

I. Choropleth Map (단계구분도)
oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

III. Choropleth Map 그리기!

ooo

IV. Some Improvements - colors
oooooooooooo

V. Some Improvements
oooooooooooo

L16. Geospatial data (3)

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr



서울과학기술대학교 데이터사이언스학과

I. Choropleth Map (단계구분도)

oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비

oooooooooooo

III. Choropleth Map 그리기!

ooo

IV. Some Improvements - colors

oooooooooooo

V. Some Improvements

oooooooooooo

I. Choropleth Map (단계구분도)

II. 지역정보를 포함한 최종 데이터셋 준비

III. Choropleth Map 그리기!

IV. Some Improvements - colors

V. Some Improvements - add text

I. Choropleth Map (단계구분도)

●○○○○○○○○○○

II. 지역정보를 포함한 최종 데이터 셋 준비

○○○○○○○○○○○

III. Choropleth Map 그리기!

○○○

IV. Some Improvements - colors

○○○○○○○○○○○

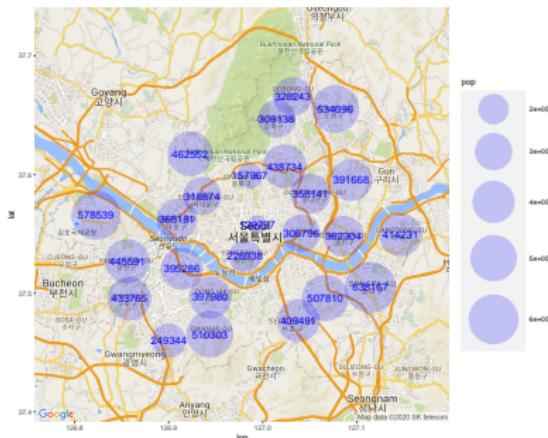
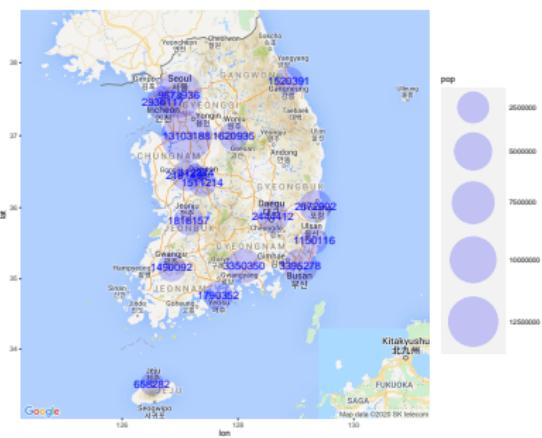
V. Some Improvements

○○○○○○○○○○○○○○

I. Choropleth Map (단계구분도)

Review & Motivation

- 지난 강의에서는 아래의 3가지 데이터 구성요소를 이용
 - 지도 (via `get_map()`)
 - 위치 정보 (`kr_latlon.csv`)
 - 지역정보 (`kr_pop_tidy.csv` or `seoul_pop_tidy.csv`)



Discussion on the previous graphics

- 특정 지점에 대한 정보를 표현하기에 좋다. (e.g. 주유소의 위치를 지도에 표현)
 - 그러나 인구수와 같은 행정구역별 통계량은 구획별 **boundary**가 명시된 지도에 표현하는 것이 더 좋다.

Choropleth map

- Choropleth map is a *special type of heat map* with *geographical boundaries*.
 - So to speak, the canvas includes *boundary* information.

데이터 사용 방식

- 지도위에 표시 (previous)
 1. 지도 (via `get_map()`)
 2. 위치 정보 (`kr_latlon.csv`)
 3. 지역정보 (`kr_pop_tidy.csv` or `seoul_pop_tidy.csv`)
 - Choropleth (this lecture note)
 1. Choropleth base
 - 새로운 형태의 canvas로 써, 기존 방식의 ‘1.지도’와 ‘2. 위치 정보’가 모두 포함됨.
 2. 지역정보

How to get the base canvases?

한국 지도 다운받기

```
library(raster) # For Geographic Data Analysis and Modeling
```

- GADM(Database of Global Administrative Areas)에 저장된 행정구역
지리정보를 다운받는다.
 - level=1은 광역시·도
 - level=2는 광역시의 구, 시·군 단위

```
kr_map_lv11 <- getData('GADM', country='kor', level=1)
kr_map_lv12 <- getData('GADM', country='kor', level=2)
```

- 이렇게 받아온 지리정보를 **shape** 파일이라고 한다.
 - 좌표의 정보가 모두 담겨 있기 때문에 `ggmap()`을 사용하지 않고 `ggplot()`을 사용할 수 있다!
 - (빠르고 쉽다는 큰 장점이 있다)

I. Choropleth Map (단계구분도)

○○○○●○○○○○○

II. 지역정보를 포함한 최종 데이터셋 준비

○○○○○○○○○○○○

III. Choropleth Map 그리기!

○○○

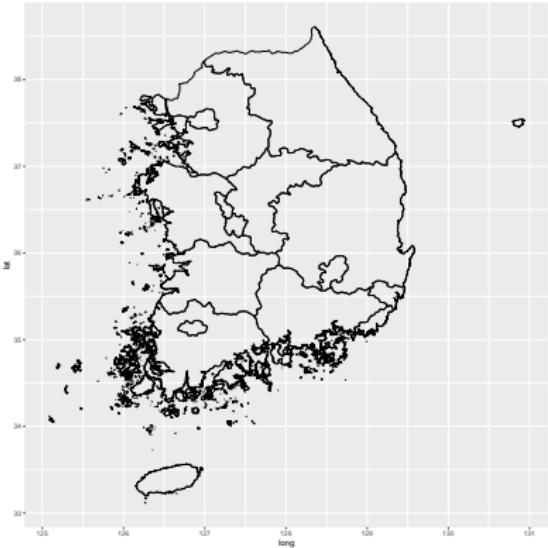
IV. Some Improvements - colors

○○○○○○○○○○○○

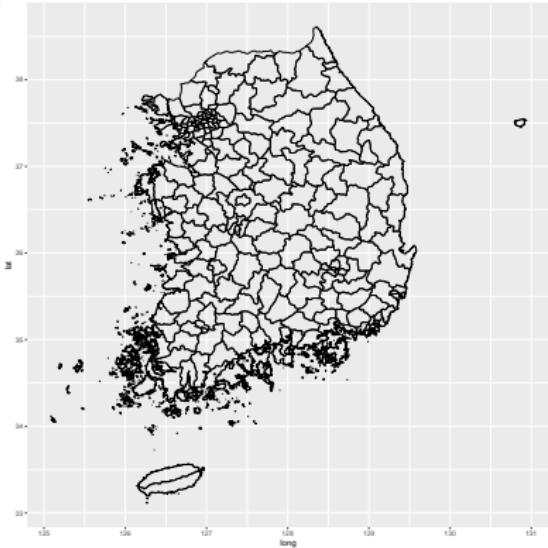
V. Some Improvements

○○○○○○○○○○○○○○○○○○

```
ggplot(kr_map_lv11,  
       aes(x=long, y=lat, group=group)) +  
  geom_path()
```



```
ggplot(kr_map_lv12,  
       aes(x=long, y=lat, group=group)) +  
  geom_path()
```



서울 지도 만들기

- 앞의 kr_map_lv11은 대한민국 전체지도로 사용하기에 무방하다.
- 서울 지도를 만들기 위해서는 kr_map_lv12에서 subsetting을 해주어야 한다.
- getData()로 받은 데이터의 클래스는 SpatialPolygonsDataFrame이다.

```
class(kr_map_lv12)

## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

- SpatialPolygonsDataFrame는 .shp 확장자를 가진 'shape' 파일을 이용해서도 생성 가능하다.
 - raster:::shapefile()을 이용하면 .shp 확장자의 파일을 불러올 수 있다.
 - 공공 데이터 포털 등에 여러가지 목적의 shp 파일이 제공된다.

oooooooo●oooo

oooooooooooo

○○○

oooooooooooo

oooooooooooo

- SpatialPolygonsDataFrame 클래스는 @data와 @polygon의 서로 연결된 하위 구조를 가지고 있다.
- 하위 구조인 @data는 data.frame이기 때문에 subset, join등의 여러가지 처리가 가능하다.

```
kr_map_lv12@data %>% str()
```

```
## 'data.frame': 229 obs. of 13 variables:
## $ GID_0    : chr "KOR" "KOR" "KOR" "KOR" ...
## $ NAME_0   : chr "South Korea" "South Korea" "South Korea" "South Korea" ...
## $ GID_1    : chr "KOR.1_1" "KOR.1_1" "KOR.1_1" "KOR.1_1" ...
## $ NAME_1   : chr "Busan" "Busan" "Busan" "Busan" ...
## $ NL_NAME_1: chr "부산광역시 | 釜山廣域市" "부산광역시 | 釜山廣域市" "부산광역시 | 釜山廣域市"
## $ GID_2    : chr "KOR.1.1_1" "KOR.1.2_1" "KOR.1.3_1" "KOR.1.4_1" ...
## $ NAME_2   : chr "Buk" "Busanjin" "Dong" "Dongnae" ...
## $ VARNAME_2: chr NA NA NA NA ...
## $ NL_NAME_2: chr "북구| 北區" "부산진구| 釜山鎮區" "동구| 東區" "동래구| 東萊區" ...
## $ TYPE_2   : chr "Gu" "Gu" "Gu" "Gu" ...
## $ ENGTTYPE_2: chr "District" "District" "District" "District" ...
## $ CC_2     : chr NA NA NA NA ...
## $ HASC_2   : chr NA NA NA NA ...
```

- kr_map_lv12@data의 변수 NAME_1를 이용해서 kr_map_lv12를 subsetting 한다.

