

# GGMAP을 이용한 데이터시각화

박지은, 오민석, 김현주, 김봉석

# 목차

1 ggmap 시작하기: API 키 사용하기

2 위치정보 찾기: geocode

3 위치정보를 활용한 시각화와 실습



1

GGMAP 시작하기: API 키 사용하기

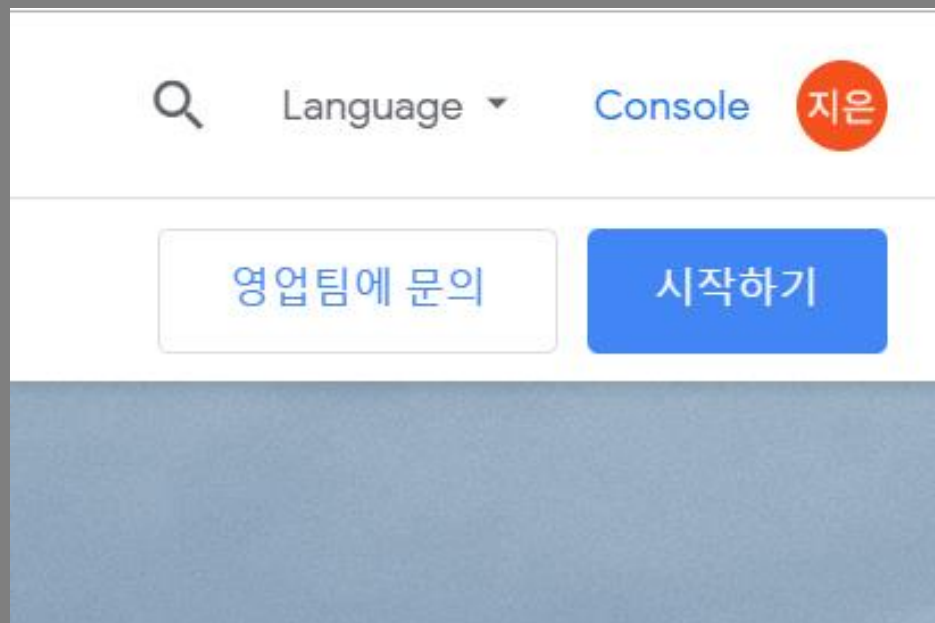
# Google Maps Platform

## 1. 로그인하기

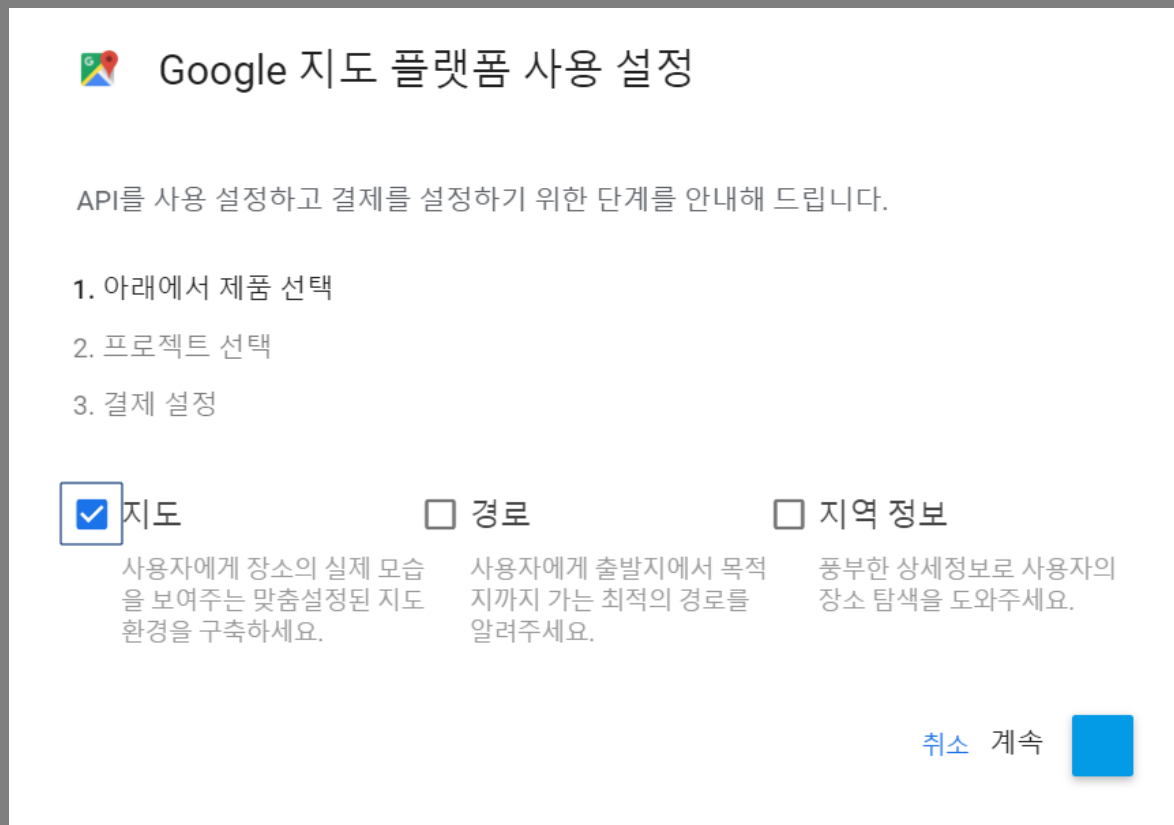


1

## GGMAP 시작하기: API 키 사용하기



2. 오른쪽 상단 시작하기  
버튼 클릭!



3. 지도 클릭! > 계속

1

GGMAP 시작하기: API 키 사용하기

## 4. NEXT



### Enable Google Maps Platform

To enable APIs or set up billing, we'll guide you through a few tasks:

1. Pick product(s) below
- 2. Select a project**
3. Set up your billing

Projects allow you to use APIs, add collaborators, and manage permissions.

Enter new project name

My Project

BACK

CANCEL

NEXT

1

## GGMAP 시작하기: API 키 사용하기

프로젝트 'My Project 806'의 결제를 사용 설정할 수 있습니다.

귀하는 결제 계정의 관리자가 아닙니다. 이 프로젝트에서 결제를 사용 설정하려면 새 결제 계정을 만들거나 결제 계정 관리자에게 문의하여 결제를 사용 설정해 달라고 요청하세요. [자세히 알아보기](#)

[취소](#) [결제 계정 만들기](#)

### 5. 결제 계정 만들기

1

## GGMAP 시작하기: API 키 사용하기

## 6. 정보 입력하기

Google Cloud Platform 무료로 사용해 보기

1/2단계

국가

대한민국

서비스 약관

☒ Google Cloud Platform 무료 평가판 서비스 약관을 읽었으며 이에 동의합니다.

계속 진행하려면 체크박스를 선택하세요.

이메일 업데이트

Google Cloud 및 Google Cloud 파트너가 보내는 뉴스, 제품 업데이트, 특별 이벤트에 대한 정기적인 이메일을 수신하겠습니다.

☐ 예☒ 아니요

AGREE AND CONTINUE

## 모든 Cloud Platform 제품에 액세스

Firebase, Google Maps API 등 서비스를 구축하고 실행할 수 있습니다.

이름 및 주소

## \$300의 무료 크레딧

가입하여 Google Cloud Platform에서 사용할 수 있는 \$300 크레딧을 받습니다.

결제 옵션

자동 결제

비용이 발생한 후에만 서비스를 결제합니다. 청구 기준일에 도달하거나 지난 자동 결제일로부터 30일이 경과하면 다음 달부터 이 금액에 비용이 자동 청구됩니다.

## 무료 평가판 종료 후

신용카드를 요청하는 이유입니다. 사용자가 유료로 전환하지 않는 한 요금이 청구되지 않습니다.

결제 수단

카드 번호

#

MM

YY

CVC

카드 소유자 이름

-

-

-

☒ 신용카드 또는 체크카드 주소가 위의 주소와 동일합니다.

무료 평가판 시작하기

## 모든 Cloud Platform 제품에 액세스

Firebase, Google Maps API 등을 포함해 앱, 웹사이트, 서비스를 구축하고 실행하는 데 필요한 모든 기능을 이용할 수 있습니다.

## \$300의 무료 크레딧

가입하여 Google Cloud Platform에서 12개월간 사용할 수 있는 \$300 크레딧을 받아 보세요.

## 무료 평가판 종료 후 자동 청구되지 않음

신용카드를 요청하는 이유는 자동 가입을 방지하기 위해서입니다. 사용자가 유료 계정으로 직접 업그레이드하지 않는 한 요금이 청구되지 않습니다.

1

## GGMAP 시작하기: API 키 사용하기



### Google Maps Platform 사용 설정

#### API 사용 설정

Google Maps Platform API 7개가 사용 설정되고 구현을 위한  
성됩니다.

## 7. API 사용 설정



### Google Maps Platform 사용 설정

You're all set!

You're ready to start developing!

YOUR API KEY

AIzaSyAzf9Cxm39UyIchhIBPIdmSvnqav81-yil



To improve your app's security, restrict this key's usage in the [API Console](#).

