시각화

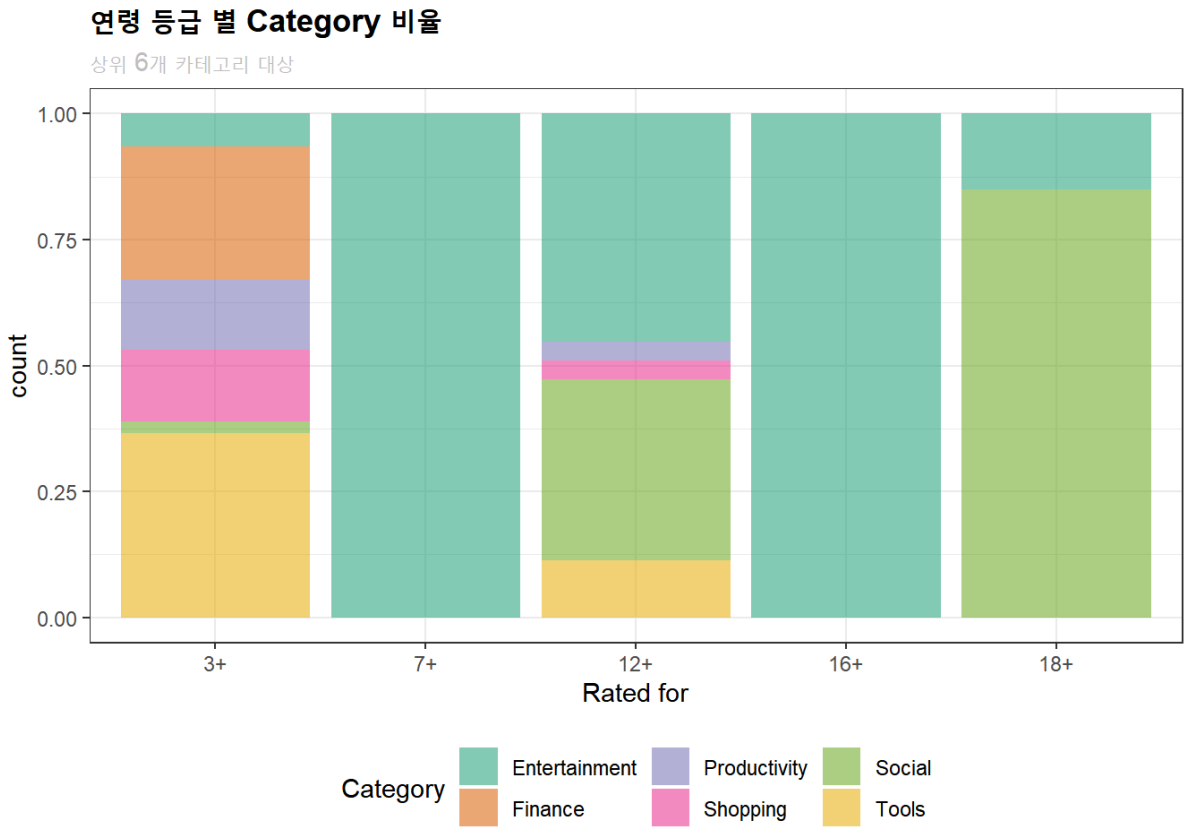
막대그래프 그리기

```
ggplot (data, aes()) + geom_bar(stat=’identity’)
```

꽉 찬 막대그래프(비율 그래프 그리기)

```
top6_df %>%
  ggplot(aes(x=`Rated for`,fill=Category))+
  geom_bar(position='fill',alpha=0.55)+
  scale_fill_brewer(palette='Dark2')+
  theme_bw()+
  labs(title='연령 등급 별 Category 비율',
       subtitle='상위 6개 카테고리 대상')+
  theme(plot.title=element_text(face='bold'),
```

```
        plot.subtitle=element_text(color='grey'),
        legend.position='bottom')
```

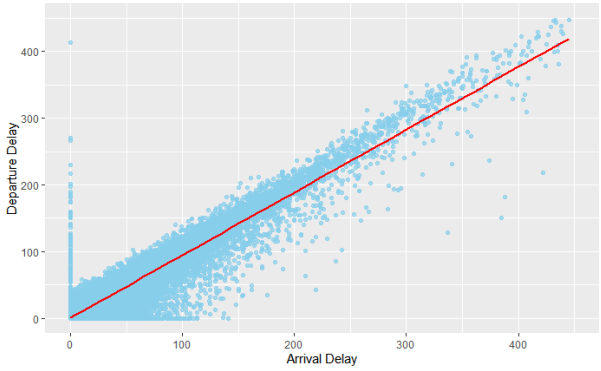


geom_text

```
geom_text(aes(x=col,label=percentage),#percentage는 문자열 데이터
            position=position_stack(vjust=0.5))
```

geom_smooth

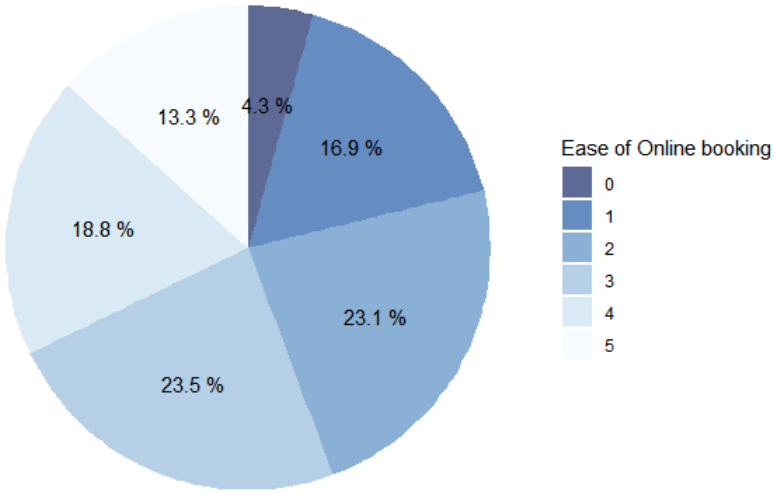
```
train %>%
  ggplot(aes(`Arrival Delay`,`Departure Delay`))+
  geom_point(color='skyblue',alpha=0.7)+
  scale_x_continuous(limits=c(0,450))+
  scale_y_continuous(limits=c(0,450))+
  geom_smooth(method='lm',colour='red')
```



원형 차트 만드는 법

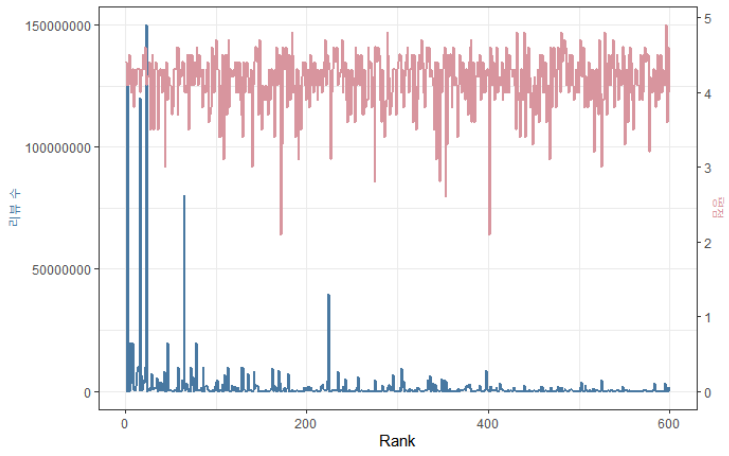
```
c_chart<-function(full=T,data,var,text=T,alpha=1){

# full : 꽉 찬 원형 그래프 / 도넛 그래프 지정
# data : 불러온 데이터 지정
# var : 시각화하고자 한 변수 선택
# text : 그래프에 주석 넣을건지 말건지
# alpha : 그래프의 투명도 지정

  graph<-data %>%
    group_by({{var}}) %>%
    summarise(count=n()) %>%
    mutate(percent=count/sum(count),
           ymax=cumsum(percent),
           ymin=ymax-percent,
           label=paste((round(percent,3)*100) %>% as.character,'%'),
           labelpos=ymax-percent/2) %>%
    ggplot()+
    geom_rect(aes(xmin=2,xmax=4,ymin=ymin,ymax=ymax,fill={{var}}),
              alpha=alpha)+
    coord_polar(theta='y')+
    theme_void()

  if(full==T){graph<-graph}else{
    graph<-graph+xlim(0,4)}

  if(text==T){
    graph<-graph+geom_text(aes(x=3.2,y=labelpos,label=label))}else{
      graph<-graph
    }

  return(graph)
  }
```

```
col <- hcl.colors(6, palette ="Blues")

c_chart(data=train,
        var=`Ease of Online booking`,
        text=T,full=T,alpha=0.75)+
  scale_fill_manual(values=col)

# 일반 ggplot 문법대로 + 해서 색상 지정, 제목 추가를 하면 된다
```



이중축 그래프 그리기

```
ratio<-max(data$review_num)/max(data$`Star Rating`)

data %>% ggplot(aes(x=Rank))+
  geom_line(aes(y=review_num),color='#4878A1',size=0.71)+
  geom_line(aes(y=ratio*`Star Rating`),color='#D8959E',size=0.71)+
  scale_y_continuous(name='리뷰 수',
                     sec.axis=sec_axis(trans=~./ratio,name='평점'))+
  theme_bw()+
  theme(axis.title.y.left = element_text(color='#4878A1'),
        axis.title.y.right = element_text(color='#D8959E'))
```