```
seoul_map_lv12 <- kr_map_lv12[kr_map_lv12$NAME_1 == "Seoul",]
seoul_map_lv12@data %>% str()

## 'data.frame': 25 obs. of 13 variables:
## $ GID_0    : chr "KOR" "KOR" "KOR" "KOR" ...
## $ NAME_0   : chr "South Korea" "South Korea" "South Korea" "South Korea" ...
## $ GID_1    : chr "KOR.16_1" "KOR.16_1" "KOR.16_1" "KOR.16_1" ...
## $ NAME_1   : chr "Seoul" "Seoul" "Seoul" "Seoul" ...
## $ NL_NAME_1: chr "서울특별시" "서울특별시" "서울특별시" "서울특별시" ...
## $ GID_2    : chr "KOR.16.1_1" "KOR.16.2_1" "KOR.16.3_1" "KOR.16.4_1" ...
## $ NAME_2   : chr "Dobong" "Dong-daemun" "Dongjak" "Eun-pyeong" ...
## $ VARNAME_2: chr NA NA NA NA ...
## $ NL_NAME_2: chr "도봉구|道峰區" "동대문구|東大門區" "동작구|銅雀區" "은평구|恩平區" ...
## $ TYPE_2   : chr "Gu" "Gu" "Gu" "Gu" ...
## $ ENGTTYPE_2: chr "District" "District" "District" "District" ...
## $ CC_2     : chr NA NA NA NA ...
## $ HASC_2   : chr NA NA NA NA ...
```

oooooooooooo●○

oooooooooooo

○○○

oooooooooooo

oooooooooooo

- (주의) `SpatialPolygonsDataFrame`는 엄밀한 의미에서의 `data.frame`은 아니기 때문에 `dplyr::filter()`를 사용할 수는 없다.

```
# Following Line causes error
```

```
seoul_map_lv12 <- kr_map_lv12 %>% filter(NAME_1 == "Seoul")
```

oooooooooooo●

oooooooooooo

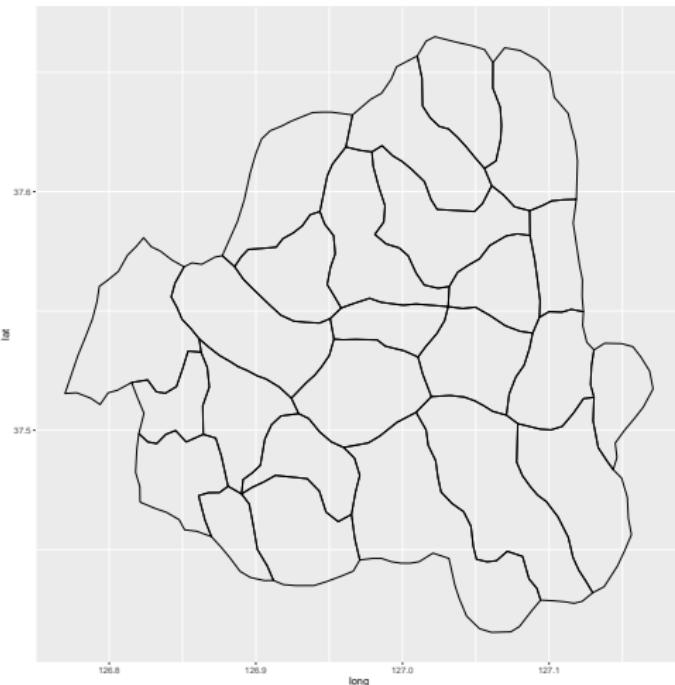
○○○

oooooooooooo

oooooooooooo

- 서울의 지도가 만들어졌다!

```
ggplot(seoul_map_lv12,  
       aes(x=long, y=lat, group=group)) +  
  geom_path()
```



I. Choropleth Map (단계구분도)
○○○○○○○○○○○○

II. 지역정보를 포함한 최종 데이터 셋 준비
●○○○○○○○○○○

III. Choropleth Map 그리기!
○○○
○○○○○○○○○○
○○○○○○○○○○○○

IV. Some Improvements - colors
V. Some Improvements

II. 지역정보를 포함한 최종 데이터 셋 준비

지역정보 데이터셋 준비

```

kr_pop <- read_csv(
  "data/kr_pop_tidy.csv",
  locale = locale('ko', encoding='euc-kr'))
kr_pop_2018 <- kr_pop %>%
  filter(year==2018 &
        category=="총인구 (명)" &
        state!="전국") %>%
  dplyr::select(state, pop)
head(kr_pop_2018)

## # A tibble: 6 x 2
##   state      pop
##   <chr>     <dbl>
## 1 서울특별시 9673936
## 2 부산광역시 3395278
## 3 대구광역시 2444412
## 4 인천광역시 2936117
## 5 광주광역시 1490092
## 6 대전광역시 1511214

```

```

seoul_pop <- read_csv(
  "data/seoul_pop_tidy.csv",
  locale = locale('ko', encoding='euc-kr'))
seoul_pop$district <-
  str_trim(seoul_pop$district)
seoul_pop_2018 <- seoul_pop %>%
  filter(year==2018 &
        category=="총인구 (명)") %>%
  dplyr::select(district, pop)
head(seoul_pop_2018)

## # A tibble: 6 x 2
##   district      pop
##   <chr>     <dbl>
## 1 종로구      157967
## 2 중구       129797
## 3 용산구      226938
## 4 성동구      306796
## 5 광진구      362304
## 6 동대문구    358141

```

데이터 셋 조인 (Canvass와 인구정보)

Strategy Development

- 지난 예제에서는 `ggmap()`에 지도 객체를 넣고, `ggplot()`에 데이터를 넣었다.
 - Choropleth Map에서는 `ggmap()`을 사용하지 않기 때문에 관련된 모든 데이터를 하나의 객체로 만들어서 `ggplot()`에 넣어야 한다.
 - 즉, 아래와 같은 데이터 조인이 선행되어야 한다.
 - (전국) `kr_map_lv11`과 `kr_pop_2018`의 조인
 - (서울) `seoul_map_lv12`와 `seoul_pop_2018`의 조인
 - 그런데, `kr_map_lv11`과 `seoul_map_lv12`는
 - `data.frame`이 아닌 `SpatialPolygonsDataFrame`이 아니다.
 - 그렇기 때문에 `join`이 불가능하다. (12페이지 참고)
 - 그렇기 때문에 `SpatialPolygonsDataFrame`를 `data.frame`로 변환해 주는 `ggplot::fortify()`함수를 사용한다.

Step 1. `ggplot:::fortify()`로 `kr_map_df` 생성

```
kr_map_df <- fortify(kr_map_lv11)  
head(kr_map_df, 2)
```

```

##      long     lat order hole piece id group
## 1 129.0179 35.07678     1 FALSE    1 1 1.1
## 2 129.0179 35.07653     2 FALSE    1 1 1.1

```

```
str(kr_map_df)
```

```
## 'data.frame': 339940 obs. of 7 variables:  
## $ long : num 129 129 129 129 129 ...  
## $ lat : num 35.1 35.1 35.1 35.1 35.1 ...  
## $ order: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ hole : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ piece: Factor w/ 926 levels "1","2","3","4",... 1 1 1 1 1 1 1 1 1 1 ...  
## $ id   : chr "1" "1" "1" "1" ...  
## $ group: Factor w/ 1561 levels "1.1","1.2","1.3",... 1 1 1 1 1 1 1 1 1 1 ...
```

- kr_map_df는 kr_map_lv11의 data.frame 버전으로서
 - 한국내의 339940개의 지점들에 대해
 - 위경도 정보 (long, lat)와
 - 구획 정보 (id, group)를 담고 있다.

Step 2. kr_map_lvl1@data의 메타 정보와 결합

- kr_map_df에는 우리가 이해할 수 있는 구획 정보가 없어졌다.
- 광역시도에 관한 메타 정보는 데이터 프레임 kr_map_lvl1@data에 담겨 있었으므로 kr_map_df와 결합한다.

```
str(kr_map_df)
```

```
## 'data.frame':    339940 obs. of  7 variables:  
## $ long : num  129 129 129 129 129 ...  
## $ lat  : num  35.1 35.1 35.1 35.1 35.1 ...  
## $ order: int  1 2 3 4 5 6 7 8 9 10 ...  
## $ hole : logi  FALSE FALSE FALSE FALSE FALSE ...  
## $ piece: Factor w/ 926 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ...  
## $ id   : chr  "1" "1" "1" "1" ...  
## $ group: Factor w/ 1561 levels "1.1","1.2","1.3",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```

sapply(kr_map_df, function(x) length(unique(x)))

##   long     lat order  hole piece    id group
## 31731 35549 160773      2    926    17 1561

str(kr_map_lv1@data)

## 'data.frame': 17 obs. of 10 variables:
## $ GID_0    : chr "KOR" "KOR" "KOR" "KOR" ...
## $ NAME_0   : chr "South Korea" "South Korea" "South Korea" "South Korea" ...
## $ GID_1    : chr "KOR.1_1" "KOR.2_1" "KOR.3_1" "KOR.4_1" ...
## $ NAME_1   : chr "Busan" "Chungcheongbuk-do" "Chungcheongnam-do" "Daegu" ...
## $ VARNAME_1: chr "Pusan|Busan Gwang'yeogssi|Pusan-gwangyoksi|Fusan" "Chungchongbuk-Do|Chungcheongbugdo|Ch'ungch'ong-bukto|Chusei Hoku-do|North Chungchong|Ch'ungch'ong-bukto" "Chungchongnam-Do|Ch'ungch'ong-namdo|Chusei Nan-do|South Chungchong|Ch'ungch'ong-namdo" "Taegu|Daegu Gwang'yeogssi|Taegu-gwangyoksi|Taikyu" ...
## $ NL_NAME_1: chr "부산광역시 | 釜山廣域市" "충청북도 | 忠淸北道" "충청남도 | 忠淸南道" ...
## $ TYPE_1   : chr "Gwangyeoksi" "Do" "Do" "Gwangyeoksi" ...
## $ ENGTTYPE_1: chr "Metropolitan City" "Province" "Province" "Metropolitan City" ...
## $ CC_1     : chr NA NA NA NA ...
## $ HASC_1   : chr "KR.PU" "KR.GB" "KR.GN" "KR.TG" ...

```

- kr_map_df\$id의 unique values가 17개이므로 조인의 key variable로 사용한다.
 - kr_map_lv1@data\$GID_1가 가장 비슷하게 생겼기에 key variable로 사용하기 좋아보인다.
 - 확인은 아래와 같이 하면된다.