DONE



## 1

## GGMAP 시작하기: API 키 사용하기

```
library('ggmap')
register_google(key='AIzaSyDglGuNAZE7fzqEr6er5EObrs_vDIUZGb8')
names <- c("용두암", "성산일출봉", "정방폭포",
           "중문관광단지", "한라산1100고지", "차귀도")
addr <- c("제주시 용두암길 15",
          "서귀포시 성산읍 성산리",
          "서귀포시 동홍동 299-3",
          "서귀포시 중문동 2624-1",
          "",
          "서귀포시 섹달동 산1-2",
          "제주시 한경면 고산리 125")
gc <- geocode(enc2utf8(addr))

gc
# A tibble: 6 x 2
#   lon lat
#   <dbl> <dbl>
# 1 127. 33.5
# 2 127. 33.5
# 3 127. 33.3
# 4 126. 33.3
# 5 126. 33.4
# 6 126. 33.3

df <- data.frame(name=names,
                  lon=gc$lon,
                  lat=gc$lat)
cen <- c(mean(df$lon), mean(df$lat))
map <- get_googlemap(center=cen,
                     maptype="roadmap",
                     zoom=10,
                     size=c(640,640),
                     marker=gc)

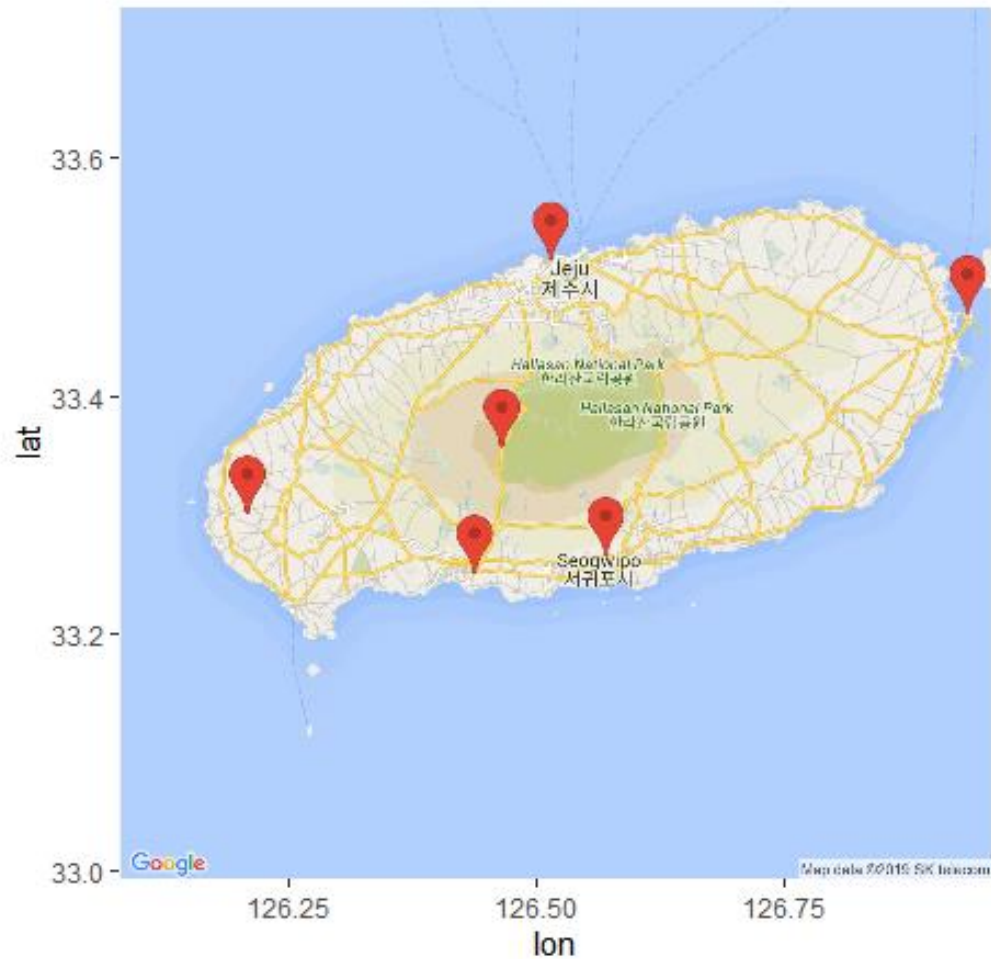
ggmap(map)
```

## 8. 코드 실행해보기

1

## GGMAP 시작하기: API 키 사용하기

이 결과가 나오면 완성!

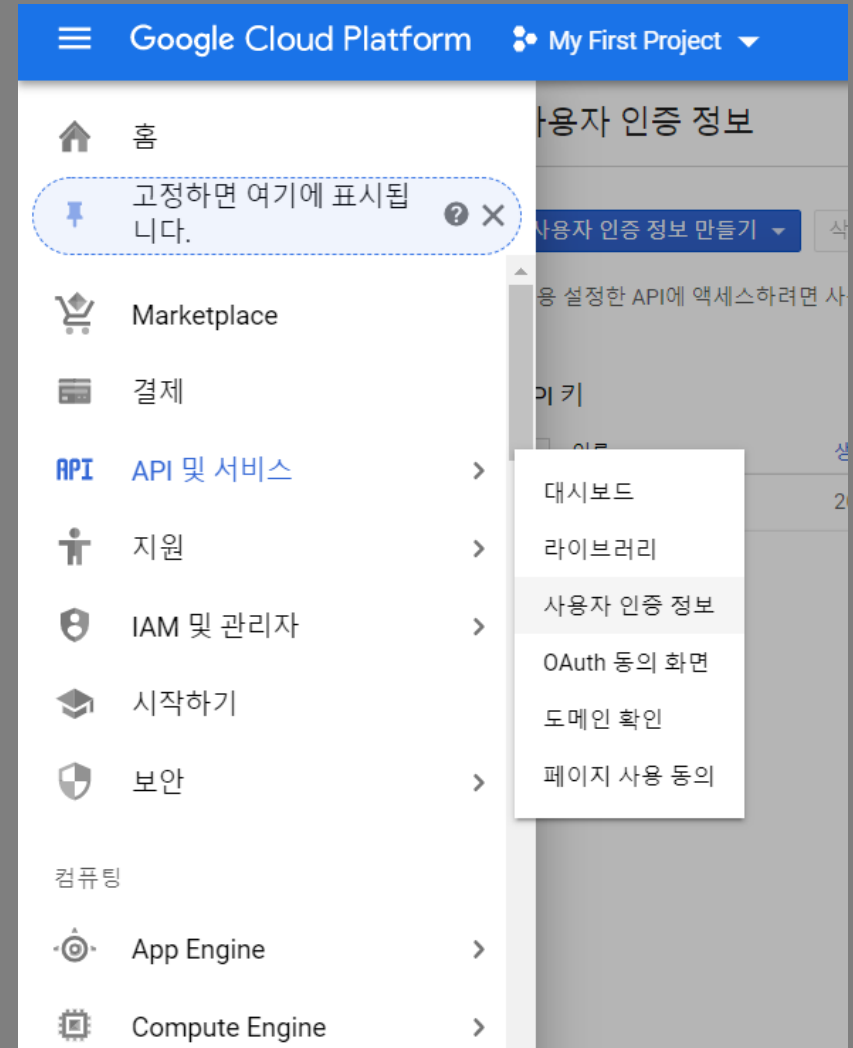
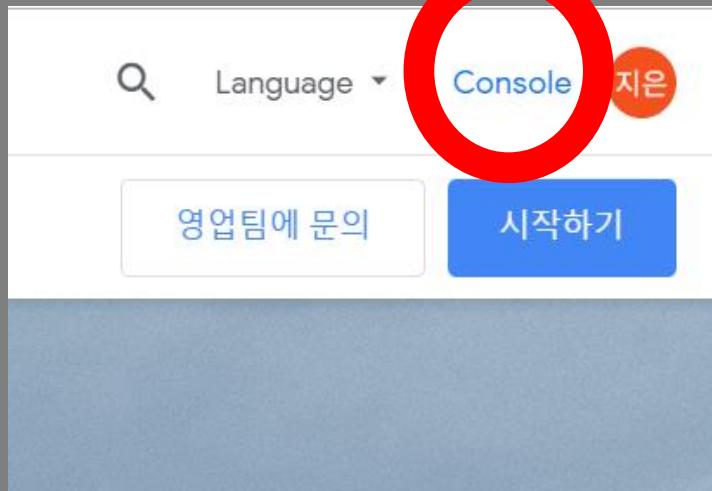


