

# 데이터 시각화와 전처리

박찬엽

2017년 7월 26일

## 목차

### 1. 과제 확인

### 2. ggplot2

- 그림을 그리는 구조
- gapminder
- layer 다루기
- 저장하기
- 실습

### 3. 한국 지도 그리기

- 좌표계와 지도 데이터
- ggmap으로 배경 받기
- kormap으로 shp 데이터 경험하기

### 4. ggplot2+extra

### 5. 과제

# 과제 확인

과제 확인



# ggplot2란

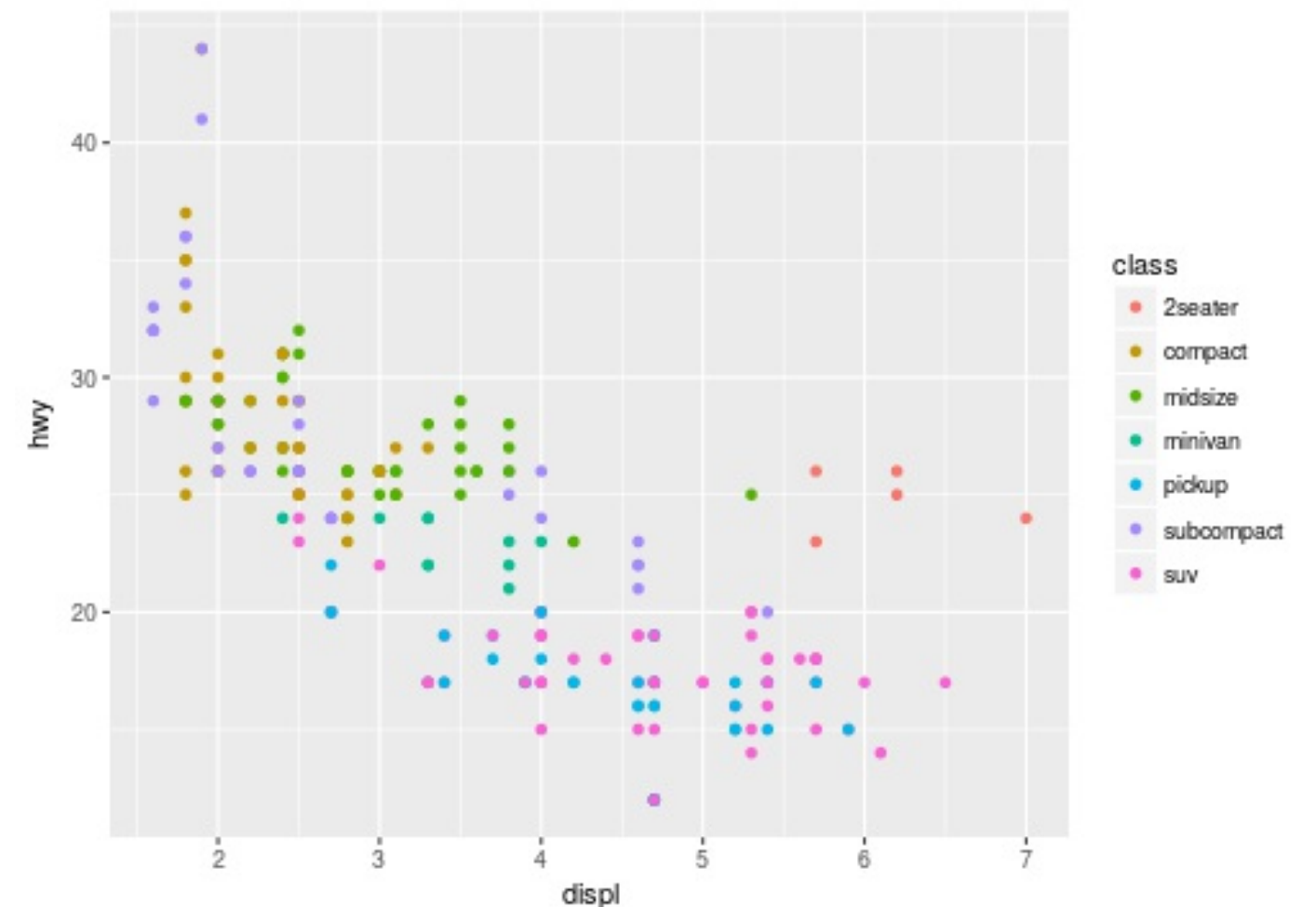
**The Grammar of Graphics**를 따르는 방식으로 차트를 출력물 수준으로 만들 수 있게 보조해주는 패키지. 그래서 **GGplot** 이라고 이름을 지은 것이고, 상세한 동작방식은 **여기**를 참고하세요. 데이터를 기하 객체(geometric object)의 미적 속성(aesthetic attributes)에 연결하는 방법을 사용합니다.

```
library(ggplot2)
ggplot(data=mpg) +
  geom_point(aes(displ, hwy,
                  colour = class))
```

# ggplot2란

**The Grammar of Graphics**를 따르는 방식으로 차트를 출력할 수준으로 만들 수 있게 보조해주는 패키지. 그래서 **GGplot** 이라고 이름을 지은 것이고, 상세한 동작방식은 [여기](#)를 참고하세요. 데이터를 기하 객체(geometric object)의 미적 속성(aesthetic attributes)에 연결하는 방법을 사용합니다.

```
library(ggplot2)
ggplot(data=mpg) +
  geom_point(aes(displ, hwy,
                 colour = class))
```



# 공부 자료

- 패키지 공식문서
- cheat sheet
- R for data science
- 한글 자료 1
- 한글 자료 2

# 용어 설명

- 기하 객체(geometric object) : 차트를 구성할 수 있는 그림의 형태들. bar, dot, line 등이 있음. geom\_\*() 형태의 함수로 layer를 구성함.
- 미적 속성(aesthetic attributes) : 각 기하 객체들의 모양을 결정하기 위한 요소들. 공통적으로 x, y, color 등이 있음.
- 연결(mapping) : 데이터의 컬럼을 필요한 미적 속성에 해당함을 명시적으로 작성하는 것.
- ggplot 자료형 : ggplot()함수로 생성하는 R 객체로 그림을 그리기 위한 정보를 포함하고 있음.
- 계층(layer) : 제공된 데이터와 연결 정보를 바탕으로 그려진 그림. + 연산자를 통해 계층을 추가하여 겹쳐서 그리는 것이 가능.
- + 연산자 : 계층을 추가할 때 데이터와 연결 정보, 지금까지 작성된 계층 정보를 전달하는 연산자. 파이프 연산자(%>%)와 비슷함.
- 좌표계(coordinate system) : 대표적인 x-y 좌표계를 사용하며 극좌표계(polar) 등으로 파이차트를 작성할 수 있음.
- 축척(scale) : 데이터를 표시하는 방식으로 연속형 자료형에서 고려함.



# ggplot 객체

`ggplot`은 그림 그리는 정보를 가지는 `layer`들을 겹쳐서 차트를 그리는 방식을 이용합니다. 대부분은 자동으로 지정을 해주고, 필수로 작성해야 하는 `layer`의 요소는 데이터, 연결정보, 기하 객체입니다.

```
layer = ggplot(data, aes(<mapping>)) + geom_*
```

# ggplot 객체

`ggplot`은 그림 그리는 정보를 가지는 `layer`들을 겹쳐서 차트를 그리는 방식을 이용합니다. 대부분은 자동으로 지정을 해주고, 필수로 작성해야 하는 `layer`의 요소는 데이터, 연결정보, 기하 객체입니다.

```
layer = ggplot(data, aes(<mapping>)) + geom_*
```

여러 `layer`를 겹칠 때 데이터와 연결정보 중에 공통으로 사용하는 것은 `ggplot()`함수로 작성하여 `geom_요소만 + 연산자로 연결하여 layer를 겹치는 방식을` 사용합니다.

```
ggplot(data, aes(<mapping>)) + # 공통 정보
  geom_요소1 +                 # 레이어 1의 구성
  geom_요소2                   # 레이어 2의 구성
```

# gapminder 데이터 소개

gapminder는 움직이는 버블 차트로 유명한 데이터셋입니다. 코드북과 `str`을 확인하면서 데이터를 확인하겠습니다.

```
if (!require("gapminder")){install.packages("gapminder")}
```

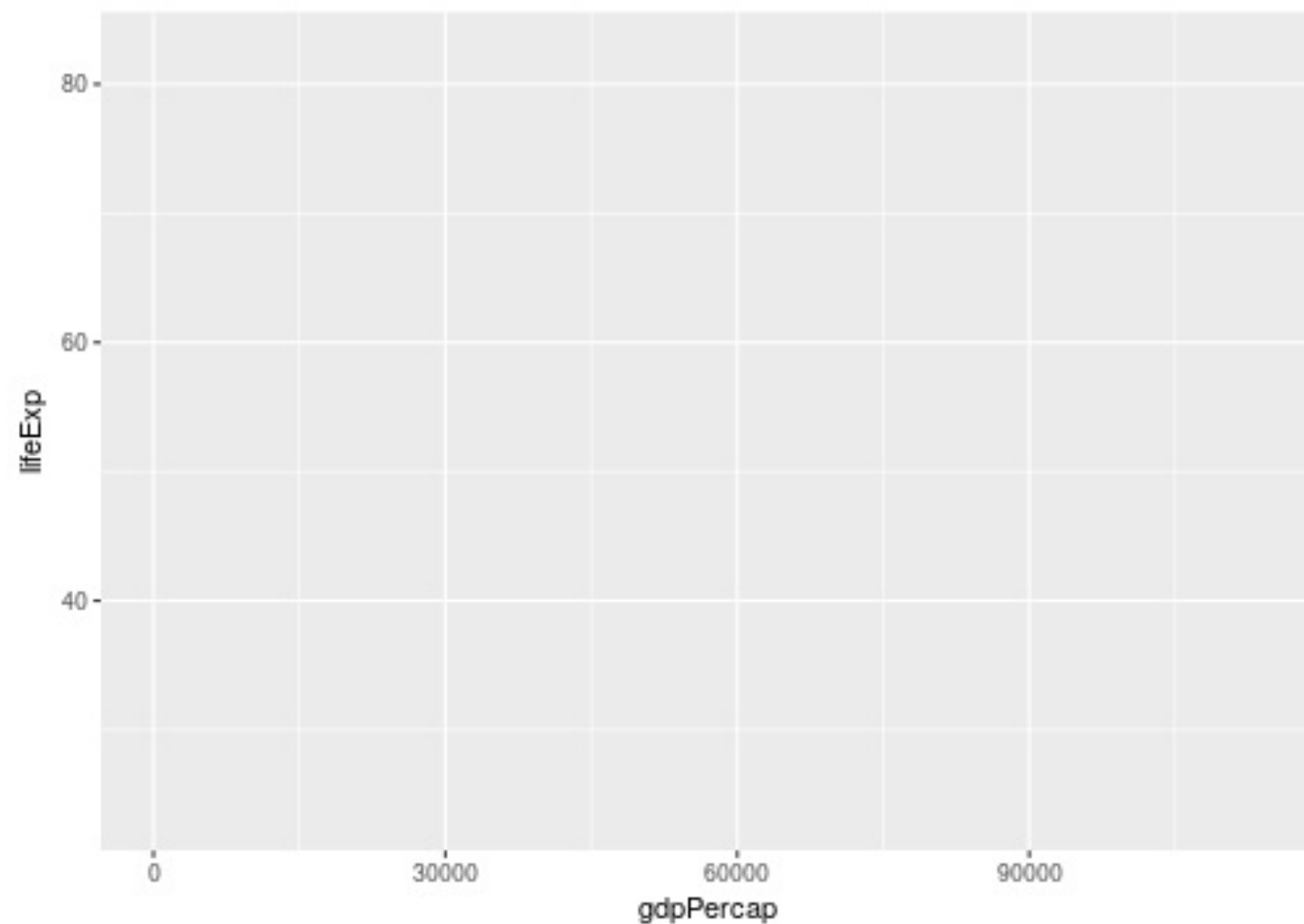
```
## Loading required package: gapminder
```

```
library(gapminder)
str(gapminder)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1704 obs. of  6 variables:
## $ country   : Factor w/ 142 levels "Afghanistan",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ continent: Factor w/ 5 levels "Africa","Americas",...: 3 3 3 3 3 3 3 3 3 3 ...
## $ year      : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
## $ lifeExp   : num  28.8 30.3 32 34 36.1 ...
## $ pop       : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 22
## $ gdpPercap: num  779 821 853 836 740 ...
```

# ggplot 객체 만들기

```
p <-ggplot(gapminder,  
           aes(x = gdpPercap, y = lifeExp))  
p
```

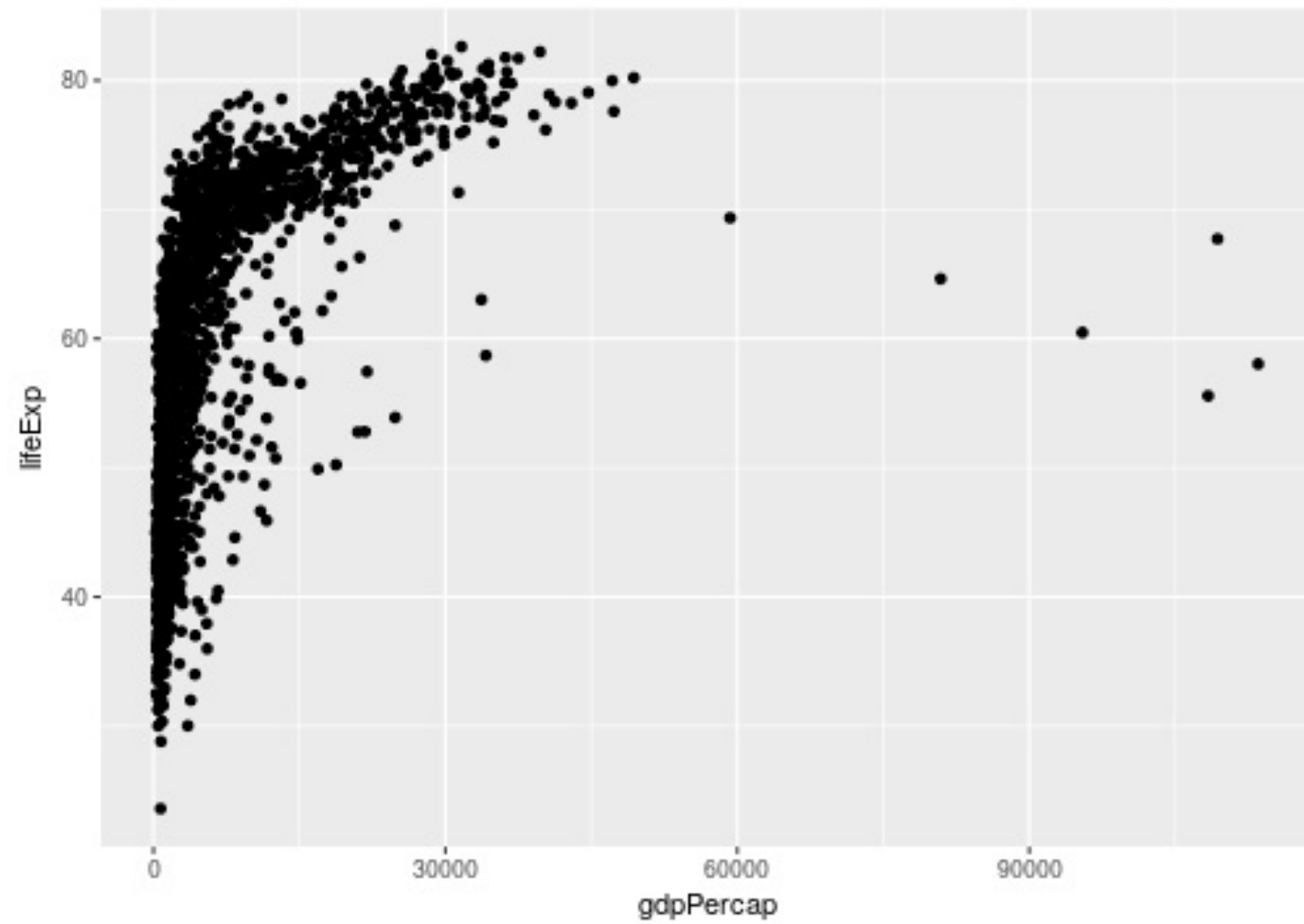


```
summary(p)
```

```
## data: country, continent, year, lifeExp, pop, gdp
## mapping:  x = gdpPercap, y = lifeExp
## faceting: <ggproto object: Class FacetNull, Facet
##   compute_layout: function
##   draw_back: function
##   draw_front: function
##   draw_labels: function
##   draw_panels: function
##   finish_data: function
##   init_scales: function
##   map: function
##   map_data: function
##   params: list
##   render_back: function
##   render_front: function
##   render_panels: function
##   setup_data: function
##   setup_params: function
##   shrink: TRUE
##   train: function
##   train_positions: function
##   train_scales: function
##   vars: function
##   super:  <ggproto object: Class FacetNull, Fac
```

