,크기)를 맵핑하여 플로팅한다.
- ✓ 그래픽 생성 기능과 통계변환을 포함할 수 있다.
- ✓ ggplot2의 기본 함수 `qplot()`-`aesthetics`(크기,모양,색상)과 `geoms`(점,선 등)으로 구성

ggplot2 패키지

➤ ggplot2 레이어 구조 이해하기



ggplot2 레이어 구조

ggplot2 패키지

➤ ggplot2 package 및 데이터 셋 설치

```
# ggplot2 패키지는 데이터프레임 데이터를 이용(변환 요구)
install.packages("ggplot2")
library(ggplot2)
str(mpg)
head(mpg)
# manufacturer model displ year cyl   trans drv  cty   hwy fl   class

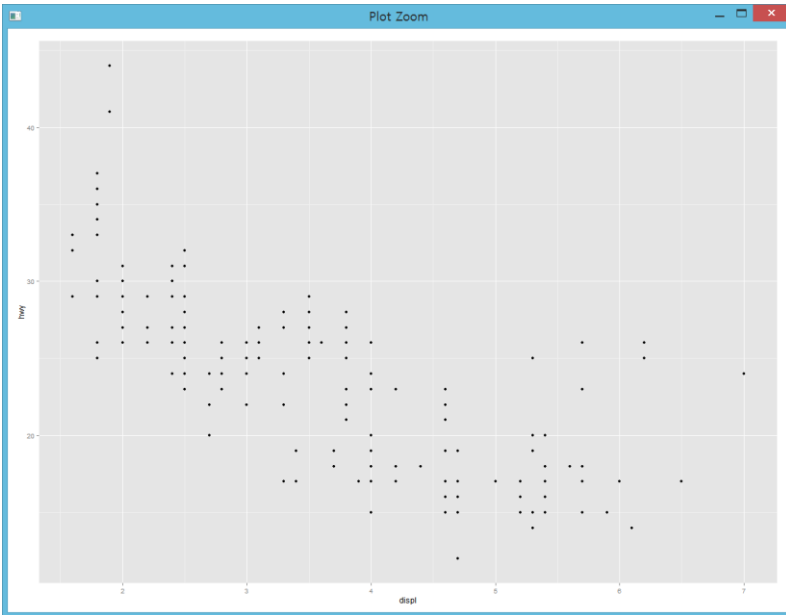
##### mpg 데이터셋 #####
# ggplot2에서 제공하는 데이터셋
# 'data.frame':   234 obs. of  11 variables:
# 주요 변수 : displ:엔진크기, cyl : 실린더수,
#           drv(구동방식) -> 사륜구동(4), 전륜구동(f), 후륜구동(r)
#####
```

ggplot2 패키지

➤ `ggplot(x, y, data)` 함수 이용 플로팅

`ggplot(displ, hwy, data=mpg)` # mpg 데이터셋의 displ과 hwy 변수 이용
displ(x):엔진크기, hwy(y):고속도로 주행

`ggplot(displ, hwy, data=mpg, color=drv)` # drv 변수값으로 색상 적용
drv -> 사륜구동(4) : red, 전륜구동(f):green, 후륜구동(r):blue,



ggplot2 패키지

➤ 1개 변수 대상 히스토그램 플로팅

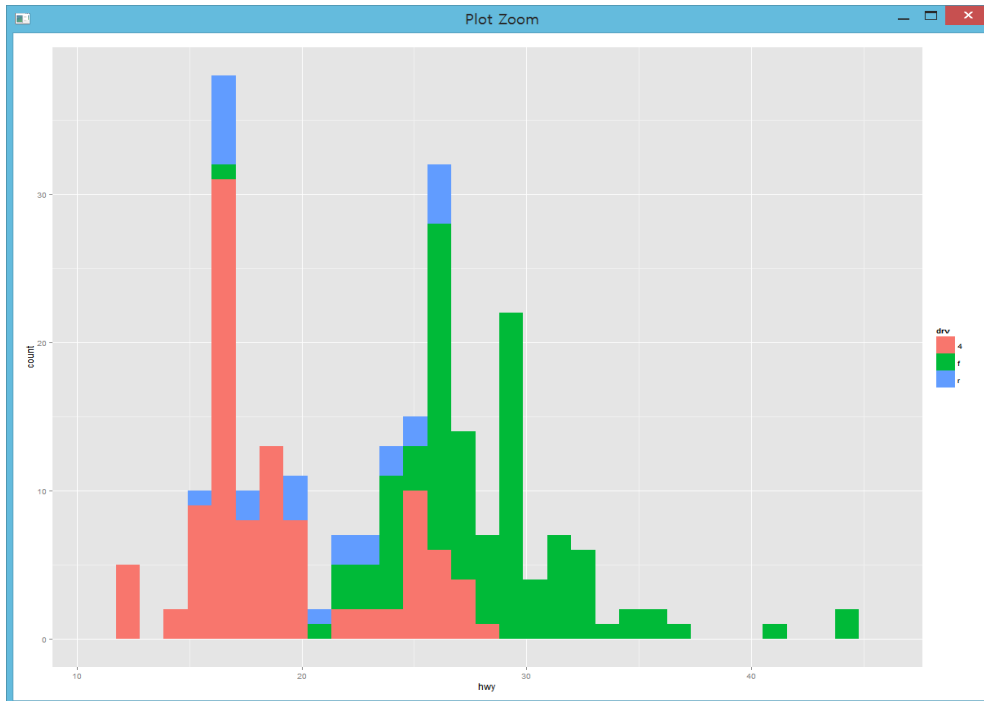
`qplot(hwy, data=mpg, fill=drv) # hwy 변수 대상 drv 색상 적용`

`# 각 그룹별 히스토그램 분포를 플로팅 한다.(fill : 색채우기)`

`# drv 기준으로 hwy의 빈도수를 플로팅`

`# 예) drv가 f이고, hwy가 29인 경우가 22회가 발생함`

`# 설명) 고속도로 주행 속도가 29이고, 전륜구동인 경우 22회 발생`

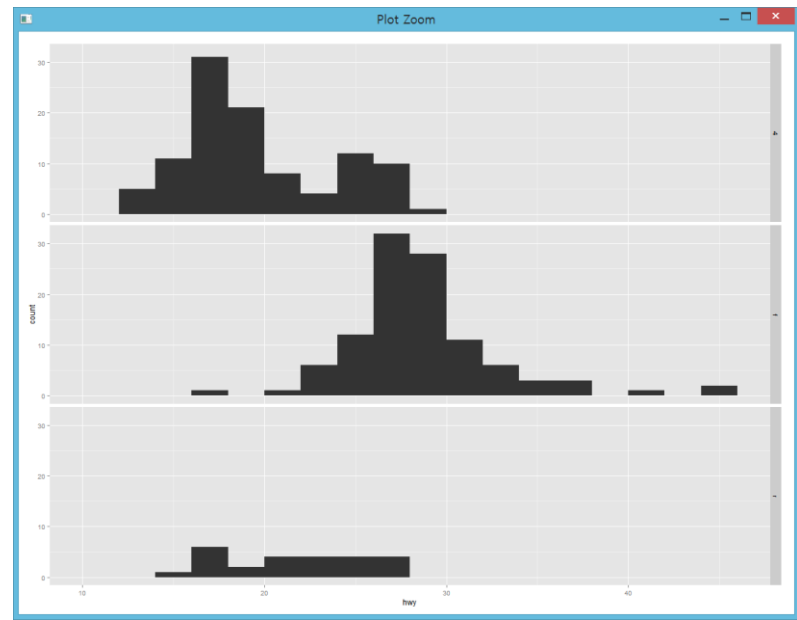
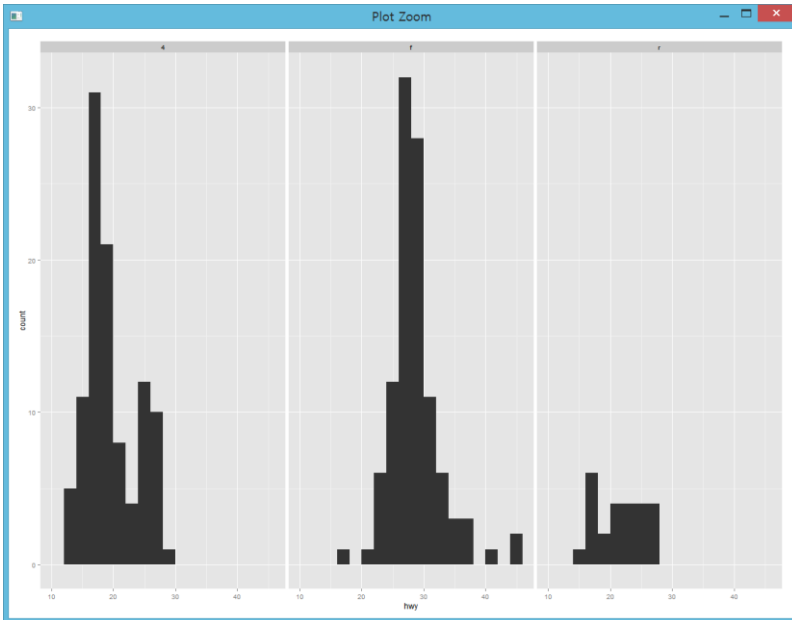