1

## GGMAP 시작하기: API 키 사용하기

### < API 키 확인하고 싶을 때! >

1. Console 창 들어가기
2. 메뉴 바에서 API 및 서비스
3. 사용자 인증 정보



1

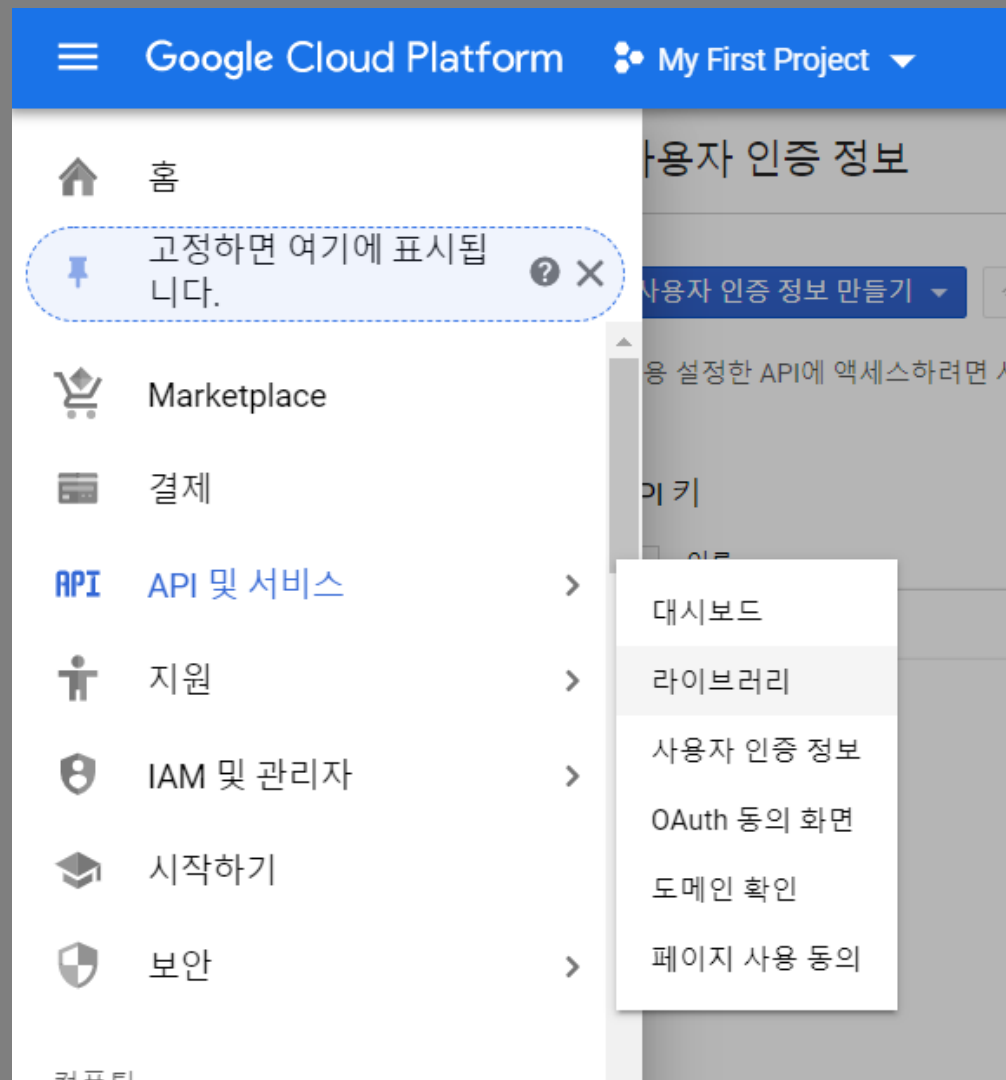
## GGMAP 시작하기: API 키 사용하기

Q. 지금까지 했는데 안됩니다!!

1

## GGMAP 시작하기: API 키 사용하기

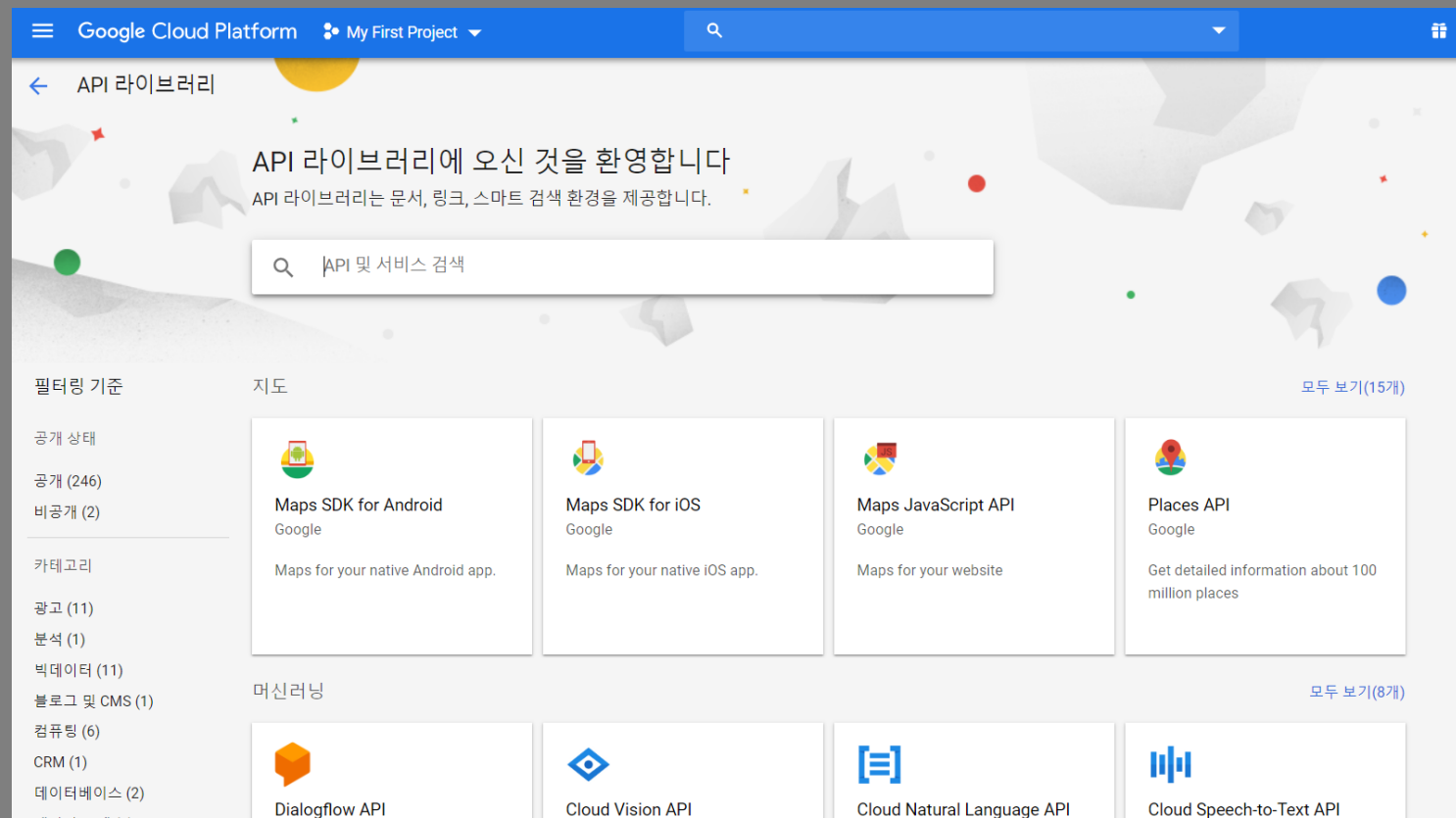
1. 메뉴 바에서 API 및 서비스
2. 라이브러리



1

## GGMAP 시작하기: API 키 사용하기

## 3. 라이브러리 창 &gt; 카테고리에서 '지도'선택



1

## GGMAP 시작하기: API 키 사용하기


### 4. 최대한 많이 깔아주세요

필터링 기준

카테고리


지도 ✕

검색결과 15개




**Directions API**  
Google

Directions between multiple locations.




**Distance Matrix API**  
Google

Travel time and distance for multiple destinations.




**Geocoding API**  
Google

Convert between addresses and geographic coordinates.




**Geolocation API**  
Google

Location data from cell towers and WiFi nodes.




**Maps Elevation API**  
Google

Elevation data for any point in the world.




**Maps Embed API**  
Google

Make places easily discoverable with interactive Google Maps.




**Maps JavaScript API**  
Google

Maps for your website




**Maps SDK for Android**  
Google


Maps for your native Android app.




**Maps SDK for iOS**



**Maps Static API**



**Places API**



**Roads API**

1

GGMAP 시작하기: API 키 사용하기

## 5. 사용설정 클릭하기



### Time Zone API

Google

Time zone data for anywhere in the world.

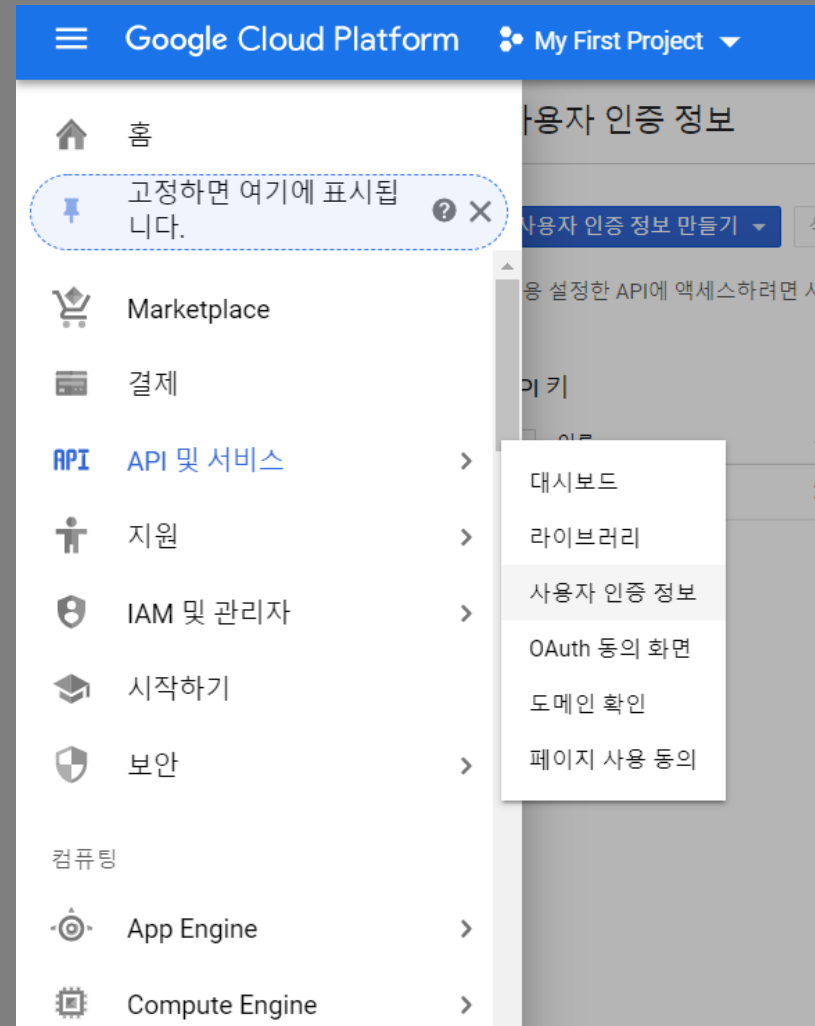
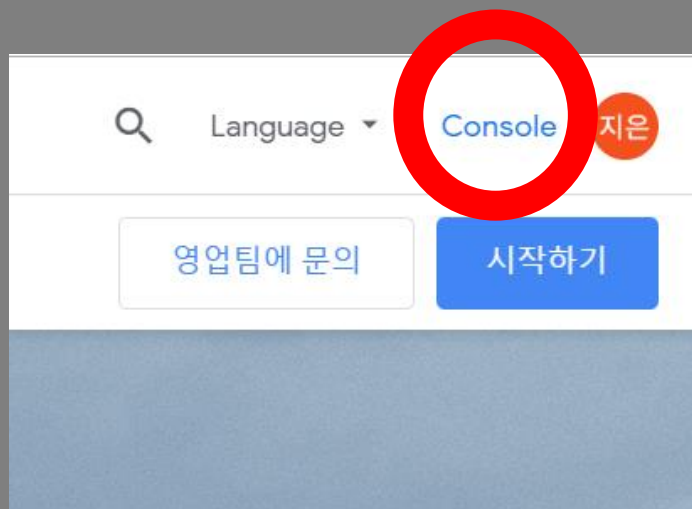
사용 설정



1


## GGMAP 시작하기: API 키 사용하기

6. Console 창 들어가기
7. 메뉴 바에서 API 및 서비스
8. 사용자 인증 정보



1

## GGMAP 시작하기: API 키 사용하기

API 키			
<input type="checkbox"/> 이름	생성일 ▾	제한사항	키
<input type="checkbox"/> API 키 1	2019. 9. 11.	API 20개	AlzaSyDglGuNAZE7fzqEr6er5EObrs_vDIUZGb8 

9. API 키 오른쪽 끝의 편집 아이콘 클릭

1

## GGMAP 시작하기: API 키 사용하기

## 10 . API 제한사항 -&gt; 키제한

아래에 API 선택!  
방금 사용활성화 한 기능들 모두 선택

## API 제한사항

API 제한사항은 이 키를 호출할 수 있는 사용 설정된 API를 지정합니다.