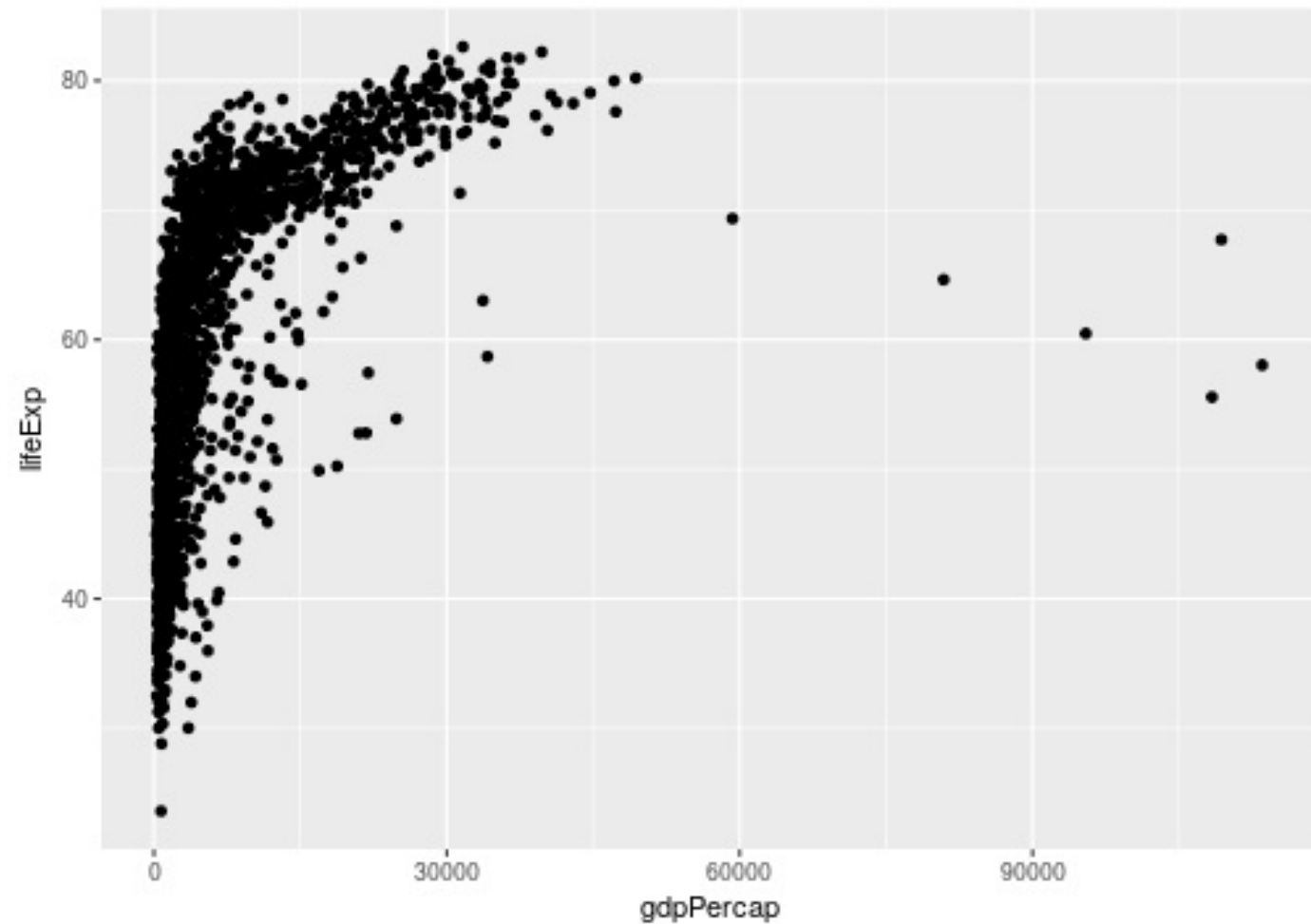
# geom 추가 하기 point

```
p_point <- p + geom_point()  
p_point
```



# geom 추가 하기 point

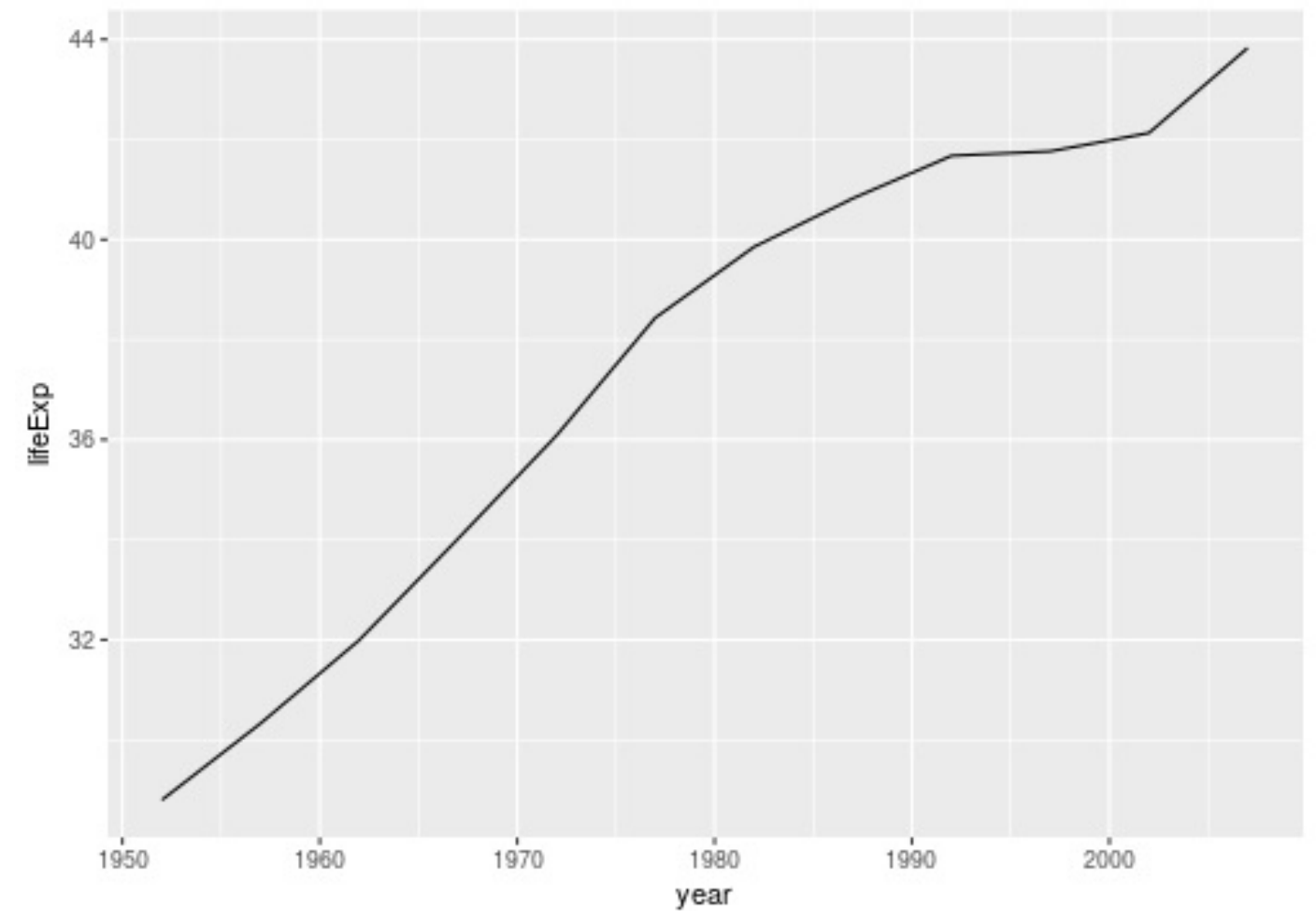
```
p_point <- p + geom_point()  
p_point
```



`summary(p_point)`를 이용해 `summary(p)`와 어떻게 달라졌는지 확인해 보세요.

# 실습 1

1. `gapminder`에서 `country`가 `Afghanistan`인 데이터만 뽑아서 `gap_af`로 저장하세요.
2. `gap_af`를 x축은 `year`, y축은 `lifeExp`로 line 차트를 그리세요.



# ggplot 내장 함수 사용시 장점

ggplot은 매우 많은 함수를 제공하고 있으며 대부분 자동으로 지원하지만 명시적으로 필요한 부분을 수정할 수 있습니다. + 연산자로 수정하고자 하는 부분을 추가하는 방식이며 추가된 함수가 자동으로 작성된 것보다 우선 하는 방식으로 적용됩니다.

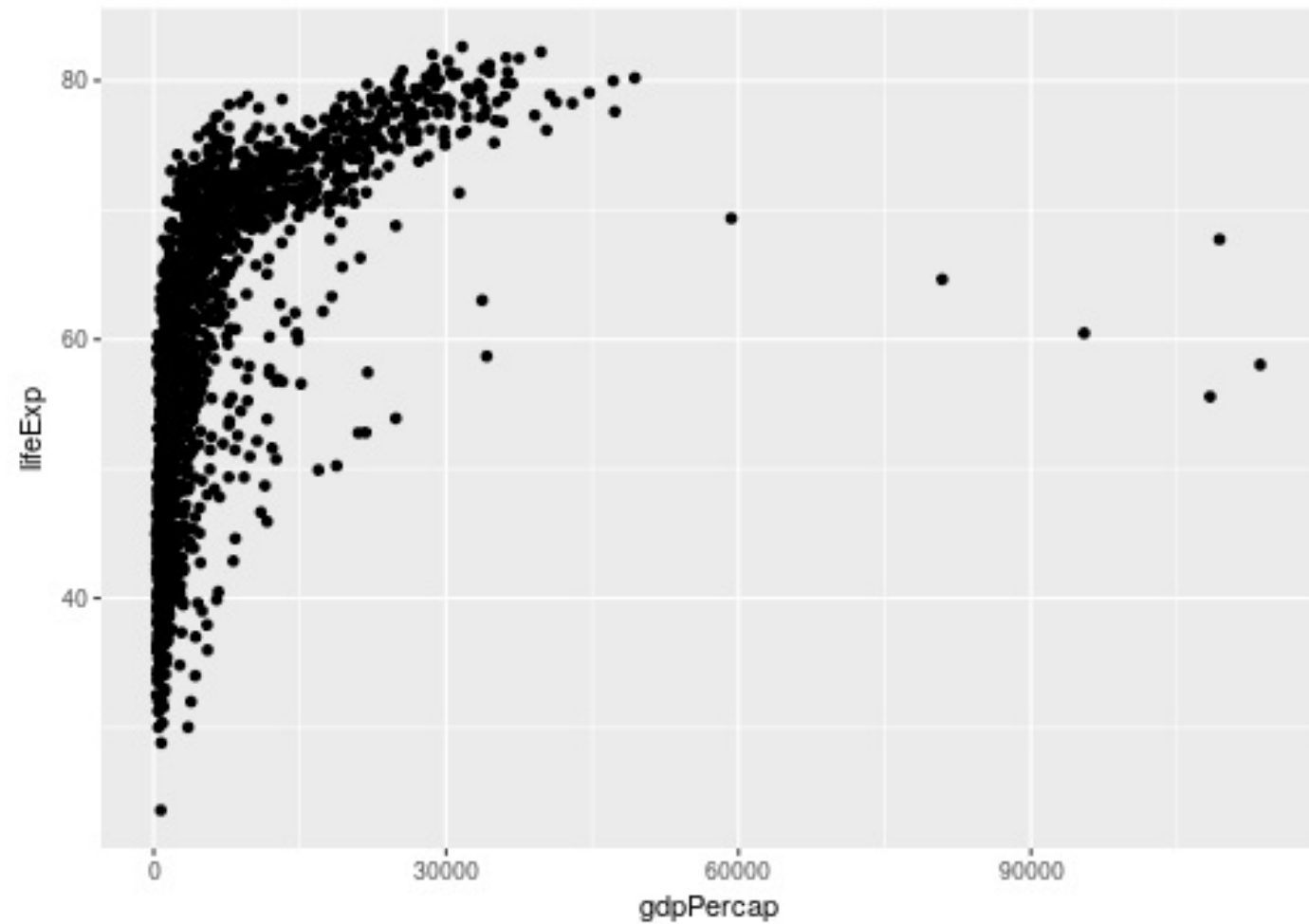
이 때 몇 가지 장점이 있는데,

1. 데이터 + 연결 정보를 가진 ggplot 객체를 수정 없이 재사용할 수 있음
2. 코드를 재사용함으로써 작성량을 줄일 수 있음
3. 파이프 연산자와 같이 가독성이 높아짐

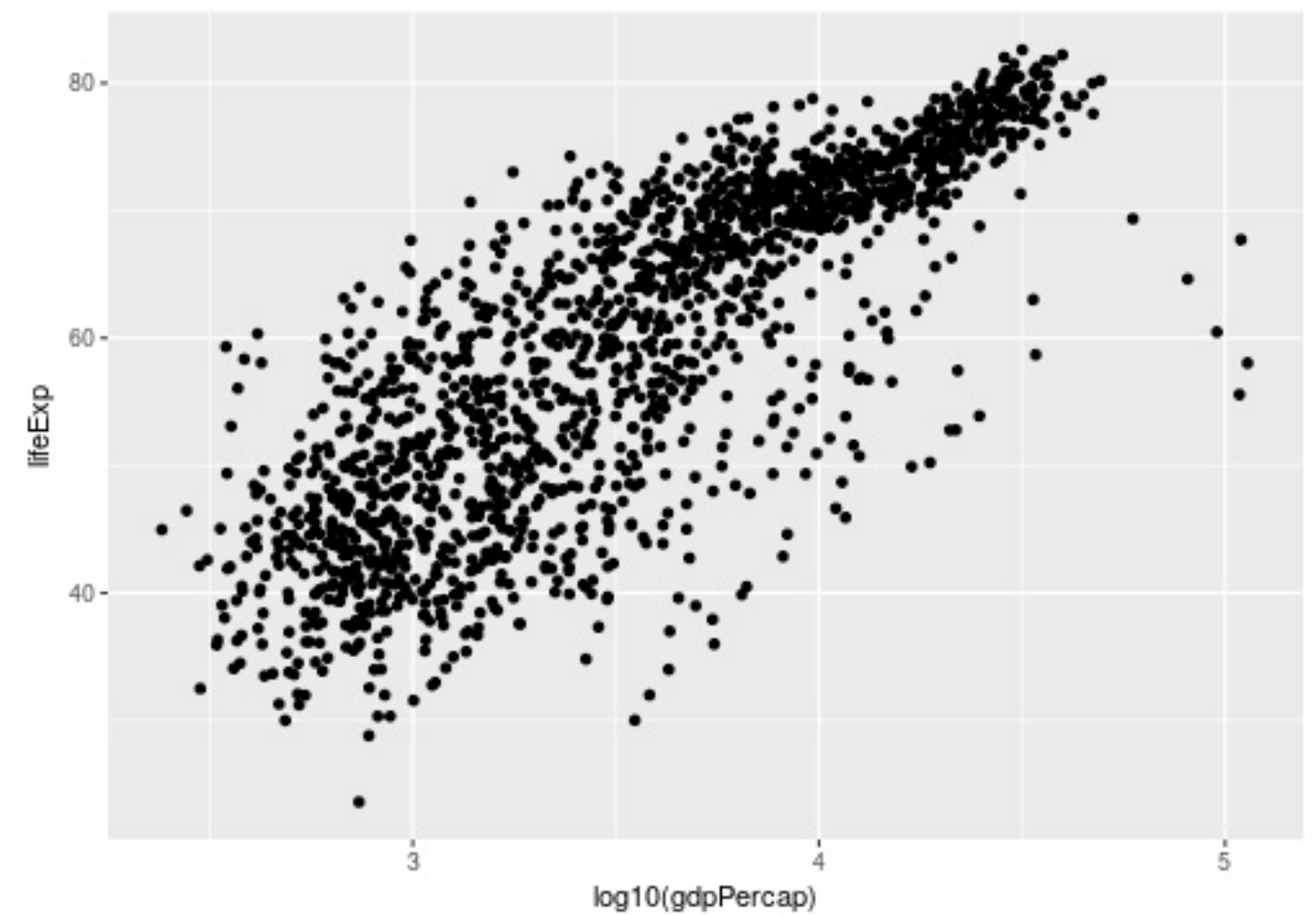


# scale 변경하기

p\_point

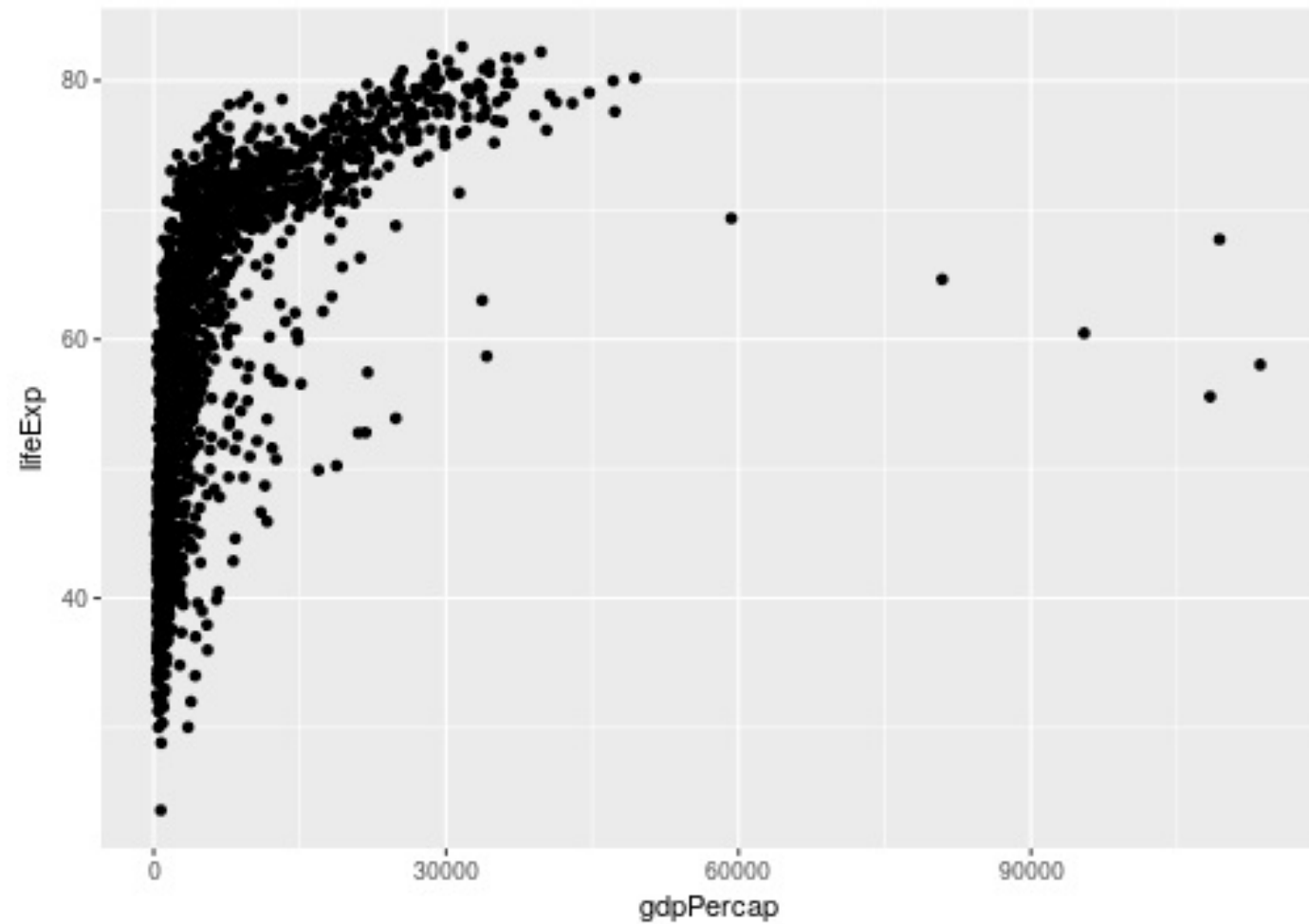


```
ggplot(gapminder) +  
  geom_point(aes(x = log10(gdpPercap),  
                 y = lifeExp))
```