ggplot2 패키지

- 1개 변수 대상 히스토그램 플로팅- binwidth=막대굵기

```
qplot(hwy, data=mpg, facets=~ drv, binwidth=2) # 열 단위
```

```
qplot(hwy, data=mpg, facets=drv~., binwidth=2) # 행 단위
```



ggplot2 패키지

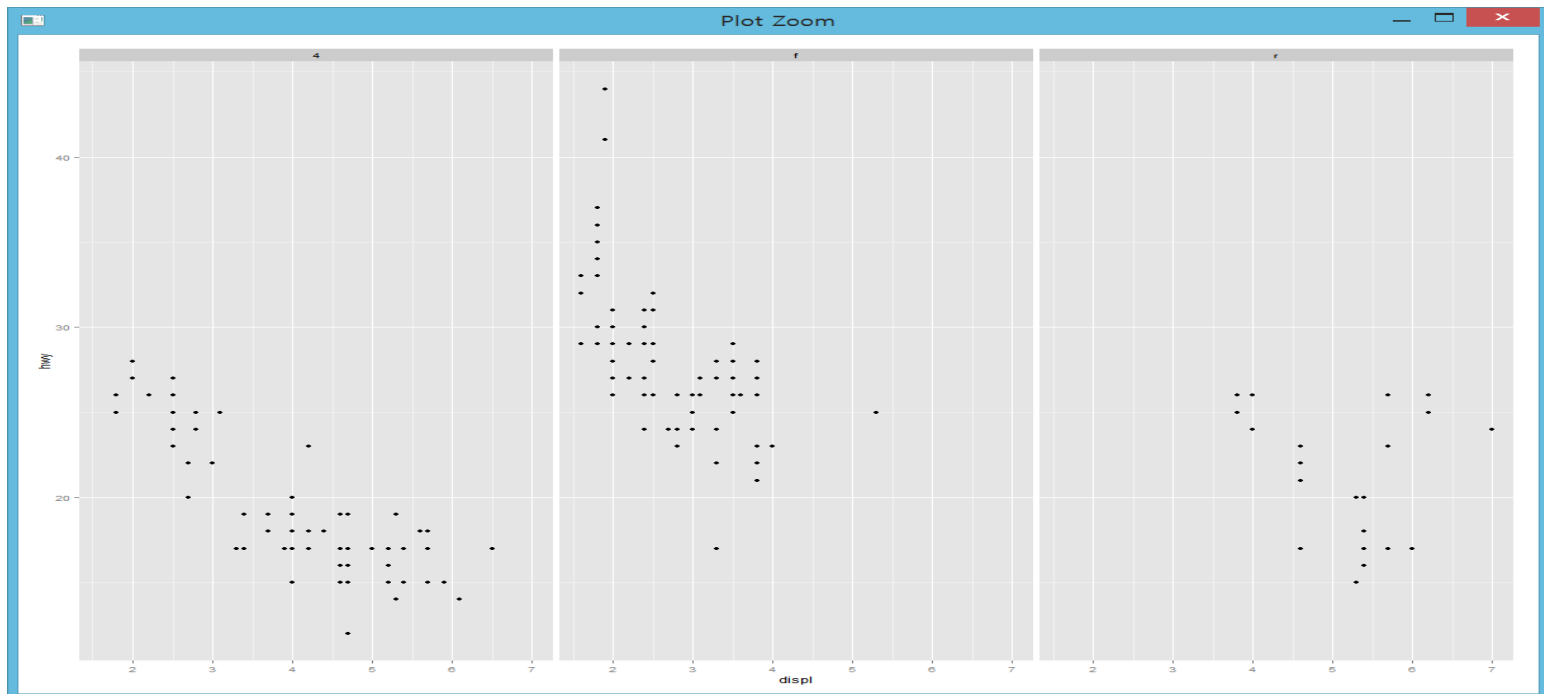
➤ 2개 변수 간의 관계분석

엔진크기와 고속도로 주행속도와의 관계를 구동방식으로 구분

```
qplot(displ, hwy, data=mpg, facets=~ drv)
```

facets : dataset에서 특정 부분집합만 취합해서 플로팅

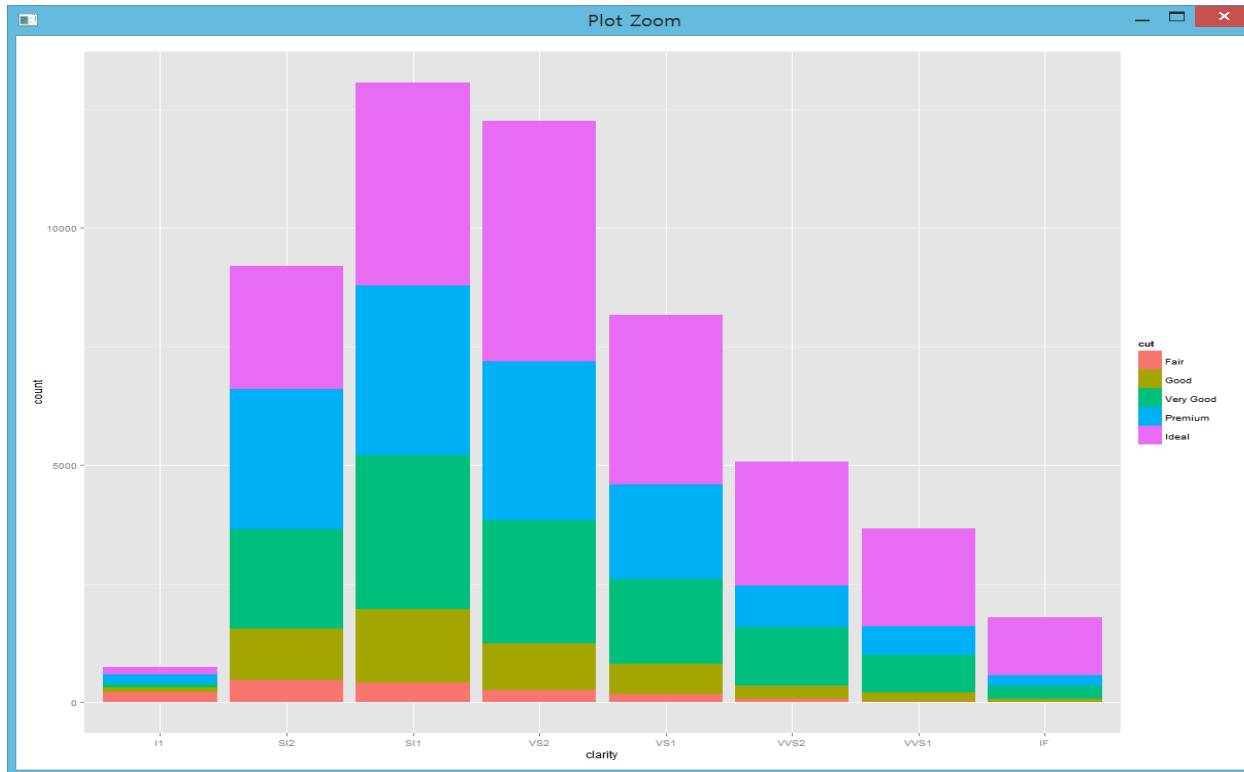
Factor변수에 의해서 그려진다. - drv값을 요인변수로 사용



ggplot2 패키지

➤ `geom="bar"`

`head(diamonds)` # ggplot2에서 제공하는 데이터 셋
`qplot(clarity, data=diamonds, fill=cut, geom="bar")`