그래프의 디테일 설정 (예. aesthetics)

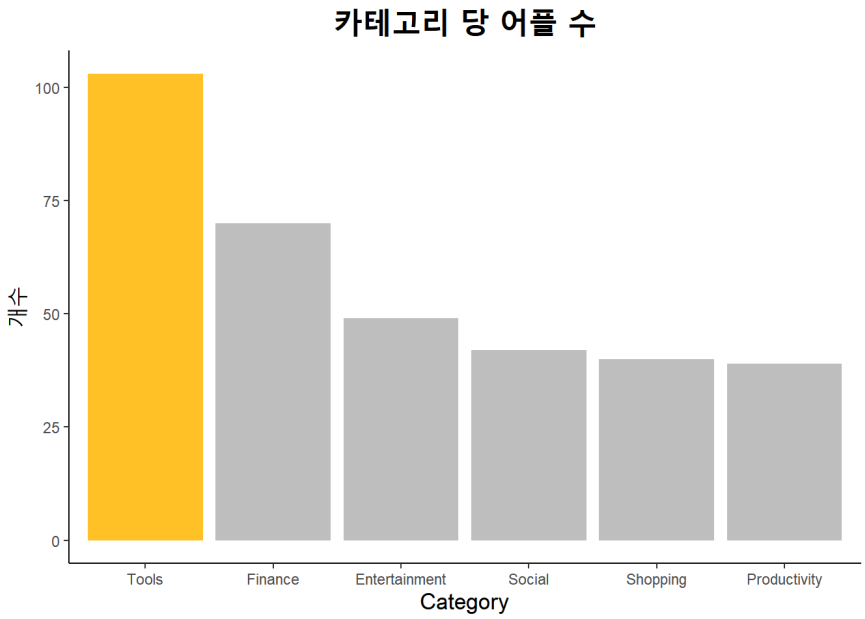
그래프 색칠하기

scale_fill_manual

```
color=c() # 색깔들 지정해주기
ggplot( aes(fill='색칠할 값')) + scale_fill_manual(values=color)
```

각 값에 따른 매핑도 가능

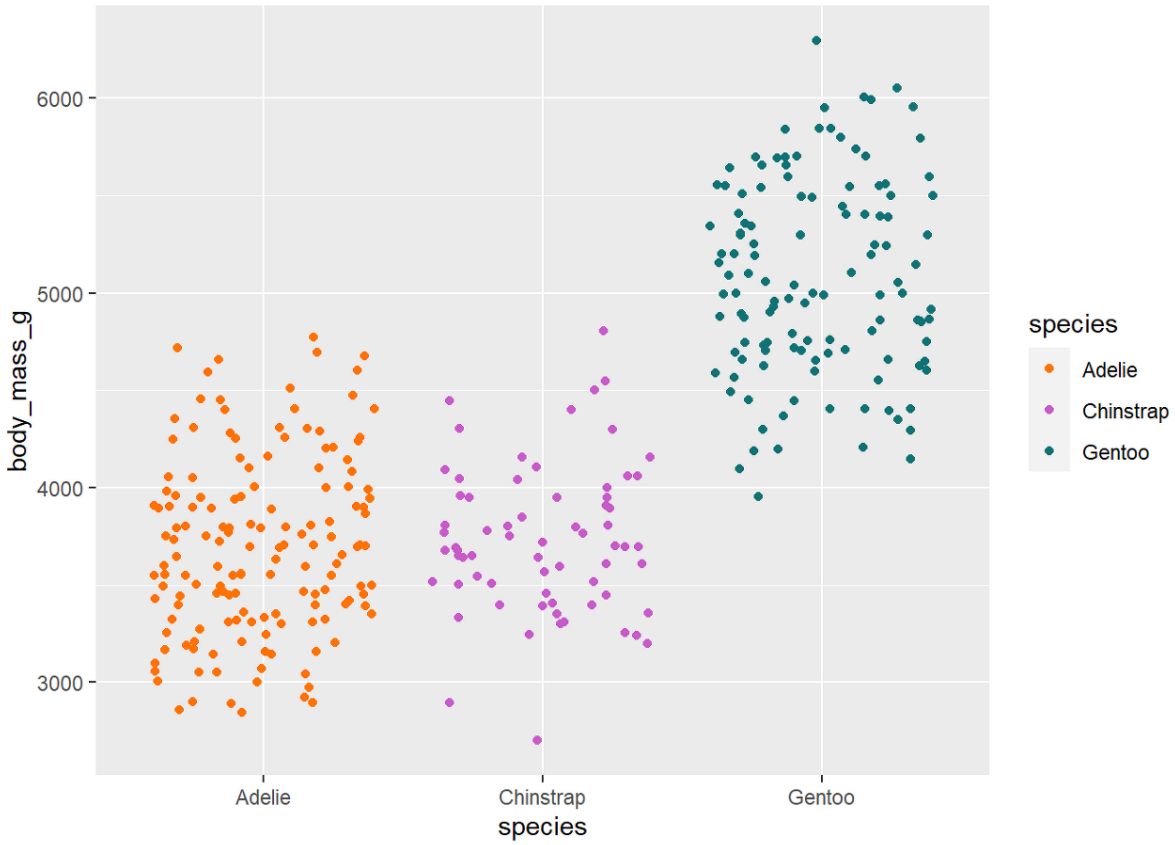
```
#시각화 전 그룹화, 색깔 변수 만들기
top6 <- top6_df %>%
  group_by(Category) %>%
  summarise(n=n()) %>%
  mutate(color=ifelse(n==max(n),'0','1'))

#시각화
top6 %>%
  ggplot(aes(x=reorder(Category,-n), y=n,fill=color)) +
  geom_col() +
  xlab('Category') + ylab("개수") +
  ggtitle("카테고리 당 어플 수") +
  theme_classic()+
  scale_fill_manual(values = c('0'='goldenrod1', '1'='grey'),guide='none') +
  theme(plot.title=element_text(face="bold",size=20,hjust = 0.5),
        axis.title.y  = element_text(size=15),
        axis.title.x = element_text(size=13))
```



scale_color_manual

특정한 색깔을 골라서 점을 찍으려면 이렇게 팔레트를 지정하면 된다.

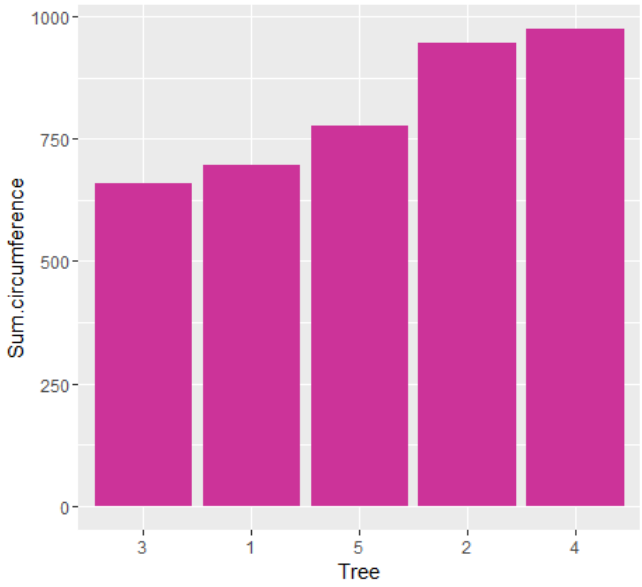
```
c('Adelie' = '#ff7204',
  'Chinstrap' = '#c85aca',
  'Gentoo' = '#0f7375') -> color_values

penguins %>%
  ggplot(aes(x = species,
             y = body_mass_g,
             color = species)) +
  geom_jitter() +
  scale_color_manual(values = color_values)
```



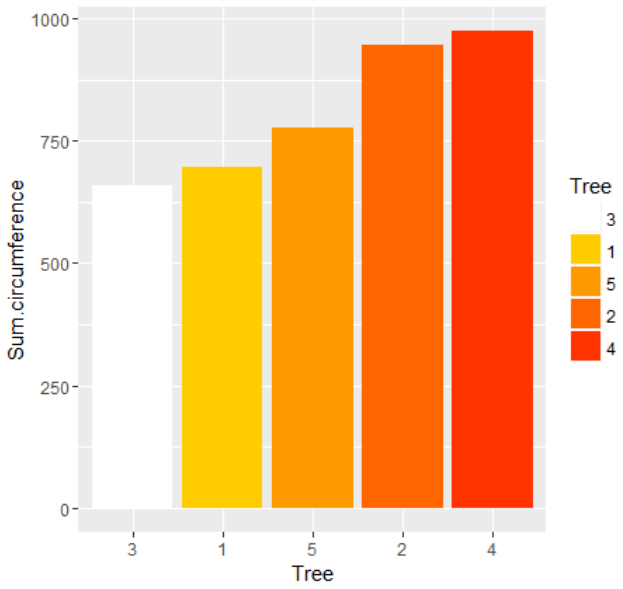
color/palette

1. 단일 색상 지정

```
library(dplyr)
library(ggplot2)

Orange%>%
  group_by(Tree)%>%
  summarize(Sum.circumference=sum(circumference))%>%
  ggplot(aes(Tree, Sum.circumference))+
  geom_bar(stat='identity',fill='#CC3399')
```