```
unique(kr %>% df$id)
```

```
## [1] "1"  "10" "11" "12" "13" "14" "15" "16" "17" "2"   "3"  "4"  "5"  "6"  "7"  
## [16] "8"  "9"
```

```
unique(kr_map_lv1@data$GID_1)
```

```
## [1] "KOR.1_1"  "KOR.2_1"  "KOR.3_1"  "KOR.4_1"  "KOR.5_1"  "KOR.6_1"  
## [7] "KOR.7_1"  "KOR.8_1"  "KOR.9_1"  "KOR.10_1" "KOR.11_1" "KOR.12_1"  
## [13] "KOR.13_1" "KOR.14_1" "KOR.15_1" "KOR.16_1" "KOR.17_1"
```

- 아래 코드는 복잡해 보이지만, 그다지 어렵지 않다.
 - kr_map_lv1@data의 GID_1 변수의 형태를 kr_map_df\$id와 같이 바꾸어 준 후에 join하면 된다.

```
kr_map_df <- kr_map_df %>% left_join(  
  kr_map_lv1@data %>%  
    separate(GID_1, into=c("dummy1", "code"), sep=".") %>%  
    separate(code, into=c("id", "dummy2"), sep="_") %>%  
    separate(NL_NAME_1, into=c("state", "dummy3"), sep="[]") %>%  
    mutate(state = str_trim(state)) %>%  
    dplyr::select(-dummy1, -dummy2, -dummy3)  
)
```

Step 3. 비슷한 방법으로 seoul_map_df를 만든다.

- kr_map_df와 seoul_map_df는 level이 다르므로 변수 이름등이 다를 수 있다는 것에 주의한다.
 - 실제로 key 변수를 확보하기 위해서 @polygons파트에 접근하여 id를 생성하였다.
 - (희망과 달리 노가다를…)

```
seoul_map_df <- fortify(seoul_map_lvl2)
seoul_map_df <- seoul_map_df %>% left_join(
  seoul_map_lvl2@data %>%
    # retrieve ID from @polygon
    mutate(id = sapply(seoul_map_lvl2@polygons, function(x) x@ID)) %>%
    separate(NL_NAME_2, into=c("district", "dummy1"), sep="|") %>%
    dplyr::select(-dummy1)
)
```

인구정보와 결합

```
kr_map_df <- kr_map_df %>% left_join(kr_pop_2018)
kr_map_df %>%
  dplyr::select(long, lat, order, hole, group, state, pop) %>%
  head(3)

##       long      lat order  hole group      state    pop
## 1 129.0179 35.07678     1 FALSE  1.1 부산광역시 3395278
## 2 129.0179 35.07653     2 FALSE  1.1 부산광역시 3395278
## 3 129.0176 35.07653     3 FALSE  1.1 부산광역시 3395278

seoul_map_df <- seoul_map_df %>% left_join(seoul_pop_2018)
seoul_map_df %>%
  dplyr::select(long, lat, order, hole, group, district, pop) %>%
  head(3)

##       long      lat order  hole group district    pop
## 1 127.0617 37.65409     1 FALSE 182873.1   도봉구 328243
## 2 127.0616 37.64340     2 FALSE 182873.1   도봉구 328243
## 3 127.0667 37.63531     3 FALSE 182873.1   도봉구 328243
```

I. Choropleth Map (단계구분도)
○○○○○○○○○○○○

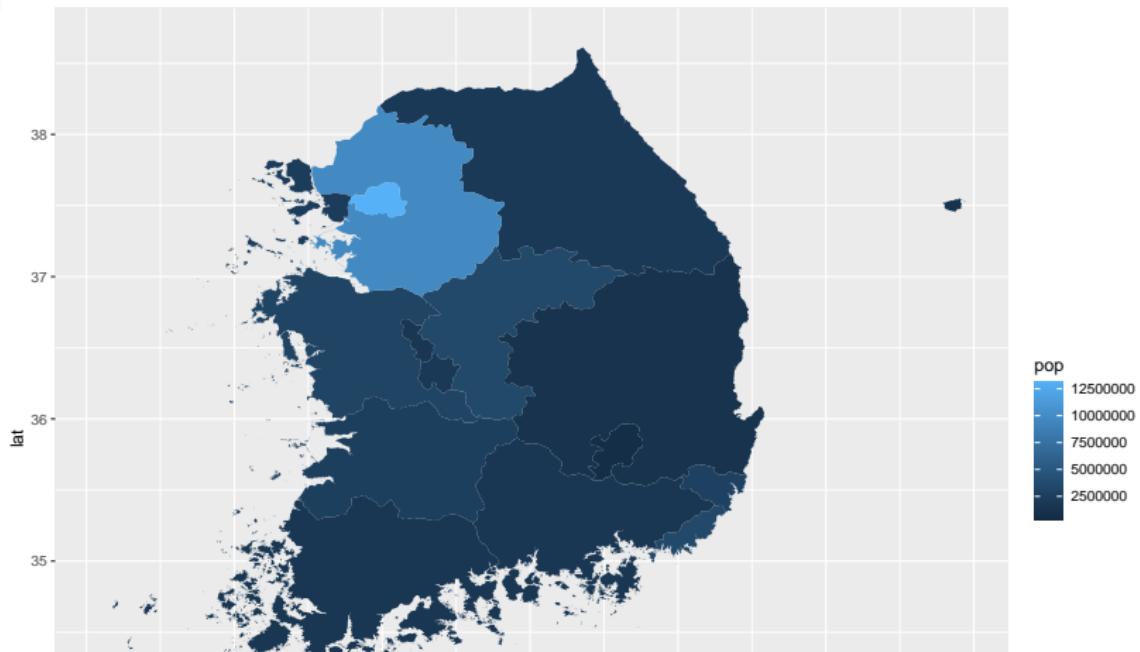
II. 지역정보를 포함한 최종 데이터 셋 준비
○○○○○○○○○○○○

III. Choropleth Map 그리기!
●○○
○○○○○○○○○○○○
○○○○○○○○○○○○○○

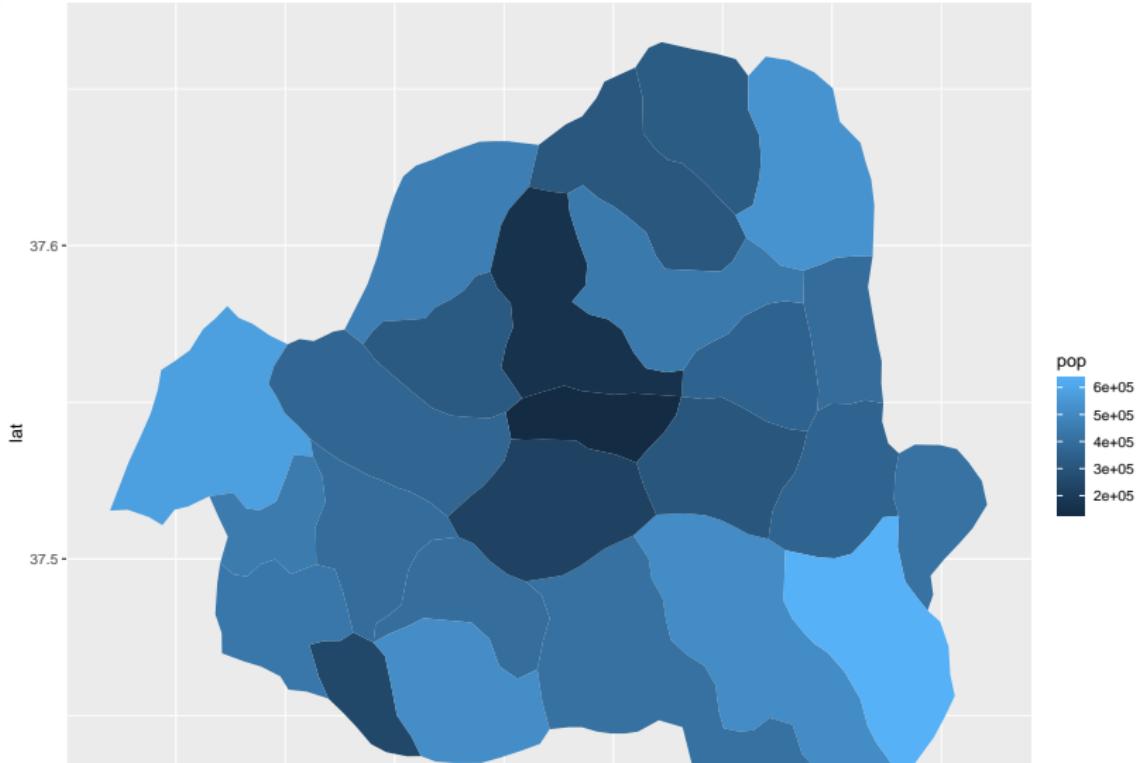
III. Choropleth Map 그리기!

Minimal Examples