- ☐ 키 제한 안함  
이 키는 모든 API를 호출할 수 있습니다.
- ☒ 키 제한

ⓘ Google Places API에는 아직 API 제한사항이 적용되지 않습니다. 이 키에서 해당 API의 사용을 제한하려면 API를 사용 중지하세요.

API 20개 ▼

## 선택한 API:

Stackdriver Trace API  
Stackdriver Monitoring API  
Stackdriver Logging API  
Stackdriver Debugger API  
Service Usage API  
Service Management API  
Google Cloud Storage JSON API  
Google Cloud APIs  
Cloud Storage  
Cloud SQL  
Cloud Datastore API  
BigQuery Storage API

## 1

## GGMAP 시작하기: API 키 사용하기

```
library('ggmap')
register_google(key='AIzaSyDg1GuNAZE7fzqEr6er5EObrs_vDIUZGb8')
names <- c("용두암", "성산일출봉", "정방폭포",
           "중문관광단지", "한라산1100고지", "차귀도")
addr <- c("제주시 용두암길 15",
          "서귀포시 성산읍 성산리",
          "서귀포시 동홍동 299-3",
          "서귀포시 중문동 2624-1",
          "서귀포시 섹달동 산1-2",
          "제주시 한경면 고산리 125")
gc <- geocode(enc2utf8(addr))

gc
# A tibble: 6 x 2
#   lon lat
#   <dbl> <dbl>
# 1 127.  33.5
# 2 127.  33.5
# 3 127.  33.3
# 4 126.  33.3
# 5 126.  33.4
# 6 126.  33.3

df <- data.frame(name=names,
                  lon=gc$lon,
                  lat=gc$lat)
cen <- c(mean(df$lon), mean(df$lat))
map <- get_googlemap(center=cen,
                     maptype="roadmap",
                     zoom=10,
                     size=c(640,640),
                     marker=gc)

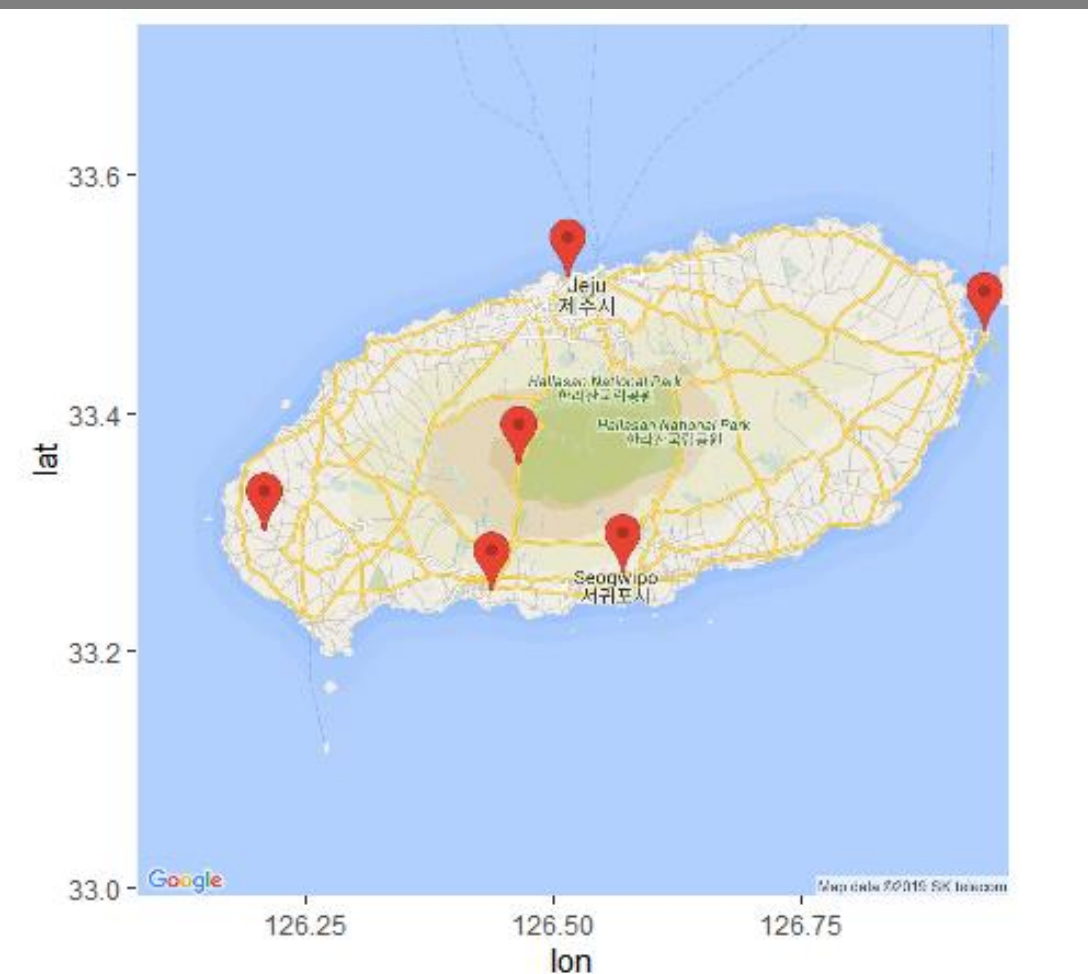
ggmap(map)
```

11. 아까 코드 재실행

1

## GGMAP 시작하기: API 키 사용하기

이 결과가 나오면 완성!



# Let's take a break!

API 때문에 고생하셨어요ㅠㅠ 잠시 쉬었다해요!

2

## 위치정보 찾기: geocode

```
gc
# A tibble: 6 x 3
#   lon lat
#   <dbl> <dbl>
# 1 127.  33.5
# 2 127.  33.5
# 3 127.  33.3
# 4 126.  33.3
# 5 126.  33.4
# 6 126.  33.3
```

lon: longitude (위도) - 세로

Lat: latitude (경도) - 가로

2

## 위치정보 찾기: geocode

```
gc
# A tibble: 6 x 2
#   lon lat
#   <dbl> <dbl>
# 1 127. 33.5
# 2 127. 33.5
# 3 127. 33.3
# 4 126. 33.3
# 5 126. 33.4
# 6 126. 33.3
```

class가 tibble로만 보이는 게 아니라  
dataframe까지 뜨는 것을 확인할 수 있음

```
> class(gc)
[1] "tbl_df"      "tbl"        "data.frame"
```



## 2 위치정보 찾기: geocode

### tibble vs. dataframe

```
> str(gc)
Classes 'tbl_df', 'tbl' and 'data.frame':      6 obs. of  2 variables:
 $ lon: num  127 127 127 126 126 ...
 $ lat: num  33.5 33.5 33.3 33.3 33.4 ...
```

```
> class(iris)
[1] "data.frame"
> heae(iris)
Error in heae(iris) : could not find function "heae"
> class(iris)
[1] "data.frame"
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4          0.2  setosa
2          4.9         3.0          1.4          0.2  setosa
3          4.7         3.2          1.3          0.2  setosa
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> as.data.frame(gc)
   lon   lat
1 126.5117 33.51496
2 126.9324 33.46993
3 126.5672 33.26632
4 126.4348 33.25143
5 126.4625 33.35777
6 126.2057 33.30110
```

```
> gc
# A tibble: 6 x 2
   lon   lat
<dbl> <dbl>
1  127.  33.5
2  127.  33.5
3  127.  33.3
4  126.  33.3
5  126.  33.4
6  126.  33.3
```

## 2

## 위치정보 찾기: geocode

tibble은 dataframe을 좀 더 편하게 (한눈에 알아보기 쉽게) 변형시킨 새로운 데이터 형식(class), dataframe과 호환.

dataframe과의 차이점)

- dataset의 character가 factor로 변하지 않음
- R에서 허용하지 않는 형태의 변수이름(e.g. ':') 사용가능
- Printing시 모두 출력하는 dataframe과는 다르게 처음 10줄과 화면에 맞는 변수만을 보여줌

```
> tb<-tibble(
+   `:)` = "smile",
+   ` ` = "space",
+   `2000` = "숫자"
+ ); tb
# A tibble: 1 x 3
#   `:)` ` ` `2000`
#   <chr> <chr> <chr>
1 smile space 숫자
```

```
> as_tibble(iris)
# A tibble: 150 x 5
#   Sepal.Length Sepal.Width Petal.Length Petal.Width
#   <dbl>         <dbl>         <dbl>         <dbl>
1         5.1           3.5           1.4           0.2
2         4.9           3           1.4           0.2
3         4.7           3.2           1.3           0.2
4         4.6           3.1           1.5           0.2
5         5           3.6           1.4           0.2
6         5.4           3.9           1.7           0.4
7         4.6           3.4           1.4           0.3
8         5           3.4           1.5           0.2
9         4.4           2.9           1.4           0.2
10        4.9           3.1           1.5           0.1
# ... with 140 more rows, and 1 more variable:
#   Species <fct>
```

2

## 위치정보 찾기: geocode

geocode(location="장소이름",output=c("latlon", "latlon", "more", "all") ...)