2. 개별 색상 지정



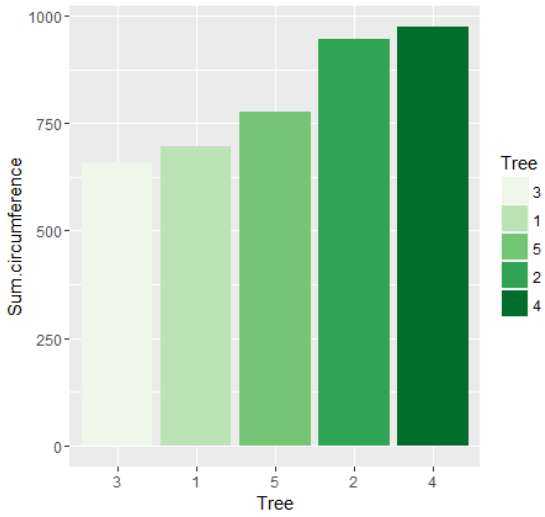

```
p<-Orange%>%
    group_by(Tree)%>%
    summarize(Sum.circumference=sum(circumference))%>%
    ggplot(aes(Tree, Sum.circumference, fill=Tree))+
    geom_bar(stat='identity')

p + scale_fill_manual(values = c( "#FFFFFF", "#FFCC00", "#FF9900",
                                  "#FF6600", "#FF3300"))
```

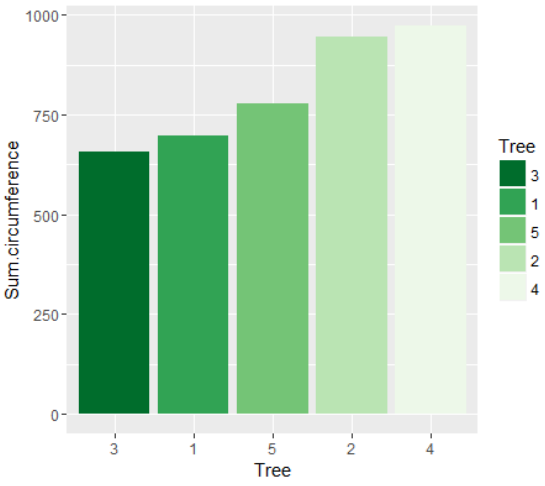
3. color palette chart

```
p<-Orange%>%
    group_by(Tree)%>%
    summarize(Sum.circumference=sum(circumference))%>%
    ggplot(aes(Tree, Sum.circumference, fill=Tree))+
    geom_bar(stat='identity')
```

```
p + scale_fill_brewer(palette = "Greens")
```

그래프의 종류에 따라 fill과 color를 다르게 적용
line 그래프면 scale_color_ brewer를 입력



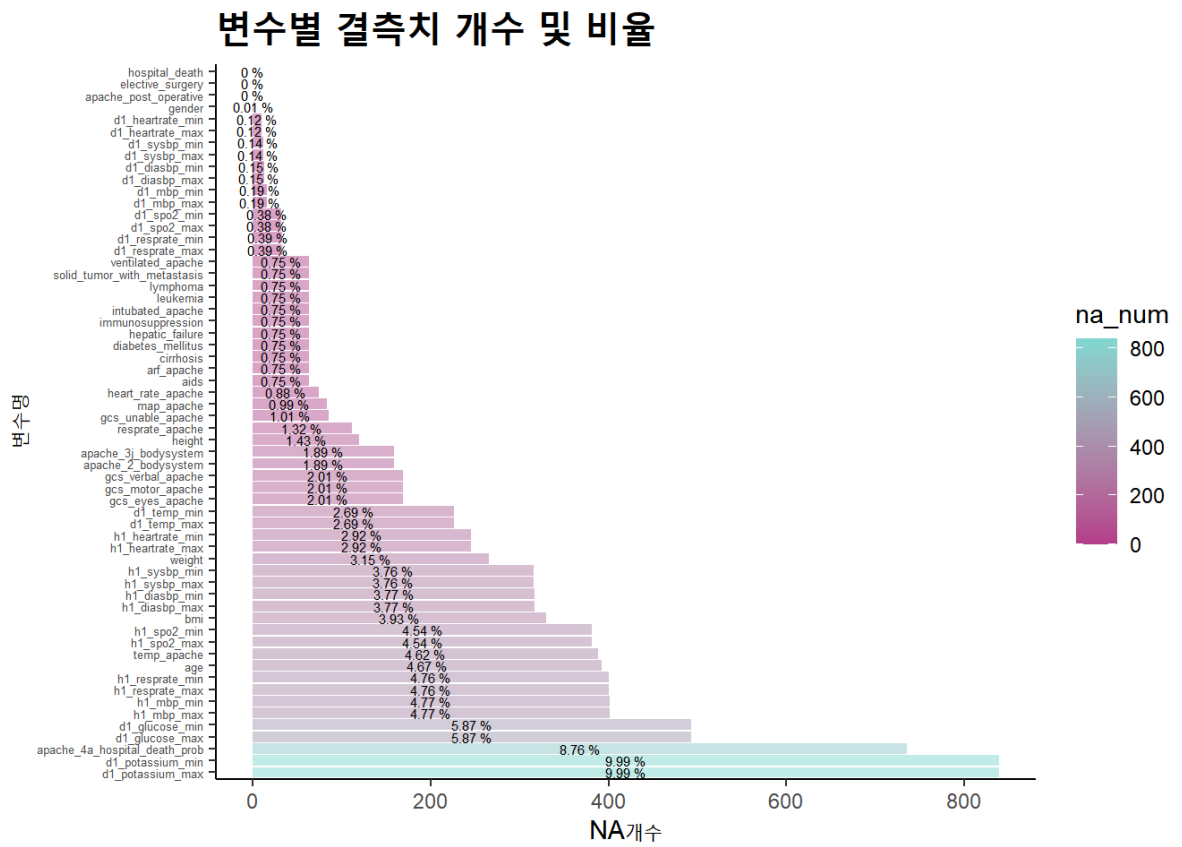



```
direction = -1 이라는 코드를 적용해주면 팔레트에서 적용되는 색상의 순서를 반대로 바꿔줄 수 있음
p + scale_fill_brewer(palette = "Greens", direction = -1)
```

color setting

scale_fill_gradient

```
#options(repr.plot.width = 15, repr.plot.height =15)

nadata %>%
  ggplot(aes(reorder(colnames,-na),na))+
  geom_bar(aes(fill = na),stat='identity',alpha=0.5)+
  scale_fill_gradient(low='#B43C8A',high='#81D8D0')+
  labs(title='변수별 결측치 개수 및 비율',
       x='변수명',y='NA개수',fill='na_num')+
  geom_text(aes(label=percentage),size=2,
            position=position_stack(vjust=0.5))+
  theme_classic()+
  theme(plot.title=element_text(face='bold',size=18),
        axis.text.y= element_text(size=5))+
  coord_flip()
```



플랏 배경색

- 배경을 하얀색으로 (grid 有) : `theme_bw()`
- `theme_classic()`
- 이 외게 플랏 바깥선, 모눈의 색도 바꿀 수 있음 theme()을 통해서
- `theme_void()` : 그래프 말고는 싹 다 없앰

그래프 제목

1. 그래프 제목, x, y 축 제목 설정하기

- `labs` 함수 사용 : `labs(title , x , y, subtitle, fill )` * fill은 범례제목
- `ggtitle()`, `xlab()`, `ylab()`

2. 그래프 제목 세부사항 설정

```
theme(plot.title = element_text(hjust = 0.5)) # hjust 인자에 0~1사이 입력

theme(plot.subtitle=element_text(size=15,color="grey"))

theme(axis.title=element_text(size=10))

# 이중축의 경우
theme(axis.title.y.left =element_text(),
      axis.title.y.right =element_text(colour="#D8959E"))
```

- `element_text()` : size, face(='bold') , angle, color, vjust(0이면 아래 1이면 위) ,hjust(좌우)

- 제목 류 글자 없애기 : element_blank() (element_text 대신 쓰면 됨)

축바꾸기

`coord_flip()`

alpha를 이용해 투명도 조절하기

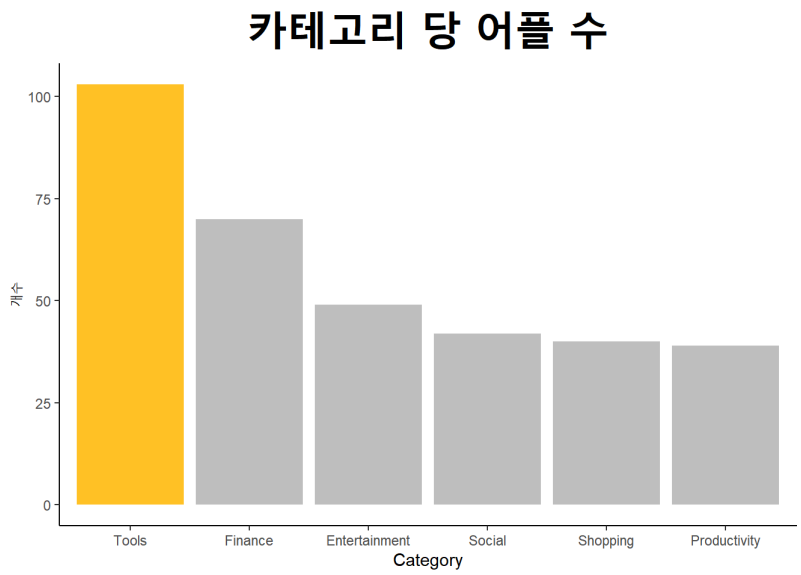
```
geom_bar(stat='identity',alpha=0.3)
geom_col(position='fill', alpha=0.7)
```

