빅데이터를이용한통계그래픽스

HW #2

1678226 유찬미 (Chanmi Yoo)

2019년 10월 4일

###### smallDia.csv 파일에는 7개의 변수를 가진 다이아몬드에 대한 자료가 들어있다.

# Load Packages  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.4.4

## -- Attaching packages ---------------------------------------------------------------------------------------------------- tidyverse 1.2.1 --

## √ ggplot2 3.0.0 √ purrr 0.2.5  
## √ tibble 1.4.2 √ dplyr 0.7.6  
## √ tidyr 0.8.1 √ stringr 1.3.1  
## √ readr 1.2.1 √ forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'readr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'stringr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.4

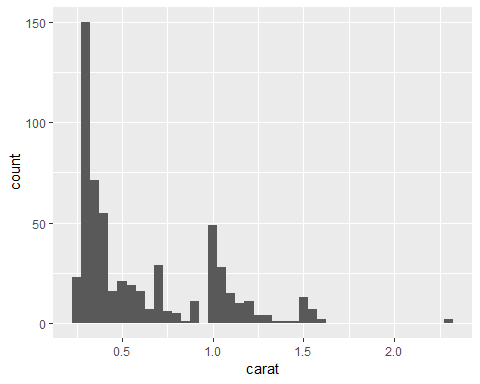
## -- Conflicts ------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(ggplot2)  
  
# Load Data  
smallDia <- read.csv("smallDia.csv")  
head(smallDia)

## carat cut color clarity depth table price  
## 1 0.72 Premium I IF 63.0 57 2795  
## 2 0.60 Very Good G IF 61.6 56 2800  
## 3 0.56 Very Good E IF 61.0 59 2833  
## 4 0.58 Very Good E IF 60.6 59 2852  
## 5 0.62 Fair F IF 60.1 61 2861  
## 6 0.63 Premium E IF 60.3 62 2879

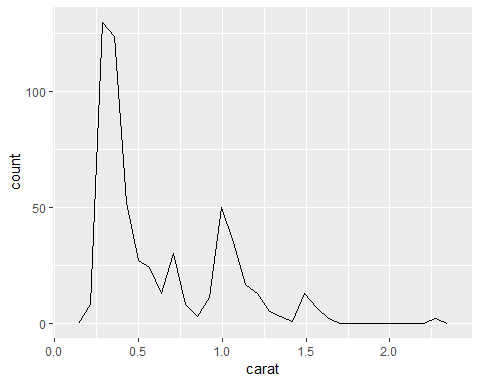
###### 1) 각 변수의 분포를 변수의 특징에 맞는 그림을 그리고 해석하시오.

# 1) carat 변수(연속형)  
ggplot(smallDia, aes(carat)) +  
 geom\_histogram(binwidth = 0.05)



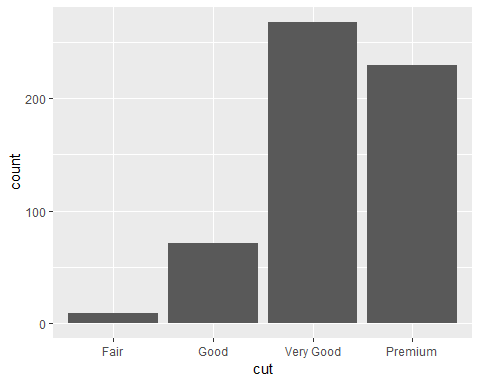
ggplot(smallDia, aes(carat)) +  
 geom\_freqpoly()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



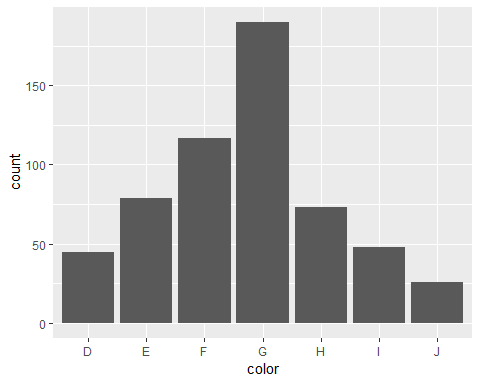
해석: 몇 개의 peak가 보이는데, 무게가 0.3인 경우와 1인 경우가 눈에 띄게 많다.