```
fig1 <- kr_map_df %>% arrange(group, order) %>%  
  ggplot(aes(x=long, y=lat, group=group)) +  
  geom_polygon(aes(fill=pop))  
  
fig1
```



```
fig2 <- seoul_map_df %>% arrange(group, order) %>%  
  ggplot(aes(x=long, y=lat, group=group)) +  
  geom_polygon(aes(fill = pop))  
  
fig2
```



I. Choropleth Map (단계)

III. Choropleth Map 그리기!

IV. Some Improvements - colors

V. Some Improvements

IV. Some Improvements - colors

I. Choropleth Map (단계별 색상)

II. 지역정보를 포함한 최종 데이터 셋 준비 0000000000

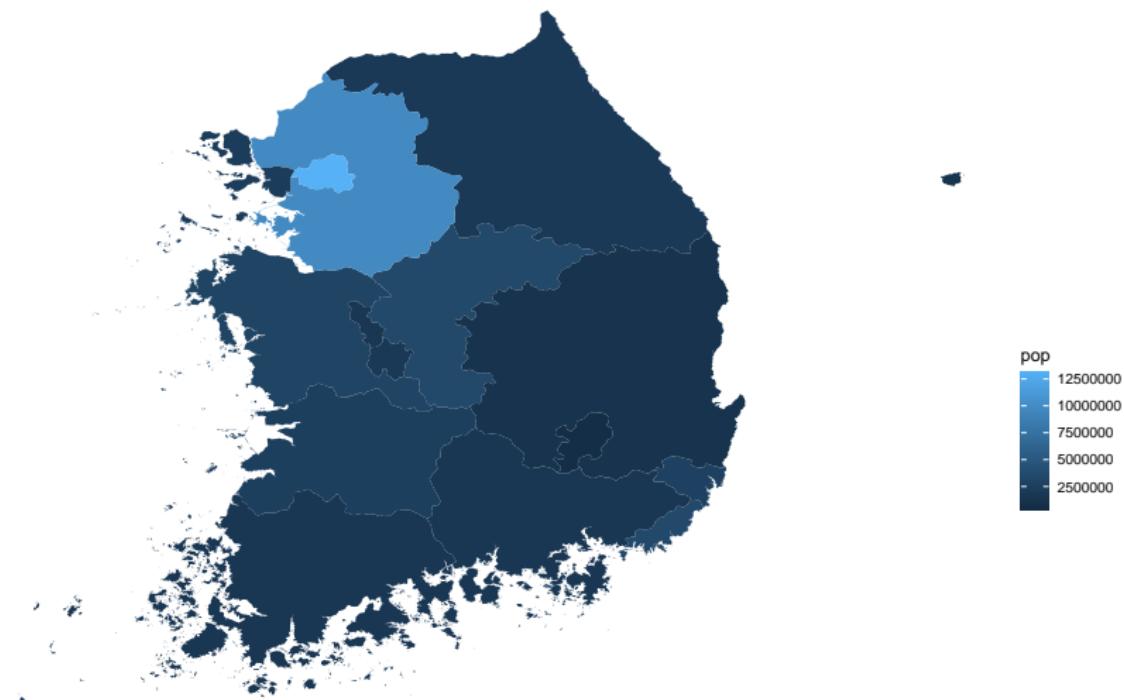
III. Choropleth Map 그리기

IV. Some Improvements - colors

V. Some Improvements

*theme_void()*로 위경도 표시 없애기

```
fig1 + theme_void()
```



I. Choropleth Map (단계도)

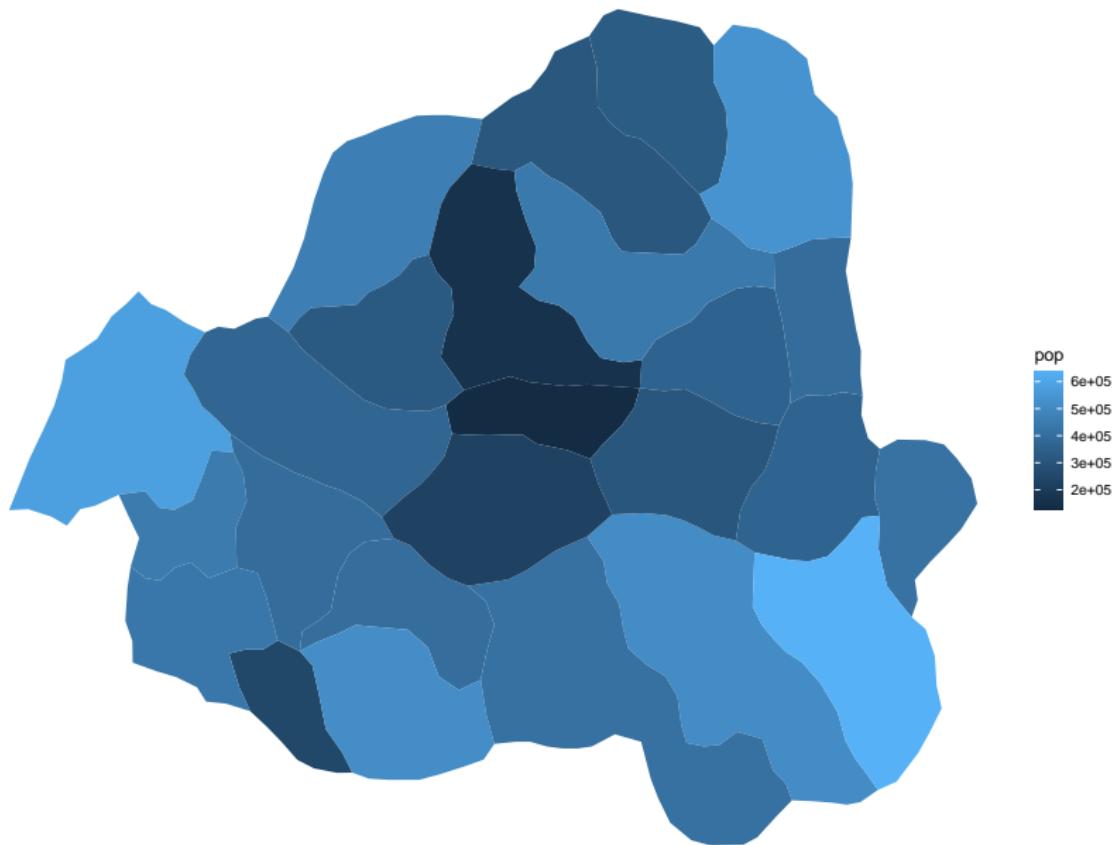
I. Choropleth Map (단계구분도) II. 지역 정보를 포함한 최종 데이터 셋 준비

III. Choropleth Map 그리기

!! IV. Some Improvements - colors

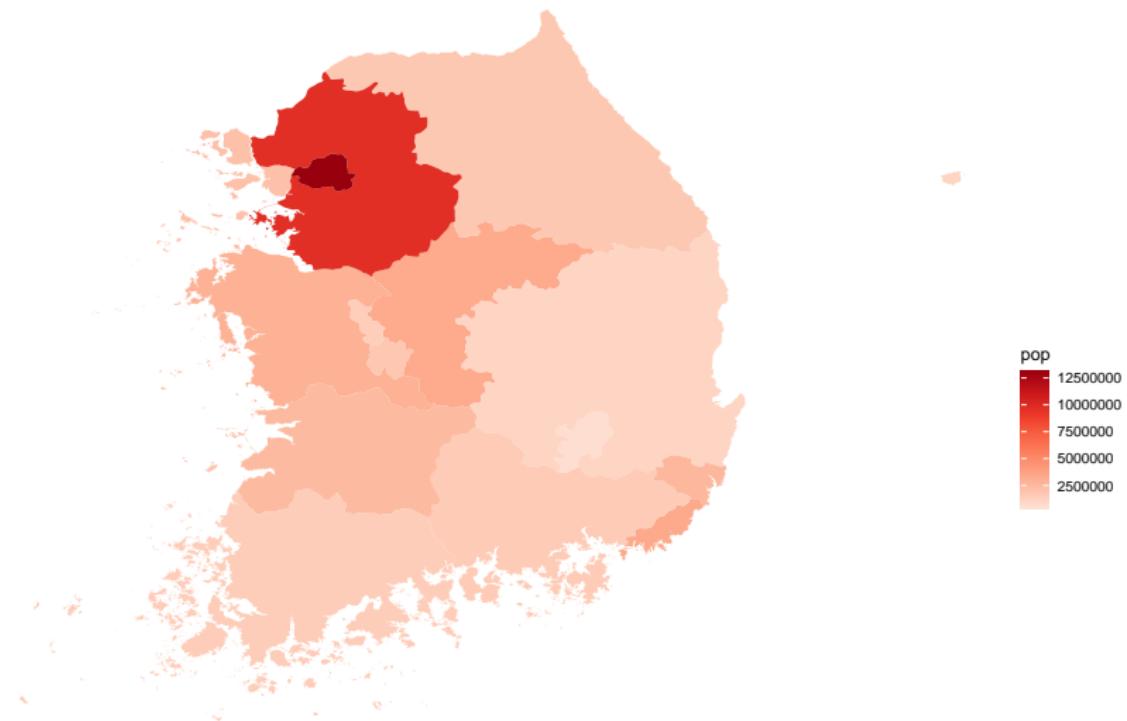
V. Some Improvements

fig2 + theme_void()



`scale_fill_distiller()` - 단일 color의 alpha