범례

reorder 함수를 이용해 그래프 정렬하기

```
ggplot(rate,aes(reorder(Category,mean),mean,fill=Category))
ggplot(top6,aes(x=reorder(Category,-num),y=num,fill=reorder(Category,-num)))

# 내림차순 의 경우 '-' 표시 해주기
```

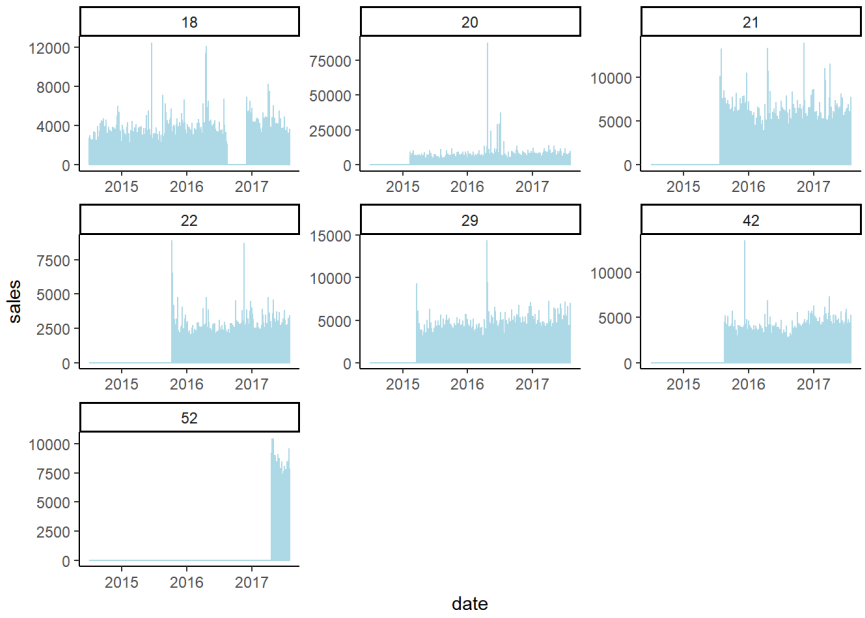
```
top6<-top6_df %>% group_by(Category) %>% summarise(num=n())

color<-c('goldenrod1',rep('grey',5))

ggplot(top6,aes(x=reorder(Category,-num),y=num,fill=reorder(Category,-num)))+
  geom_bar(stat='identity')+
  scale_fill_manual(values=color)+
  xlab('Category')+ylab('개수')+theme_classic()+
  ggtitle("카테고리 당 어플 수")+
  theme(plot.title=element_text(size=30,face="bold",hjust=0.5),legend.position='none')
```



ggplot x축, y축 범위 지정

facet_wrap()

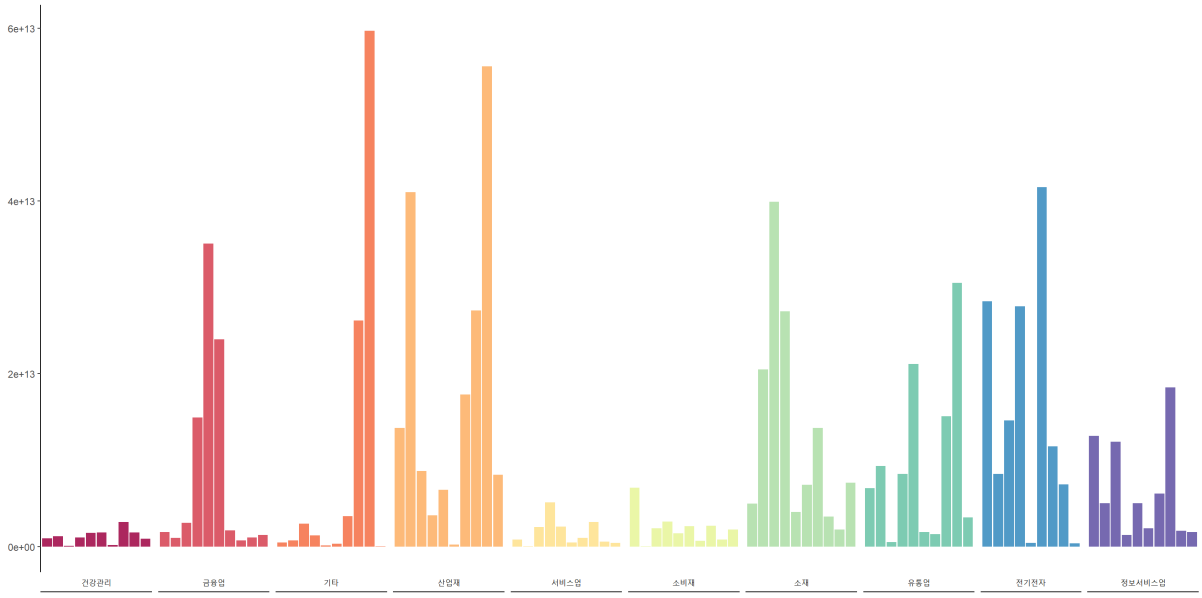
```
train %>%
  filter(store_nbr %in% percent10) %>%
  ggplot(aes(x=date,y=sales))+
  geom_line(color='lightblue')+
  facet_wrap(vars(store_nbr),nrow=3,scale='free')+
  theme_classic()
```



Argument	사용예시	의미
`vars()`	`vars(컬럼이름K)`	면분할 기준이 될 컬럼 지정 * 컬럼K는 범주형(factor) 변수여야 함
`nrow`	`nrow=숫자`	출력되는 면분할 그래프의 서브플롯 행 개수 지정
`ncol`	`ncol=숫자`	출력될 면분할 그래프의 서브플롯 열 개수 지정
`labeller`	`labeller='label_both'`	서브플롯에 변수 이름(컬럼K,..) 표시 여부 지정
`as.table`	`as.table=TRUE`	TRUE / FALSE 서브플롯 정렬 순서 지정
`scales`	`scales='fixed'`	free / fixed / free_x / free_y (default : fixed) 각 서브플롯의 x축, y축 통일(fixed) 여부 지정
`strip.position`	`strip.position='top'`	top / bottom / left / right 각 서브플롯 라벨의 위치 지정

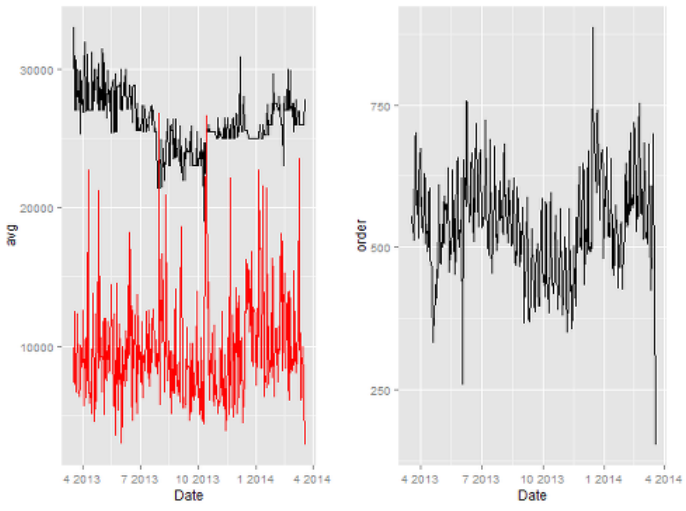
```
data %>% filter(회사명!='삼성전자') %>%
  ggplot(aes(x = 회사명,  y = `2021매출액`, fill = 대분류))  +
  geom_col(position = "dodge",alpha=0.85) +
  facet_grid(~대분류, scales = "free_x", space = "free_x",switch = "x") +
  scale_fill_brewer(palette="Spectral")+
  theme_classic()+
  theme(legend.position='none',
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        strip.background = element_blank())
```

참고



gridExtra()

```
require('gridExtra')
require('ggplot2')

p1 <- ggplot(HH1, aes(x=Date, y=avg) + geom_line()
p2 <- ggplot(HH1, aes(x=Date, y=order) + geom_line()

grid.arrange(p1, p2, ncol=2)
```

