1. 지도의 시각화

지도의 시각화는 R에서 쉽게 할수 있는 매우 강력한 시각화 방법이다. 사실 지도의 시각화는 엑셀이나 파워포인트, 포토샾이나 일러스트 같은 대중적으로 많이 사용되는 툴에서도 가능하겠지만 매우 번거로운 작업임에 틀림없다. 또 단순한 지도의 시각화라면 다소의 시간과 많은 정신적 스트레스를 감소하고라도 할 수 있겠지만 만약 200개가 넘는 시군구 지역별 시각화를 해야한다면 이는 단순히 시간과 정신적 스트레스과 바꿀수 있는 작업이 아닐 것이다. 이때는 전문가를 찾아 의뢰하는 것이 해결책이겠지만 항상 전문가들은 바쁘기때문에 시간이 많이 소요되고 비싸다. R은 지도의 시각화에 생각보다 많은 기능이 지원된다. 이 기능을 잘 사용하면 지도의 시각화에 시간을 절약하고 정신적 스트레스를 급감시켜 생명이 늘어나는 기쁨을 누릴지도 모른다.

# 지도의 생성

지도의 시각화를 위해서는 먼저 지도를 그릴 수 있어야 한다. 또 지도를 그리려면 지도에 관련된 데이터가 필요할 것이다. R이든 다른 툴에서든 지도를 그리기 위해 제공되는 데이터 타입은 여러가지가 있지만 여기서는 가장 많이 사용되는 Shape 파일(\*.shp)과 Shape 파일을 사용하여 JSON 형태로 저장한 geojson 파일을 불러읽어 지도를 그리는 방법을 설명하겠다. 여기에서 설명하는 방식 외에 다양한 방식으로 지도를 시각화 할 수 있다. 따라서 여기여 소개하는 방법외에도 자신이 가장 편한 방법을 사용하는 것이 가장 좋은 방법이다.

## Shape 파일

Shape 파일은 대부분의 GIS(Geographical Information System)에서 사용되는 지형 벡터 데이터 파일 포맷이다. 확장자를 \*.shp로 설정하는데 선, 점, 다각형(Polygon)으로 지형을 벡터의 형태로 표현하는 텍스트 파일이다. 하지만 Shape 파일은 보통 \*.shp, \*.shx, \*.dbf, \*.kml, \*.prj 등의 파일이 한 세트처럼 제공되는 것이 일반적이다. 그렇다고 이 네개의 파일을 모두 쓰는 것은 아니고 사용하는 응용에 따라 선택하여 사용할 수 있다.  
우리나라 지도의 Shape 파일은 인터넷에서 다운로드 받을 수 있다.[[1]](#footnote-20) 다운로드 받은 파일은 2021년 1월 업데이트된 시도 파일이다. 이 두 파일은 각각 \*.shp, \*.shx, \*.dbf, \*.prj의 네 개의 파일로 구성된 압축파일이고 이중 \*.shp만을 사용하도록 하겠다.

Shape 형태의 지도 데이터를 불러 들이는데는 여러가지 방법이 있지만 여기서는 sf패키지의 st\_as\_sf()를 사용하여 sf 객체로 변환한다. sf 패키지는 Simple Feature라고 불리는 점, 선, 다각형, 멀티 포인트, 멀티 라인 등 2차원 기하요소로 구성된 지형의 저장 및 처리 방법을 지원하는 패키지이다. Simple Feature는 OGC(Open Geospatial Consortium)와 ISO( International Organization for Standardization )에서 표준화 포맷으로 채택되어 널리 사용되고 있다. 이 Simple Feature로 표현된 객체를 ggplot 레이어로 생성하기 위해서는 ggplot2패키지의 geom\_sf()를 사용한다.

시도 행정구역의 shape 파일(TL\_SCCO\_CTPRVN.shp)을 R로 불러 들이기 위해서는 먼저 sf 패키지의 read\_sf()를 사용할 수 있다. read\_sf()로 읽어들인 Shape 파일은 앞서 설명한 sf(Simple Feature) 객체로 저장된다.