# 2) cut 변수(범주형)  
ggplot(smallDia, aes(cut)) +  
 geom\_bar() +  
 scale\_x\_discrete(limit=c("Fair", "Good", "Very Good", "Premium")) # 배열순서 변경



해석: cutting 질이 Fair 등급인 다이아몬드는 별로 없고, Very Good과 Premium 등급인 다이아몬드의 개수가 많다.

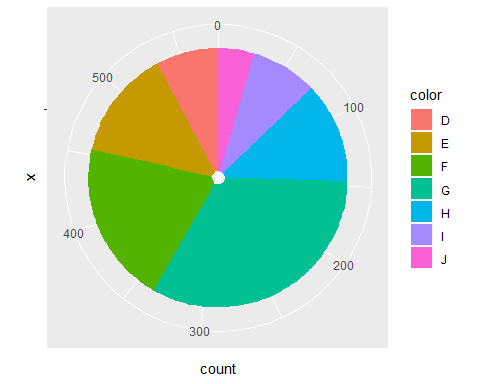
# 3) color (범주형)  
ggplot(smallDia, aes(color)) +  
 geom\_bar()



table(smallDia$color)

##   
## D E F G H I J   
## 45 79 117 190 73 48 26

color\_num <- data.frame(color= c('D', 'E', 'F', 'G', 'H', 'I', 'J' ),  
 count = c(45 , 79, 117, 190, 73 ,48, 26) )  
ggplot(color\_num, aes("", count, fill = color)) +  
 geom\_bar(stat="identity") +  
 coord\_polar("y")

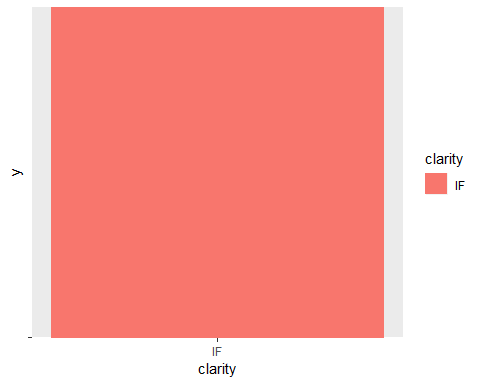


해석: 색상별 개수 차이가 있다. G 색상이 가장 많고, J 색상이 가장 적다.

# 4) clarity(범주형)  
table(smallDia$clarity)

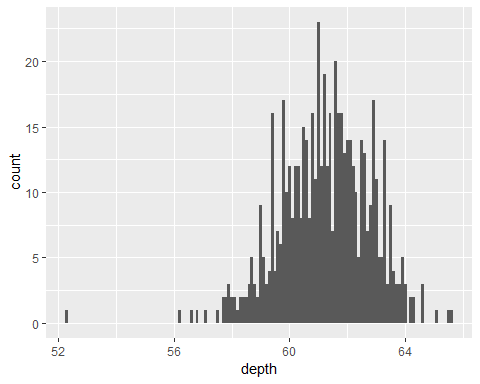
##   
## IF   
## 578

ggplot(smallDia, aes(clarity, "", fill=clarity)) +  
 geom\_bar(stat = "identity")



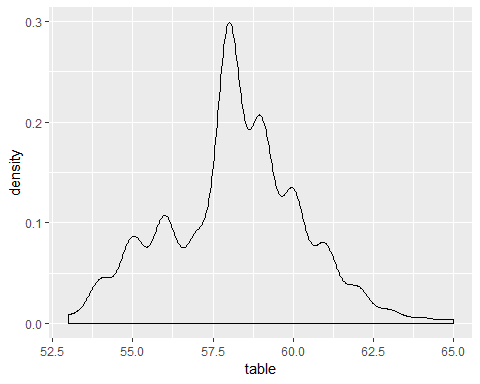
해석: 주어진 자료의 모든 다이아몬드는 투명도가 IF로 동일하다.

# 5) depth (연속형)  
ggplot(smallDia, aes(depth)) +  
 geom\_histogram(binwidth = 0.1)



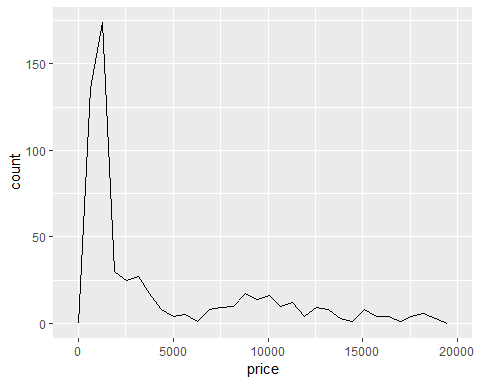
해석: 61을 기준으로 전반적으로 대칭 구조를 따르며, 매우 작은 이상치가 있다.