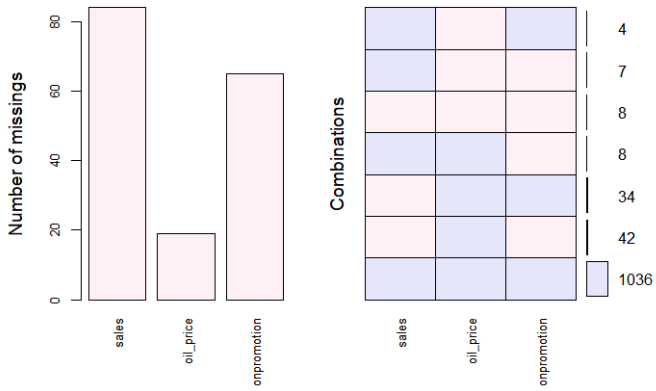


클러스터링 시각화 함수

NA 패턴 시각화

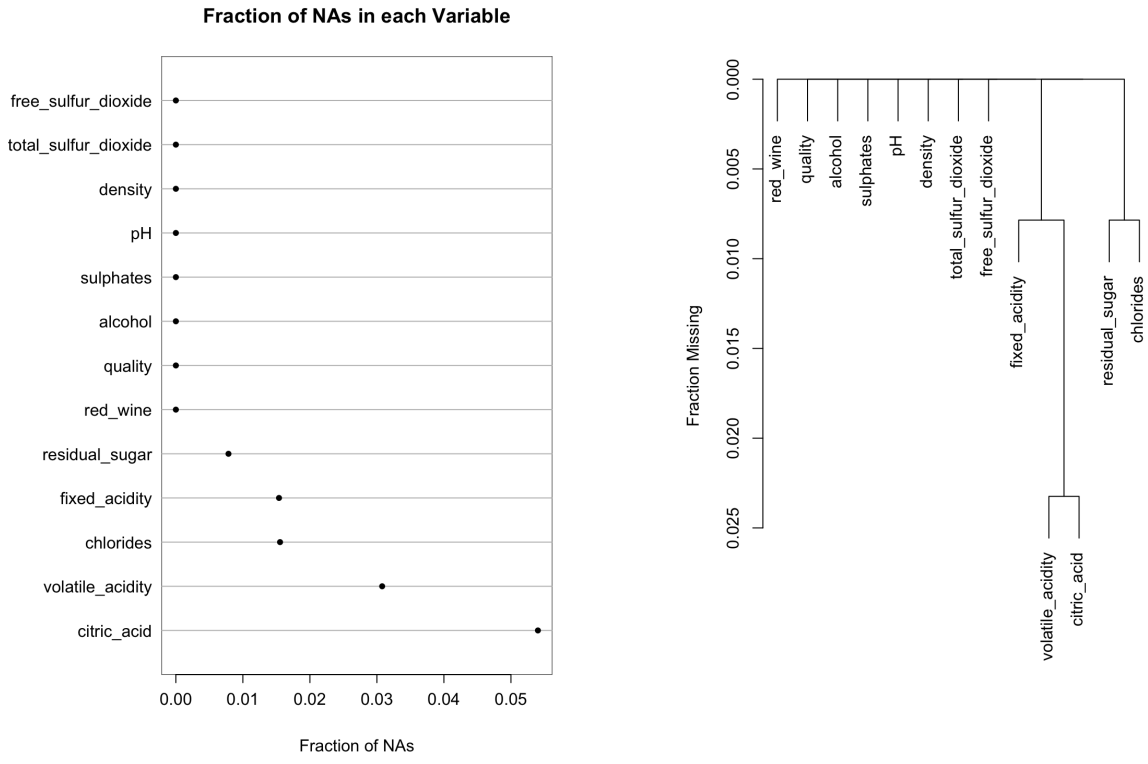
VIM 패키지 이용

```
library(VIM)

aggr(data,
     col=c('lavender','lavenderblush'),
     numbers=TRUE,
     prop=FALSE,
     cex.axis=0.8)
```



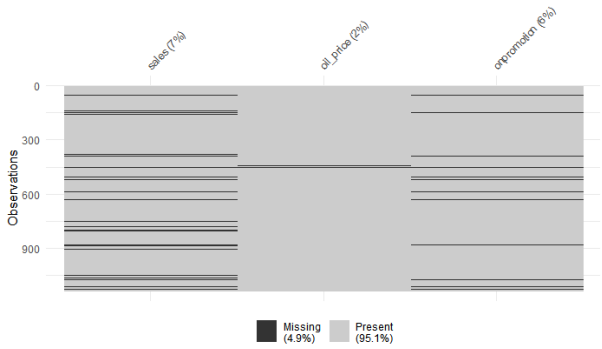
Hmisc 패키지 이용

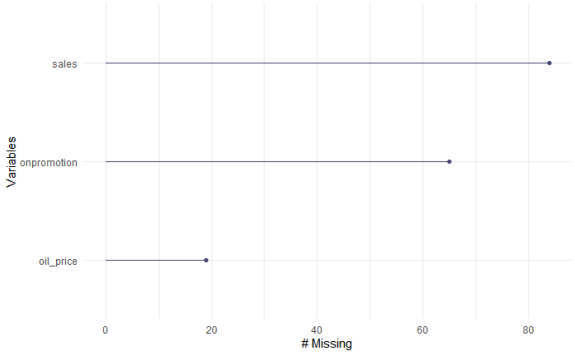
```
par(mfrow = c(1,2))
na_patterns <- Hmisc::naclus(missing_df)
Hmisc::naplot(na_patterns, 'na per var')
plot(na_patterns)
```

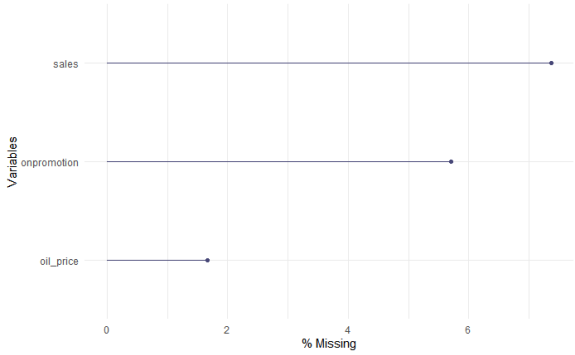


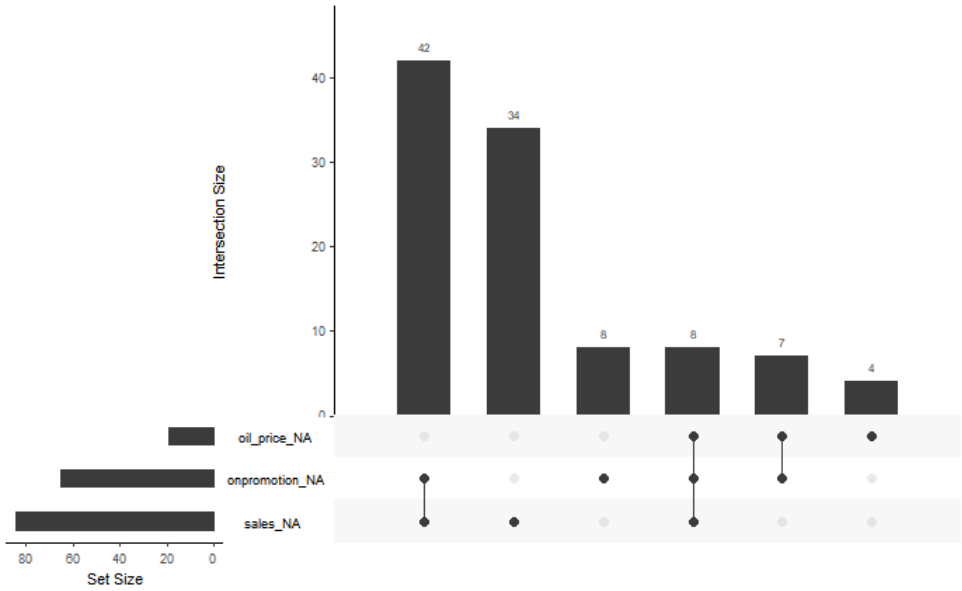
naniar 패키지 이용

```
library(naniar)
vis_miss(data) #파이썬 msno
gg_miss_var(data)
gg_miss_var(data,show_pct=TRUE)
gg_miss_upset(data)
```









상관관계 플랏

corrplot title 위치 조정

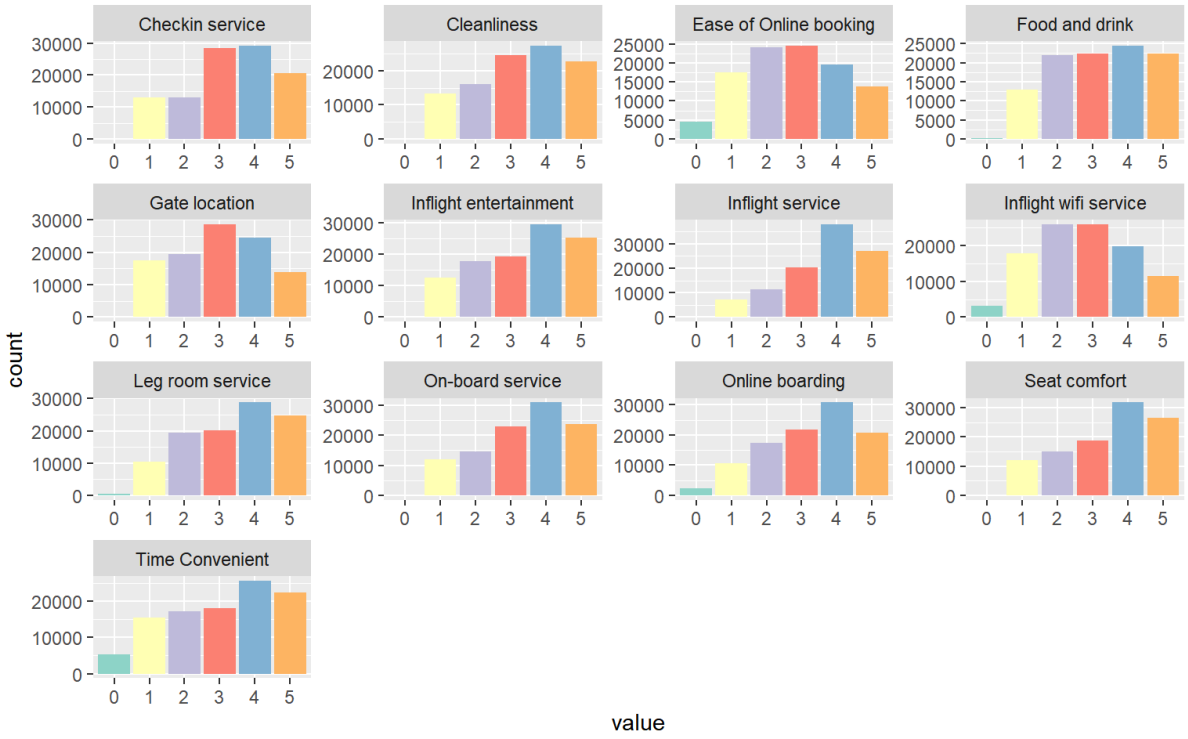
변수들의 분포 시각화 하는 법

크게 두가지 방법이 있다.