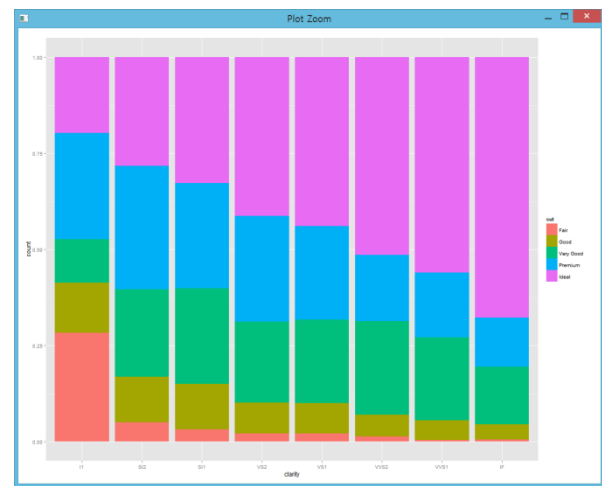
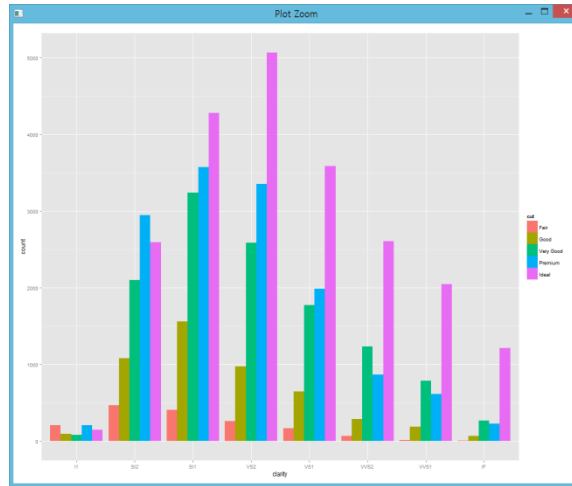
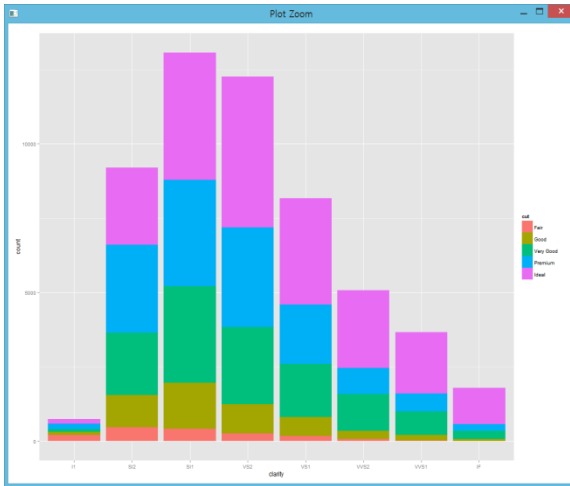
ggplot2 패키지

➤ 다양한 bar 차트 유형

```
qplot(clarity, data=diamonds, geom="bar", fill=cut, position="stack")
```

```
qplot(clarity, data=diamonds, geom="bar", fill=cut, position="dodge")
```

```
qplot(clarity, data=diamonds, geom="bar", fill=cut, position="fill")
```



ggplot2 패키지

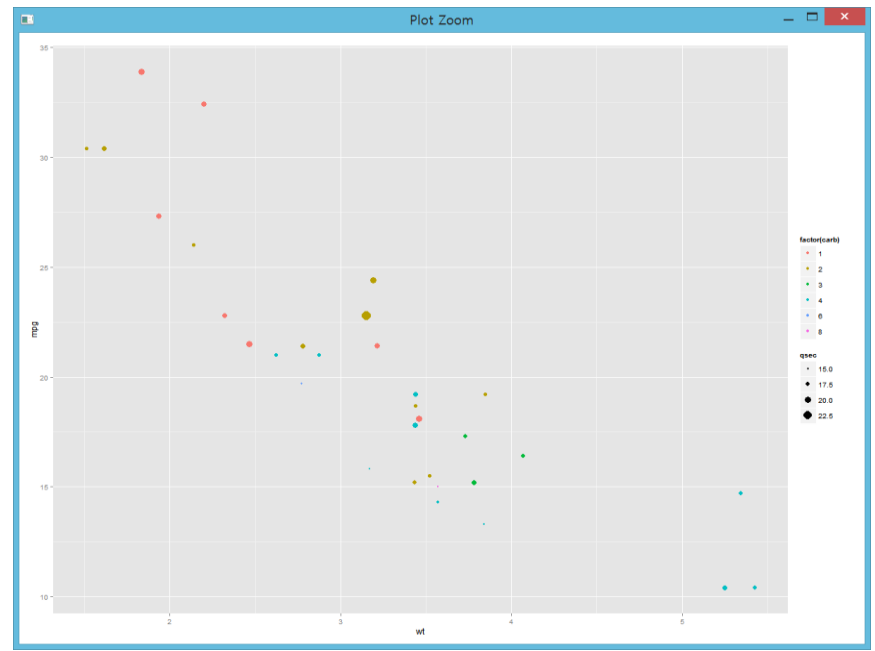
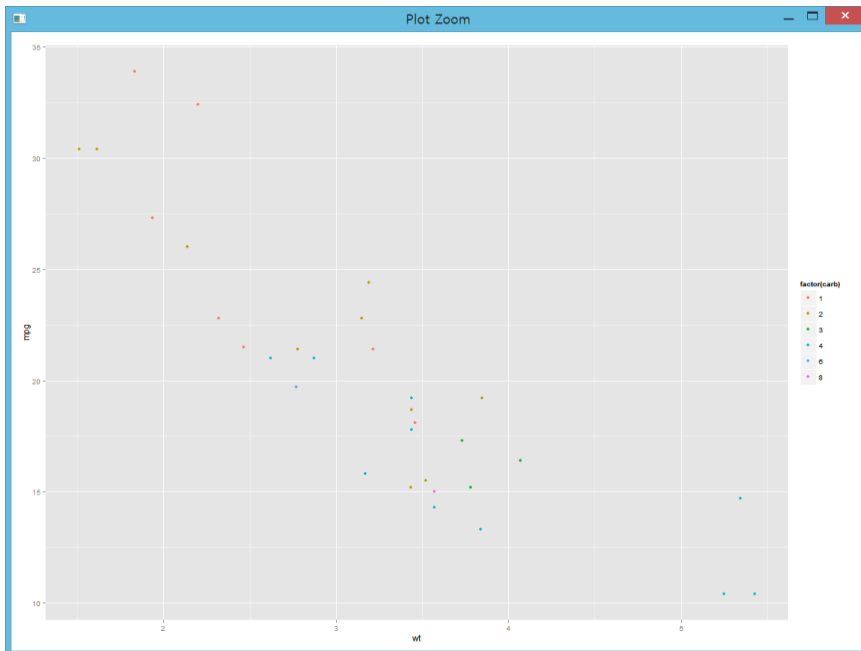
➤ 색상과 크기 적용

`head(mtcars)` # ggplot2에서 제공하는 데이터 셋

qsec 변수 값으로 point 크기 지정

`qplot(wt, mpg, data=mtcars, color=factor(carb))` # 색상 적용

`qplot(wt, mpg, data=mtcars, size=qsec, color=factor(carb))` # 크기 적용



ggplot2 패키지

➤ geom="point"