```
fig1 + theme_void() + scale_fill_distiller(palette = "Reds", direction = 1)
```



I. Choropleth Map (단계구분도)
oooooooooooo

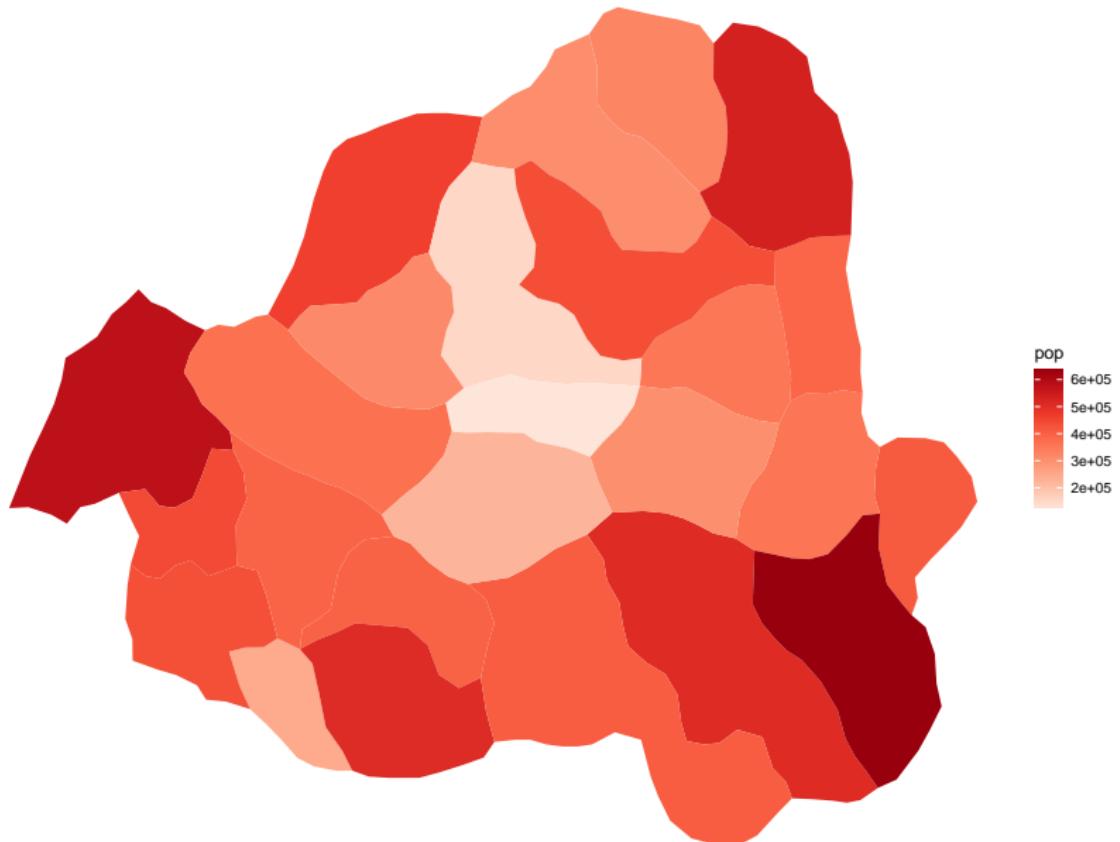
II. 지역정보를 포함한 최종 데이터 셋 준비
oooooooooooo

III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooo●oooooooo

V. Some Improvements
oooooooooooooooo

```
fig2 + theme_void() + scale_fill_distiller(palette = "Reds", direction = 1)
```



I. Choropleth Map (단계구분도)
oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

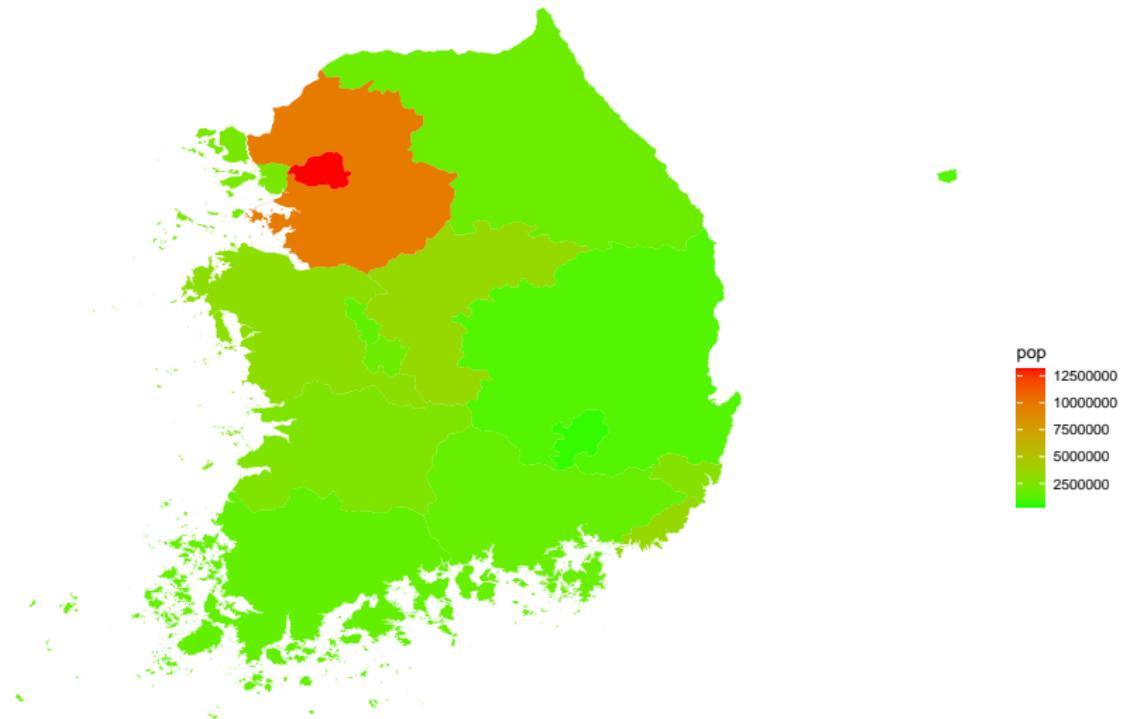
III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooo●oooo

V. Some Improvements
oooooooooooo

scale_fill_gradient() - 두 가지 색상

```
fig1 + theme_void() + scale_fill_gradient(low="green", high="red")
```



I. Choropleth Map (단계구분도)
oooooooooooo

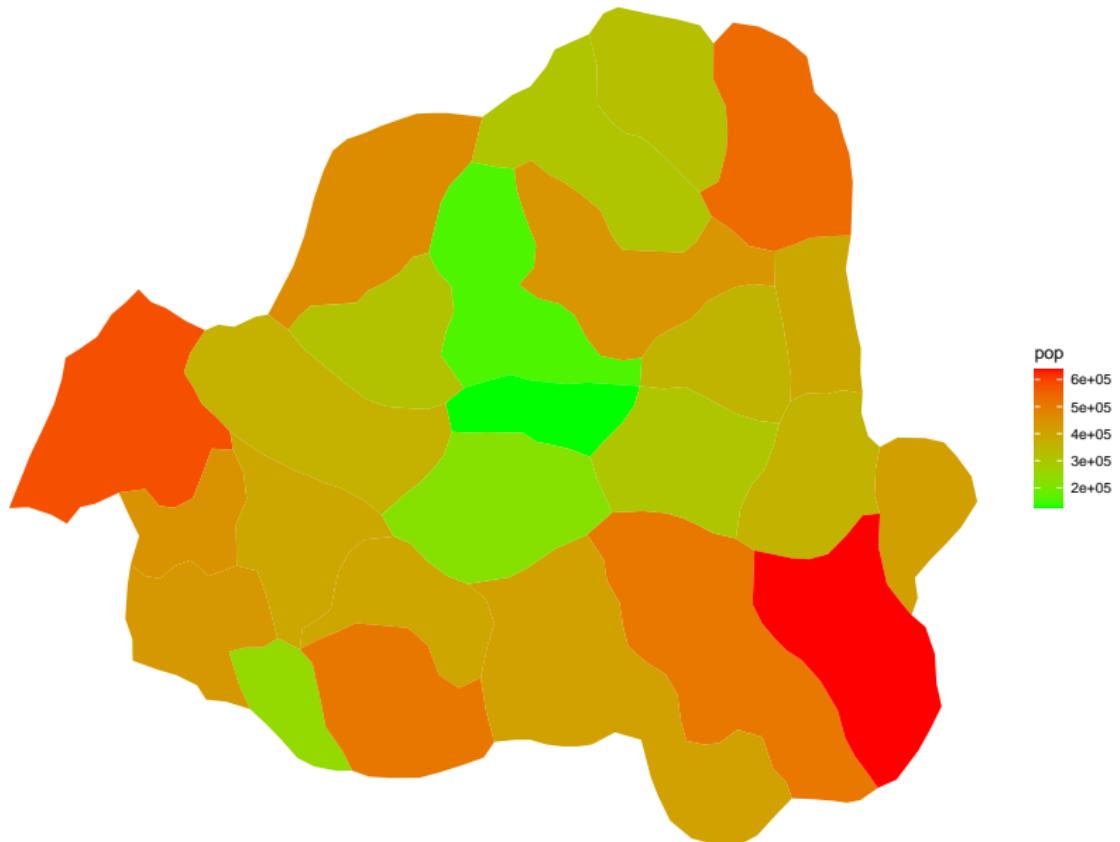
II. 지역정보를 포함한 최종 데이터 셋 준비
oooooooooooo

III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooooooo●oooo

V. Some Improvements
oooooooooooooooo