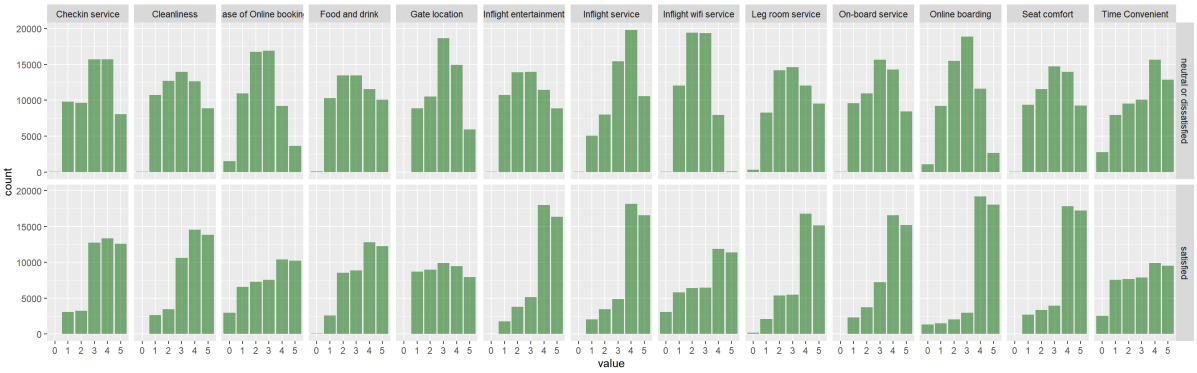
gather() & facet_wrap()/facetgrid()를 이용한 시각화

1. 범주형 자료의 경우

```
train %>% select(all_of(levels_6) )%>%
  gather %>%
  group_by(key,value) %>%
  summarise(count=n()) %>%
  ggplot(aes(value,count))+
  geom_bar(aes(fill=value),stat='identity')+
  scale_fill_brewer(palette='Set3')+
  facet_wrap(vars(key),ncol=4,scales='free')+
  theme(legend.position='none')
```

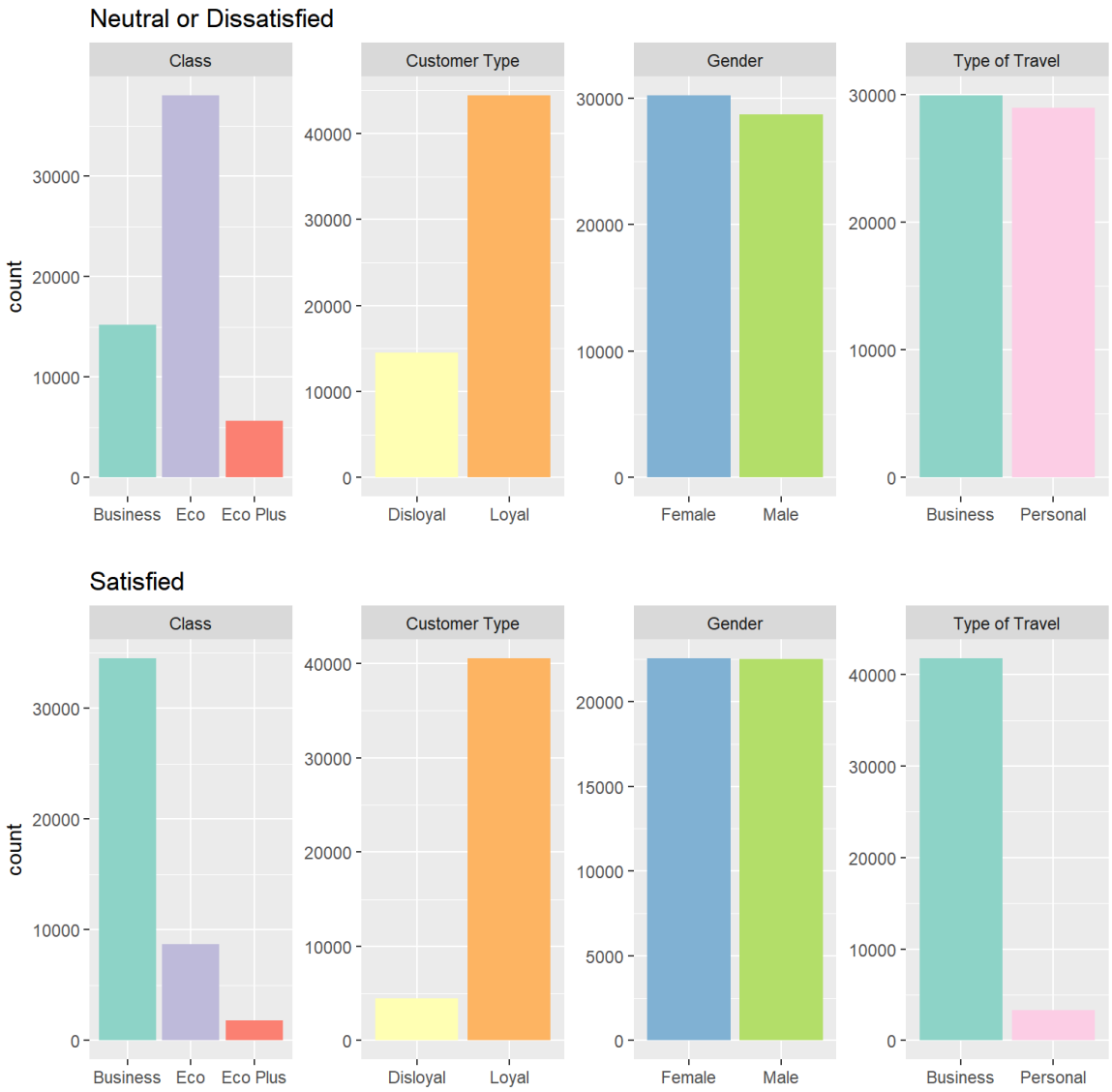



```
train %>% select(c(levels_6,'satisfaction')) %>%
  gather(levels_6,key='key',value='value') %>%
  group_by(satisfaction,key,value) %>%
  summarise(count=n()) %>%
  ggplot(aes(value,count))+
  geom_bar(aes(fill=value),stat='identity',
           fill='darkgreen',alpha=0.5)+
  facet_grid(satisfaction~key)+
  theme(legend.position='none')
```



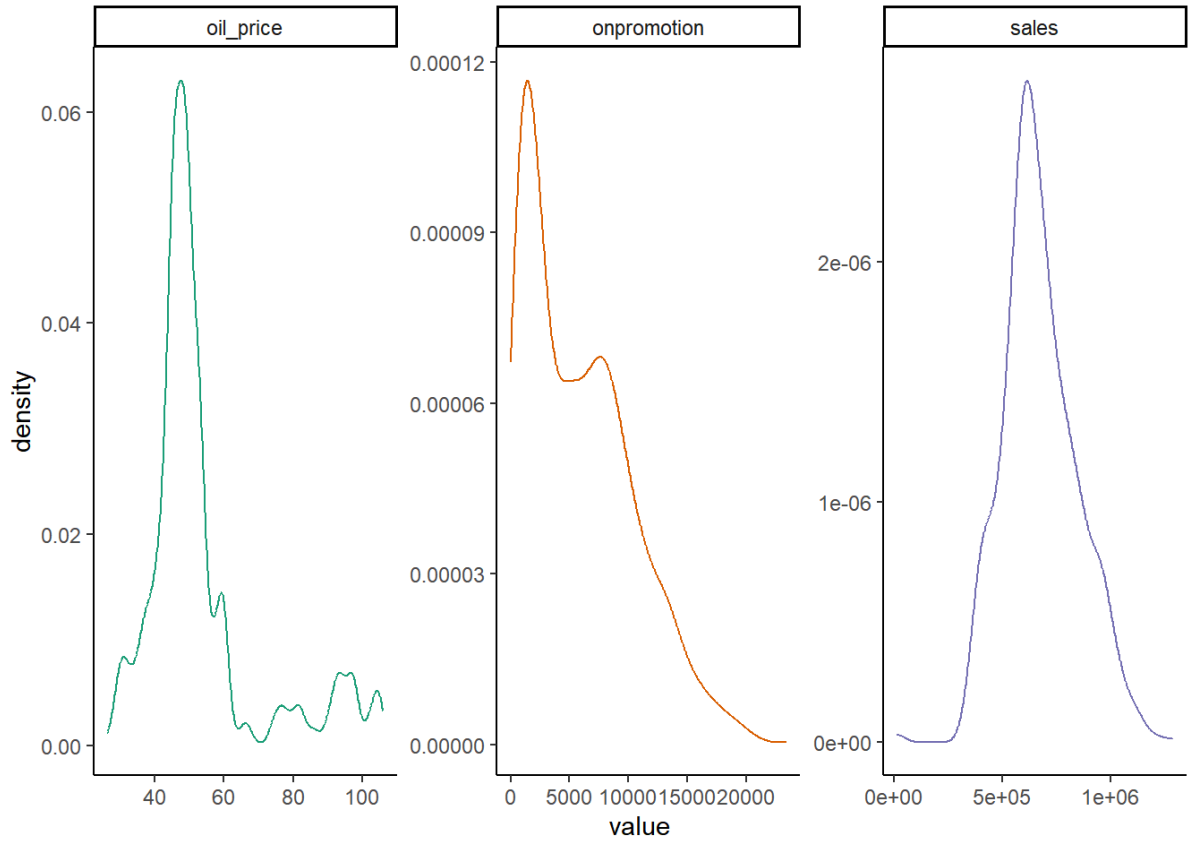
▼ 조금 긴 코드

```
nd<-train %>% filter(satisfaction=='neutral or dissatisfied') %>%
  select(all_of(under_6)) %>%
  gather %>%
  group_by(key,value) %>%
  summarise(count=n()) %>%
  ggplot(aes(value,count))+
  geom_bar(aes(fill=value),stat='identity')+
  scale_fill_brewer(palette='Set3')+
  facet_wrap(vars(key),ncol=4,scales='free')+
  theme(legend.position='none')+
  labs(x='',title='Neutral or Dissatisfied')

sa<-train %>% filter(satisfaction=='satisfied') %>%
  select(all_of(under_6))%>%
  gather %>%
  group_by(key,value) %>%
  summarise(count=n()) %>%
  ggplot(aes(value,count))+
  geom_bar(aes(fill=value),stat='identity')+
  scale_fill_brewer(palette='Set3')+
  facet_wrap(vars(key),ncol=4,scales='free')+
  theme(legend.position='none')+
  labs(x='',title='Satisfied')

library(ggpubr)
ggarrange(nd,sa,ncol=1,legend='none')
```