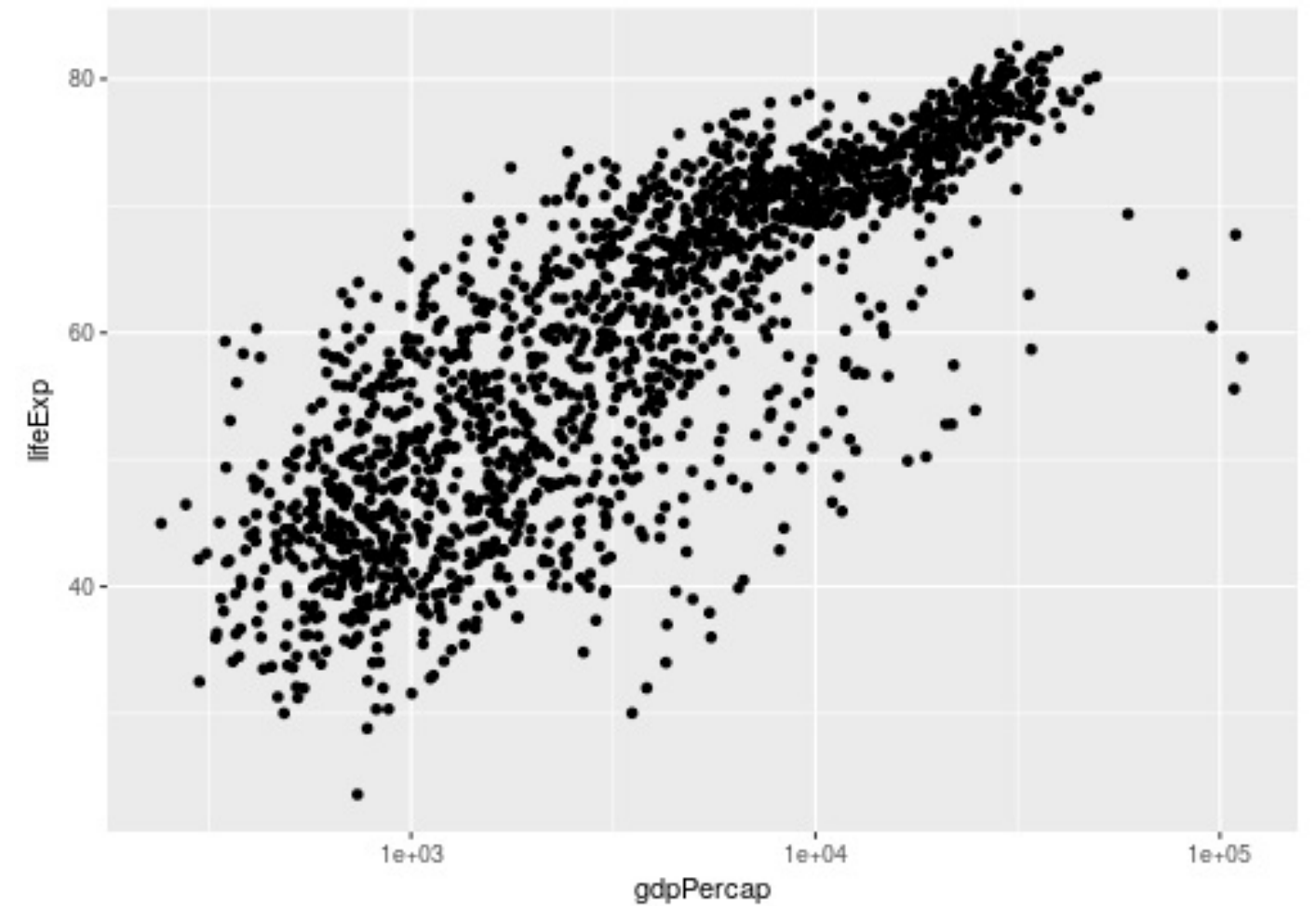


# scale\_\* 함수로 변경하기

```
p_point
```



```
(p_point_log10 <- p_point + scale_x_log10())
```



# aes의 공통 정보

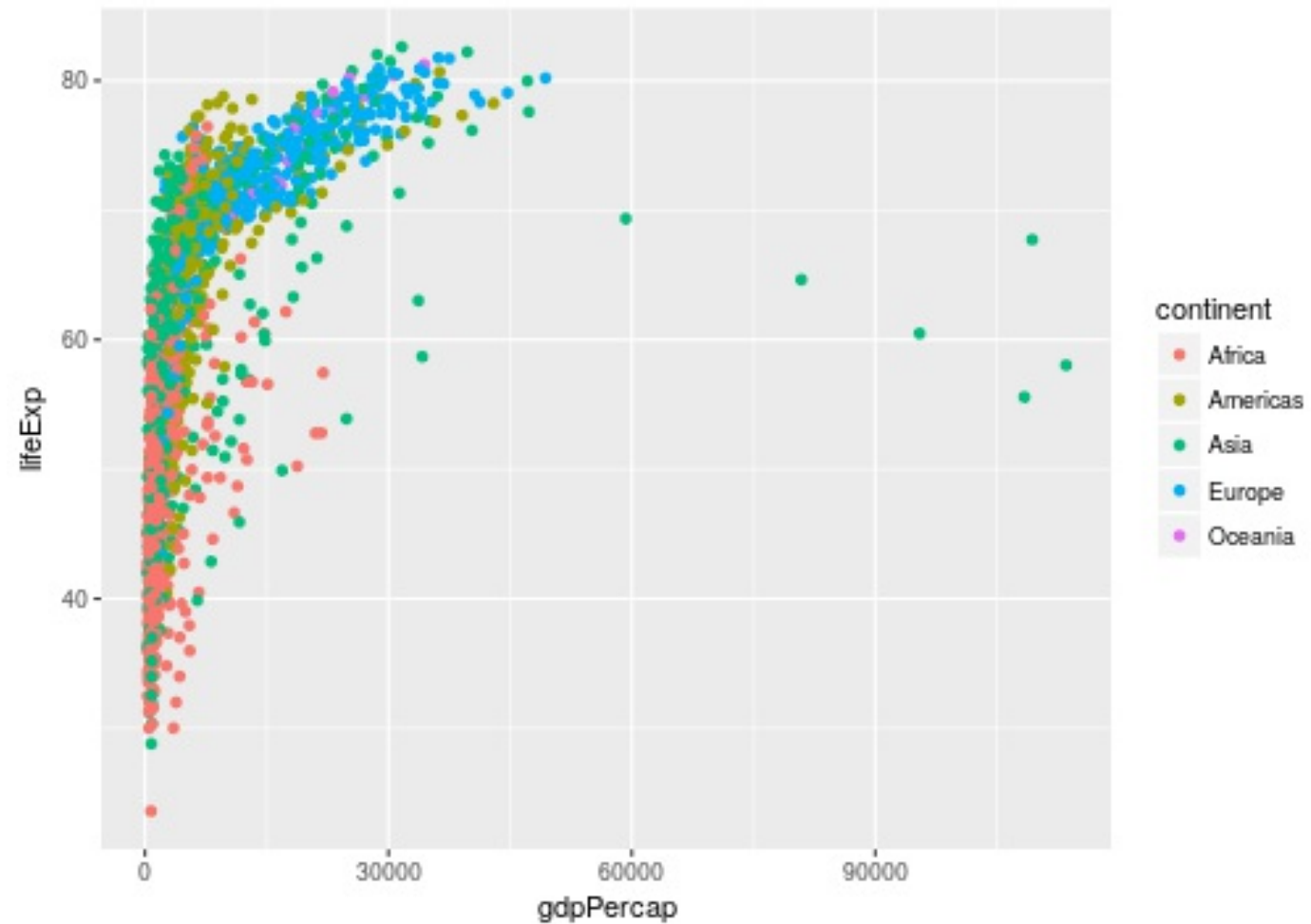
ggplot [cheatsheet](#)에 보면 각 geom에 따른 aes를 설명하고 있습니다. 그 중 많이 보이는 몇 가지를 소개합니다.

- x: x-y 좌표계에서 x축으로 표시할 데이터를 뜻합니다.
- y: x-y 좌표계에서 y축으로 표시할 데이터를 뜻합니다.
- color: 색 정보로써 각 기하 객체가 가질 색을 구분하는데 사용합니다. 보통 명목형 데이터가 사용됩니다.
- group: color의 하위 호환 aes로 데이터는 묶어서 표시하지만 색으로 추가 구분하지는 않습니다.
- alpha: 투명도를 뜻하며 기하 객체의 투명도를 연속형 데이터 표시로 추가 사용하기도 합니다.
- size : 크기를 뜻하며 기하 객체의 크기를 연속형 데이터 표시로 추가 사용하기도 합니다.

# color 정보를 aes를 이용해서 mapping 하기

```
(p_point_color <- p +  
  geom_point(aes(color = continent)))
```

summary(p\_point\_color)로 색 정보가 추가 된것을 확인해 보세요.



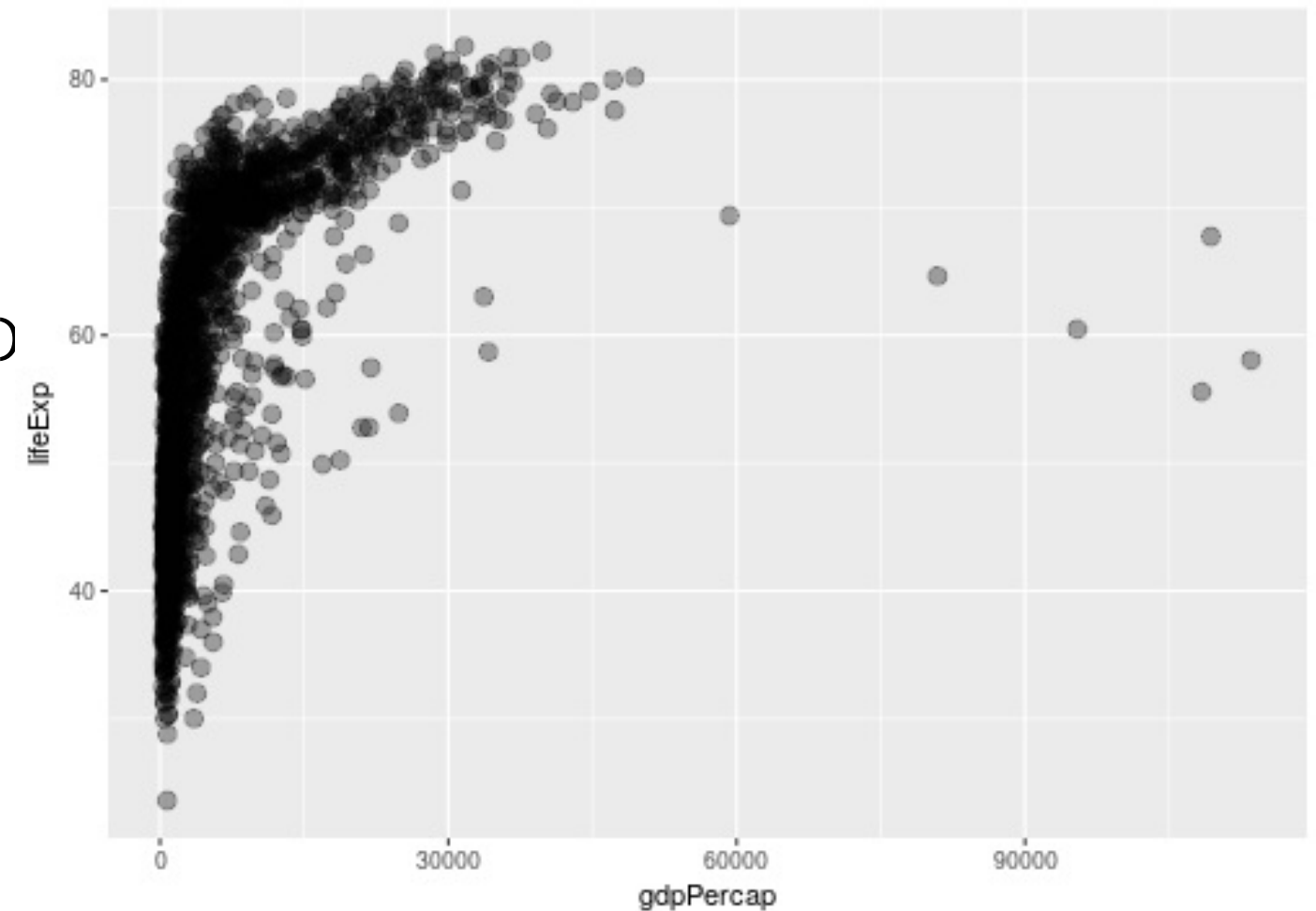
# geom 특징 조절하기

기하 객체의 각 특징은 굳이 데이터에 매핑 될 필요없이 조절할 수도 있습니다.

아래 두 `summary`를 확인해서 어떻게 정보가 다른지 확인해 보세요.

```
summary(p + geom_point(alpha = (1/3), size = 3 ))  
summary(p + geom_point(aes(alpha = (1/3), size = 3))
```

```
p + geom_point(alpha = (1/3), size = 3)
```



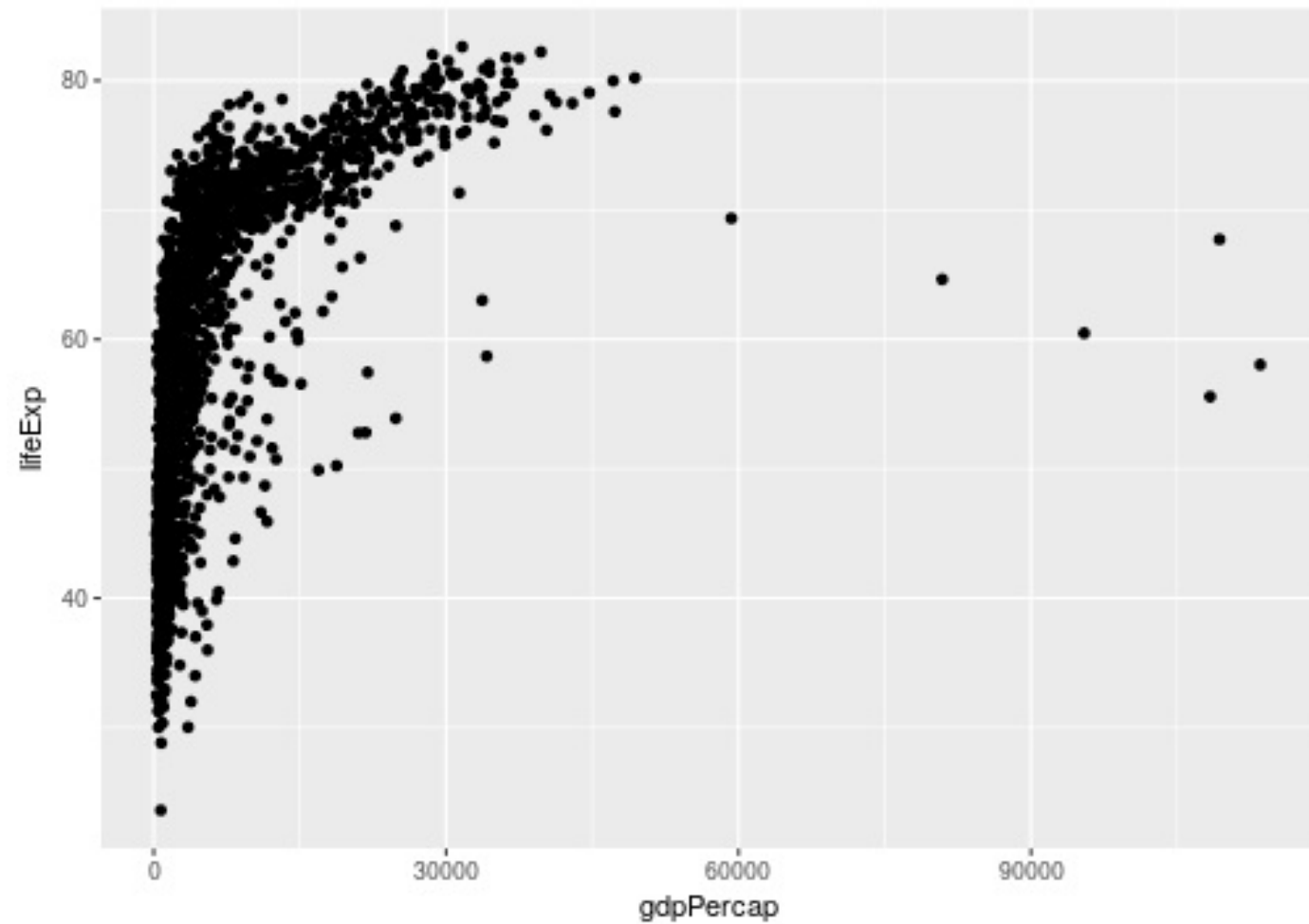
# stat\_\*() 함수로 layer 추가하기

stat\_\*() 함수는 geom\_\*() 함수의 특별한 형태로써 통계적인 기능들이 추가되어 동작합니다. geom\_\*() 함수 내에서도 통계 함수들을 사용하여 구현할 수도 있습니다.