read\_sf(dsn,layer, …, query = NA, options = NULL, quiet = FALSE, geometry\_column = 1L, type = 0, promote\_to\_multi = TRUE, int64\_as\_string = FALSE, check\_ring\_dir = FALSE, fid\_column\_name = character(0), drivers = character(0), wkt\_filter = character(0), quiet = TRUE, stringsAsFactors = FALSE, as\_tibble = TRUE)  
- dsn : 데이터 소스 이름 설정  
- layer : 레이어 이름, 특별한 지정이 없다면 dsn의 확장자를 뗀 파일 이름을 사용  
- … : st\_as\_sf()에 사용할 매개변수  
- options : 데이터 소스를 읽을 때 사용할 옵션 설정  
- geometry\_coulumn : 지형정보가 들어있는 열 이름 설정  
- as.tibble : sf 객체를 tibble로 설정할지에 대한 논리값 설정하는 논리값  
- type : simple feature type의 ISO 번호  
- promote\_to\_multi : 전체 다각형을 Multi형 다각형으로 설정하는 논리값  
- int64\_as\_string : 문자열 수치를 int64로 변환하는 논리값  
- check\_ring\_dir : 다각형 링의 방향을 설정하는 논리값  
- fid\_column\_name : Feature ID로 사용할 열 이름 설정  
- wkt\_filter : WKT의 표현에 사용할 필터 설정

Shape 파일을 읽어 sf 객체로 저장하는 코드는 다음과 같다.

## sf 패키지 설치  
if(!require(sf)) {  
 install.packages('sf')  
 library(sf)  
}  
  
## read\_sf()을 사용하여 TL\_SCCO\_CTPRVN.shp 파일을 읽어옴(옵션은 한글깨짐을 방지하기 위한 인코딩값, 띄어쓰기 주의)  
spdf\_shp <- st\_read('파일 저장 경로/TL\_SCCO\_CTPRVN.shp'), options = 'ENCODING=CP949')  
  
## spdf\_shp 클래스 확인  
class(spdf\_shp)  
  
## spdf\_shp 열 이름 확인  
names(spdf\_shp)

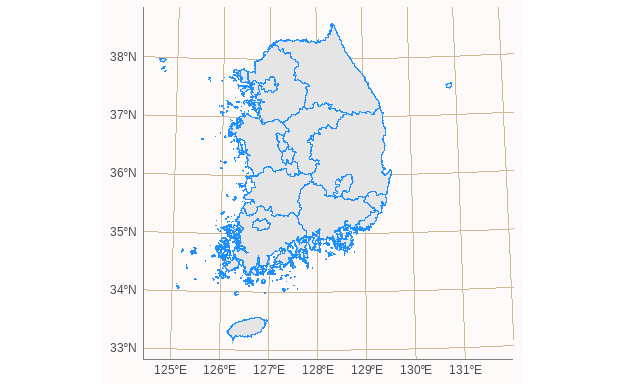
## options: ENCODING=CP949   
## Reading layer `TL\_SCCO\_CTPRVN' from data source   
## `D:\R\git\datavisualization\chap10\TL\_SCCO\_CTPRVN.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 17 features and 3 fields  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: 746110.3 ymin: 1458754 xmax: 1387950 ymax: 2068444  
## Projected CRS: PCS\_ITRF2000\_TM

## [1] "sf" "data.frame"

## [1] "CTPRVN\_CD" "CTP\_ENG\_NM" "CTP\_KOR\_NM" "geometry"

이제 불러들인 지도 객체를 ggplot2를 사용하여 그려보도록 하겠다.

## sf 객체(Simple Feature)는 별다른 X, Y축의 매핑 없이 geom\_sf() 레이어를 생성할 수 있다.   
spdf\_shp |> ggplot() +   
 ## X축을 long(경도), Y축을 lat(위도), group을 group, color를 id로 매핑하고 fill을 white로 설정한 geom\_polygon 레이어 생성   
 ## simple feature 객체를 사용하여 geom\_sf 레이어를 생성  
 geom\_sf(color = 'dodgerblue')



**실행결과 10-**Shape 파일 지도의 시각화

## geojson 파일

GeoJSON은 위치정보를 갖는 점을 기반으로 체계적으로 지형을 표현하기 위해 설계된 개방형 공개 표준 형식이다. 지형 정보를 자바스그립트 오브젝트 노테이션(JAVA Script Object Notation, JSON)을 활용하여 사용하는 파일 포맷으로 OpenLayers, Leaflet, MapServer, Geoforge 소프트웨어, GeoServer, GeoDjango, GDAL, Safe Software FME 등 많은 매핑 및 GIS 소프트웨어 패키지에서 지원하고있다[[2]](#footnote-23)

geojson 파일의 확장자는 json으로 붙는다. geojson 파일은 앞서 설명한 Shape 파일에서부터 변환하여 생성할 수 있지만 여기서는 인터넷에 게시된 우리나라 시도, 시군구 geojson 파일을 다운로드하여 사용했다.[[3]](#footnote-25)

geojson 파일은 Shape의 \*.shp 파일에 비해 파일 사이즈가 작고 처리속도도 Shape 파일보다 매우 빠르다는 장점이 있다.