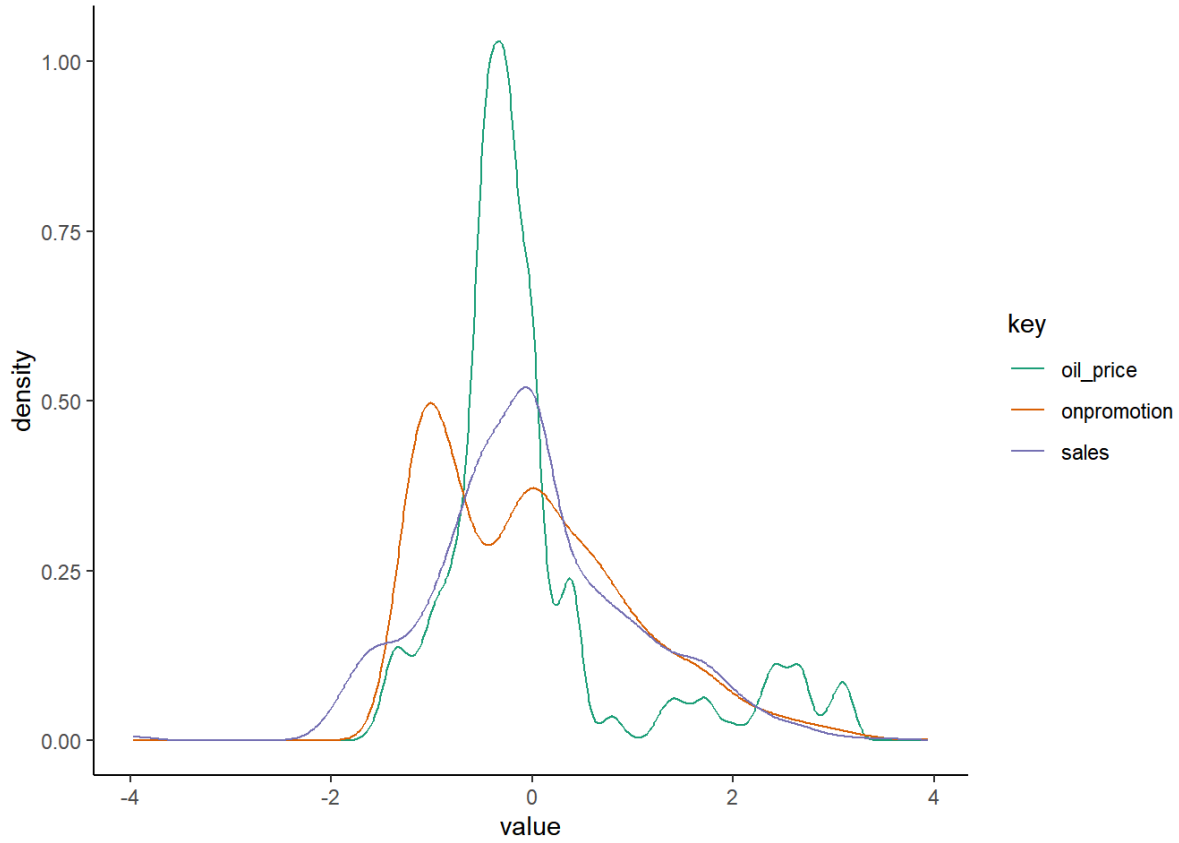


2. 수치형 변수

```
data %>% gather %>% ggplot(aes(x=value,color=key))+
  geom_line(stat='density')+
  theme_classic()+
  scale_color_brewer(palette="Dark2")+
  theme(legend.position = "none")+
  facet_wrap(vars(key),ncol=3,scales="free")
```




```
data %<>% scale %>% as.data.frame

data %>% gather %>%
  ggplot(aes(x=value,color=key))+
  geom_line(stat='density')+
  theme_classic()+
  scale_color_brewer(palette="Dark2")
```



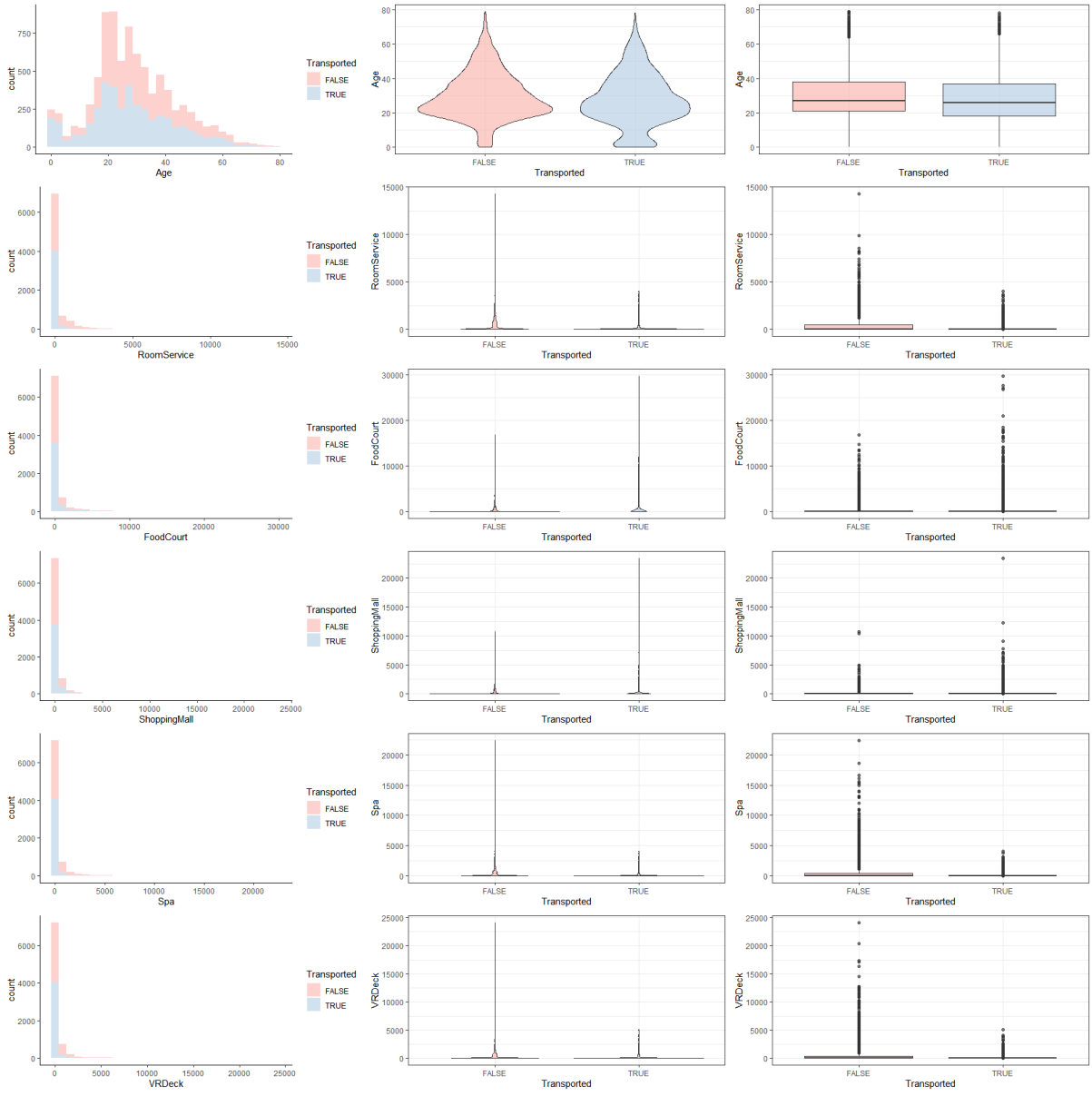
lapply와 grid.arrange(grobs=list)를 이용해서 시각화하기