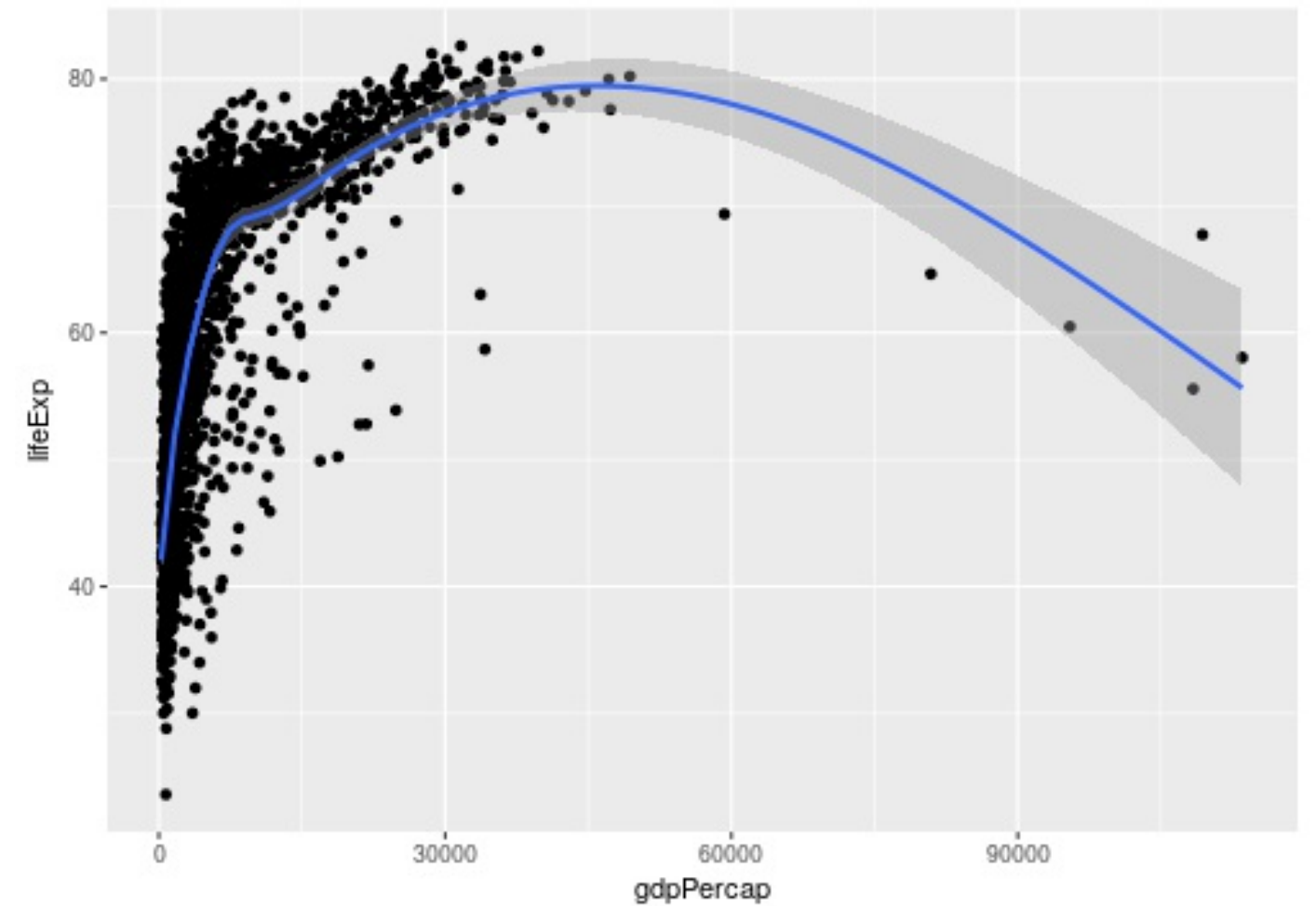
stat\_\*() 함수의 [공식 설명서](#)를 참고하세요.

# stat\_smooth로 간단히 추세선 추가하기

p\_point



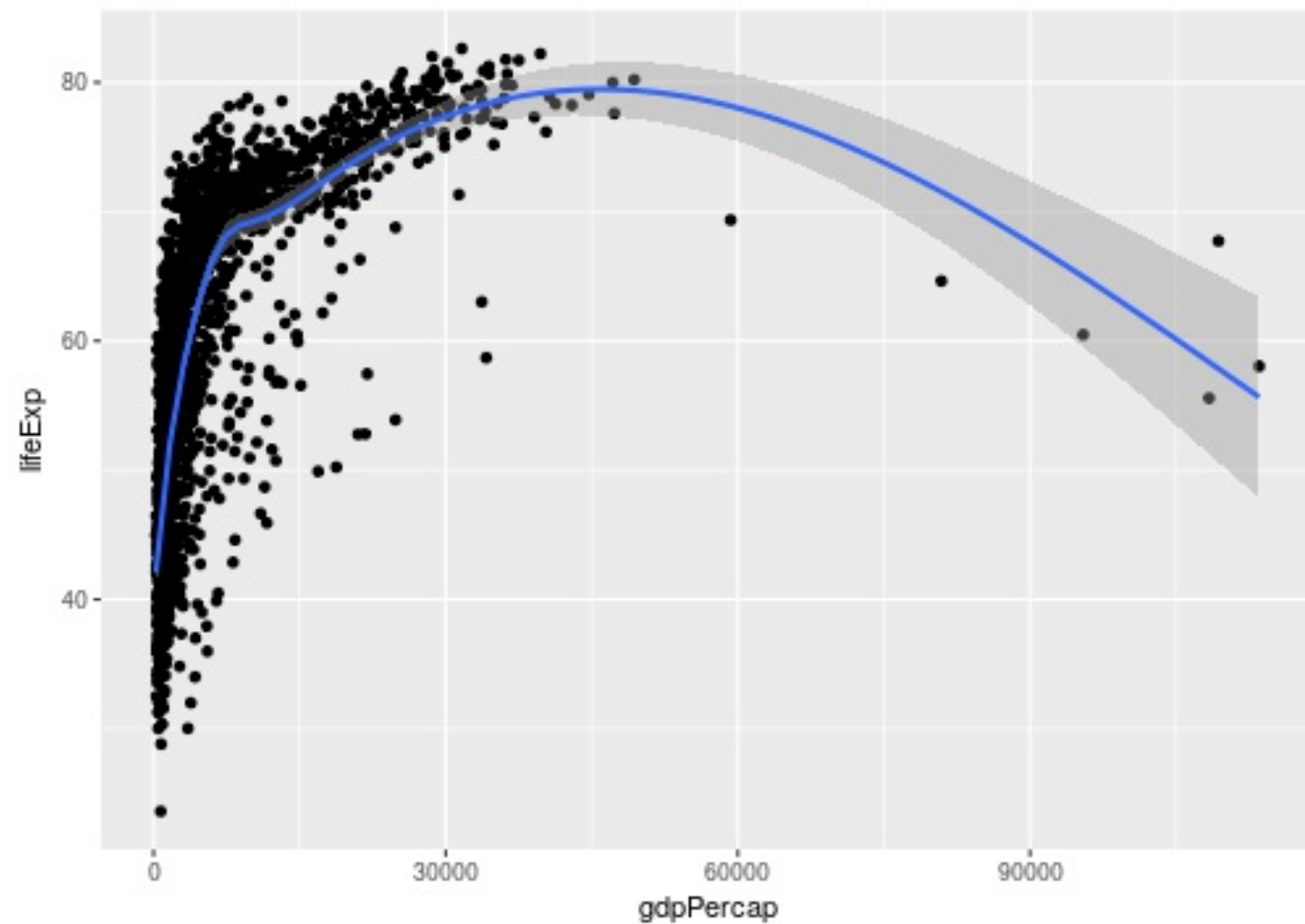
p\_point + stat\_smooth()



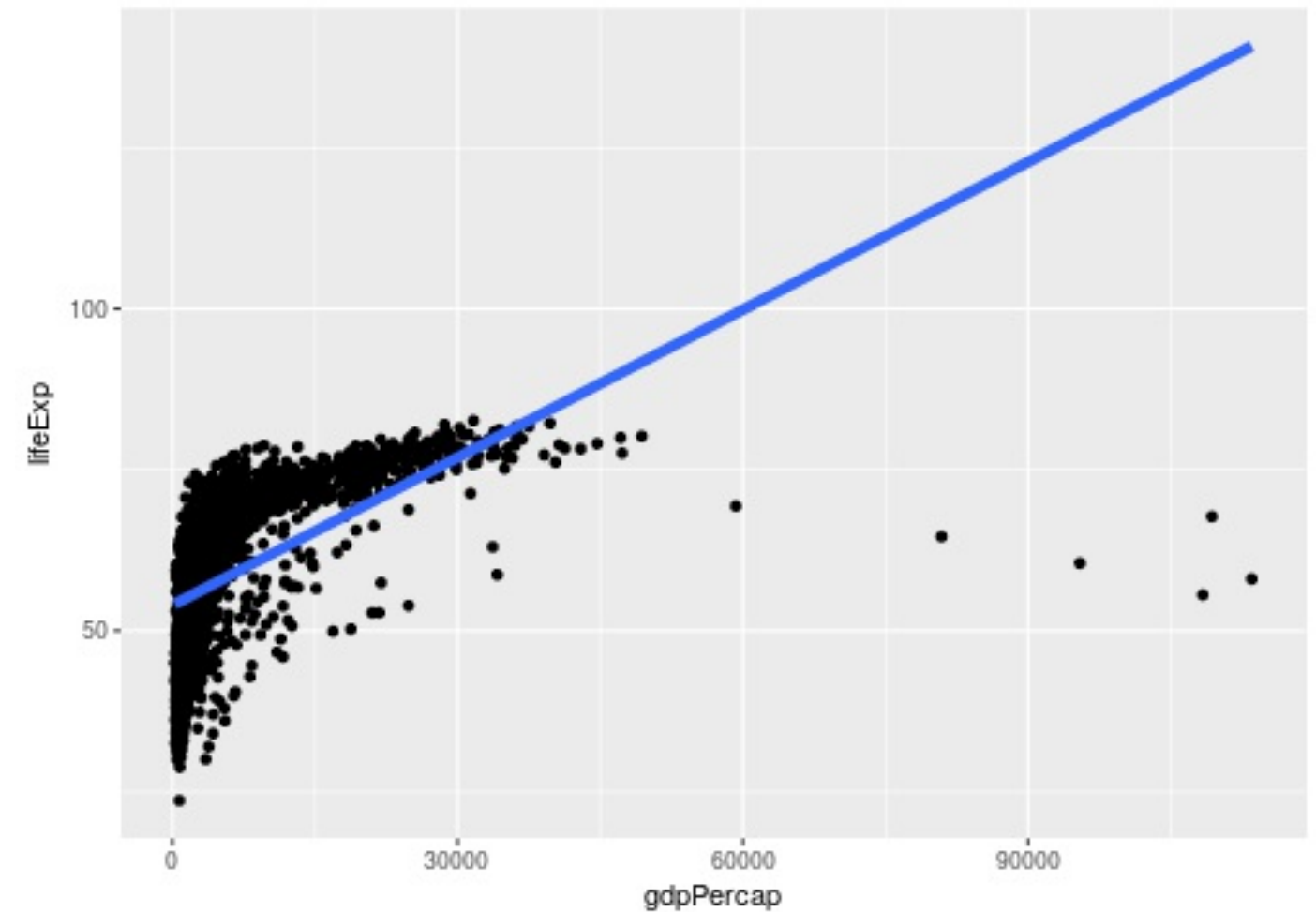
# method를 lm(선형회귀)로 변경

```
p_point + stat_smooth()
```

```
## `geom_smooth()` using method = 'gam'
```



```
p_point + geom_smooth(lwd = 2,  
                      se = FALSE,  
                      method = "lm")
```



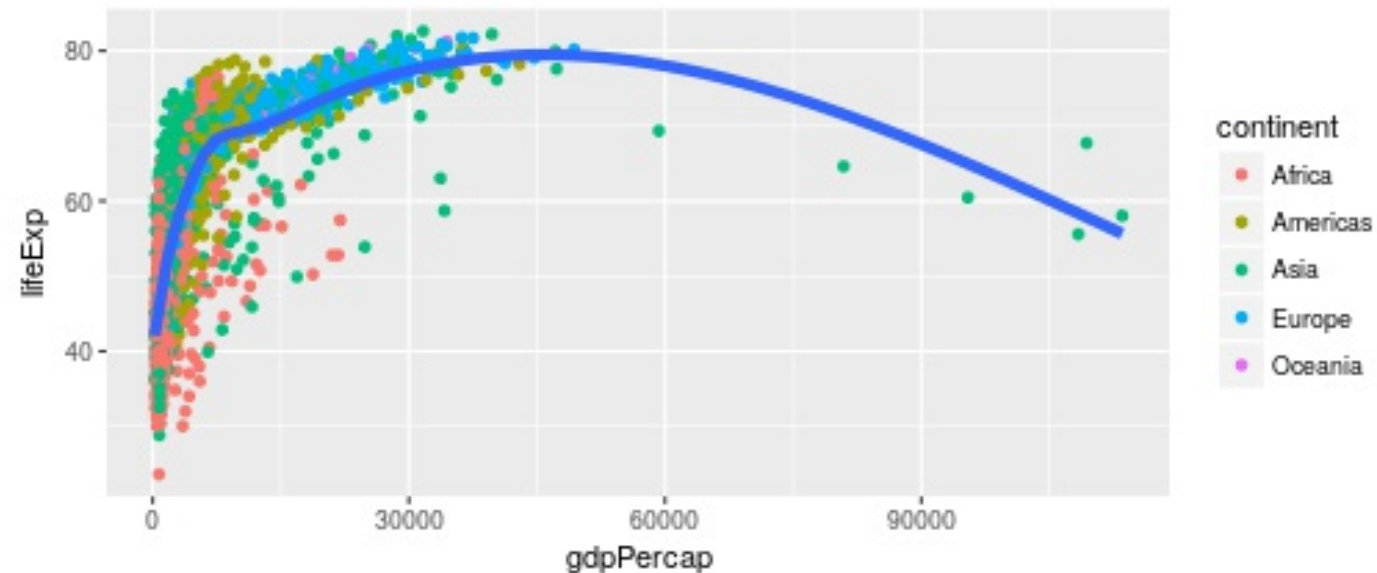


# 색정보를 전체에 반영하기

layer를 추가할 때 작성한 mapping 정보는 이후에 더해지는 layer에는 반영되지 않습니다. 그렇기 때문에 공통으로 사용하는 mapping 정보는 최초 `ggplot` 객체 생성시 `aes()` 함수로 반영해야 합니다.