\*자기학교이름 넣어서 출력해보기!\*

```
> geocode("Ewha womans University","latlon")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Ewha+womans+University&key=xxx
# A tibble: 1 x 2
  lon lat
  <dbl> <dbl>
1 127. 37.6
```

output="latlon" : tibble 형식으로 위도와 경도 출력

2

## 위치정보 찾기: geocode

```
> geocode("Ewha womans University","latlon")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Ewha+womans+University&key=xxx
# A tibble: 1 x 3
  lon lat address
<dbl> <dbl> <chr>
1 127. 37.6 52 ewhayeodae-gil, daehyeon-dong, seodaemun-gu, seoul, south korea
```

output="latlon": tibble 형식으로 위도와 경도, 영어주소를 출력

2

## 위치정보 찾기: geocode

```
> geocode("Ewha womans University","more")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Ewha+Womans+University&key=xxx
# A tibble: 1 x 9
  lon lat type loctype address north south east west
  <dbl> <dbl> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>
1 127. 37.6 establishment rooftop 52 ewhayeodae-gil, daehyeon-dong, seodaemun-gu, seoul, south~ 37.6 37.6 127. 127.
```

output= "more": tibble 형식으로 위도와 경도, 주소 출력(방위와 국가범위까지)

## 2

## 위치정보 찾기: geocode

```
> geocode("Ewha Womans University","all")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Ewha+Womans+University&key=xxx
$results
$results[[1]]
$results[[1]]$address_components
$results[[1]]$address_components[[1]]
$results[[1]]$address_components[[1]]$long_name
[1] "52"

$results[[1]]$address_components[[1]]$short_name
[1] "52"

$results[[1]]$address_components[[1]]$types
$results[[1]]$address_components[[1]]$types[[1]]
[1] "premise"

$results[[1]]$address_components[[2]]
$results[[1]]$address_components[[2]]$long_name
[1] "Ewhayeodae-gil"

$results[[1]]$address_components[[2]]$short_name
[1] "Ewhayeodae-gil"

$results[[1]]$address_components[[2]]$types
$results[[1]]$address_components[[2]]$types[[1]]
[1] "political"

$results[[1]]$address_components[[2]]$types[[2]]
[1] "sublocality"

$results[[1]]$address_components[[2]]$types[[3]]
[1] "sublocality_level_4"

$results[[1]]$address_components[[3]]
$results[[1]]$address_components[[3]]$long_name
[1] "Daehyeon-dong"

$results[[1]]$address_components[[3]]$short_name
[1] "Daehyeon-dong"

$results[[1]]$address_components[[3]]$types
$results[[1]]$address_components[[3]]$types[[1]]
[1] "political"

$results[[1]]$address_components[[3]]$types[[2]]
[1] "sublocality"
```

output="all": 행정구역 분류, 우편번호 등  
세세한 정보까지 확인가능  
각 정보들이 리스트로 구별되어 있음

2

## 위치정보 찾기: geocode

## 한글로 주소출력하기

location=en2utf8("지역명&amp;language=ko")

```
> geocode(location=enc2utf8("이화여자대학교&language=ko"), output="latlon")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=%EC%9D%B4%ED%99%94%EC%97%AC%EC%9E%90%EB%8C%80%ED%95%99%EA%B5%90&language=ko&key=xxx
# A tibble: 1 x 3
  lon lat address
<dbl> <dbl> <chr>
1 127. 37.6 대한민국 서울특별시 서대문구 대현동 이화여대길 52
```

```
> geocode(location=enc2utf8("이화여자대학교&language=ko"), output="more")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=%EC%9D%B4%ED%99%94%EC%97%AC%EC%9E%90%EB%8C%80%ED%95%99%EA%B5%90&language=ko&key=xxx
# A tibble: 1 x 9
  lon lat type loctype address north south east west
<dbl> <dbl> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>
1 127. 37.6 establishment rooftop 대한민국 서울특별시 서대문구 대현동 이화여대길 52 37.6 127. 127.
```

## 2 위치정보 찾기: geocode

```
> geocode(location=enc2utf8("이화여자대학교&language=ko"),output="more")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=%EC%9D%B4%ED%99%94%EC%97%AC%EC%9E%90%EB%8C%80%ED%95%99%EA%B5%90&language=ko&key=xxx
# A tibble: 1 x 9
  lon lat type loctype address north south east west
  <dbl> <dbl> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>
1 127. 37.6 establishment rooftop 대한민국 서울특별시 서대문구 대현동 이화여대길 52 37.6 37.6 127. 127.
```

type: 구글이 분류한 해당 지역(point)의 type (e.g. house(주택), establishment(기관)...)

loctype: type과 비슷하지만 type의 street\_address에서 좀더 세부적인 정보제공  
e.g.

rooftop: 구체적인 geocode와 위치정보를 가진 장소

range\_interpolated: 구체적인 위치정보 없으나 주변 rooftop들의 정보들로 대체됨



## Quakes data를 이용하여 지진 규모를 발생지역에 표시

### Data 구조 확인

```
library(ggmap)
library(ggplot2)

str(quakes)
```

```
> str(quakes)
'data.frame':  1000 obs. of  5 variables:
 $ lat      : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long     : num   182 181 184 182 182 ...
 $ depth    : int    562 650 42 626 649 195 82 194 211 622 ...
 $ mag      : num    4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int     41 15 43 19 11 12 43 15 35 19 ...
```

3

## 위치정보를 이용한 시각화와 실습

### 정보 축소 및 centre 정하기

```
df <- head(quakes,100) # quakes information  
cen <- c(mean(df$long), mean(df$lat)) # map location
```

```
> cen  
[1] 179.1387 -19.9468
```

## Ggmap의 경도 지정에 따라 데이터 변경

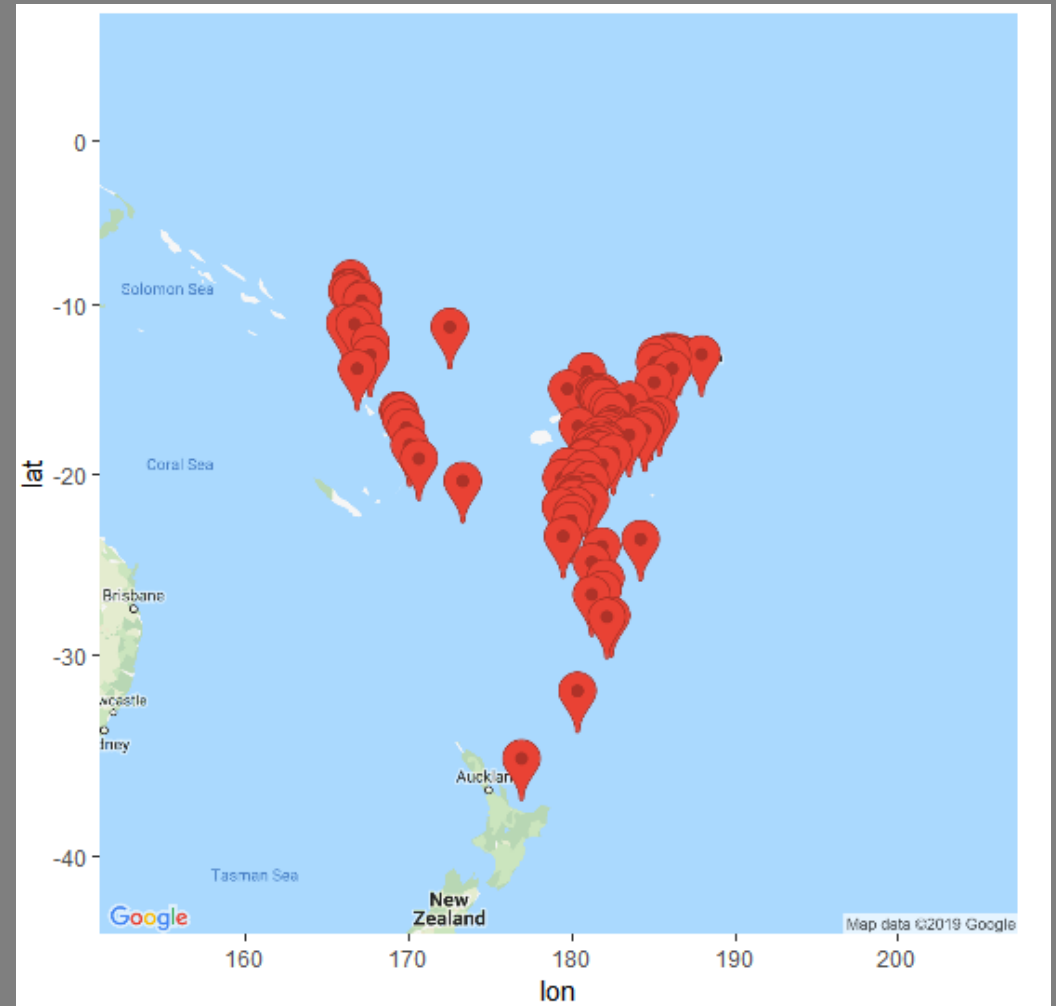
```
# 경도를 넘는 경우 변환  
# 원래는 360도인데 구글 지도에서는 -180~180  
|  
gc <- data.frame(lon=df$long, lat=df$lat) # create data frame  
gc$lon <- ifelse(gc$lon>180, -(360-gc$lon), gc$lon)
```

```
> gc$lon  
[1] -178.38 -178.97 -175.90 -178.34 -178.04 -175.69 166.10 -178.07  
[9] -178.26 179.59 -179.31 167.00 -177.89 -178.34 169.92 -175.05  
[17] 165.96 -178.50 179.78 -179.69 -178.84 166.32 -179.84 -178.00  
[25] -179.72 -178.51 167.51 -179.21 -178.53 -177.63 179.24 166.74  
[33] -174.95 -179.20 -174.00 179.33 169.23 -178.72 -178.60 169.33  
[41] 176.78 -173.90 179.82 -173.96 169.41 -177.70 -178.30 166.32  
[49] -179.92 -174.75 -177.65 -175.58 173.20 -179.33 -177.84 -177.87  
[57] -179.00 -179.40 -178.65 179.20 -178.45 -177.60 172.38 166.22  
[65] -178.59 -175.07 -178.40 179.62 -178.14 -172.19 -174.20 -175.65  
[73] 166.20 179.99 -178.77 -179.96 -175.30 167.06 -178.29 -178.89  
[81] -179.79 -179.01 -177.62 -176.60 -178.30 -175.69 170.50 179.96  
[89] -173.70 -173.56 167.53 167.06 -177.98 169.71 -174.74 -177.60  
[97] -178.89 -176.59 166.54 179.92
```

## Center 중심으로 지도 그리기

```
get_googlemap(center=cen, # on the map  
               maptype="roadmap",  
               zoom=4,  
               marker=gc) %>% ggmap()
```

Zoom : 3(continent) ~ 21(building)  
디폴트 값 : 10(city)