```
fig2 + theme_void() + scale_fill_gradient(low="green", high="red")
```



I. Choropleth Map (단계구분도)
oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

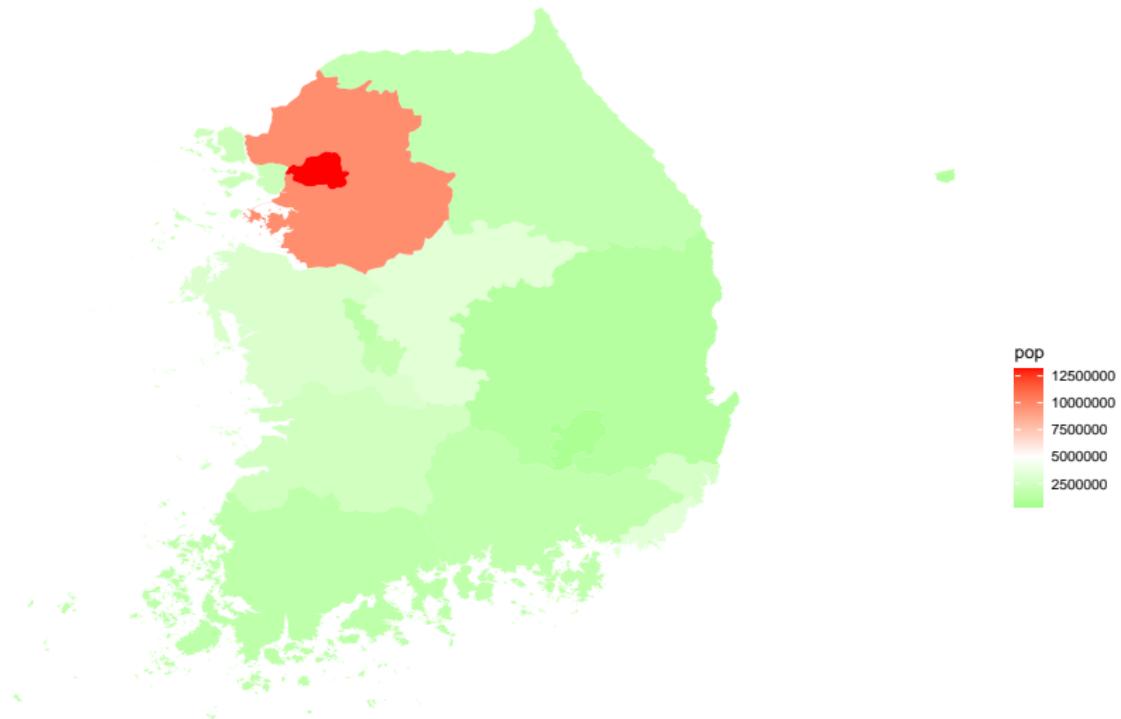
III. Choropleth Map 그리기!
○○○

IV. Some Improvements - colors
oooooooo●○○○

V. Some Improvements
oooooooooooooooo

scale_fill_gradient2() - 세 가지 색상

```
fig1 + theme_void() + scale_fill_gradient2(low="green", high="red", mid = "white", midpoint = 50)
```



I. Choropleth Map (단계구분도)
oooooooooooo

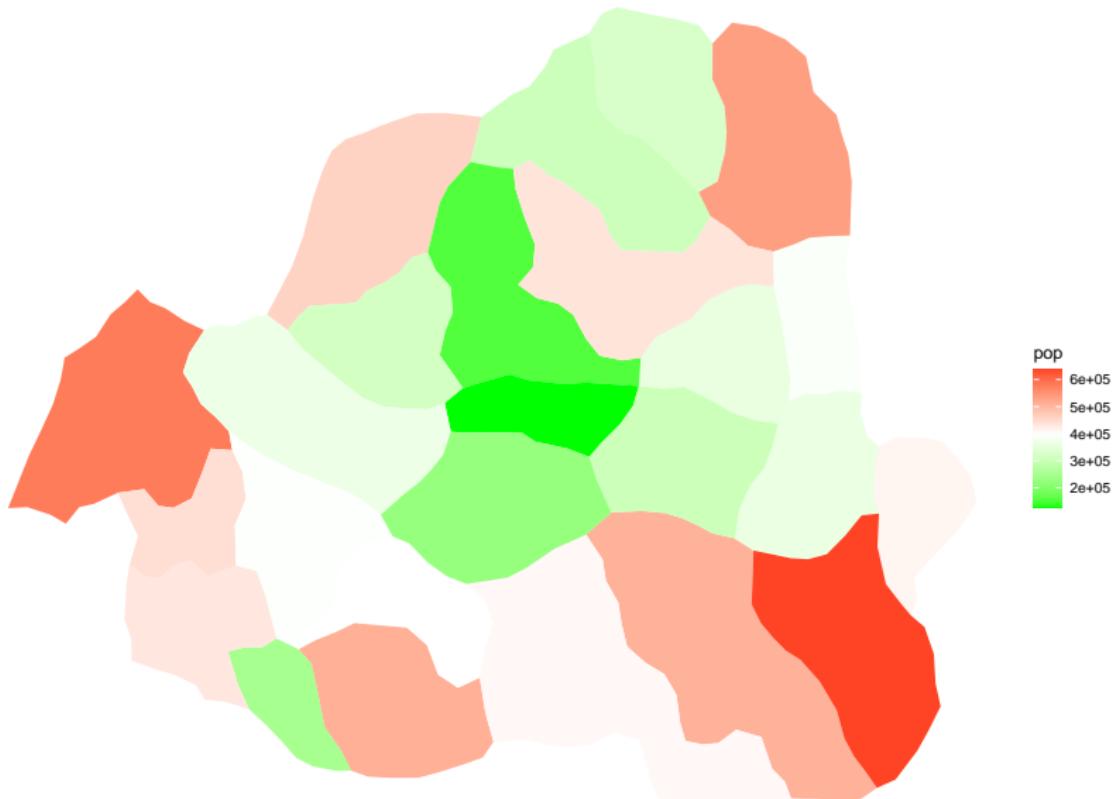
II. 지역정보를 포함한 최종 데이터 셋 준비
oooooooooooo

III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooooooo●ooo

V. Some Improvements
oooooooooooo

```
fig2 + theme_void() + scale_fill_gradient2(low="green", high="red", mid = "white", midpoint = 400000)
```



I. Choropleth Map (단계구분도)
oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

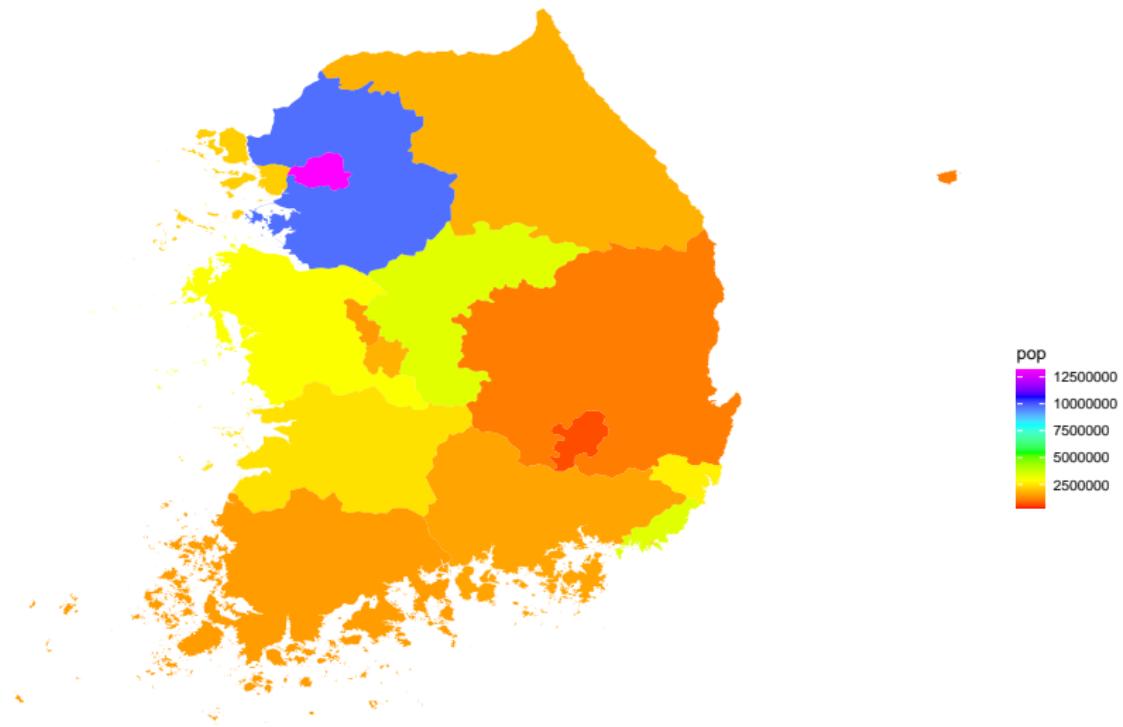
III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooooooo●○

V. Some Improvements
oooooooooooo

scale_fill_gradientn() - multiple 색상, 효용은?