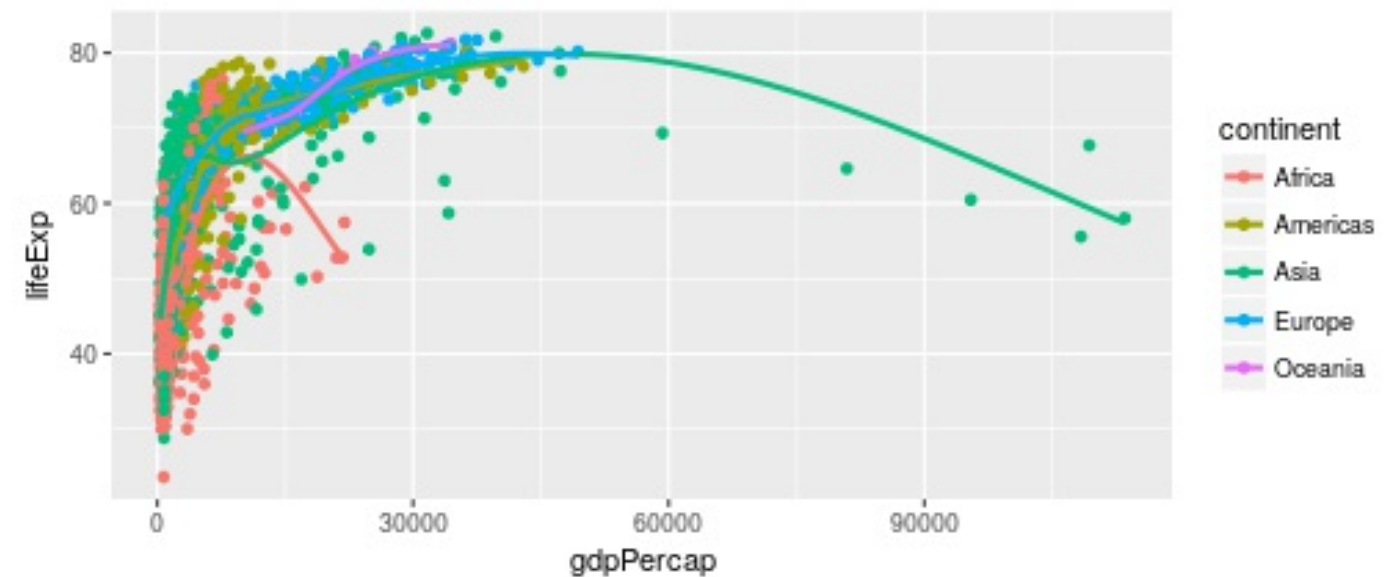
```
p_point_color +  
  geom_smooth(lwd = 2, se = FALSE)
```

```
## `geom_smooth()` using method = 'gam'
```



```
p + aes(color = continent) +  
  geom_point() +  
  geom_smooth(lwd = 1, se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```

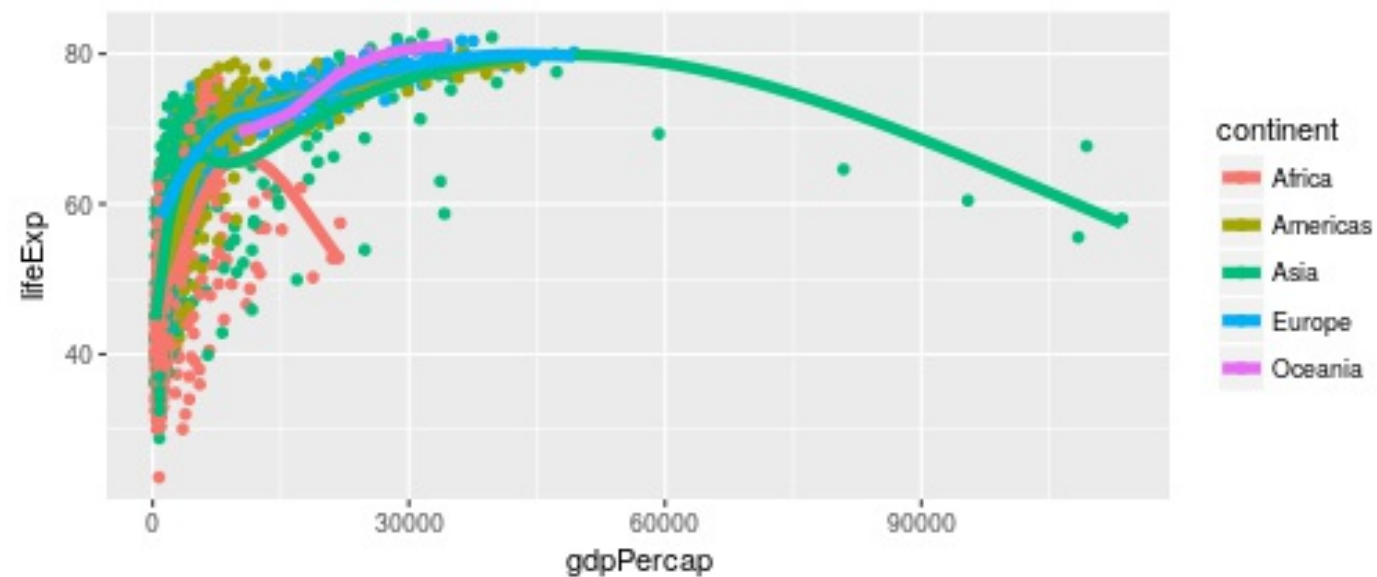


# group 과 color

group과 color가 어떻게 다르게 동작하는지 확인하세요.

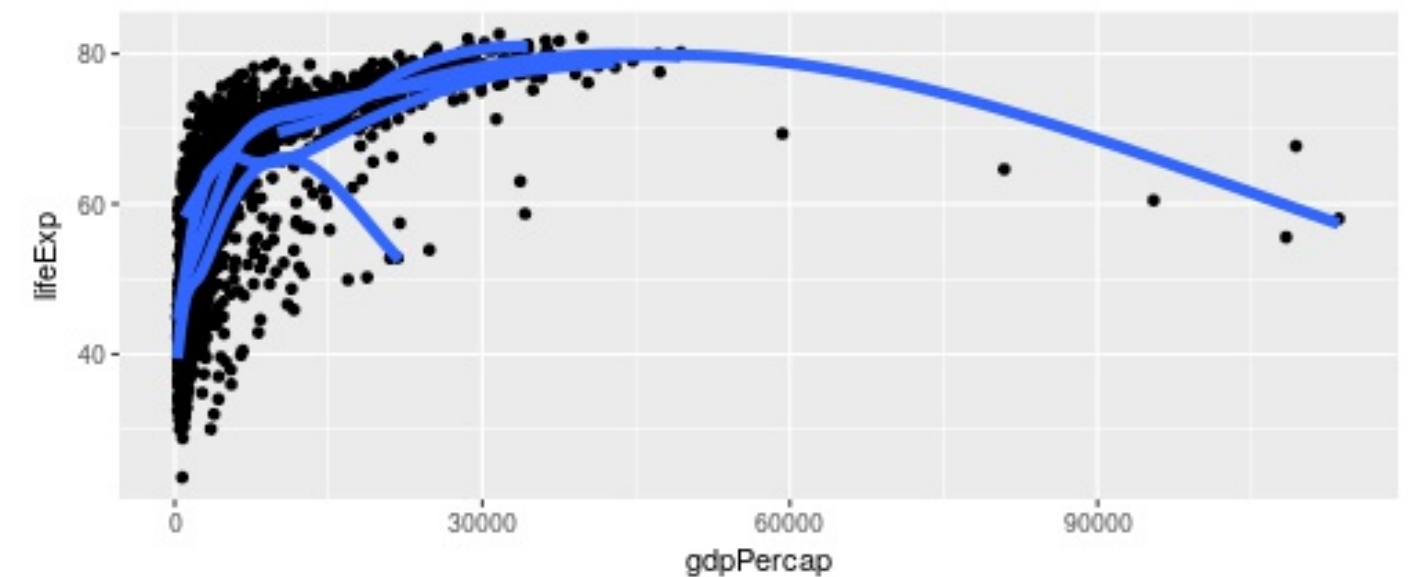
```
p + aes(color = continent) +  
  geom_point() +  
  geom_smooth(lwd = 2, se = FALSE)
```

## `geom\_smooth()` using method = 'loess'



```
p + aes(group = continent) +  
  geom_point() +  
  geom_smooth(lwd = 2, se = FALSE)
```

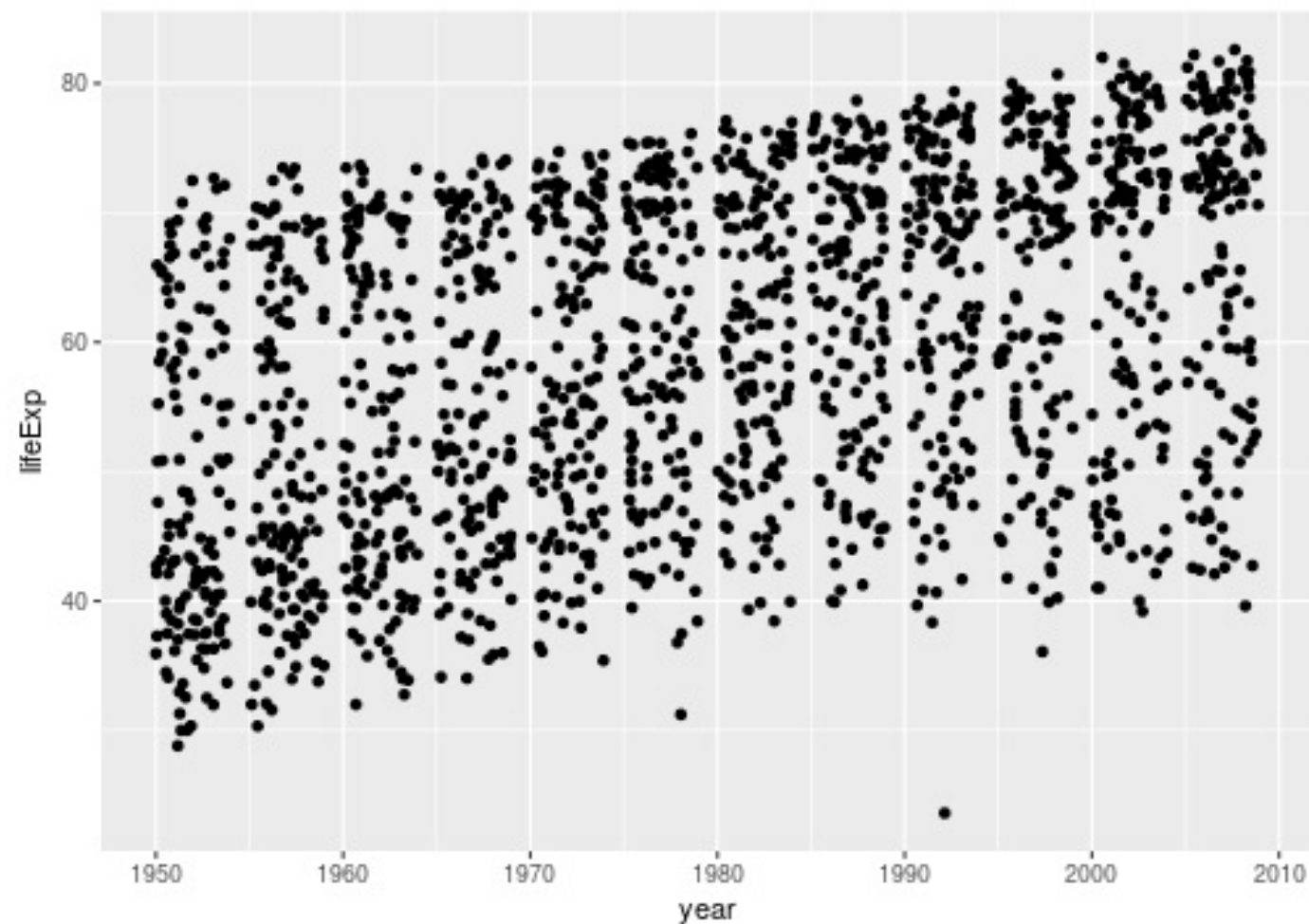
## `geom\_smooth()` using method = 'loess'



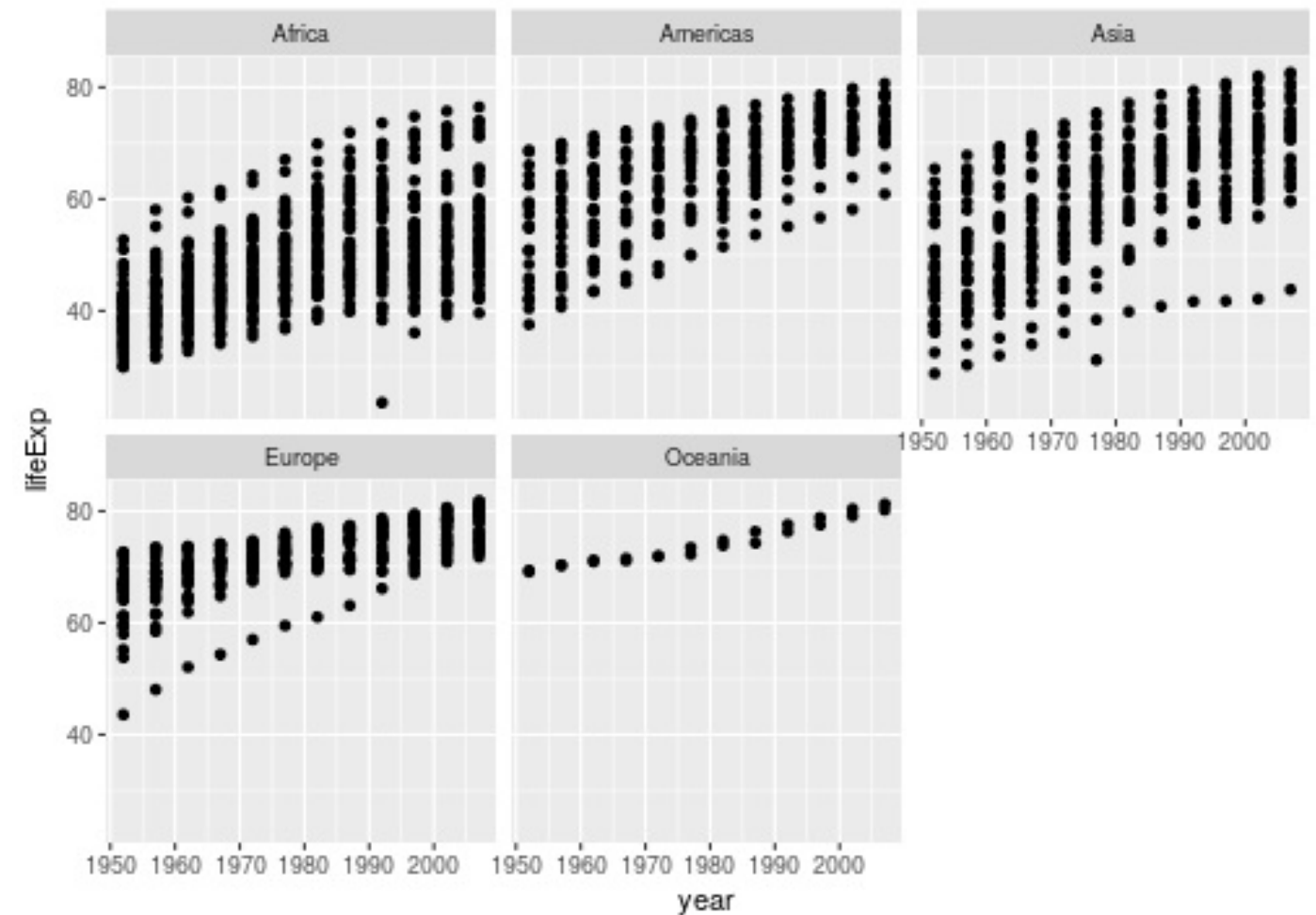
# group으로 차트 나누어 그리기

대륙을 기준으로 차트를 그룹지어 나누어 그릴 수 있습니다. `facet_*()` 함수를 참고하세요.