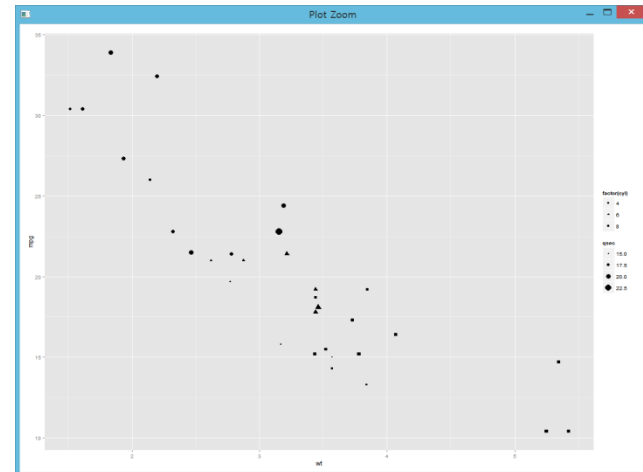
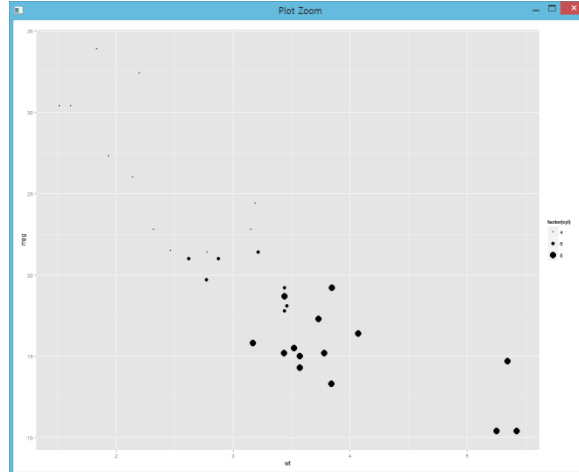
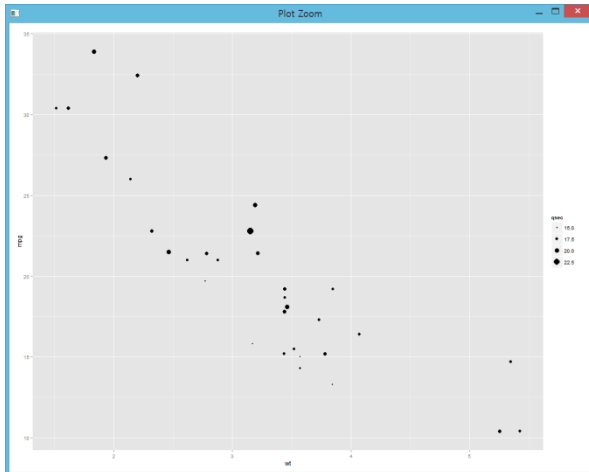
```
qplot(wt, mpg, data=mtcars, size=qsec, geom="point")
```

cyl 변수의 요인으로 point 크기 지정

```
qplot(wt, mpg, data=mtcars, size=factor(cyl), geom="point")
```

qsec 변수 값으로 point 크기 지정, cyl 변수의 요인으로 point 모양 적용

```
qplot(wt, mpg, data=mtcars, size=qsec, shape=factor(cyl), geom="point")
```

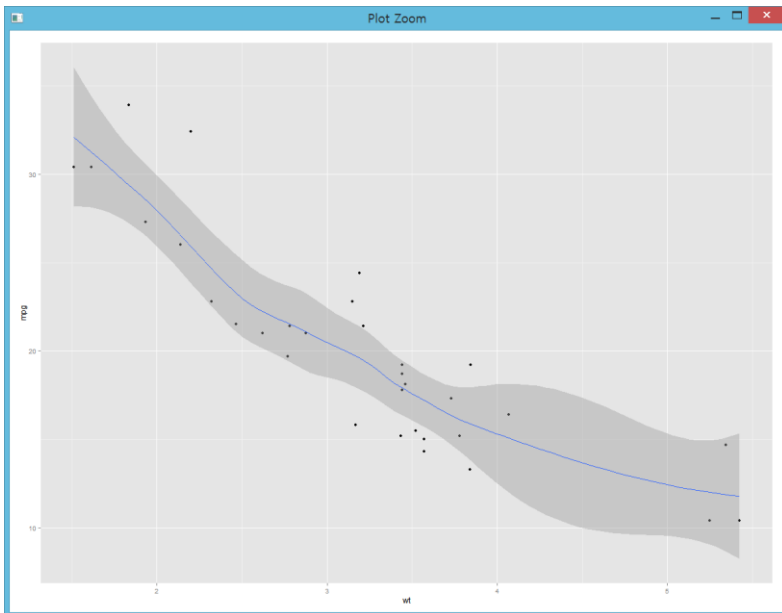


ggplot2 패키지

- `geom=c("point", "smooth")`

```
qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"))
```

```
qplot(wt, mpg, data=mtcars, color=factor(cyl),  
      geom=c("point", "smooth")) # cyl 변수 요인으로 색상 적용
```

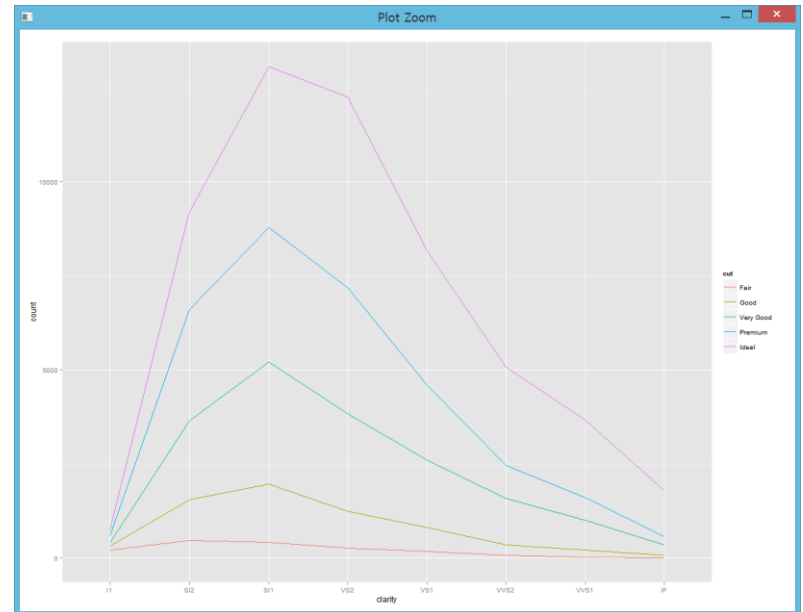
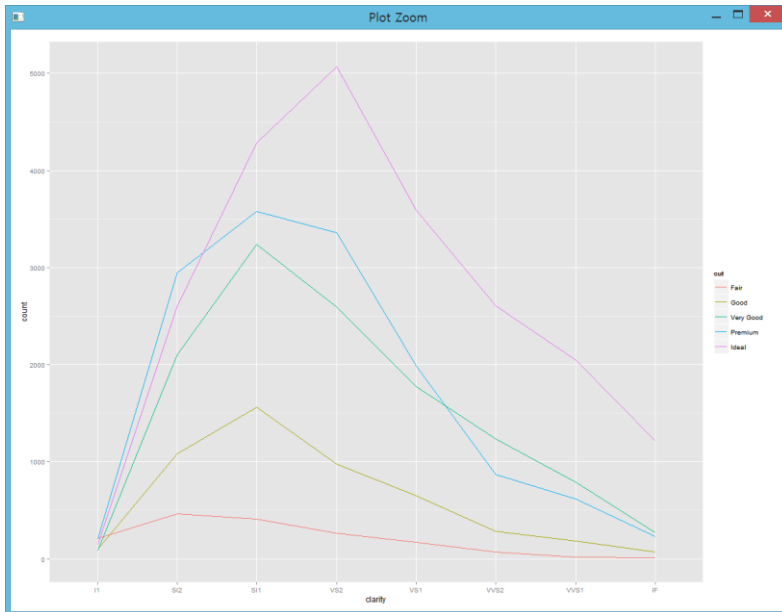


ggplot2 패키지

➤ geom="freqpoly"

```
qplot(clarity, data=diamonds, geom="freqpoly", group=cut, colour=cut,  
      position="identity")
```

```
qplot(clarity, data=diamonds, geom="freqpoly", group=cut, colour=cut,  
      position="stack")
```