수치형 변수 시각화하기

```
시각화함수 만들기

num_vis<-function(var){


  hist<-train %>%
    ggplot(aes_string(x={{var}}))+
    geom_histogram(aes(fill=Transported),alpha=0.6)+
    scale_fill_brewer(palette='Pastel1')+
    theme(legend.position='none')+
    theme_classic()


  violin<-train %>%
    ggplot()+
    geom_violin(aes_string(x='Transported',y={{var}},fill='Transported'),
              alpha=0.65)+
    theme_bw()+
    scale_fill_brewer(palette='Pastel1')+
    theme(legend.position='none')

  box<-train %>%
    ggplot()+
    geom_boxplot(aes_string(x='Transported',y={{var}},fill='Transported'),
              alpha=0.65)+
    scale_fill_brewer(palette='Pastel1')+
    theme_bw()+
    theme(legend.position='none')

  num_plot<-grid.arrange(hist,violin,box,ncol=3)
```

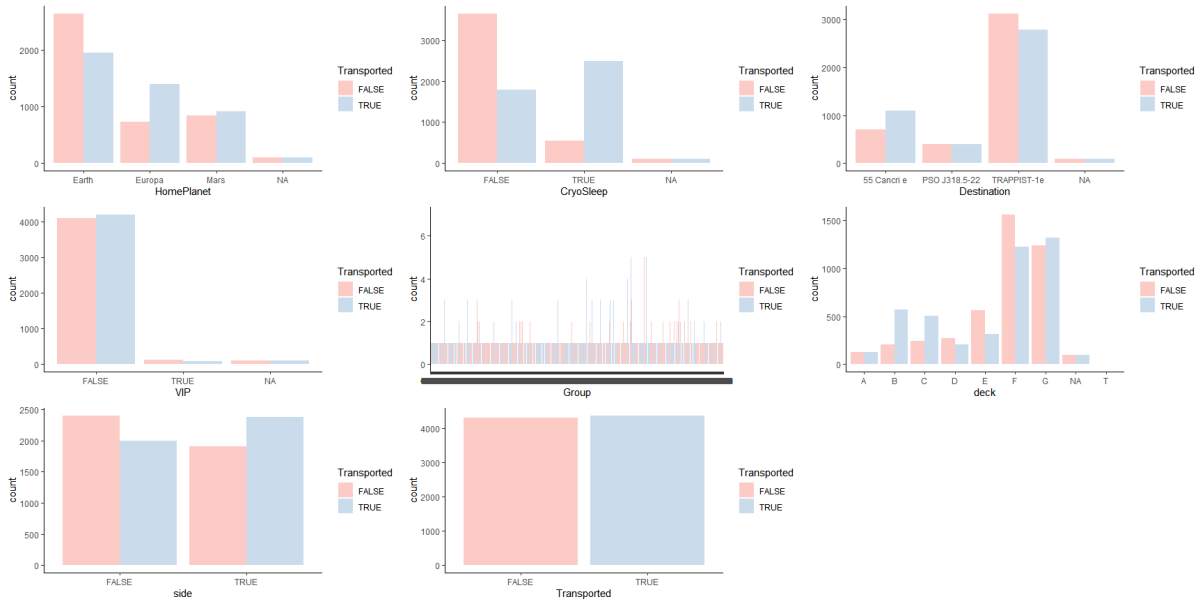
```
  return(num_plot)
  }
```

```
numeric<-c("Age","RoomService","FoodCourt","ShoppingMall","Spa","VRDeck" )

grid.arrange(grobs=lapply(numeric,num_vis),ncol=1)
```

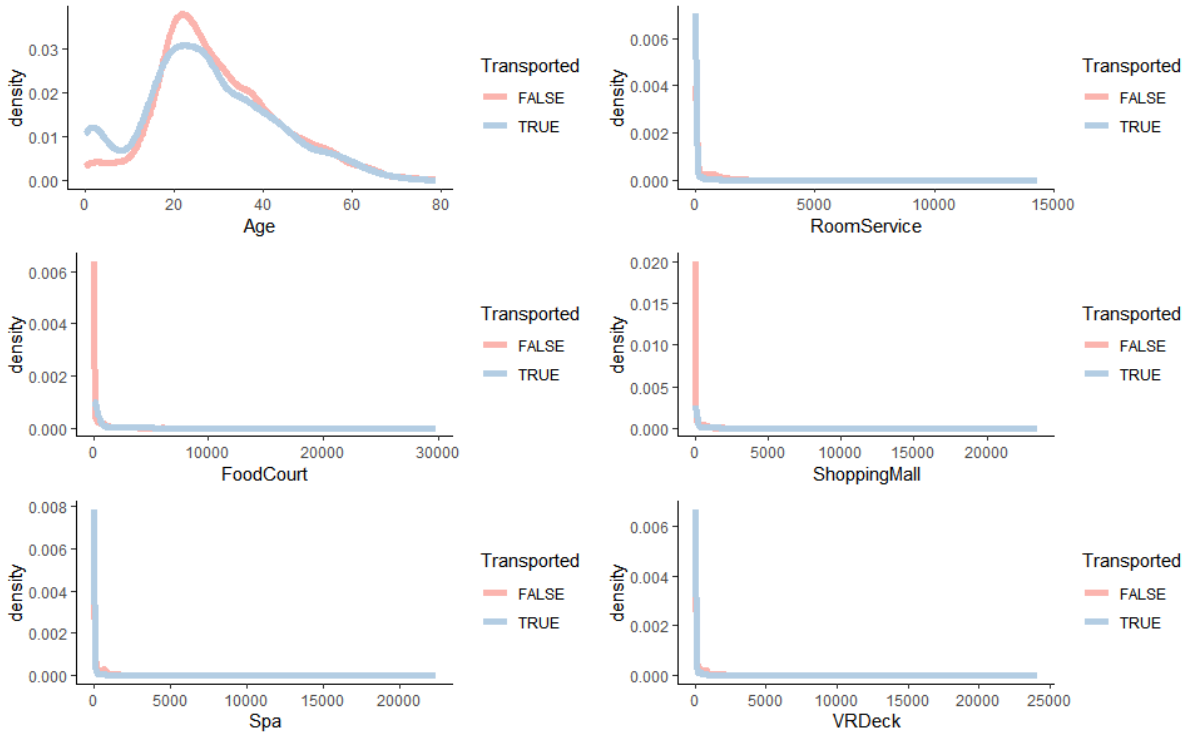



```
bar_plot<-function(var){
  bar<-ggplot(train,aes_string(x={{var}}))+
    geom_bar(aes(fill=Transported)
             ,position='dodge',alpha=0.7)+
    scale_fill_brewer(palette='Pastel1')+
    theme_classic()
```

```
  return(bar)
}

cate<-c('HomePlanet','CryoSleep','Destination',
        'VIP','Group','deck',
        'side','Transported')

plots<-lapply(cate,
              bar_plot)


grid.arrange(grobs=plots,ncol=3)
```

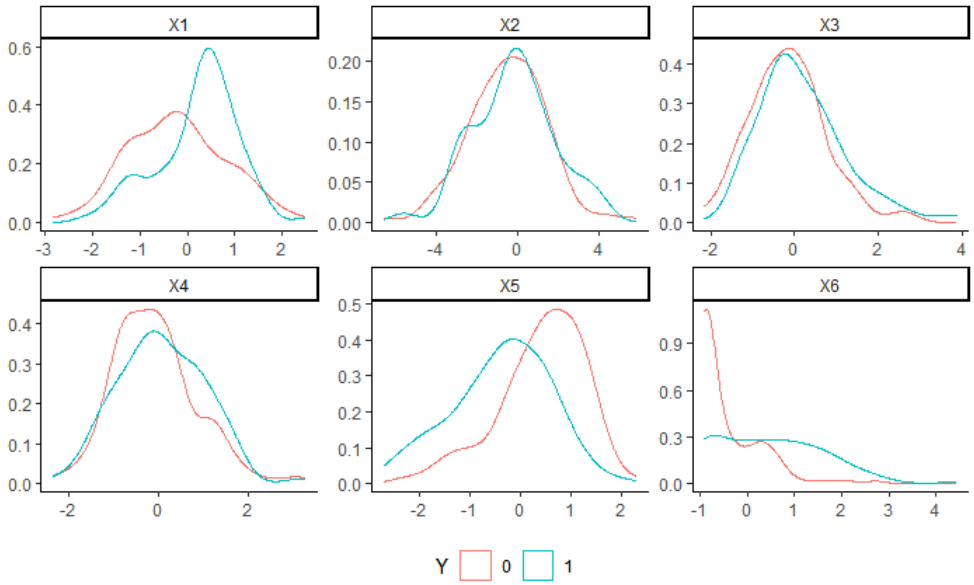



```
plots<-lapply(numeric,
              function(vars){
                ggplot(data=train,aes_string(x={{vars}},color='Transported'))+
                  geom_line(stat='density',size=2)+
                  scale_color_brewer(palette='Pastel1')+
                  theme_classic()})

grid.arrange(grobs=plots,ncol=2)
```



두 방법 비교

```
train %>%
  gather('X1':'X6',key='variable',value='value') %>%
  mutate_at('Y',as.factor) %>%
  ggplot()+
  geom_density(mapping=aes(x=value,color=Y,group=Y))+
  facet_wrap(vars(variable),ncol=3,scales="free")+
  theme_classic()+
  theme(legend.position = "bottom")+
  labs(x=NULL,y=NULL)
```



```
lapply(c("X1","X2","X3","X4","X5","X6" ),
       function(var){
         ggplot(train)+
           geom_line(stat='density',aes_string(x=var,color='Y'),
                     size=2)+
           theme_classic()})->plots

grid.arrange(grobs=plots,ncol=3)
```