```
(lp <- ggplot(gapminder) +  
  geom_jitter(aes(x = year, y = lifeExp)))
```

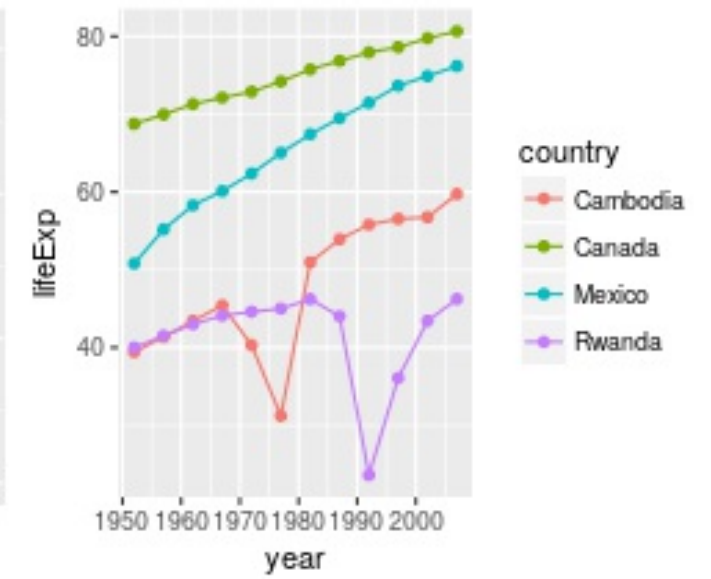
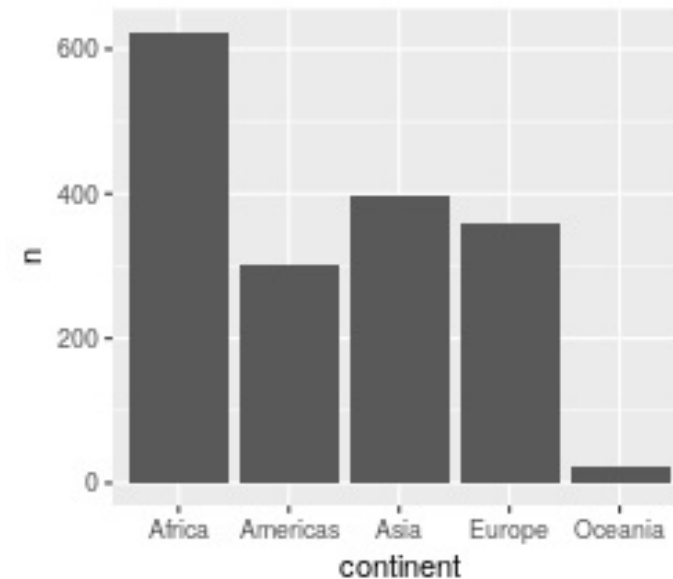
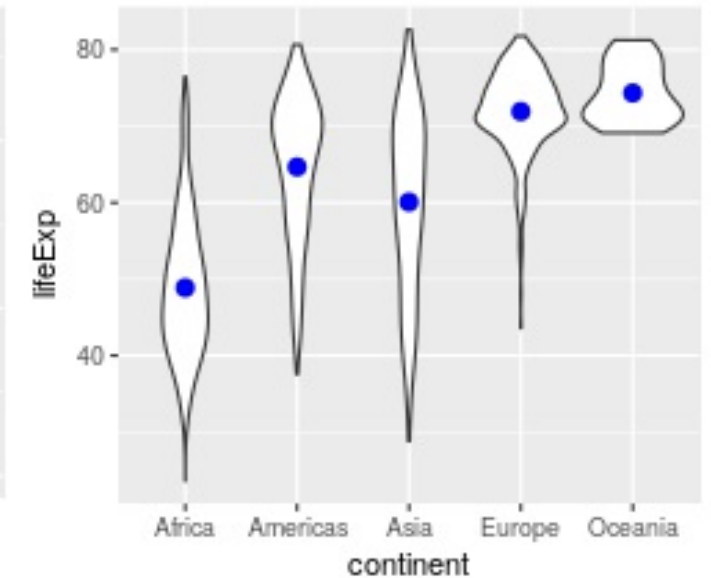
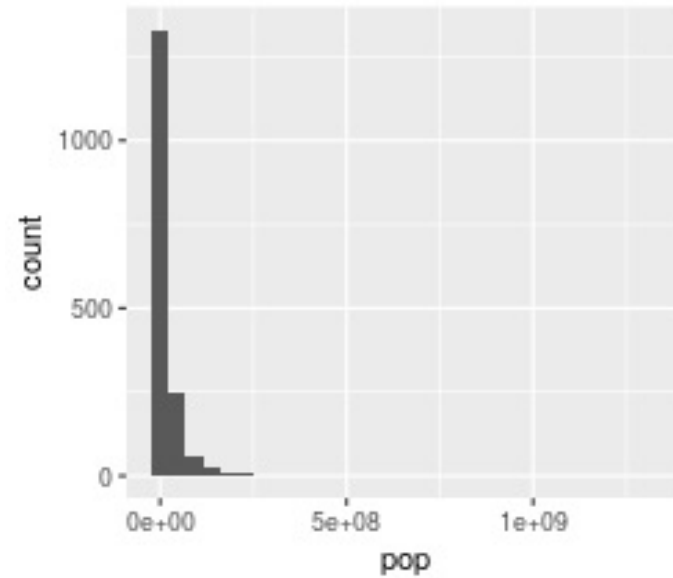


```
lp + facet_wrap(~ continent)
```



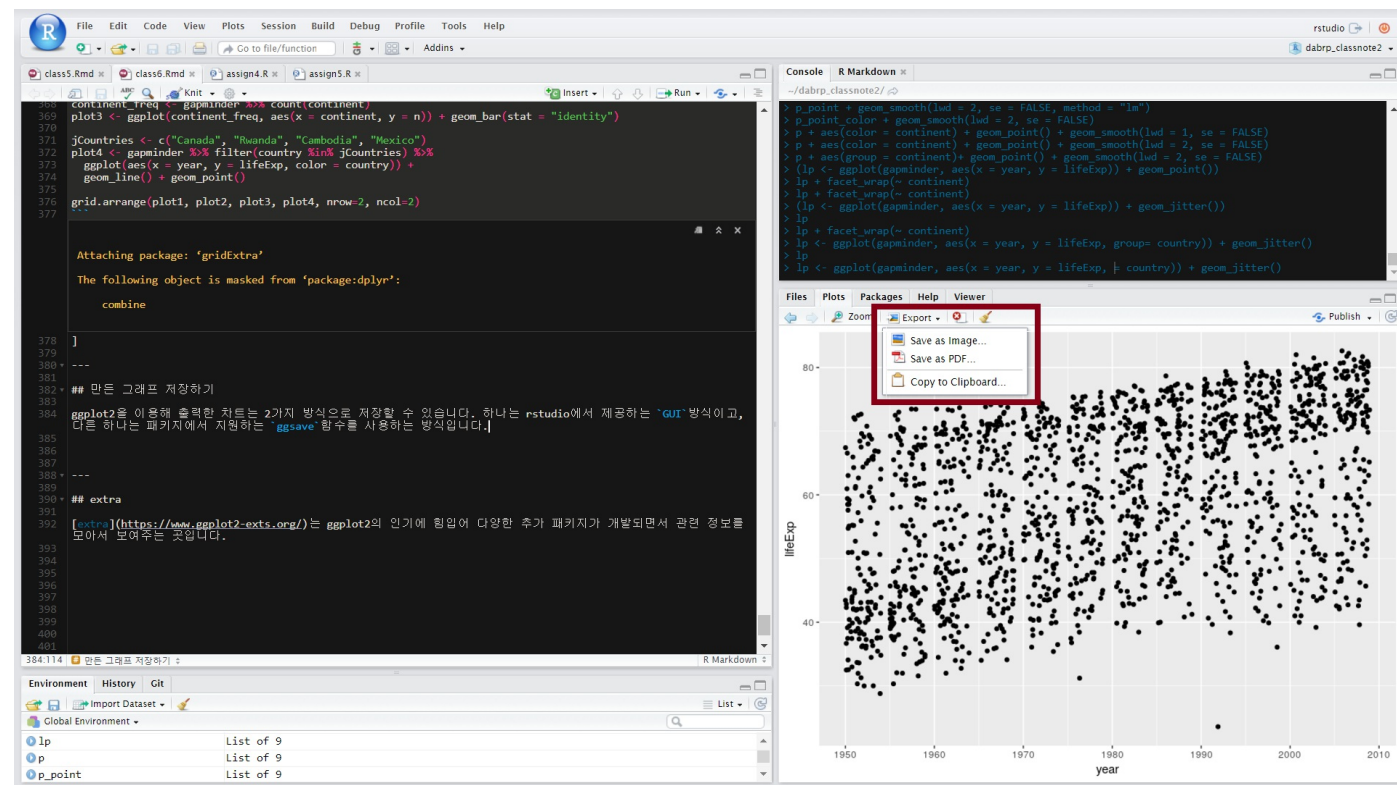
## 실습 2

1. `gapminder`에서 `pop`의 히스토그램(histogram)을 그리세요.
2. `continent`를 x축, `lifeExp`를 y축으로 하는 바이올린차트(violin)를 그리고 평균을 파란색 점으로 추가하세요.
3. 각 `continent`별로 총 몇 개의 데이터가 있는지 세어 `continent_freq`를 만들고, bar 차트를 그리세요.
4. Canada, Rwanda, Cambodia, Mexico 4 나라만 사용하여 x축은 `year`, y축은 `lifeExp`로 line 차트를 각 년도에 점을 추가하여 그리세요.



# 만든 그래프 저장하기

ggplot2을 이용해 출력한 차트는 2가지 방식으로 저장할 수 있습니다. 하나는 rstudio에서 제공하는 GUI방식이고, 다른 하나는 패키지에서 지원하는 **ggsave**함수를 사용하는 방식입니다.



```
dir.create("../ggsave", showWarnings = F)
ggsave("../ggsave/last.png")
```

## Saving 7 x 5 in image

# 여러 차트를 저장하기

계속 만드는 차트를 여러 번 저장해야 할 때 가 있습니다. 역시 **for** 문을 사용해서 저장해야 하는데, 이름을 바꾸면서 저장해야 해서 코드를 작성해 보았습니다.

```
dir.create("../ggsave", showWarnings = F)
for(i in 1:length(unique(gapminder$country))){
  gapminder %>% filter(country == gapminder$country[i]) %>%
    ggplot(aes(x = year, y = lifeExp, color = country)) +
    geom_line() + geom_point() +
    ggsave(paste0("../ggsave/", gapminder$country[i], ".png"))
  # print(paste0(i, " / ", length(unique(gapminder$country))))
}
```

# 지도 그리기

# 지도 그리기

지도는 x-y 좌표계에 위경도를 바탕으로 그림을 그리는 것과 같습니다. 점을 찍는 것 이외에도 다각형(**polygon**) 도형을 다루는 것, 외부의 지도 서비스를 이용해서 배경 이미지를 지도로 만드는 것 등의 일을 처리하게 됩니다.



# 한국에서의 좌표계

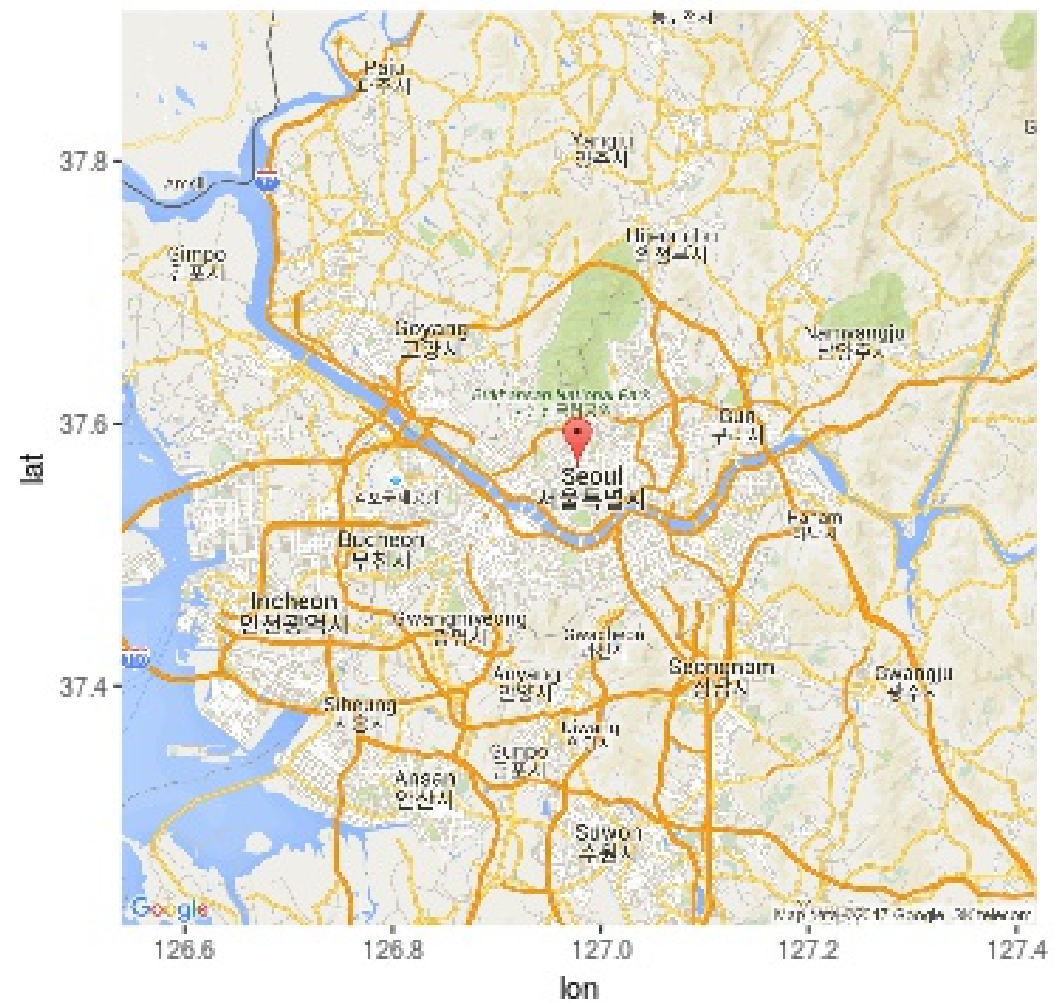
한국은 크게 UTM-K와 GPS로 알고있는데 WGS84(위경도)를 가장 많이 사용하고 있으며 지도에 관련된 api를 제공하는 서비스에서 사용할 때에 매우 중요한 인자이므로 꼭 알고 있어야 합니다.

- WGS84 : 위성이 사용하고 있는 좌표계이며 구글 지도가 표준으로 사용하고 있음.
- UTM-K : 국제 표준 좌표계중 하나인 UTM 좌표계를 한국 지리에 맞게 보정한 좌표계로 단위가 미터(m)단위라 거리에 대한 감각이 직관적이어서 많이 사용함. 대표적으로 네이버 지도가 사용함.

추가로 **좌표계에 대한 설명**을 참고해주세요.

# ggmap

**ggmap**은 구글 지도를 배경으로 ggplot을 사용할 수 있게 해주는 패키지로 내부적으로 모두 ggplot의 함수를 사용해서 문법이 호환됩니다.



# geocoding

**geocoding**이란 글자로 이루어져있는 주소를 위경도로 바꾸는 것을 뜻합니다. 반대로 반대로 위경도로 구성된 위치정보를 그 위치의 대표값(주소, 건물 이름 등)으로 바꾸는 것도 가능합니다.

ggmap은 `geocode()` 함수를 제공하고 있습니다.

# geocoding

**geocoding**이란 글자로 이루어져있는 주소를 위경도로 바꾸는 것을 뜻합니다. 반대로 반대로 위경도로 구성된 위치정보를 그 위치의 대표값(주소, 건물 이름 등)으로 바꾸는 것도 가능합니다.

ggmap은 `geocode()` 함수를 제공하고 있습니다.

```
geocode(location, output = "latlon", source = "google")
```

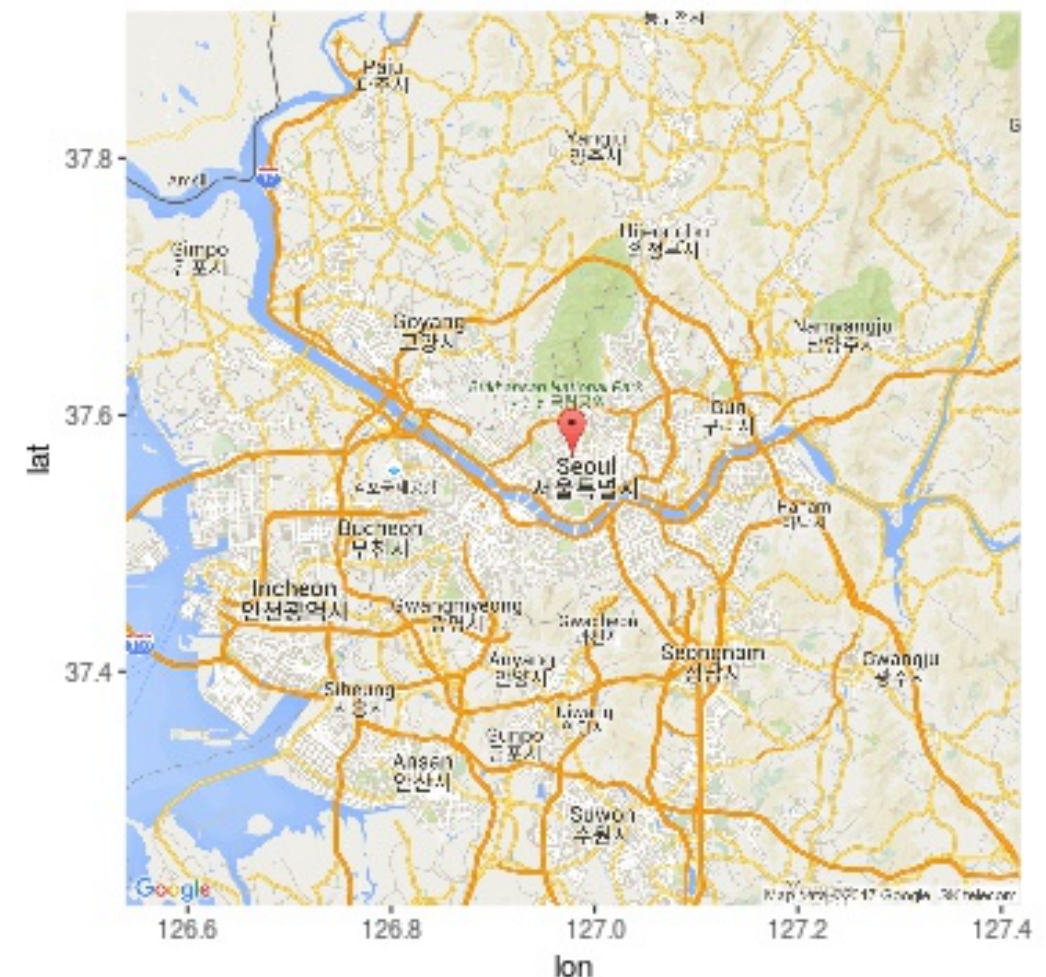
- `location`: 주소나 건물 이름에 해당하는 변환하고자 하는 값입니다.
- `output`: "latlon", "latlon", "more", "all"의 4가지 옵션이 있으며 뒤로 갈수록 정보가 상세합니다.
- `source`: 구글의 공개된 API를 사용한다는 뜻으로 대량으로 사용하기 위해서는 api-key를 등록해야 하거나 돈을 지불해야 할 수 있습니다.

# ggmap으로 ggplot 객체로 만들기

`get_googlemap()` 함수는 google 지도를 `ggmap` 객체로 가져오는 함수입니다. 그래서 추가적으로 `ggmap()` 함수를 이용해서 `ggplot` 객체로 변환해야 합니다. `windows`는 인코딩 문제가 있어서 `URLencode(enc2utf8("장소"))`의 형태로 사용해야 합니다.