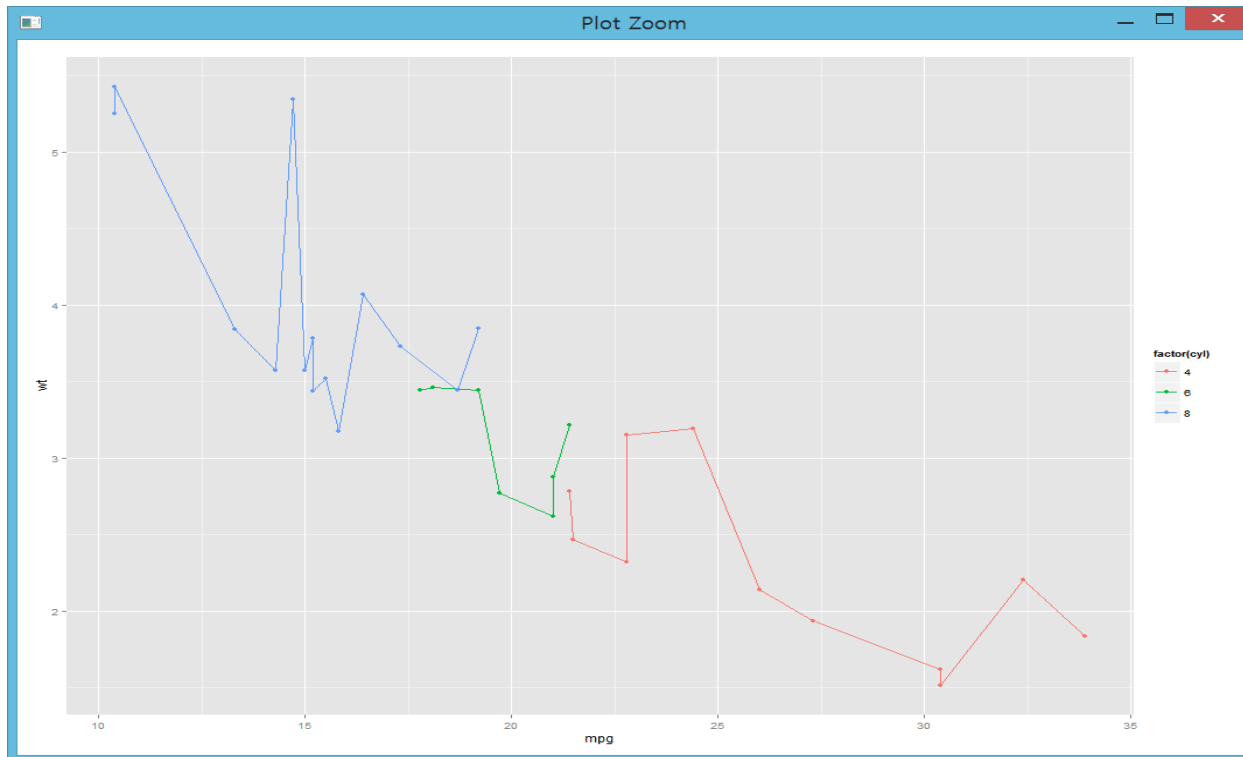


ggplot2 패키지

➤ geom_line()

```
qplot(mpg, wt, data=mtcars, color=factor(cyl), geom="point") +  
geom_line()
```

```
qplot(mpg, wt, data=mtcars, color=factor(cyl), geom=c("point","line"))
```



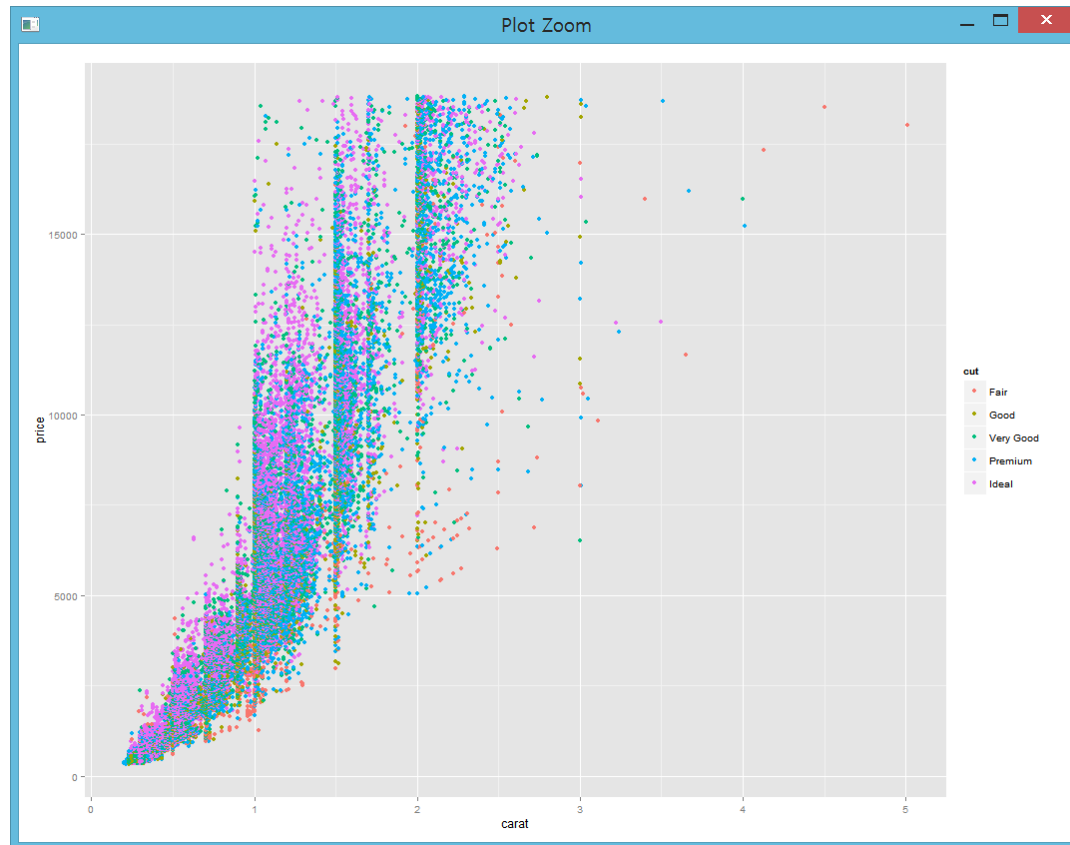
ggplot2 패키지

➤ aes(x, y, color)

```
p<-ggplot(diamonds, aes(carat, price, color=cut))
```

```
p<-p+layer(geom="point") # point 추가
```

```
plot(p)
```



ggplot2 패키지

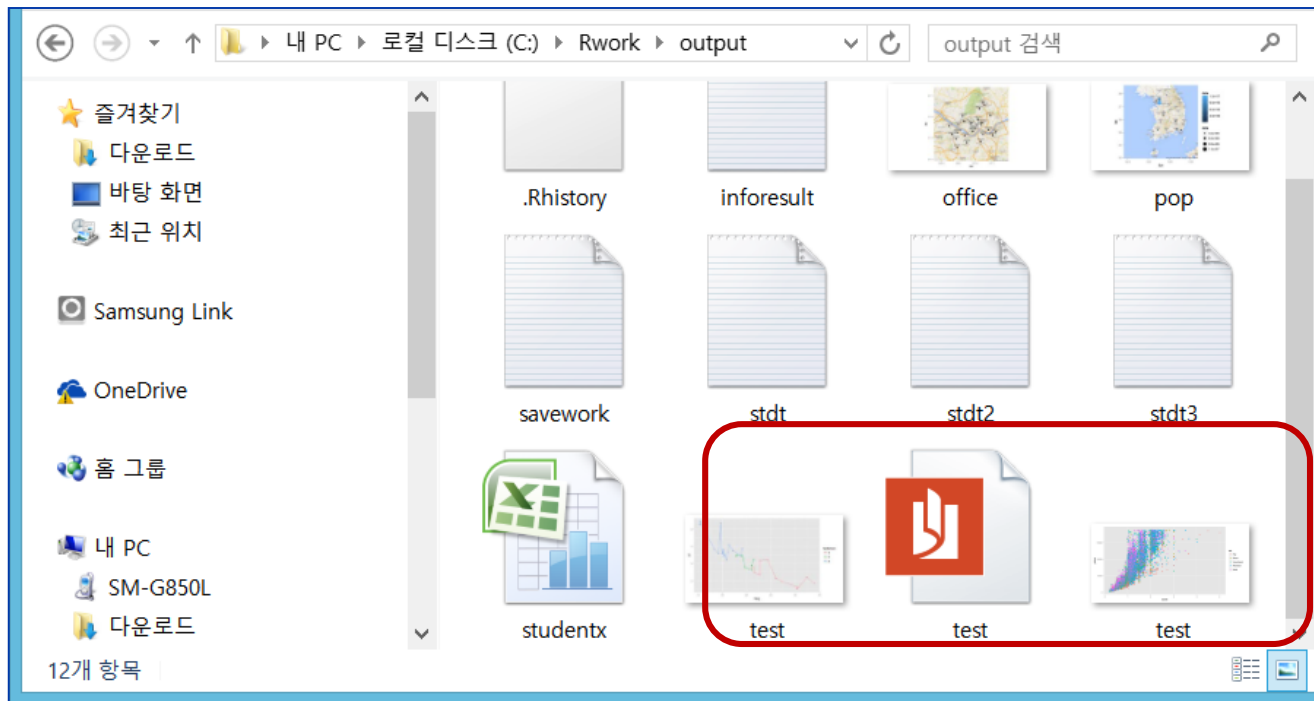
➤ 그래프 저장

`ggsave(file="C:/workspaces/Rwork/output/test.pdf")` # 가장 최근 그래프 저장

`ggsave(file="C:/workspaces/Rwork/output/test.jpg", dpi=72)`

변수에 저장된 그래프 저장

`ggsave(file="C:/workspaces/Rwork/output/test.png", plot=p, width=10, height=5)`



공간시각화

➤ 공간시각화 특징

- ✓ 공간 시각화는 지도를 기반으로 하기 때문에 위치, 영역, 시간과 공간에 따른 차이 및 변화에 대한 것을 다룸
- ✓ 위치 : 위도 및 경도, 지도에 버블로 표현
- ✓ 영역 : 데이터에 따른 색상으로 표현
- ✓ 시.공간 : 레이어 형태로 추가하여 시각화