# 6) table (연속형)  
ggplot(smallDia, aes(table)) +  
 geom\_density()



해석: 58에서 peak가 발생하며, 거의 대칭에 가깝지만 왼쪽으로 아주 살짝 치우친 분포이다.

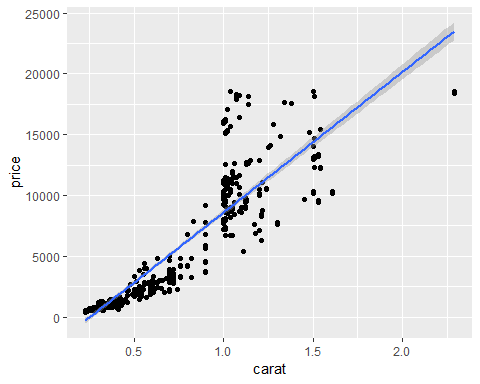
# 7) price (연속형)  
ggplot(smallDia, aes(price)) +  
 geom\_freqpoly(bins = 30)



해석: 가격은 2500 이하인 경우가 대다수였으나, 약 20000까지 간 경우도 있었다. 매우 왼쪽으로 치우친 분포이다.

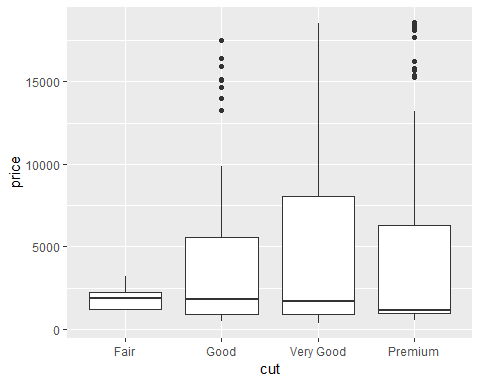
###### 2) 가격에 영향을 미치는 변수를 알아보고자 한다. 이를 위한 다양한 알맞은 그림 을 그린 후 해석하고 결론을 도출하시오.

# 1) carat과 price의 관계   
ggplot(smallDia, aes(carat, price)) +  
 geom\_point() +  
 geom\_smooth(method="lm")

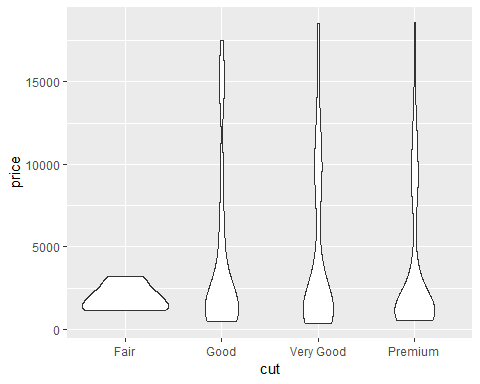


해석1) 전반적으로 price는 carat에 비례한다. 해석2) 그러나 carat(크기)가 커질수록 크기로 가격을 예측하기 어려워진다. 0.5 carat 미만일 때는 크기가 비슷하면 비슷한 가격으로 책정되었고, 0.5~1 carat일 때는 비교적 carat 크기에 비례해 가격이 책정되었지만, 1 carat을 초과하면 비슷한 크기이더라도 가격이 천차만별이다.

# 2) cut과 price의 관계   
ggplot(smallDia, aes(cut, price)) +  
 geom\_boxplot() +  
 scale\_x\_discrete(limits=c("Fair", "Good", "Very Good", "Premium"))

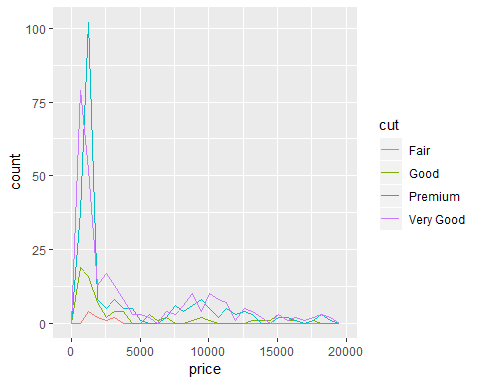


ggplot(smallDia, aes(cut, price)) +  
 geom\_violin() +  
 scale\_x\_discrete(limits=c("Fair", "Good", "Very Good", "Premium"))