```
if (!require("ggmap"))  
  {devtools::install_github("dkahle/ggmap")}  
library(ggmap)  
# for mac  
loc<-"서울"  
tar<-"서울시청"  
# for windows  
loc<-URLencode(enc2utf8("서울"))  
tar<-URLencode(enc2utf8("서울시청"))  
geocityhall<-geocode(tar)
```

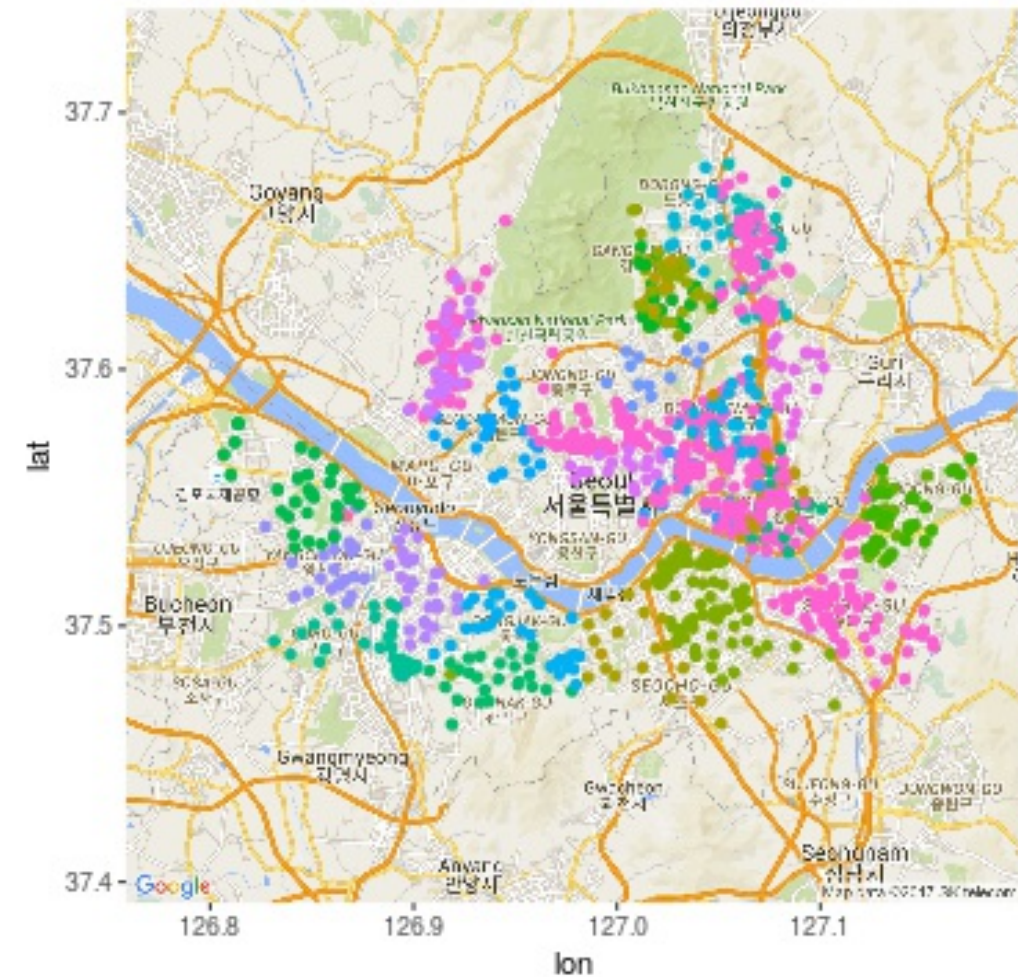
```
get_googlemap(loc,  
              maptype = "roadmap",  
              markers = geocityhall) %>%  
  ggmap()
```





# 실습 3

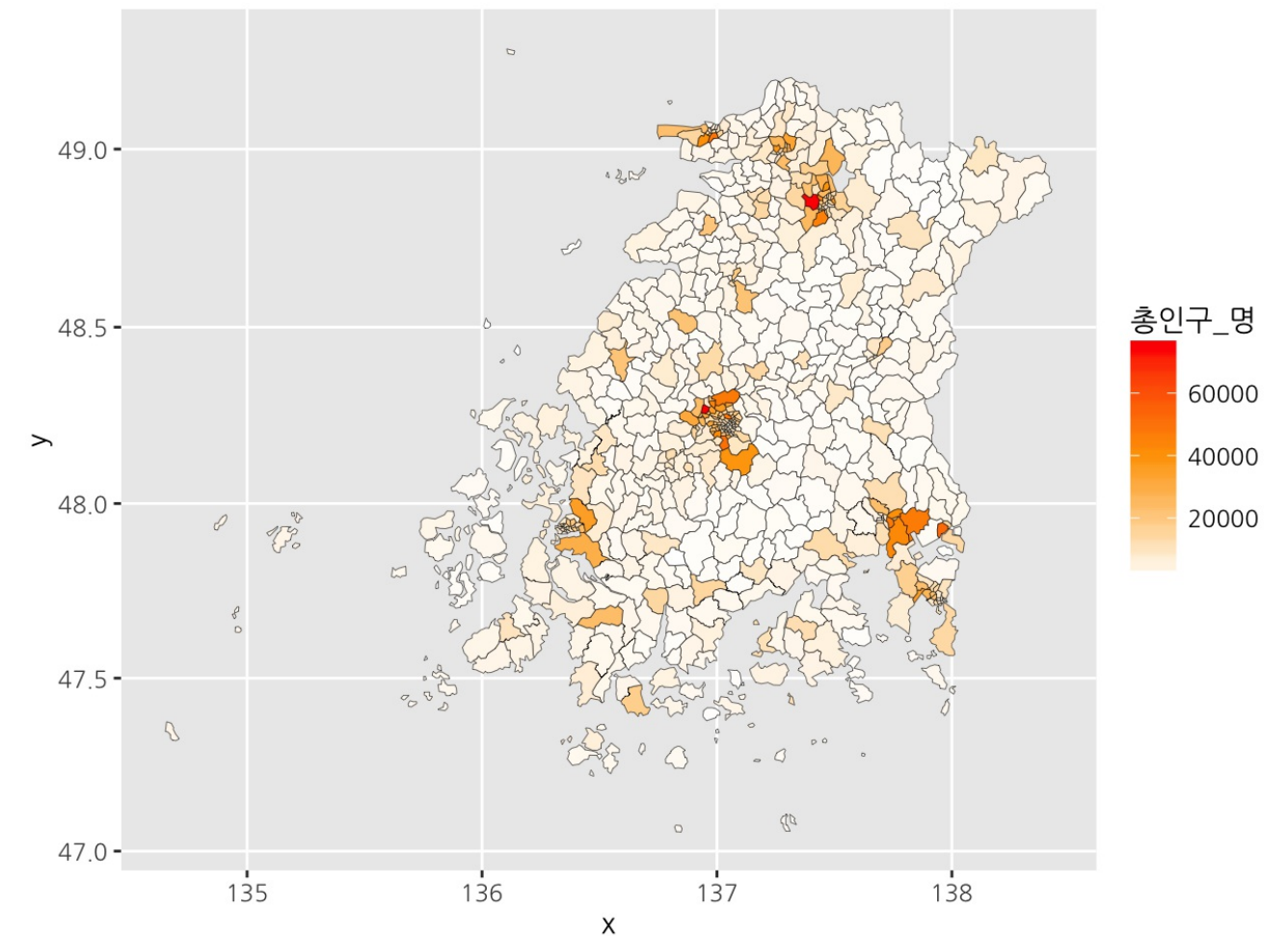
1. ./data/wifi.csv 파일을 이용해서 소재 지도로명 주소에 서울이 포함되는 wifi만 zoom 11, maptype roadmap 서울 시청 중심으로 wifi의 위치를 google 지도 위에 작성해 주세요.
  2. 관리기관명을 기준으로 색을 다르게 표시해 주세요.
- google 지도 위에 ggplot 객체를 추가하려면 아래와 같이 작성하면 됩니다.
- ```
get_googlemap(loc, maptype = "roadmap") %>%  
ggmap() +  
geom_point(data, aes(x=lon, y=lat))
```



# 한국 polygon 지도 패키지 Kormaps

**Kormaps**은 행정구역에 대한 **polygon** 정보를 미리 패키징한 것으로 인터랙티브 시각화를 지원하는 함수도 함께 제공하고 있습니다.

Kormaps는 설치 및 사용이 까다로워서 함께 진행하도록 하겠습니다. 코드는 **여기**를 참고해 주세요.



# extra

**extra**는 ggplot2의 인기에 힘입어 다양한 추가 패키지가 개발되면서 관련 정보를 모아서 보여주는 곳입니다.



# ggsci

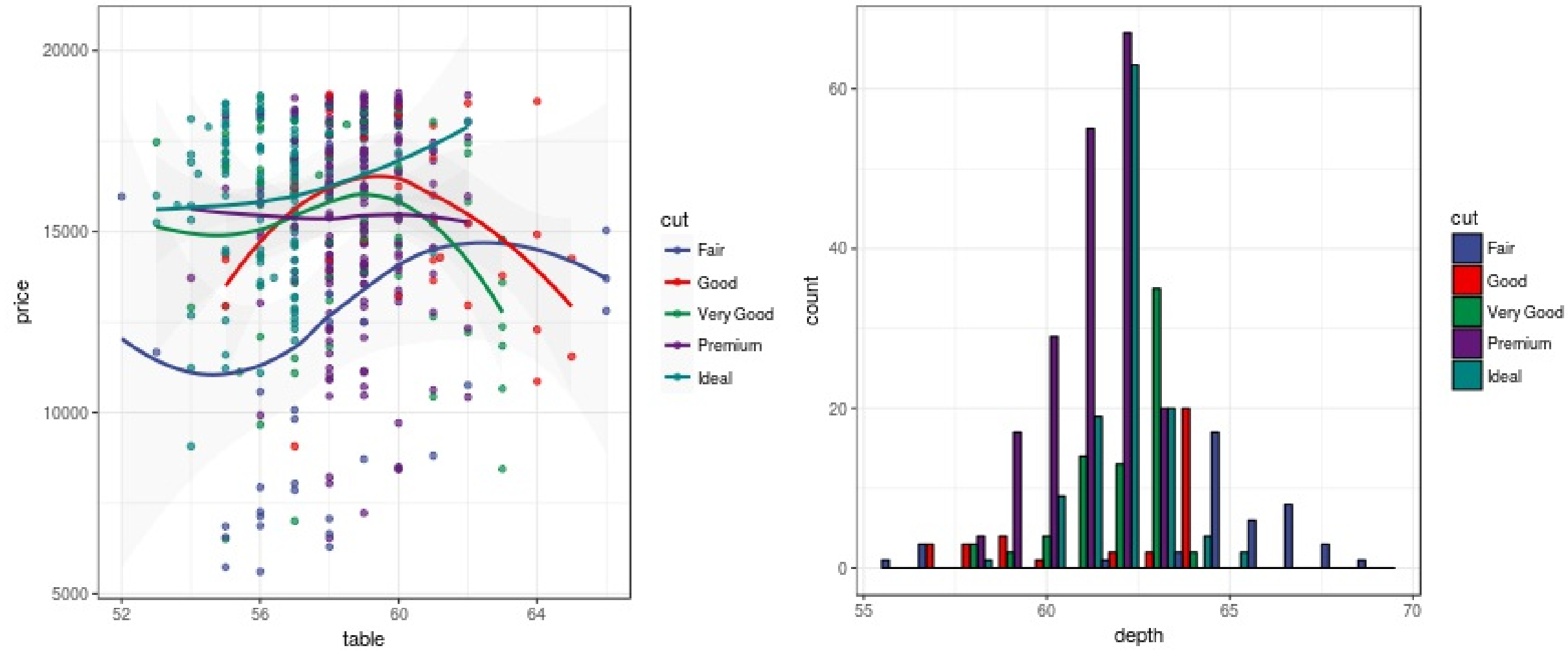
ggsci는 유명 색조합이 미리 준비되어 있는 패키지

```
if (!require(ggsci)) devtools::install_github("road2stat/ggsci")
library(ggsci)
library(gridExtra)
data("diamonds")
p1 = ggplot(subset(diamonds, carat >= 2.2),
            aes(x = table, y = price, colour = cut)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "loess", alpha = 0.05, size = 1, span = 1) +
  theme_bw()

p2 = ggplot(subset(diamonds, carat > 2.2 & depth > 55 & depth < 70),
            aes(x = depth, fill = cut)) +
  geom_histogram(colour = "black", binwidth = 1, position = "dodge") +
  theme_bw()

p1_aaas = p1 + scale_color_aaas()
p2_aaas = p2 + scale_fill_aaas()
```

```
grid.arrange(p1_aaas, p2_aaas, ncol = 2)
```



# ggthemes

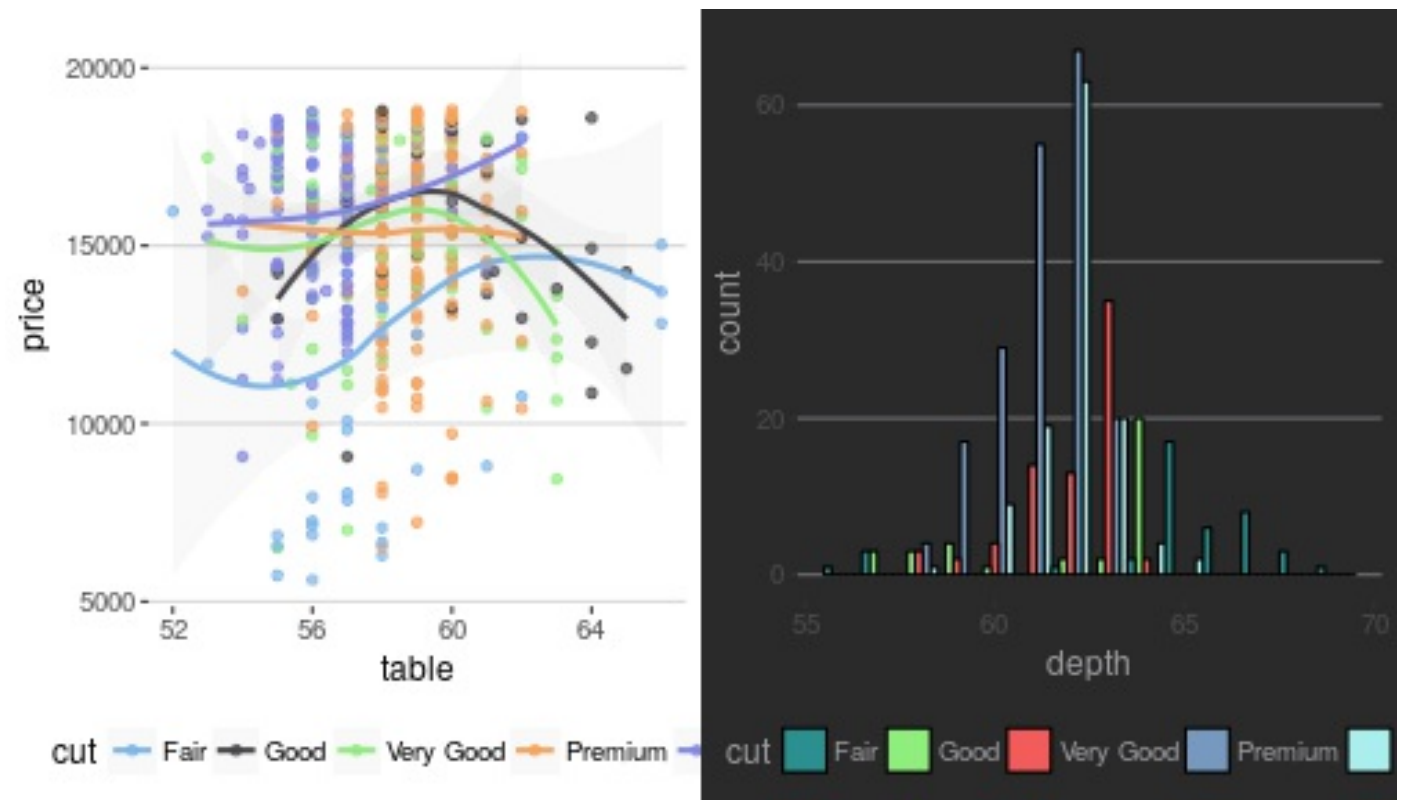
**ggthemes** 또한 **ggsci**와 같이 유명한 테마를 미리 만들어서 함수화한 패키지입니다. **ggsci**가 저널들의 디자인 테마 모음이라면 **ggthemes**는 다양한 제품 테마 모음이라고 할 수 있습니다.