공간시각화

➤ ggmap package 설치

지도 관련 패키지 설치

```
install.packages("ggmap") # 'ggmap'와 'ggplot2' 관련 패키지  
library(ggmap)
```

Warning messages:

- 1: 패키지 'ggmap'는 R 버전 3.1.3에서 작성되었습니다
- 2: 패키지 'ggplot2'는 R 버전 3.1.3에서 작성되었습니다

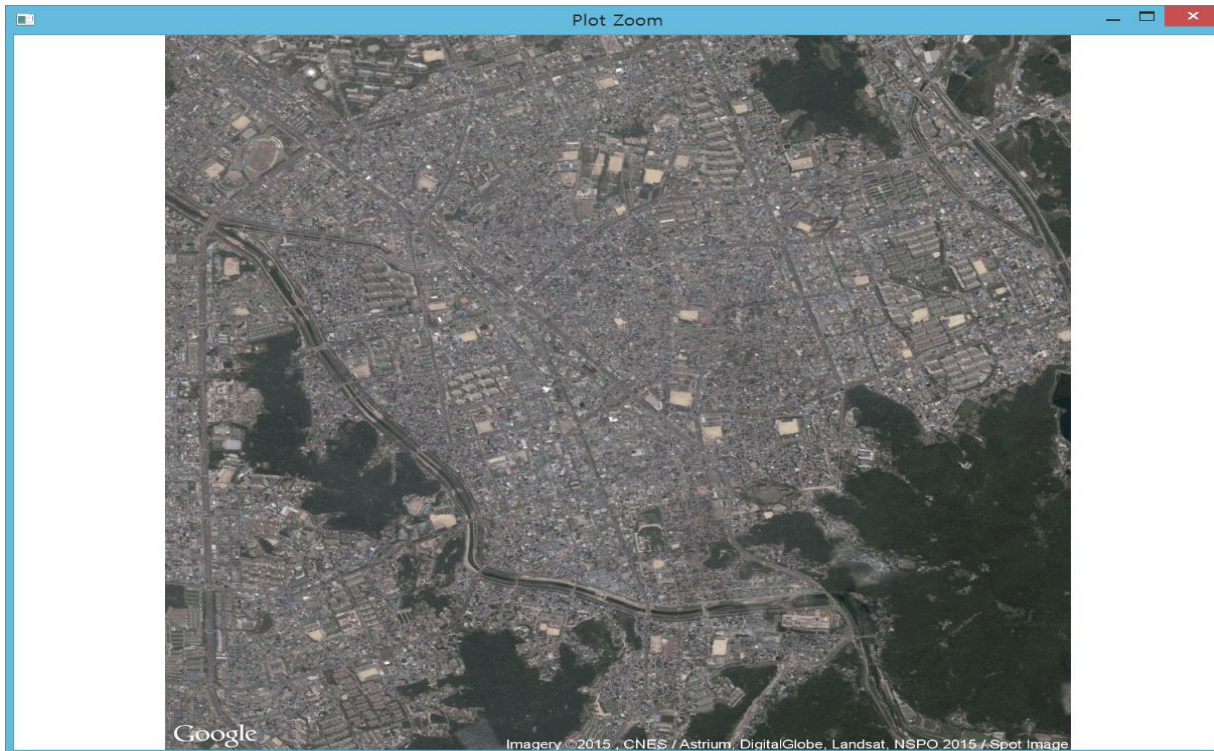
공간시각화

➤ google 지도 불러오기

```
map <- get_googlemap("Jeonju", zoom=14, maptype='satellite', scale=2)
```

```
# scale=2 : 선명도 좋아짐
```

```
ggmap(map, size=c(600,600), extent='device') #장치 허용크기로 표시
```



공간시각화

➤ get_map()과 ggmap() 함수 사용

실습 데이터 가져오기

```
loc <- read.csv("C:/workspaces/Rwork/data/seouloffice.csv",header=T)
```

```
loc # 구청명    LON(위도)    LAT(경도)
```

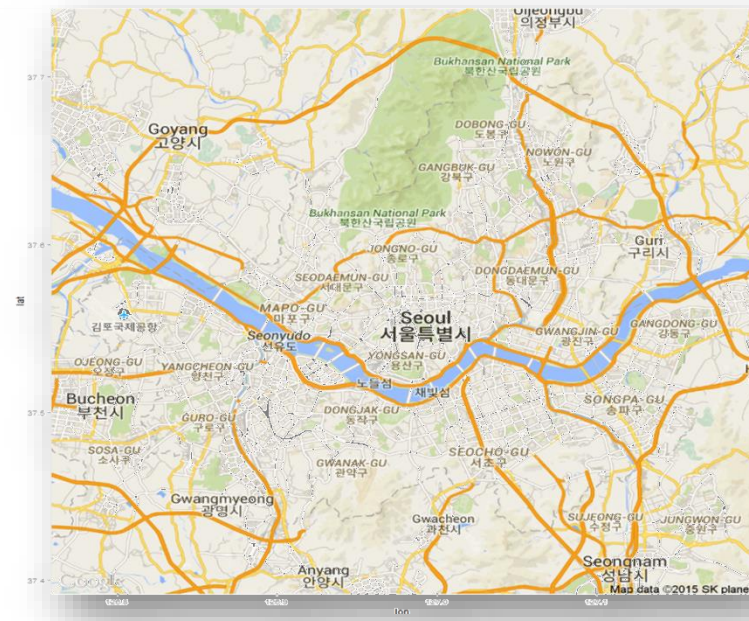
get_map("중심지역", 확대비율, 지도유형) : ggmap 패키지 제공 함수

```
kor <- get_map("seoul", zoom=11, maptype = "roadmap") # 지도정보
```