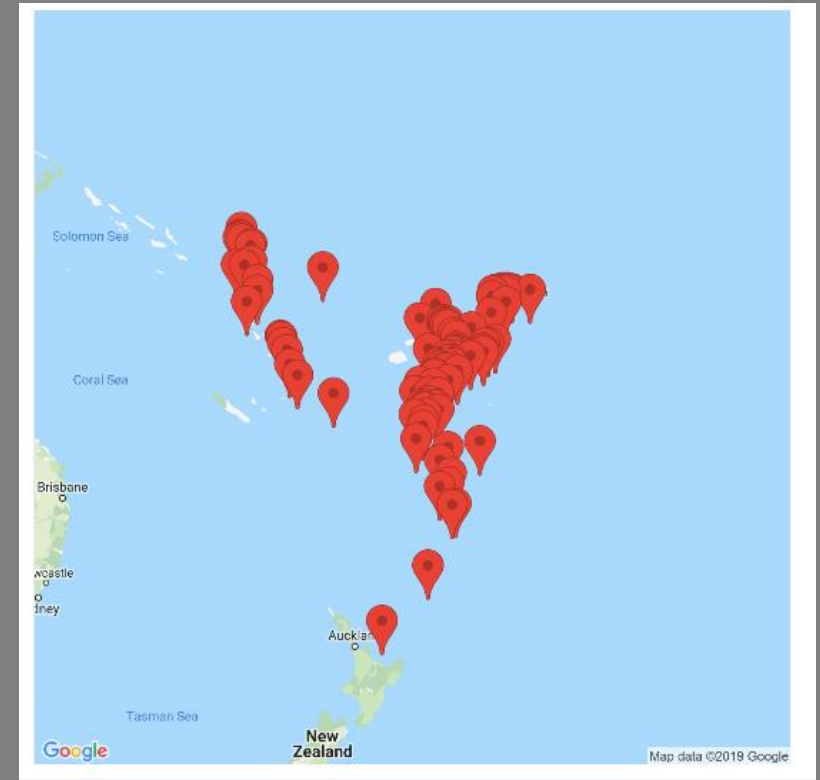
## 3

## 위치정보를 이용한 시각화와 실습

x, y 축 제목, 눈금, text 제거

```
map <- get_googlemap(center=cen, # on the map  
                      maptype="roadmap",  
                      zoom=4,  
                      marker=gc)
```

```
ggmap(map)+theme(axis.title.x=element_blank(), # x,y axis blank  
                 axis.text.x=element_blank(),  
                 axis.ticks.x=element_blank(),  
                 axis.title.y=element_blank(),  
                 axis.text.y=element_blank(),  
                 axis.ticks.y=element_blank())
```

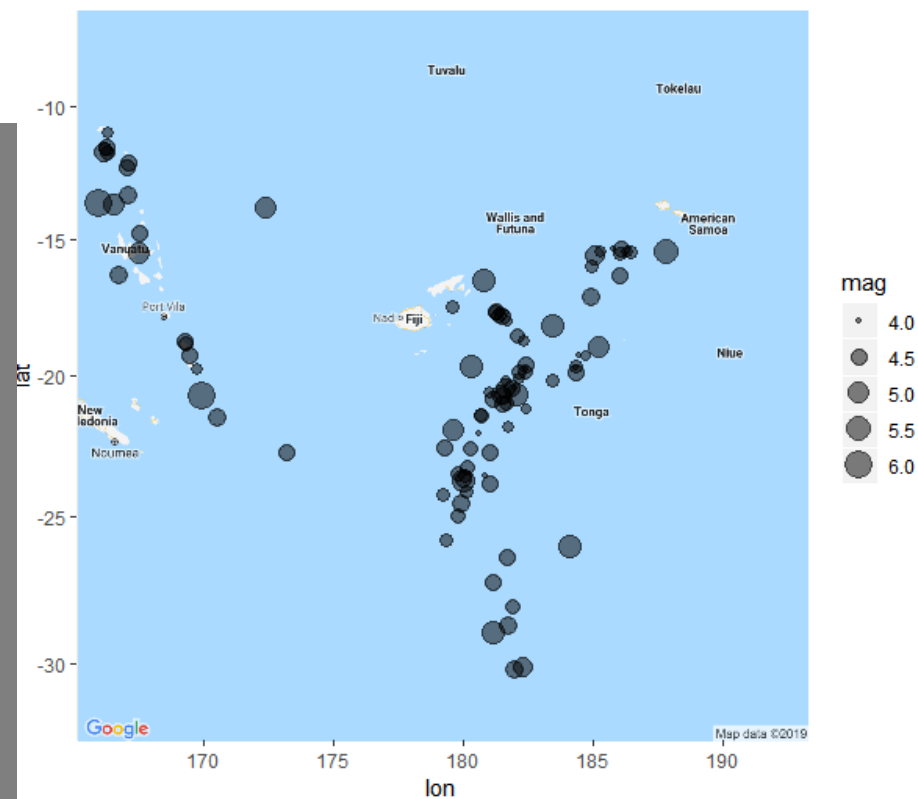


## 3

## 위치정보를 이용한 시각화와 실습

## mag에 따른 원의 크기 표시

```
# mag에 따른 원의 크기 표시|  
gmap+geom_point(data=df, # 산점도 표현  
                aes(x=long,y=lat,size=mag), # 원의 크기를 mag로 표시  
                alpha=0.5) # 투명도 표시
```

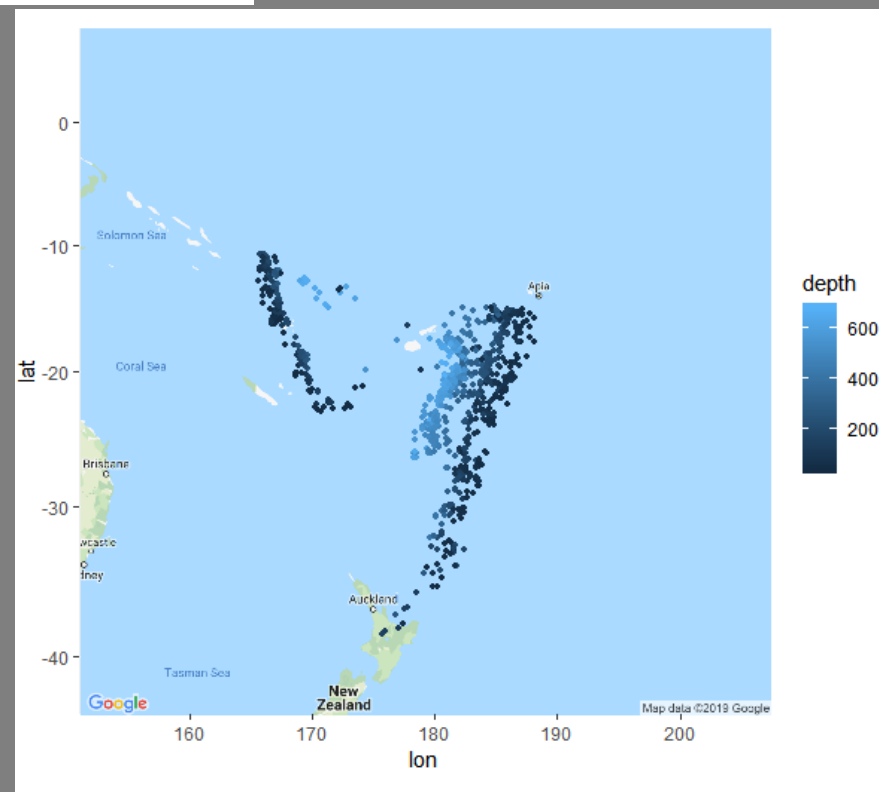


3

## 위치정보를 이용한 시각화와 실습

## # depth에 따른 색깔 표시

```
# depth에 따른 색깔 표시  
ggmap(map) + geom_point(data=quakes, aes(x=long, y=lat, col=depth), size=1)
```

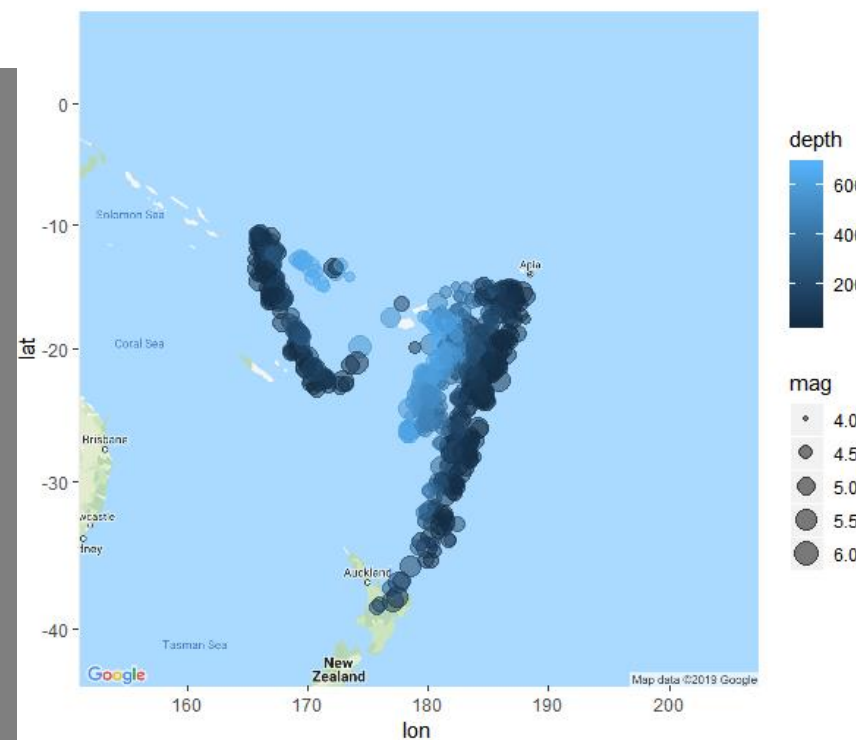


3

## 위치정보를 이용한 시각화와 실습

# size, depth에 따른 색깔 표시

```
# size, depth에 따른 색깔 표시  
ggmap(map) + geom_point(data=quakes, aes(x=long, y=lat, size=mag, col=depth), alpha=0.5)
```





## 지하철 2,3 호선 역 나타내기

## Data 불러오기 및 전처리

```
data <- read.csv('C:/Users/palt1/Desktop/R/지하철역위치.csv', header=T)
```

```
# 필요한 열 뽑기
data2 <- data[,c(2,3,8,9)]
colnames(data2) <- c('전철역명', '호선', 'x좌표', 'y좌표')
```

```
> head(data2)
  전철역명 호선 x좌표.WGS. y좌표.WGS.
1  건대입구   7   37.54069   127.0702
2  건대입구   2   37.54069   127.0702
3     구의   2   37.53708   127.0859
4     강변   2   37.53509   127.0947
5 잠실나루   2   37.52073   127.1038
6     잠실   8   37.51395   127.1022
```

```
> head(data)
  전철역코드 전철역명 호선 외부코드 사이버스테이션 x좌표 y좌표 x좌표.WGS. y좌표.WGS.
1      2729 건대입구   7      727      212 515365 1121815   37.54069   127.0702
2      212  건대입구   2      212      212 515365 1121815   37.54069   127.0702
3      213     구의   2      213      213 518867 1120790   37.53708   127.0859
4      214     강변   2      214      214 520755 1120262   37.53509   127.0947
5      215 잠실나루   2      215      215 522630 1119835   37.52073   127.1038
```