```
if (!require(ggthemes)){
  devtools::install_github("jrnold/ggthemes")}

library("ggthemes")

p1_th1 <- p1 +
  theme_hc() +
  scale_colour_hc()
p2_th2 <- p2 +
  theme_hc(bgcolor = "darkunica") +
  scale_fill_hc("darkunica")
```

```
grid.arrange(p1_th1, p2_th2, ncol = 2)
```

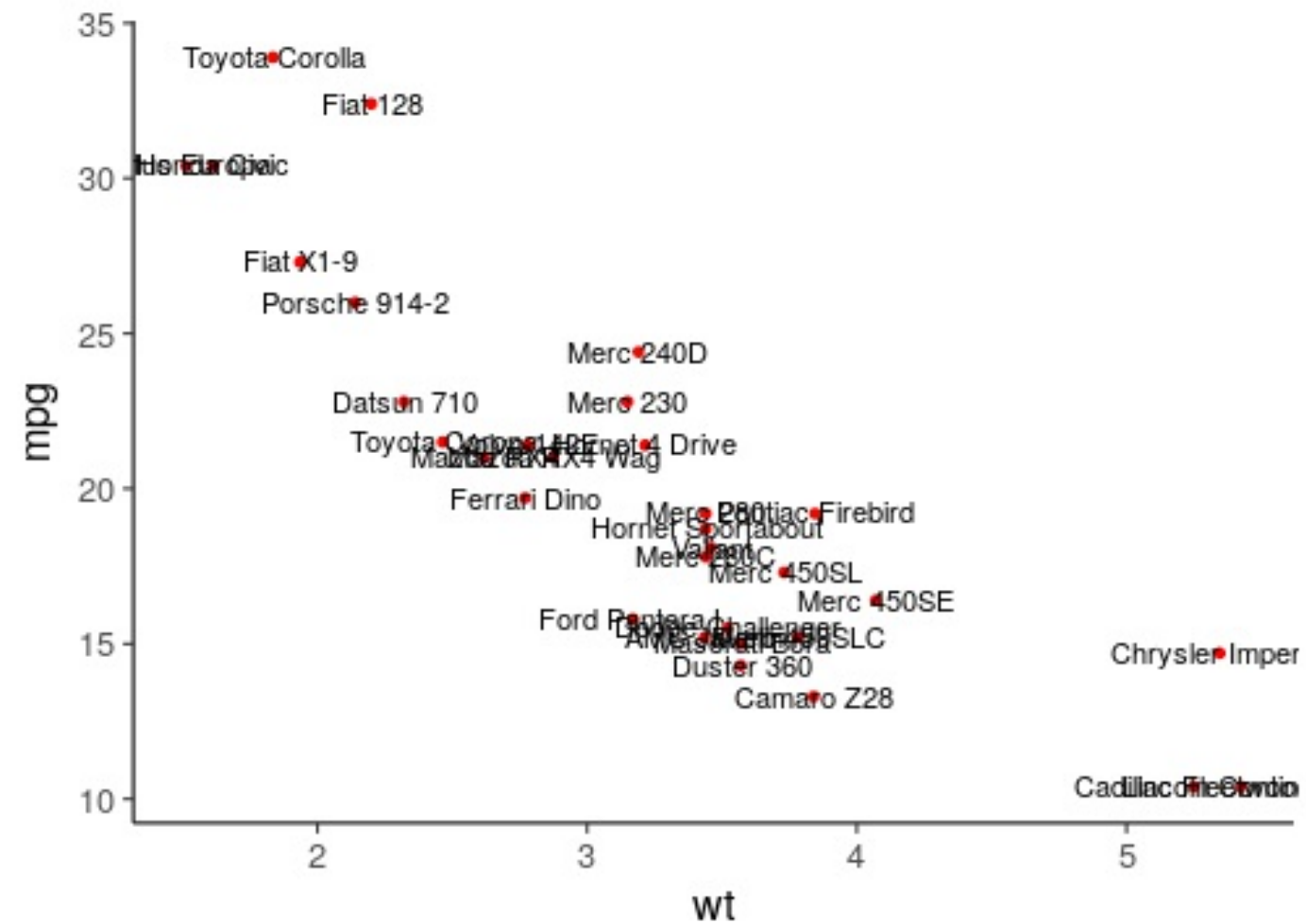


# ggrepel

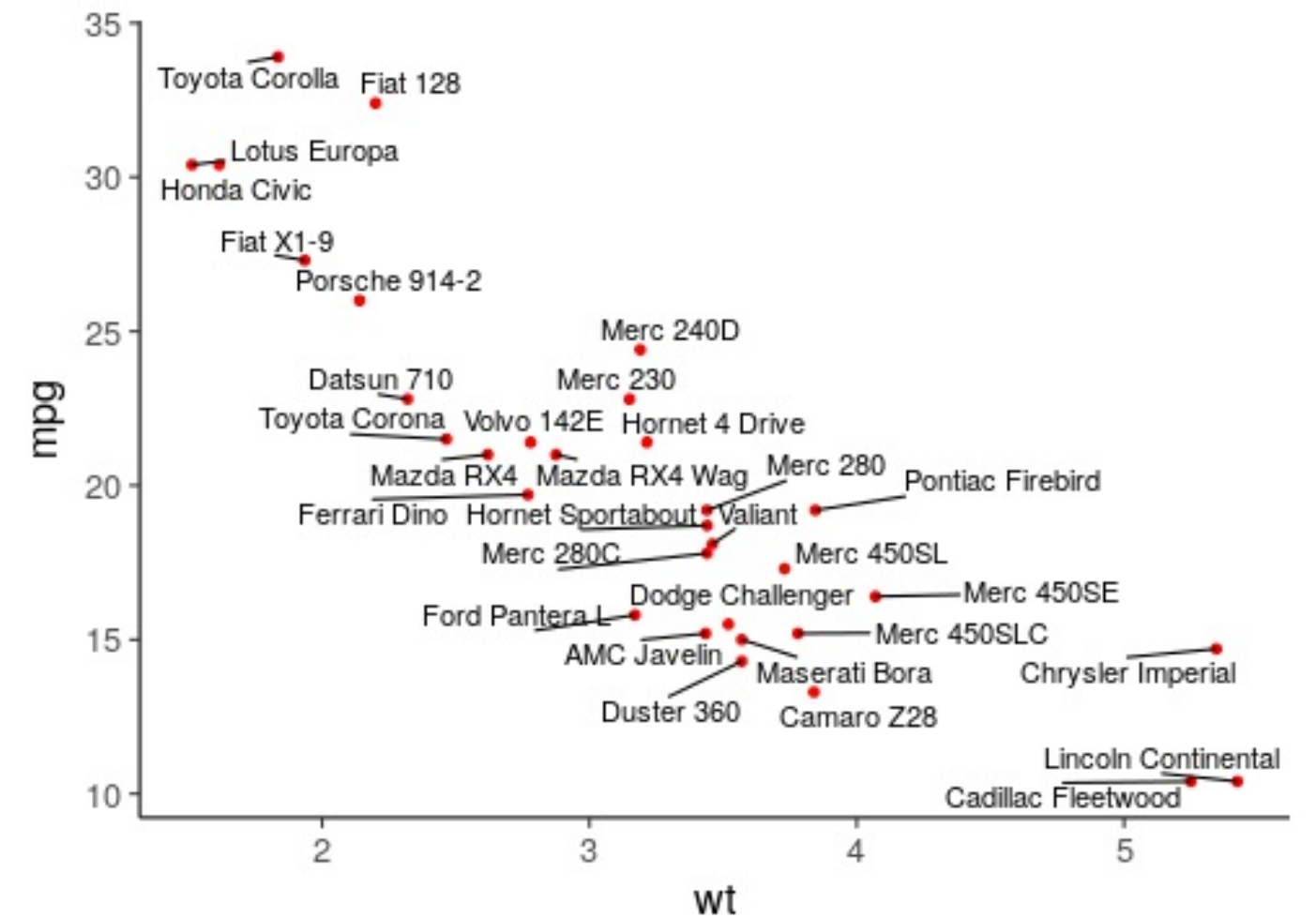
**ggrepel**는 글자를 겹치지 않게 그려주는 패키지로 보고서를 작성할 때 매우 유용합니다.

```
if (!require(ggrepel)){  
  devtools::install_github("slowkow/ggrepel")  
}  
library(ggrepel)
```

```
ggplot(mtcars) +
  geom_point(aes(wt, mpg), color = 'red') +
  geom_text(aes(wt, mpg,
                label = rownames(mtcars))) +
  theme_classic(base_size = 16)
```



```
ggplot(mtcars) +
  geom_point(aes(wt, mpg), color = 'red') +
  geom_text_repel(aes(wt, mpg,
                    label = rownames(mtcars))) +
  theme_classic(base_size = 16)
```



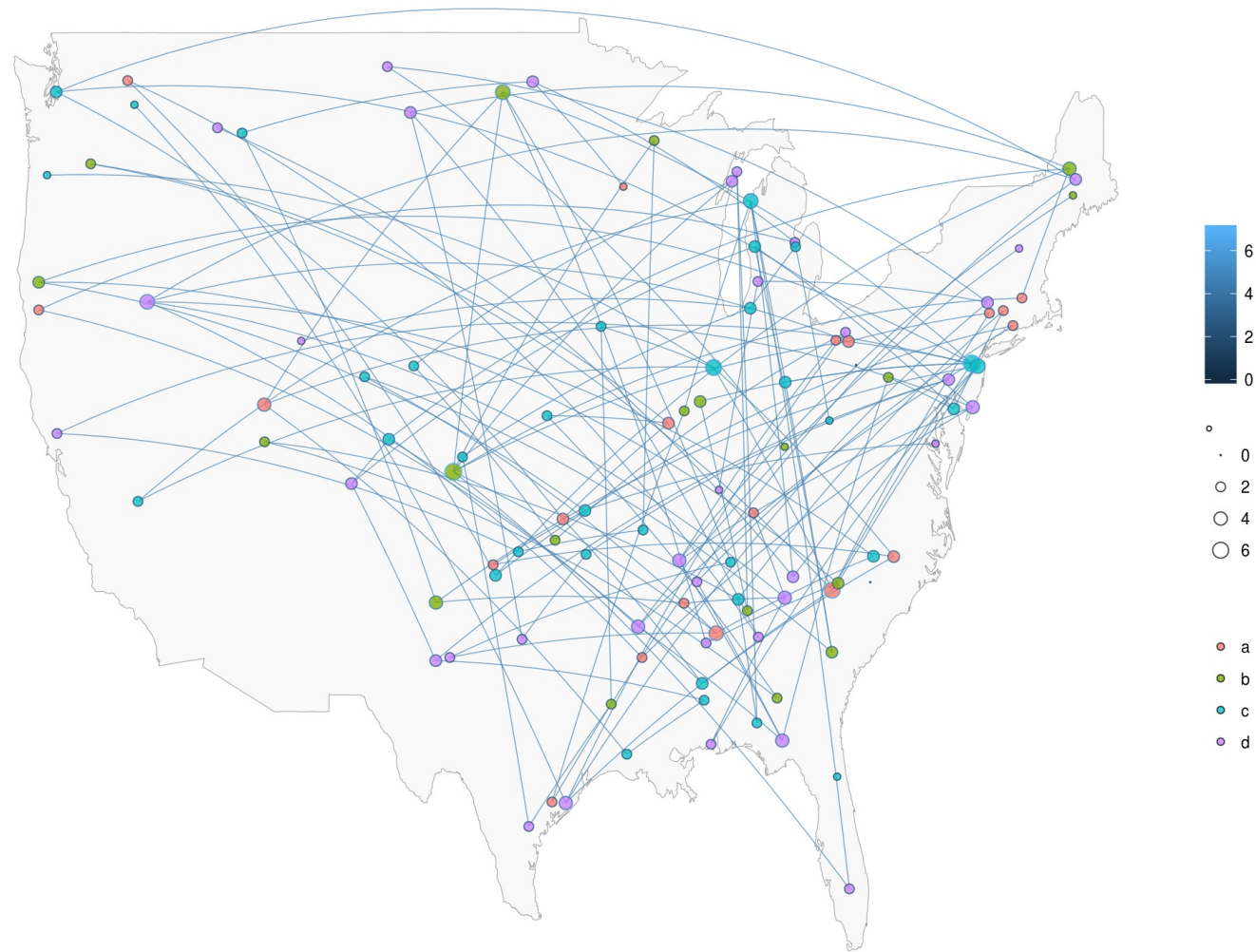
# ggfortify

**ggfortify**는 분석 결과 보고용 출력 수준 차트 생성 패키지로 R에서 제공하는 함수 결과물에 대해서 디자인된 출력 결과물을 대응하고 있습니다.

제공하는 함수 리스트는 위 링크를 참고해주세요. 위 링크의 하단에 예시를 설명한 7개의 설명서가 있으며 생존 분석, 선형회귀, 시계열 분석 등에 대해 `autoplot()` 함수 하나로 대응하는 것을 보여줍니다.

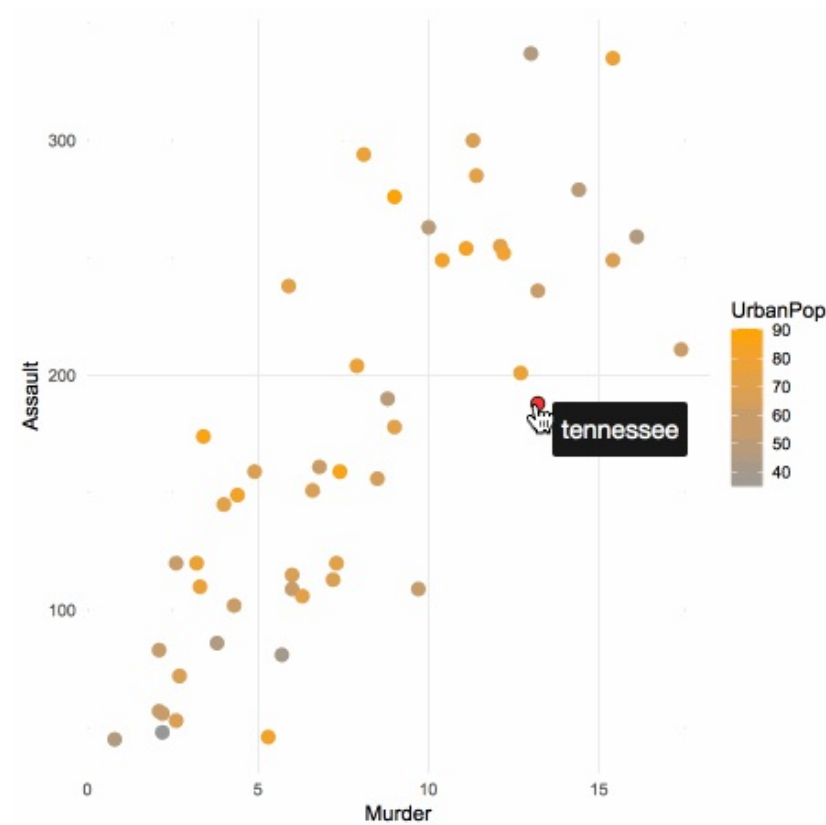
# ggally

ggally 또한 분석 결과 보고용 출력 수준 차트 생성 패키지로 ggfortify가 지원하지 않는 **network 분석**에 대해서 출력을 지원하고 있어서 소개합니다.



# ggiraph

**ggiraph**는 javascript 기반 동적 차트 생성 패키지로 클릭이나 마우스 오버 등의 동작에서 인터랙티브한 동작을 작성할 수 있게 도와주는 기능이 있습니다.





# ggedit

gui 클릭으로 ggplot 컨트롤할 수 있게 해주는 add-on으로 rstudio에서 직접 사용할 수 있습니다.

```
install.packages("ggedit")
library(ggedit)
p <- ggplot(mtcars, aes(x = hp, y = wt)) +
  geom_point() + geom_smooth(method = 'loess')
p
ggedit(p)
```

# 과제

1. `./recomen` 폴더에 있는 데이터를 사용해 주세요.
  - `"chennel.csv"`를 사용해서 x축은 `chennel`, y축은 각 채널별 사용량의 합으로 bar 차트를 그려주세요.
  - `"competitor.csv"`를 사용해서 x축은 `useDate`, y축은 그 달의 데이터 수로 line 차트를 그려주세요
  - 위와 같은 조건에서 `partner` 별로 라인과 색을 구분해서 그려주세요.
  - 축 데이터가 날짜인 경우 `scale_x_date()` 함수를 참고하세요.
2. `ggfortify` 패키지와 과제 5에서 진행한 회귀분석을 사용해 주세요.
  - [링크](#)를 참고해서 `fit` 객체를 시각화해주세요. 링크의 2번째 그림을 따라하면 됩니다.

# 과제 계속

1. "./data/wifi.csv" 데이터를 사용해주세요.
  - 서울중 각 구에 wifi가 몇개 있는지 세고 각 구 이름으로 geocoding한 위치에 갯수를 size로 하는 버블 차트를 그려주세요.
  - 시도를 기준으로 위와 같은 버블 차트를 그려주세요.
  - 시도를 기준으로 wifi의 수가 기간에 따라 얼마나 늘어났는지 확인할 수 있는 라인 차트를 그려주세요.
  - 서비스제공사명을 기준으로 통신 3사가 아닌 제공사는 기타로 처리하고, 여러 개로 작성된 것은 각 개수 만큼 분리하여 독립 wifi로 처리하여 위의 차트를 서비스제공사명으로 그룹지어서 그려주세요. 총 4개 차트가 함께 나오면 됩니다.