maptype : roadmap, satellite, terrain, hybrid

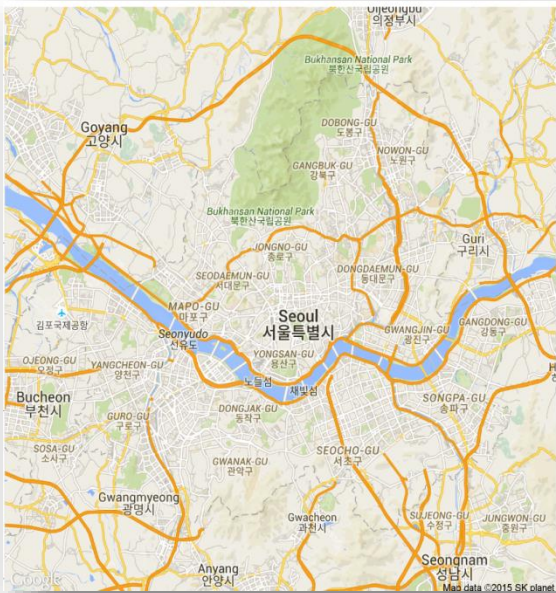
지도 정보로 지도 플로팅

```
ggmap(kor)
```



공간시각화

➤ 레이어 기법으로 공간 시각화 결과화면



레이어1 : 지도



레이어2 : 포인트 추가



레이어3 : 지역명 추가

공간시각화

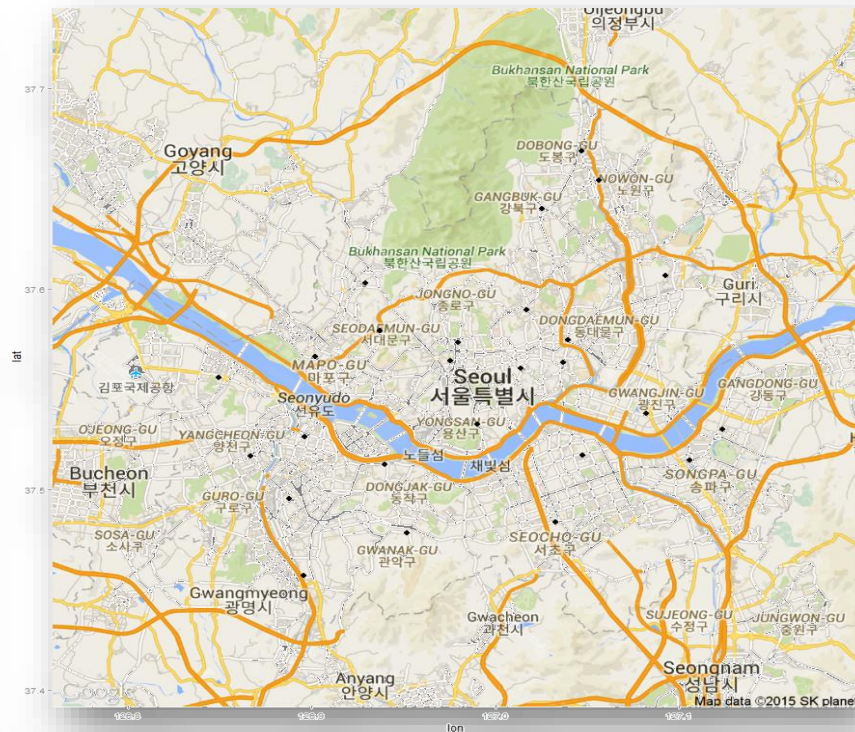
➤ 레이어 기법으로 공간 시각화

레이어1 : 지도 -> 레이어2 : 지도위에 포인트

```
ggmap(kor)+geom_point(data=loc, aes(x=LON, y=LAT),size=3)
```

ggmap(kor) : 지도정보로 지도 플로팅, + geom_point() : 포인트(◆) 추가

+ : 지도 위에 레이어 형태로 포인트 추가



공간시각화

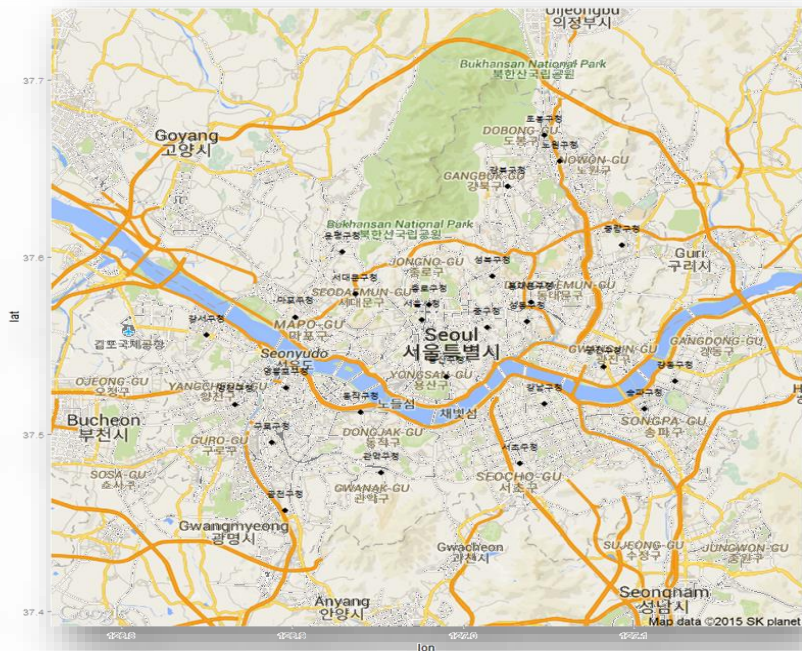
```
kor.map <- ggmap(kor)+geom_point(data=loc, aes(x=LON, y=LAT),size=3)
```

geom_point() : ggplot2에서 제공하는 함수

레이어 3: 지도 위에 포인트 위에 텍스트(구청명) 표시

```
kor.map+geom_text(data=loc, aes(x=LON, y=LAT+0.01,label=구청명),size=3)
```

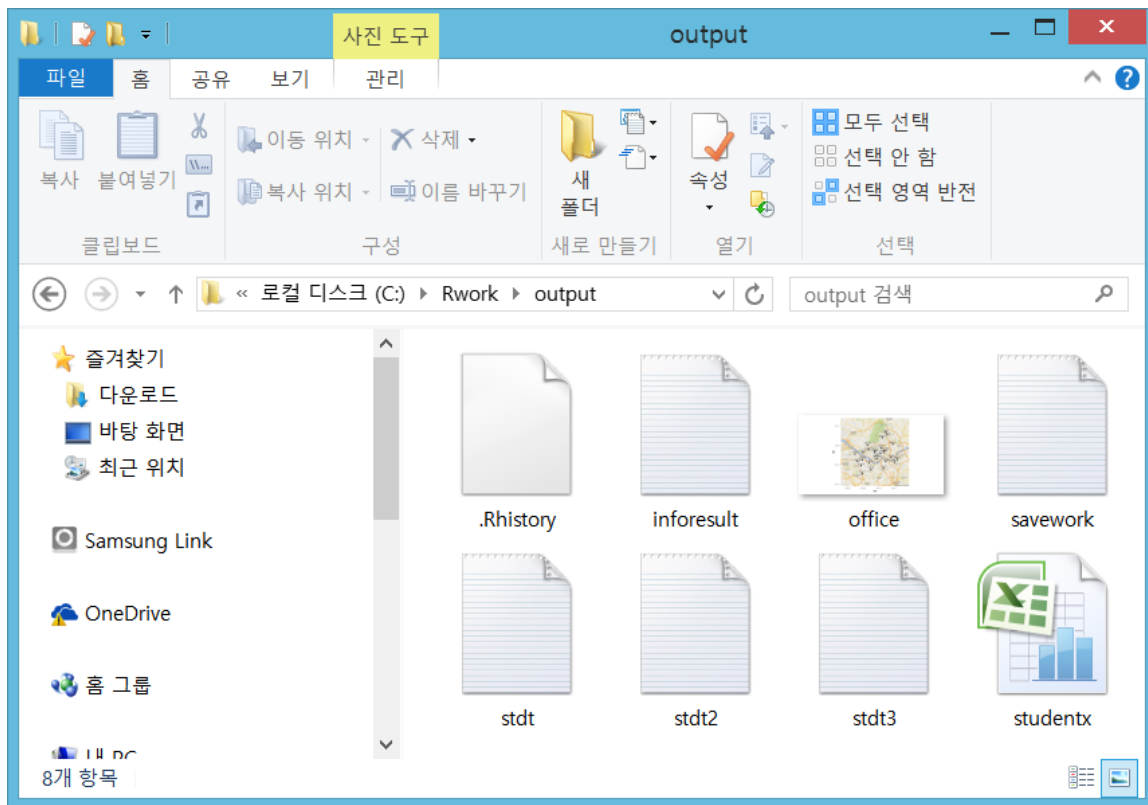
LAT+0.01 : 텍스트 위치(포인트의 0.01 위쪽)



공간시각화

➤ 지도 이미지 파일로 저장

ggsave() : 현재 plots창의 지도를 이미지 파일로 저장 함수
ggsave("C:/workspaces/Rwork/output/office.png", dpi=500)



공간시각화

➤ 공간시각화 실습

공간시각화 실습 데이터 가져오기

```
pop <- read.csv("C:/workspaces/Rwork/data/population2014.csv",header=T)
pop
```

```
lon <- pop$LON
```

```
lat <- pop$LAT
```

```
data <- pop$총인구수
```

```
# 위도,경도,총인구수 이용 데이터프레임 생성
```

```
df <- data.frame(lon,lat,data)
```

```
df
```

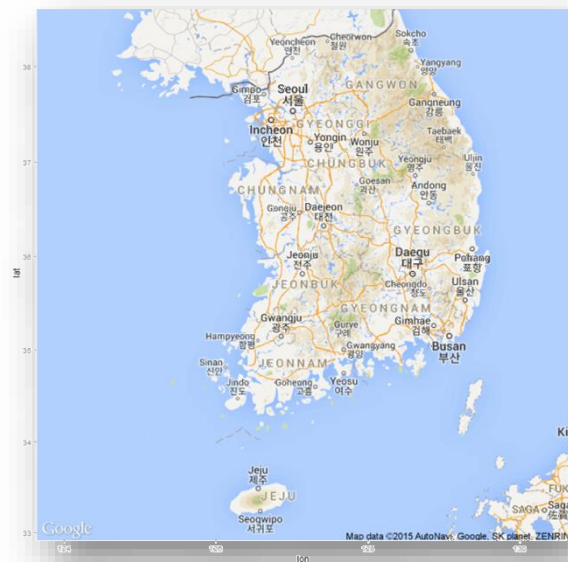
```
# 지도정보 생성
```

```
map <- get_map("Jeonju", zoom=7 , maptype='roadmap')
```

```
# 레이어1: 지도 플로팅
```

```
map1 <- ggmap(map)
```

```
map1
```