geojson 파일의 지도 데이터를 읽어오기 위해서는 geojsonio패키지의 geojson\_read()를 사용하면 가능하다. geojson\_read()로 읽어온 데이터는 Shape 파일을 읽어들인 클래스와 동일한 ‘SpatialPolygonsDataFrame’ 객체로 생성할 수 있다. 가급적이면 ‘SpatialPolygonsDataFrame’ 객체로 불러들이는 것이 Shape 파일로 불러읽은 객체의 코드를 공유할 수 있어 권장한다. 다음은 geojson\_read()로 geojson 파일을 읽어들이는 코드이다.

## rgdal 패키지 설치  
if(!require(geojsonio)) {  
 install.packages('geojsonio')  
 library(geojsonio)  
}  
  
## geojson\_read()을 사용하여 TL\_SCCO\_SIG.json 파일(시군구 행정구분)을 읽어오는데 SpatialPolygonsDataFrame 형태로 읽어옴(what = "sp")  
spdf\_geojson <- geojson\_read('Shape 파일 경로/TL\_SCCO\_SIG.json', what = "sp")  
  
## spdf\_geojson 클래스 확인  
class(spdf\_geojson)  
  
## spdf\_geojson 열 이름 확인  
names(spdf\_geojson)  
  
## spdf\_geojson 슬롯 이름 확인  
slotNames(spdf\_geojson)

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

## [1] "SIG\_CD" "SIG\_ENG\_NM" "SIG\_KOR\_NM"

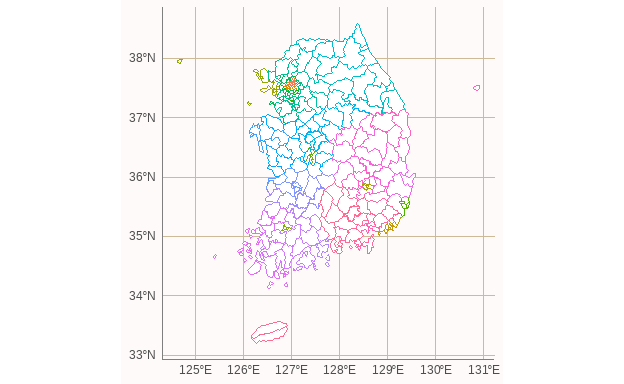
## [1] "data" "polygons" "plotOrder" "bbox" "proj4string"

위의 코드의 결과와 같이 Shape 파일을 불러들인 결과가 저장되는 클래스는 ‘SpatialPolygonDataFrame’ 이고 “CTPRVN\_CD”, “CTP\_ENG\_NM”, “CTP\_KOR\_NM”의 세개의 열이 포함되어 있다. 또 다섯개의 슬롯이 있는 것이 보인다. 슬롯은 S4 클래스[[4]](#footnote-27)에서 사용되는데 S4 클래스를 구성하는 구성요소이다.

이제 geojson으로 불러들인 지도 객체를 ggplot2를 사용하여 그려보도록 하겠다. 앞서 Shape 파일을 불러들인 spdf\_shp 객체를 ggplot2에서 사용하기 위해 st\_as\_sf()를 이용해서 sf객체(Simple Feature)로 변환했다.

## sf 패키지 설치  
if(!require(sf)) {  
 install.packages('sf')  
 library(sf)  
}  
  
## spdf\_geojson을 st\_as\_sf()을 사용하여 simple feature 객체로 저장  
sf\_spdf <- st\_as\_sf(spdf\_geojson)

ggplot() +   
 ## simple feature 객체를 사용하여 geom\_sf 레이어를 생성  
 geom\_sf(data = sf\_spdf, aes(color = SIG\_CD), fill = "white", show.legend = F)



**실행결과 10-**sf 객체의 지도의 시각화

# 지도위에 데이터 표시

R은 지도를 그리기 위해 사용하는 프로그램이 아니다. R에서의 지도의 사용은 데이터를 표현하기 위해 사용하는 것이다. 그렇다면 데이터를 어떻게 지도위에 표현할 것인가가 가장 중요하다. 지도위에 데이터를 표현하는 방법은 지역의 색을 사용하여 각각의 지역의 특성을 시각화하는 방법(Choropleth Map)과 지도에 특정 위치를 점으로 표기하는 방법(Symbol Map)을 주로 사용한다.