## 2호선 추출

```
# 2호선 추출
s_2 <- data2 %>% filter(호선=='2')

# 상행/하행 모두 표시되서 절반만 추출
N <- seq(1,44,2)
s_21 <- s_2[N,]
```

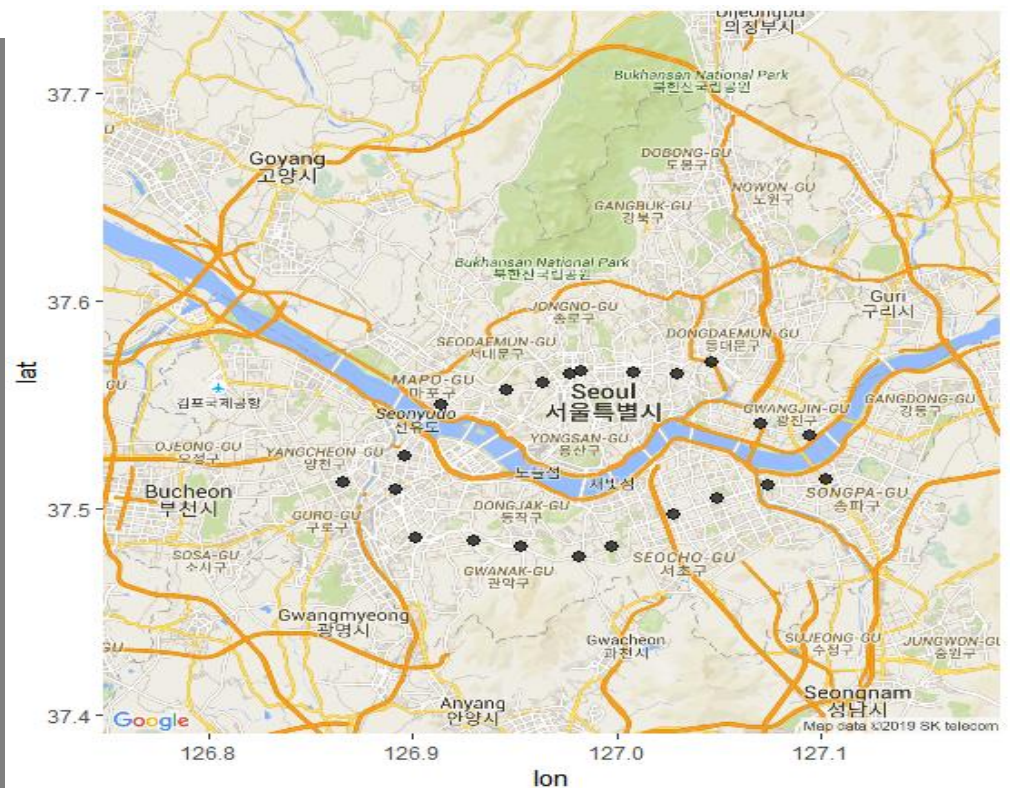
```
> s_21
```

	전철역명	호선	x좌표	y좌표
1	건대입구	2	37.54069	127.0702
3	강변	2	37.53509	127.0947
5	잠실	2	37.51395	127.1022
7	종합운동장	2	37.51100	127.0736
9	선릉	2	37.50450	127.0490
11	강남	2	37.49717	127.0279
13	방배	2	37.48143	126.9976
15	이대	2	37.55673	126.9460
17	충정로	2	37.55997	126.9637

## 2호선 위치 표시

```
# 투명도 조절 (alpha조건은 투명도 조절)
```

```
ggmap(seoul) + geom_point(data=s_21, aes(x=y좌표, y=x좌표), size=2.5, alpha=0.7)
```



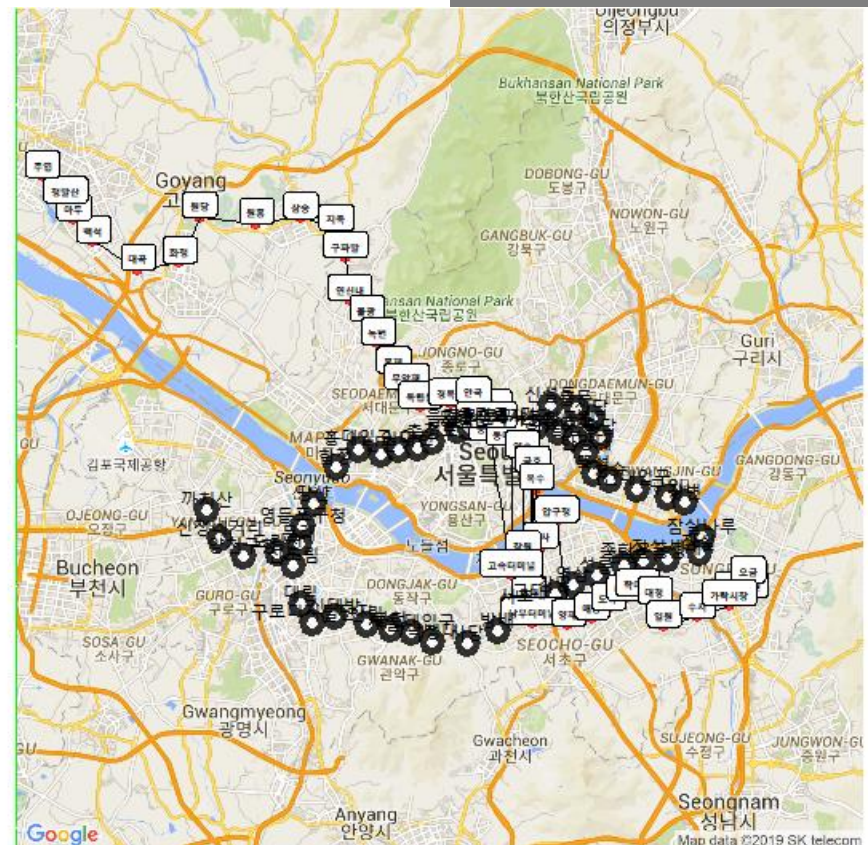
## Data 구조 확인

```
# 3호선 뽑기  
s_3 <- data2 %>% filter(호선=='3')  
center <- c(mean(s_3$y좌표), mean(s_3$x좌표))  
seoul <- get_map(center, zoom=11, maptype='roadmap')
```

## 2,3호선 표시

```
ggmap(seoul, size=c(300,300)) +
  geom_point(data=s_2, aes(x=y좌표, y=x좌표), size=2.5, alpha=0.8, col='black', fill='white', stroke=3, shape=21) + #stroke : 테두리 굵기
  geom_point(data=s_3, aes(x=y좌표, y=x좌표), size=2.5, alpha=0.7, col='red') +
  geom_line(data=s_3, aes(x=y좌표, y=x좌표), linetype=1) +
  geom_label(data=s_3, aes(x=y좌표, y=x좌표+0.005, label=전철역명, fontface='bold'), size=2) +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005, label=전철역명), size=2.8) +
  theme(axis.title.x=element_blank(), # x,y axis blank
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        panel.background = element_rect(fill='green'), legend.position='right')
```

```
# theme : legend, title, 배경, axis의 line, text 등
#          추가적인 요소 변경
```



## 실제 데이터에서 주소만 알 때

```
# 여러 지역의 마커를 표시

names <- c("용두암", "성산일출봉", "정방폭포",
           "중문관광단지", "한라산1100고지", "차귀도")
addr <- c("제주시 용두암길 15",
         "서귀포시 성산읍 성산리",
         "서귀포시 동홍동 299-3",
         "서귀포시 중문동 2624-1",
         "서귀포시 섹달동 산1-2",
         "제주시 한경면 고산리 125")
```

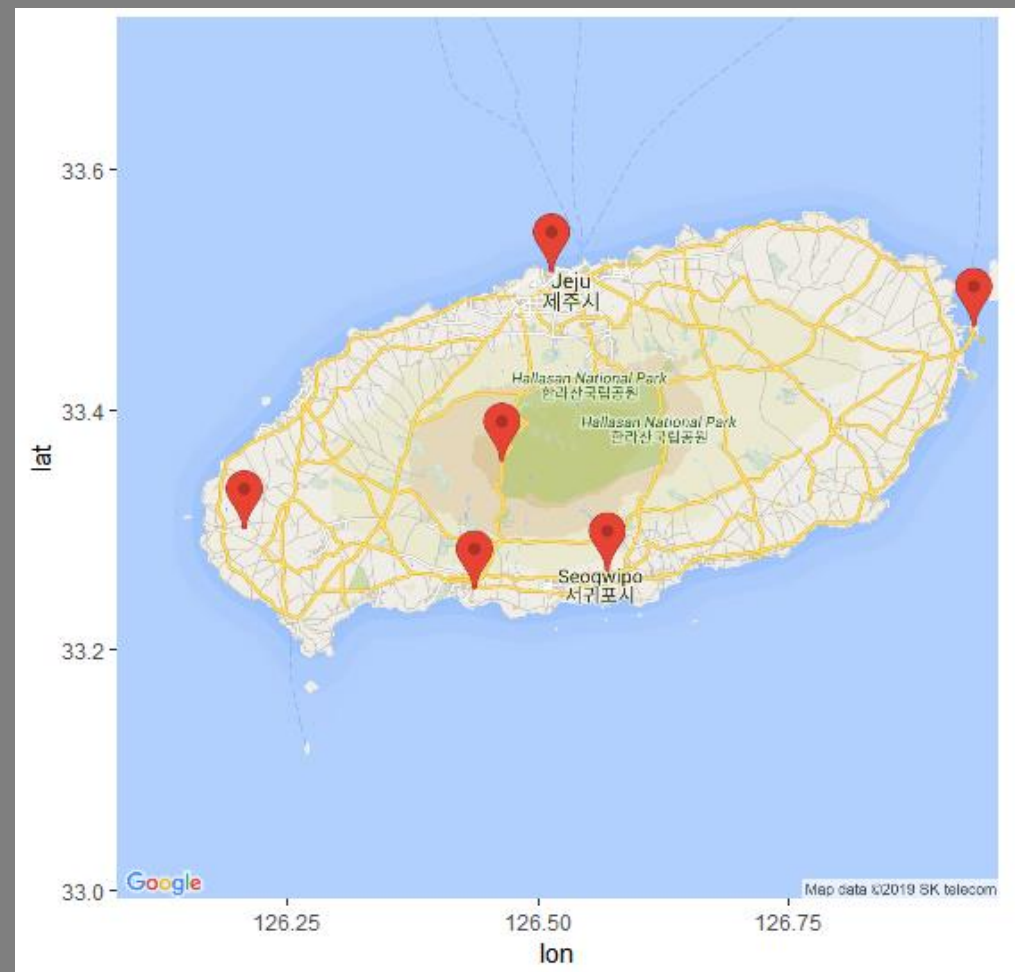
```
gc <- geocode(enc2utf8(addr))
```

```
gc
# A tibble: 6 x 2
#   lon lat
#   <dbl> <dbl>
# 1  127.  33.5
# 2  127.  33.5
# 3  127.  33.3
# 4  126.  33.3
# 5  126.  33.4
# 6  126.  33.3
```