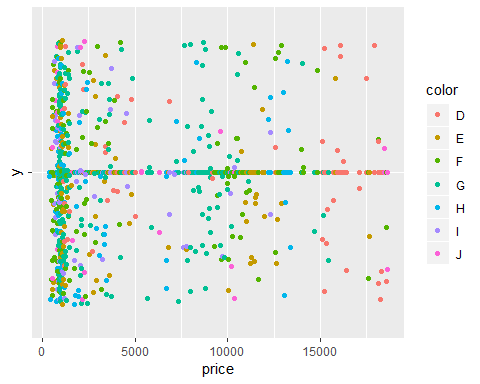
ggplot(smallDia, aes(price, color=cut)) +  
 geom\_freqpoly()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

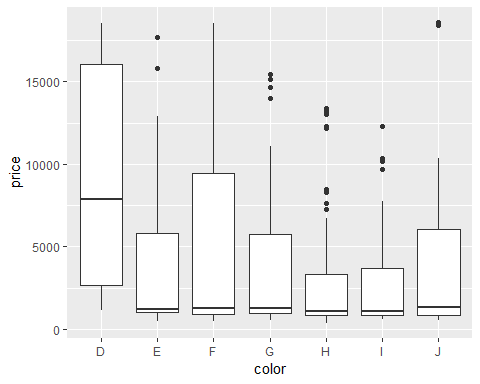


해석1) cutting 질에 상관 없이 가격이 약 2500인 경우가 많았다.  
해석2) cutting의 질은 얼마나 다이아몬드가 비싸질 수 있는가와 관련이 있었다. Fair 등급인 경우 가격이 대부분 2500 이하에서 형성되고, Very Good과 Premium인 경우 10000정도로 비싼 가격에 팔리는 경우도 많았다.

# 3) color와 price의 관계  
ggplot(smallDia, aes(price, "", color=color)) +  
 geom\_point() +  
 geom\_jitter()

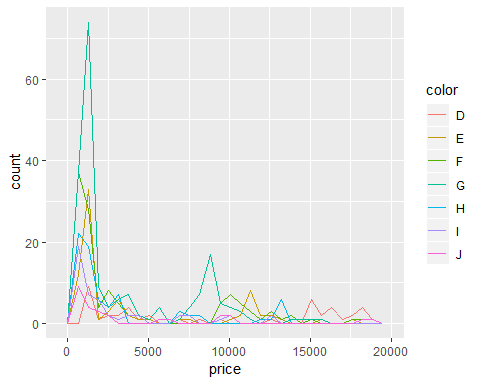


ggplot(smallDia, aes(color, price)) +  
 geom\_boxplot()



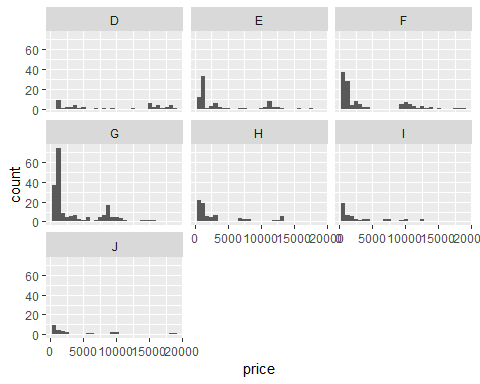
ggplot(smallDia, aes(price, color=color)) +  
 geom\_freqpoly()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



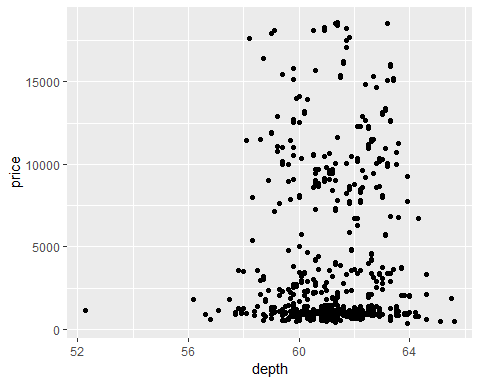
ggplot(smallDia,aes(price)) +   
 geom\_histogram() +   
 facet\_wrap(~color)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