공간시각화

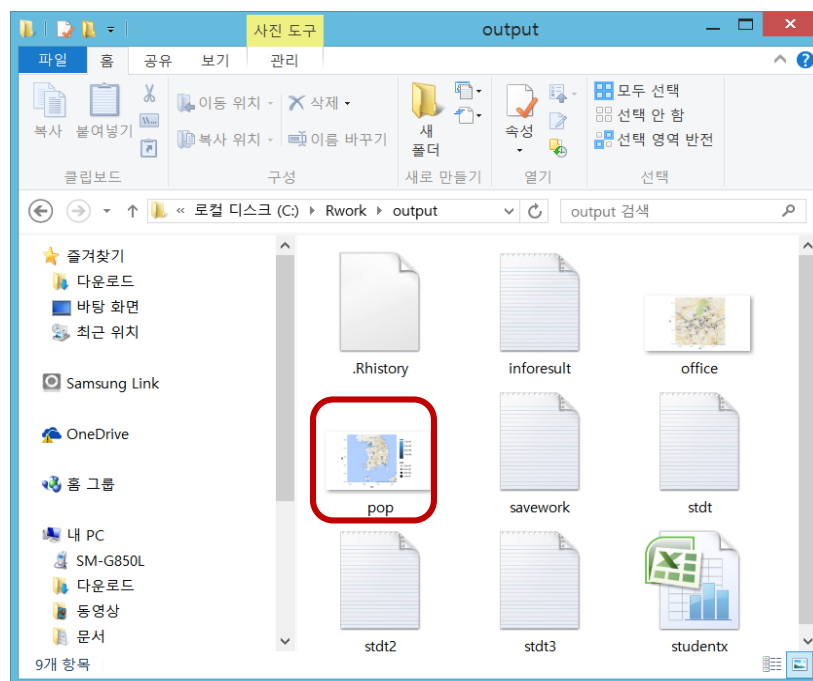
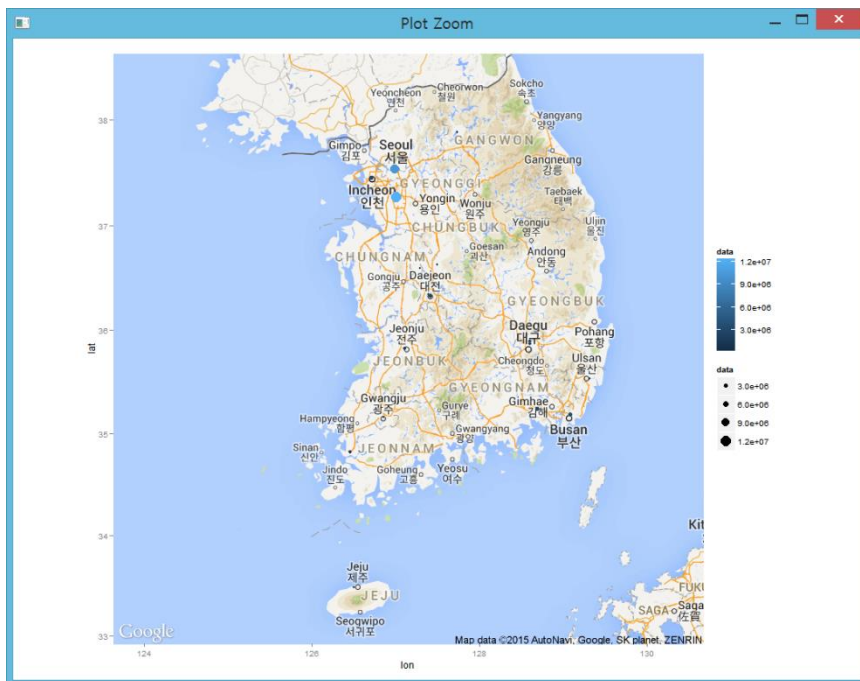
➤ 공간시각화 실습

레이어2 : 포인트 추가

```
map1 + geom_point(aes(x=lon,y=lat,colour=data,size=data),data=df)
```

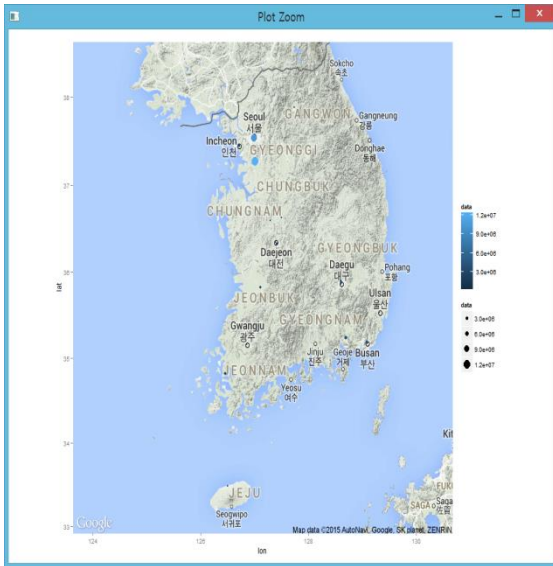
크기, 넓이, 폭, 밀도 적용 파일 저장

```
ggsave("c:/workspaces/Rwork/output/pop.png",scale=1,width=7,height=4,dp  
i=1000)
```

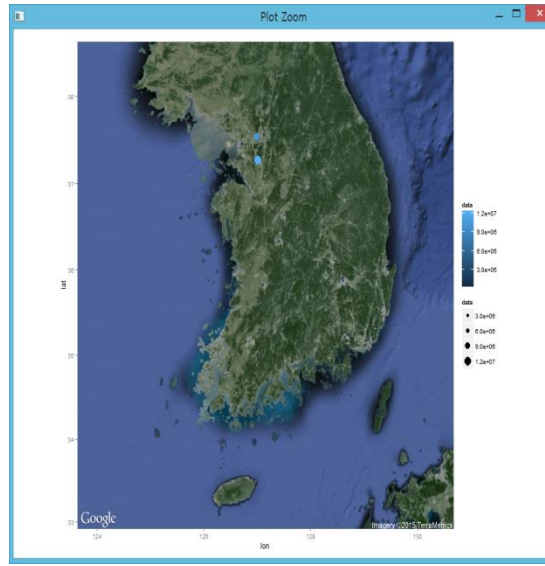


공간시각화

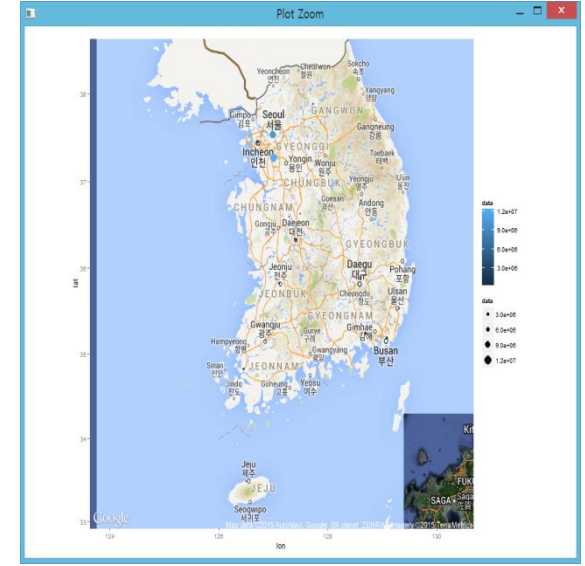
➤ 다양한 지도 유형



maptype='terrain'



maptype='satellite'



maptype='hybrid'

공간시각화

➤ stat_bin2d() 함수

stat_bin2d() : 버블모양 적용함수, 인구수(data) 기준 색 적용

```
map5 <- get_map("Jeonju", zoom=7, maptype='hybrid')
```

```
map5 <- ggmap(map5)
```

```
map5 + stat_bin2d(aes(x=lon,  
  y=lat, colour=data,  
  fill=factor(data),  
  size=data),  
  data=df)
```