## 위치 표시

```
df <- data.frame(name=names,  
                 lon=gc$lon,  
                 lat=gc$lat)  
  
|  
cen <- c(mean(df$lon), mean(df$lat))  
  
get_googlemap(center=cen,  
              maptype="roadmap",  
              zoom=10,  
              size=c(640,640),  
              marker=gc) %>% ggmap()
```



## 주요도시 인구 표시

## 데이터 불러오기

```
#1.  
data <- read.csv("C:/Users/palt1/Desktop/R/poppulation_2014 (1).csv")
```

```
head(data)
```

```
#2. 열 이름 변경
```

```
region <- data$지역명
```

```
lon <- data$lon
```

```
lat <- data$lat
```

```
pop <- data$population
```

```
data <- data.frame(region, lon, lat, pop)
```

```
head(data)
```

```
> head(data)  
  region      lon      lat  pop  
1   서울 126.9780 37.56654 9975  
2   부산 129.0756 35.17955 3452  
3   대구 128.6014 35.87144 2475  
4   인천 126.7052 37.45626 2862  
5   광주 126.8526 35.15955 1505  
6   대전 127.3845 36.35041 1553  
>
```



## 지도 불러오기

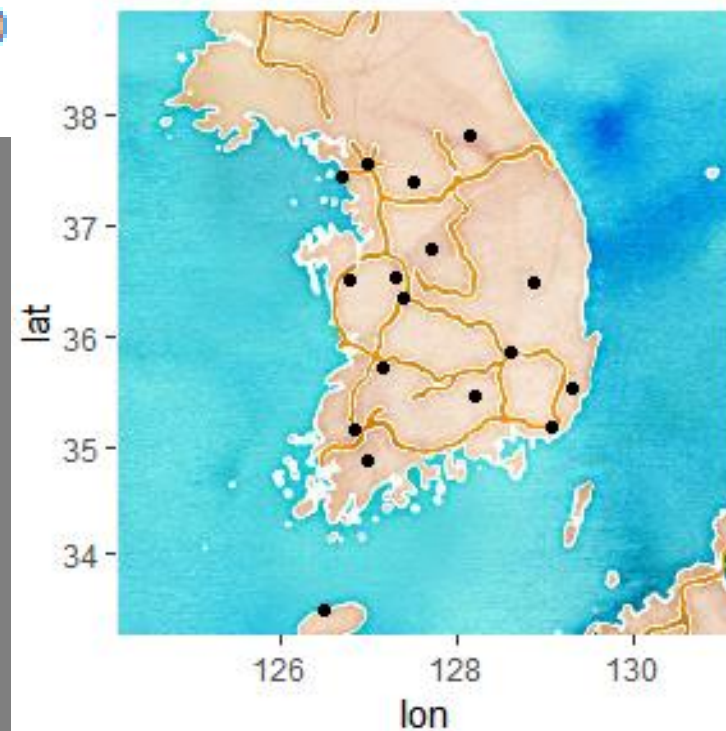
```
# 세 종 좌표 수정 (잘못 되었음)
data[8,2] <- 127.296620
data[8,3] <- 36.535268

#3. get_map
center <- c(mean(data$lon), mean(data$lat))

map <- get_map(location =center, maptype="watercolor", zoom=7)
m1 <- ggmap(map)
|
```

## 각 지역 표시하기

```
#4. geom_point  
m1 + geom_point(data=data, aes(x=lon, y=lat))
```



- 인구별 색깔, 사이즈 정하기
- labeling

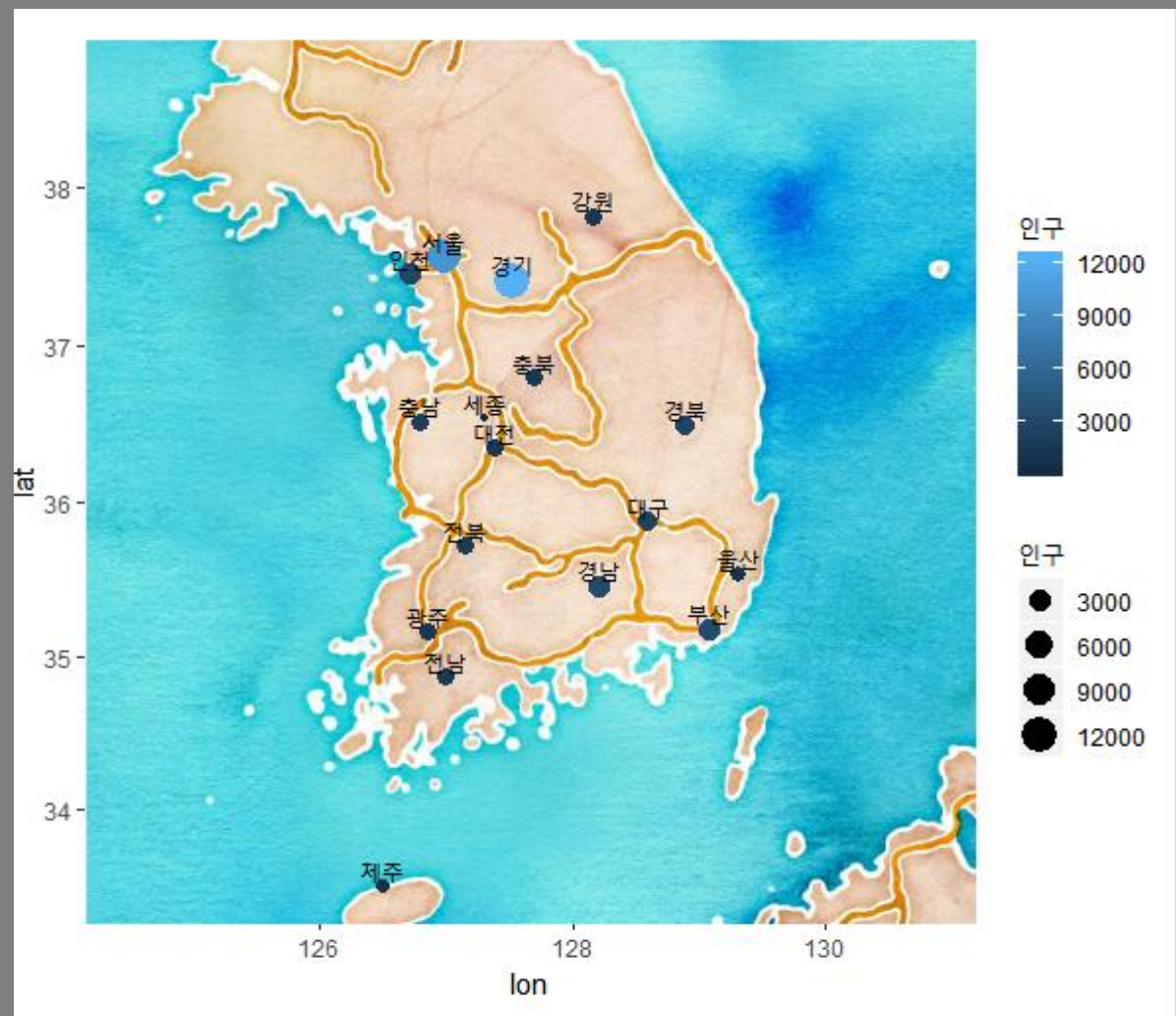
```
#5 | pop비율에 맞게 size변경 / label 넣기 / 범례제목 변경
m1 +
  geom_point(data=data,
             aes(x=lon,y=lat, color=pop, size=pop)) +
  geom_text(data=data,
            aes(x=lon, y=lat+0.1, label=region),
            size=3) +
  scale_color_continuous(name="인구") +
  scale_size_continuous(name="인구")
```

[범례 설정 및 이름 정하기]

- scale\_color/size\_continuous/discrete  
연속형/이산형에 따라 나뉘어짐

3

## 위치정보를 이용한 시각화와 실습



## Scale\_color/size\_discrete 예시

```
# data:다운
data <- read.csv("C:/Users/palt1/Desktop/R/university.csv")
head(data)
str(data)

# get_map : 지도 가져오기
library(ggmap)
map <- get_map("seoul", zoom=11, maptype="watercolor")
ggmap(map)

# data의 위도, 경도 point 찍기
# 학교이름별 색깔변화
map2 <- ggmap(map) +
  geom_point(data=data,
             aes(x=lon, y=lat, color=학교명),
             size=3)

# label 넣기 / 범례제목설정
map2 + geom_text(data=data,
                 aes(x=lon+0.01, y=lat+0.01, label=학교명),
                 size=3) +
  scale_color_discrete(name="학교명")
```

```
> head(data)
      학교명      lat      lon
1 서울대학교 37.45988 126.9519
2 중앙대학교 37.50415 126.9570
3 연세대학교 37.56578 126.9386
4 홍익대학교 37.55171 126.9261
5 숭실대학교 37.49638 126.9569
6 고려대학교 37.59080 127.0278
> |
```



## 서울특별시 커피숍 정보(주소만 표시)

번호	사업장명	소재지전차	도로명전차	인허가일자	영업상태명
1	곰미커피	서울특별시	서울특별시	20130605	운영중
2	학다방	서울특별시	서울특별시	19760205	운영중
3	남탕매점	서울특별시	서울특별시	20030315	운영중
4	솔	서울특별시	서울특별시	19810723	운영중
5	아람다방	서울특별시	서울특별시	19871022	운영중
6	우희	서울특별시	서울특별시	19760205	운영중
7	삼거리	서울특별시	서울특별시	19760422	운영중
8	대지다방	서울특별시	서울특별시	19911116	운영중

1. add = []
2. for문을 통해 add에 주소 다  
기입.
3. vector화 : add\_ = unlist(add)
4. geocode(enc2utf8(add\_))



끝!

안되거나 질문 있으시면 말씀해주세요😊