```
fig1 + theme_void() + scale_fill_gradientn(colours=rainbow(6))
```



I. Choropleth Map (단계구분도)
oooooooooooo

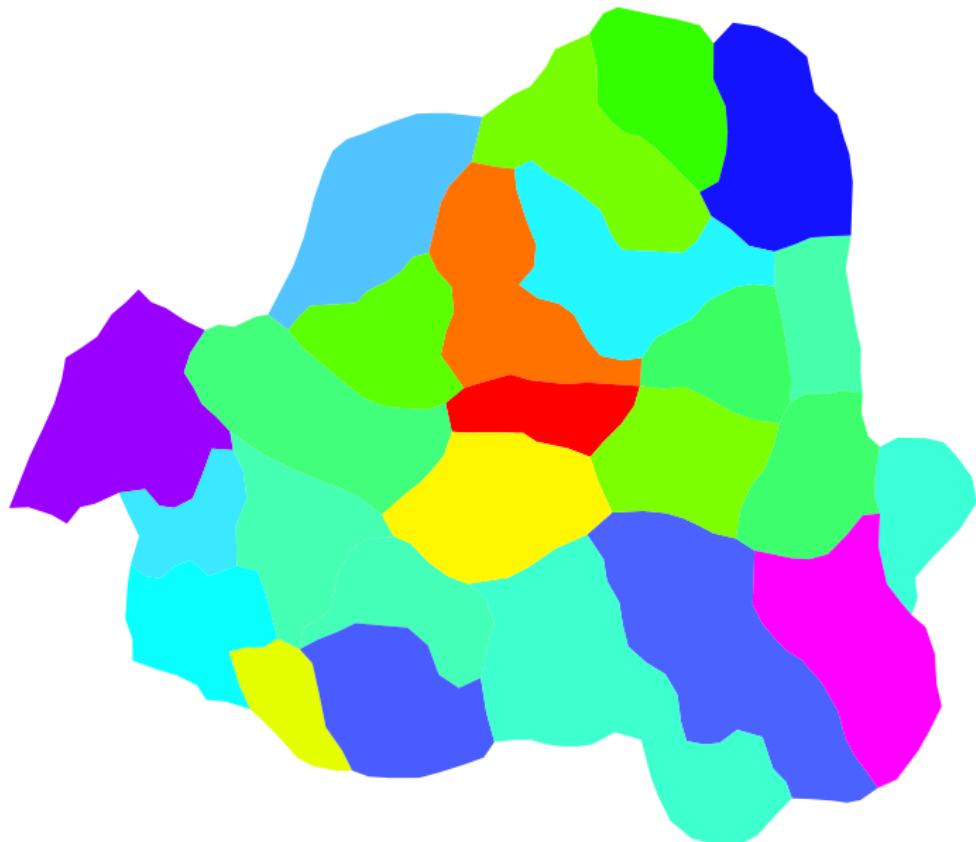
II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

III. Choropleth Map 그리기!
ooo

IV. Some Improvements - colors
oooooooooooo●

V. Some Improvements
oooooooooooo

```
fig2 + theme_void() + scale_fill_gradientn(colours=rainbow(6))
```



I. Choropleth Map (단계구분도)
○○○○○○○○○○○○

II. 지역정보를 포함한 최종 데이터 셋 준비
○○○○○○○○○○○○

III. Choropleth Map 그리기!
○○○

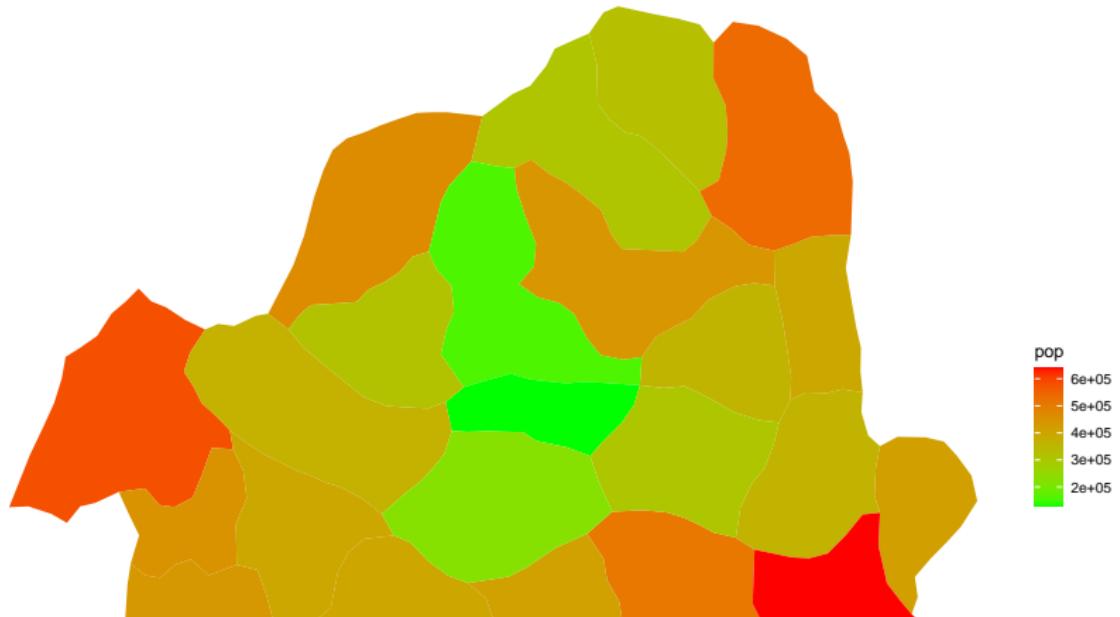
IV. Some Improvements - colors
○○○○○○○○○○○○

V. Some Improvements
●○○○○○○○○○○○○

V. Some Improvements - add text

Base

```
seoul_map_df %>% arrange(group, order) %>%
  ggplot(aes(x=long, y=lat, group=group)) +
  geom_polygon(aes(fill = pop)) + # fig2 so far
  theme_void() + scale_fill_gradient(low="green", high="red")
```



○○○○○○○○○○○○

○○○○○○○○○○○○

○○○

○○○○○○○○○○○○

○○●○○○○○○○○○○

Recreate the minimal example in L15.p14

```
kr_latlon <- read_csv(
  "data/kr_latlon.csv",
  locale = locale('ko', encoding='euc-kr'))
seoul_pop_2018 <-
  left_join(
    seoul_pop_2018,
    kr_latlon,
    by=c("district"="area"))
```

```
library(ggmap)
ggmap(seoul_map) +
  geom_text(
    data = seoul_pop_2018,
    aes(x = lon, y = lat, label=pop),
    size = 5, color = "blue")
```



Add annotation to choropleth

```
ggplot(seoul_map_df %>% arrange(group, order)) +  
  geom_polygon(aes(x=long, y=lat, group=group, fill = pop)) + # be careful with inheritance  
  theme_void(base_family='NanumGothic') + scale_fill_gradient(low="green", high="red") +  
  # ggmap(seoul_map) + # no need, because we will text on choropleth!  
  geom_text(  
    data = seoul_pop_2018,  
    aes(x = lon, y = lat, label=pop),  
    size = 5, color = "blue")
```

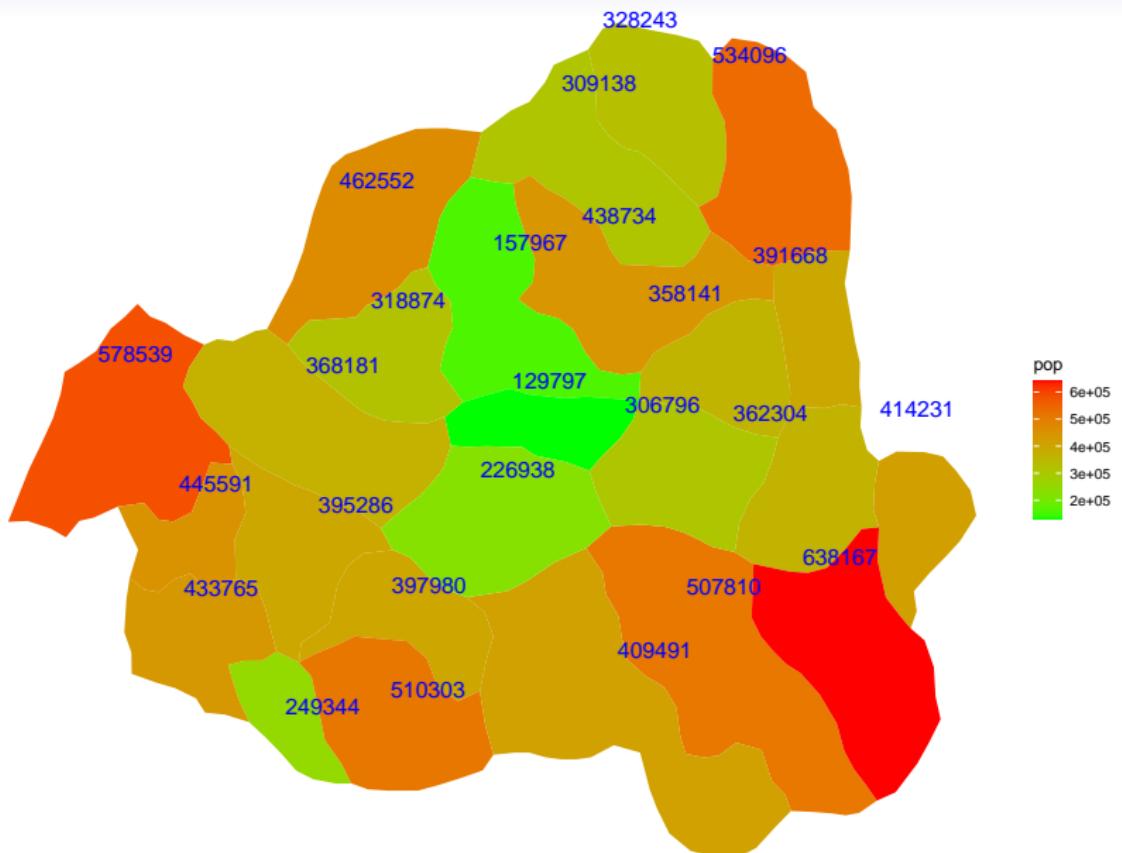
I. Choropleth Map (단계구분도) II

3. 지역정보를 포함한 최종 데이터 셋 준비

III. Choropleth Map 그리기! IV

7. Some Improvements - colors

V. Some Improvements



I. Choropleth Map (단계구분도)
oooooooooooo

II. 지역정보를 포함한 최종 데이터셋 준비
oooooooooooo

III. Choropleth Map 그리기!
ooo

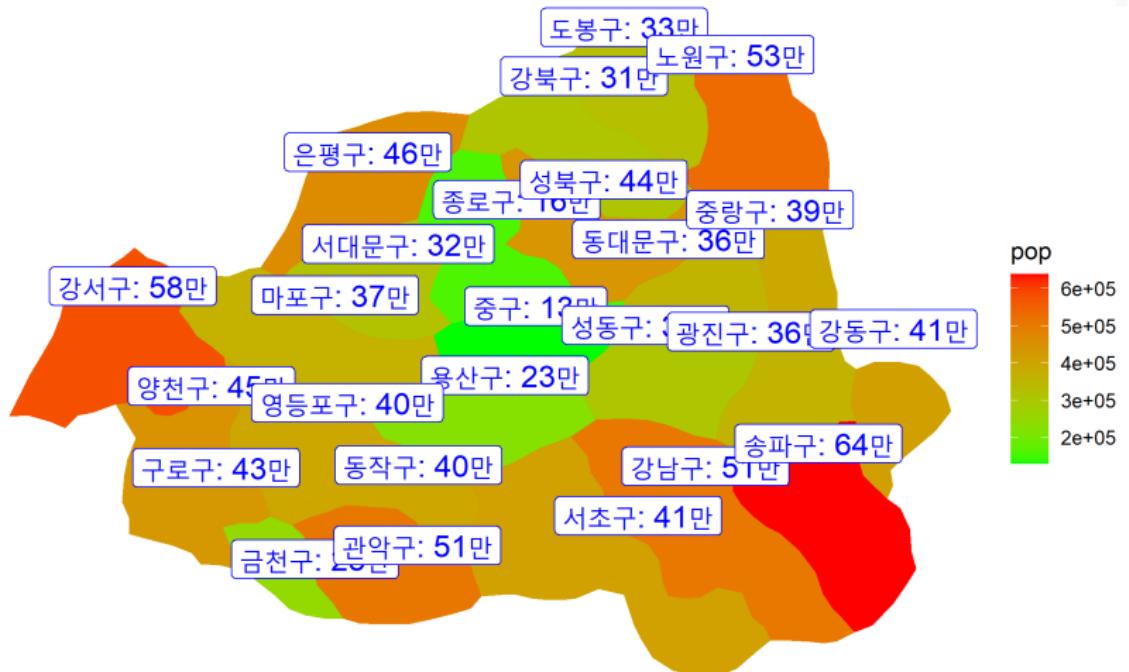
IV. Some Improvements - colors
oooooooooooo

V. Some Improvements
oooo●oooooooo

A little better?