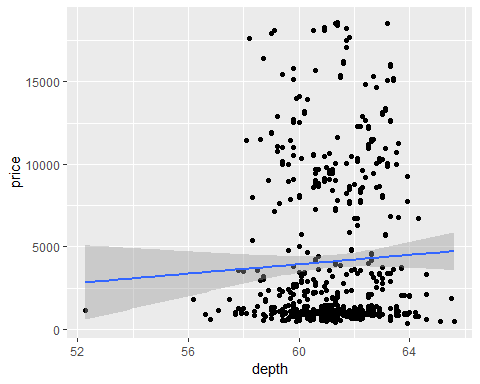


해석) 색상에 따라 대강의 가격이 형성된다. peak를 기준으로 크게 3가지 그룹으로 나눌 수 있다. H, I 색상은 싼 그룹(2500 peak), E, F, G, J는 중간 그룹(2500과 10000 peak), D는 비싼 그룹(2500과 15000 peak)으로 나누어볼 수 있다.

# 4) depth와 price의 관계  
ggplot(smallDia, aes(depth, price)) +  
 geom\_point()

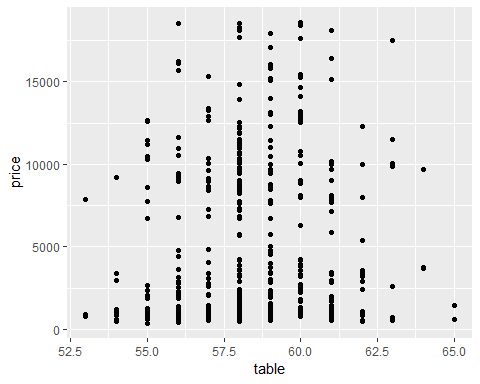


ggplot(smallDia, aes(depth, price)) +  
 geom\_point() +  
 geom\_smooth(method="lm")

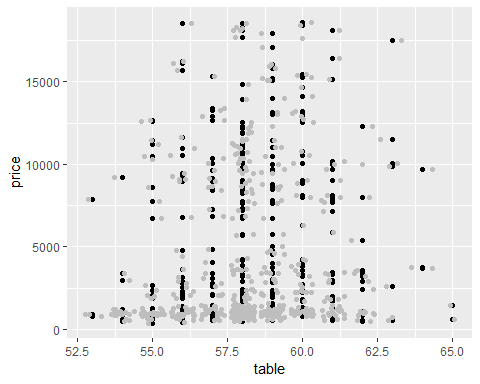


해석: 깊이와 가격 사이 특별한 패턴이 보이지 않는다.

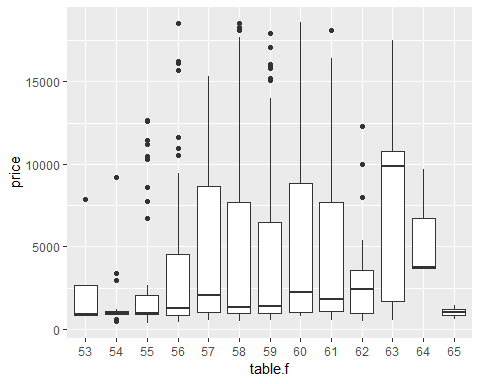
# 5) table과 price의 관계   
ggplot(smallDia, aes(table, price)) +  
 geom\_point()



ggplot(smallDia, aes(table, price)) +  
 geom\_point() +  
 geom\_jitter(color = "grey")

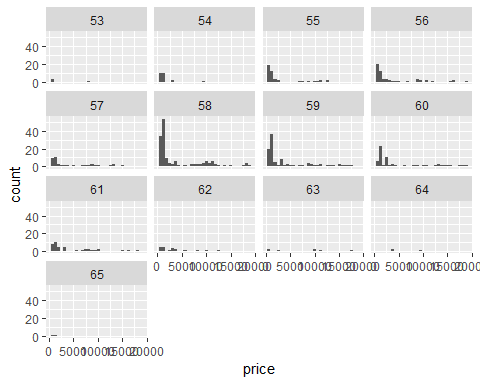


smallDia$table.f <- factor(smallDia$table)  
ggplot(smallDia, aes(table.f, price)) +  
 geom\_boxplot()



ggplot(smallDia, aes(price)) +  
 geom\_histogram() +  
 facet\_wrap(~table)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



해석1) 윗면 길이와 가격이 비례한다고 보기 어렵다.

해석2) 뒷면 길이가 너무 작거나 너무 길지 않을 때, 비싼 다이아몬드가 많았다.

# 6) 산점도 행렬   
library(GGally)

## Warning: package 'GGally' was built under R version 3.4.4

##   
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':  
##   
## nasa

ggpairs(smallDia)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