```
fig_both <-  
  ggplot(seoul_map_df %>% arrange(group, order)) +  
  geom_polygon(aes(x=long, y=lat, group=group, fill = pop)) + # be careful with inheritance  
  theme_void() + scale_fill_gradient(low="green", high="red") +  
  geom_label(  
    data = seoul_pop_2018,  
    aes(x = lon, y = lat, label=paste0(district, ":", round(pop/10^4), "만")),  
    size = 5, color = "blue")  
ggsave(fig_both, filename = "fig/fig_both.png")
```

```
include_graphics("fig/fig_both.png")
```



Let it include all years

```
# 1. Prepare `seoul_map_df`
library(raster)
kr_map_lvl2 <- getData('GADM', country='kor', level=2)
seoul_map_lvl2 <- kr_map_lvl2[kr_map_lvl2$NAME_1 == "Seoul",]
seoul_map_df <- fortify(seoul_map_lvl2)
seoul_map_df <- seoul_map_df %>% left_join(
  seoul_map_lvl2@data %>% # retrieve ID from @polygon
  mutate(id = sapply(seoul_map_lvl2@polygons, function(x) x@ID)) %>%
  separate(NL_NAME_2, into= c("district", "dummy1"), sep="|") %>%
  dplyr::select(-dummy1)
)

# 2. Prepare `seoul_pop` for all years
seoul_pop <- read_csv("data/seoul_pop_tidy.csv", locale = locale('ko', encoding='euc-kr'))
seoul_pop$district <- str_trim(seoul_pop$district)
seoul_pop <- seoul_pop %>%
  filter(category == "총인구 (명)") %>%
  dplyr::select(year, district, pop)
```

```
# 3. Let `seoul_map_df` include `seoul_pop`
seoul_map_df <- seoul_map_df %>% left_join(seoul_pop)
seoul_map_df %>%
  dplyr::select (long, lat, order, hole, group, district, year, pop) %>%
  head(3)

##       long      lat order  hole   group district year    pop
## 1 127.0617 37.65409     1 FALSE 182873.1  도봉구 2015 340095
## 2 127.0617 37.65409     1 FALSE 182873.1  도봉구 2016 336745
## 3 127.0617 37.65409     1 FALSE 182873.1  도봉구 2017 332586
```

I. Choropleth Map (단계구분도)
○○○○○○○○○○○○

II. 지역정보를 포함한 최종 데이터셋 준비
○○○○○○○○○○○○

III. Choropleth Map 그리기!
○○○

IV. Some Improvements - colors
○○○○○○○○○○○○

V. Some Improvements
○○○○○○○○●○○

4. Finally

```
fig_all_year <- ggplot(seoul_map_df %>% arrange(group, order)) +  
  geom_polygon(aes(x=long, y=lat, group=group, fill = pop)) +  
  theme_void() + scale_fill_gradient (low="green", high="red") +  
  geom_text(data = left_join(seoul_pop, kr_latlon, by = c("district"="area"))),  
            aes(x = lon, y = lat, label= paste0(district, ":", round(pop/10^4,1), "만")),  
            size = 3, color = "blue") +  
  facet_wrap(~year)  
  
ggsave(fig_all_year, filename = "fig/fig_all_year.png")
```

I. Choropleth Map (단계구분도)

○○○○○○○○○○○○

II. 지역정보를 포함한 최종 데이터셋 준비

○○○○○○○○○○○○

III. Choropleth Map 그리기!

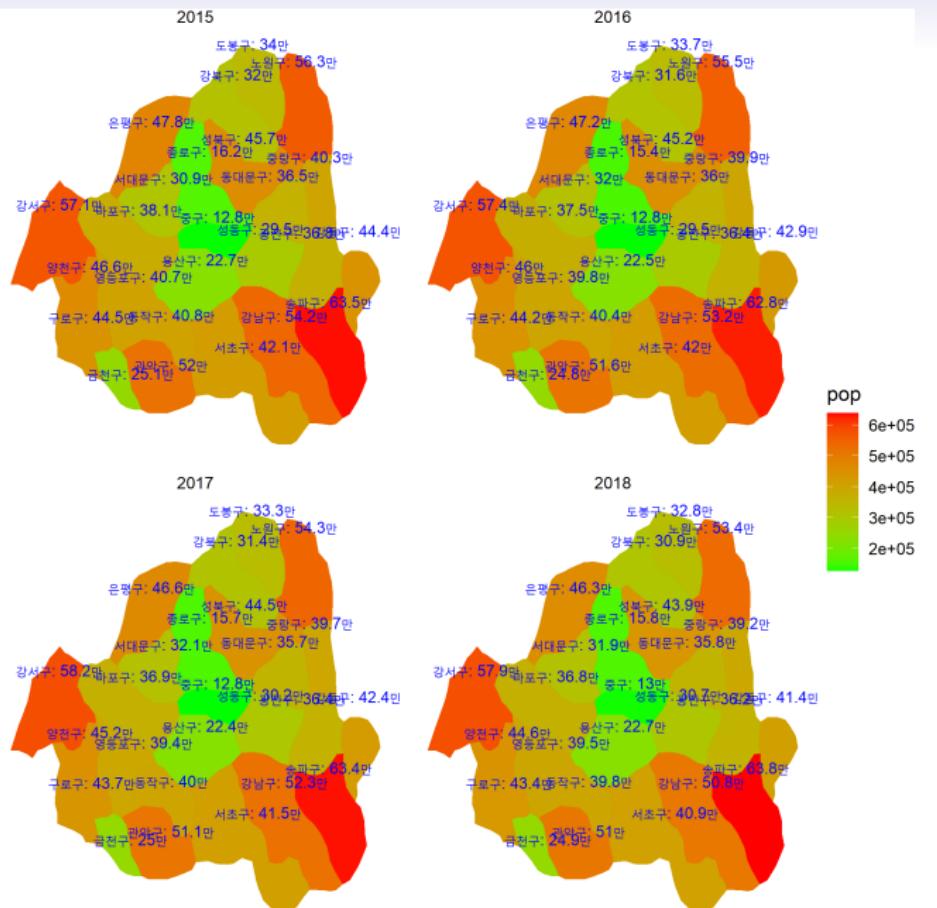
○○○

IV. Some Improvements - colors

○○○○○○○○○○○○

V. Some Improvements

○○○○○○○○○○○○●●



I. Choropleth Map (단계구분도) II. 지역정보를 포함한 최종 데이터셋 준비 III. Choropleth Map 그리기! IV. Some Improvements - colors V. Some Improvements
oooooooooooo oooooooooooo ooo oooooooooooo oooooooooooo●

"Hello"

```
## [1] "Hello"
```