## 지도 색을 사용한 데이터 표시(Choropleth Map)

색으로 데이터를 표시하는 방법은 각각의 다각형(Polygon)으로 구성되어 있는 지역의 색을 데이터에 따라 다르게 함으로서 지역간 데이터의 차이를 시각화 하는 방법이다. 지형 데이터를 이루고 있는 geometry의 내부 색(fill) 미적요소를 표현하고자 하는 데이터에 매핑함으로써 구현이 가능하다.

전국 시도 지도에 데이터를 매핑하기 위해 먼저 데이터를 전처리 하겠다. 지도에 표기할 데이터는 df\_입학생 데이터프레임의 2021년의 일반대학 입학생수를 지도 데이터와 조인해서 사용한다. 이 과정은 Shape 파일을 사용하는 방법(데이터프레임 객체)과 geojson 파일을 사용하는 방법(sp 객체)으로 나누어 설명하겠다.

### Shape 파일 사용(데이터프레임 객체)

sf패키지의 read\_sf()를 사용하여 Shape 파일의 지형 데이터를 읽어들이면 데이터가 데이터프레임 타입(tibble도 가능)으로 저장된다. 이 데이터프레임에는 시도 코드(CTPRVN\_CD), 영문 이름(CTP\_ENG\_NM), 한글 이름(CTP\_KOR\_NM), 지형 정보(geometry) 열을 가지고 있다.

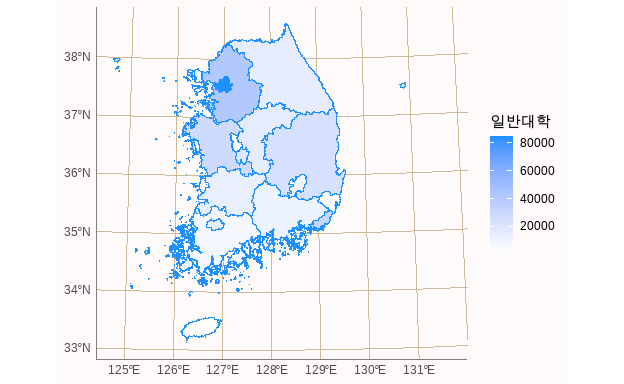
## Simple Feature 객체의 데이터 확인  
head(spdf\_shp)

## Simple feature collection with 6 features and 3 fields  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: 900494.7 ymin: 1612159 xmax: 1387950 ymax: 2068444  
## Projected CRS: PCS\_ITRF2000\_TM  
## CTPRVN\_CD CTP\_ENG\_NM CTP\_KOR\_NM geometry  
## 1 42 Gangwon-do 강원도 MULTIPOLYGON (((1163759 190...  
## 2 41 Gyeonggi-do 경기도 MULTIPOLYGON (((931607.5 18...  
## 3 48 Gyeongsangnam-do 경상남도 MULTIPOLYGON (((1053643 161...  
## 4 47 Gyeongsangbuk-do 경상북도 MULTIPOLYGON (((1171290 178...  
## 5 29 Gwangju 광주광역시 MULTIPOLYGON (((932712.7 16...  
## 6 27 Daegu 대구광역시 MULTIPOLYGON (((1107778 178...

하지만 df\_입학자 데이터에 있는 지역 이름과는 유사하지만 정확히 일치하지 않는다. 따라서 두 데이터프레임을 조인하기 위해서는 이 부분을 맞춰주어야 한다. 문자열로 조인하는 것도 좋은 방법이지만 중간 오타의 문제로 데이터가 정상적으로 조인되지 않는 경우가 흔히 발생하기 때문에 가급적 수치 데이터인 코드로 조인하는 것을 권장한다. 다음과 같이 df\_입학자에 조인을 위한 추가 열을 생성해주고 두 데이터프레임을 조인한다.

## 데이터를 필터링하고 데이터프레임 조인을 위한 추가 열 생성  
df\_입학자\_join <- df\_입학자 |> filter(연도 == '2021', 지역 != '전체') |>   
 mutate(id = case\_when(  
 지역 == '강원' ~ '42',   
 지역 == '경기' ~ '41',  
 지역 == '경남' ~ '48',  
 지역 == '경북' ~ '47',  
 지역 == '광주' ~ '29',  
 지역 == '대구' ~ '27',  
 지역 == '대전' ~ '30',  
 지역 == '부산' ~ '26',  
 지역 == '서울' ~ '11',  
 지역 == '세종' ~ '36',  
 지역 == '울산' ~ '31',  
 지역 == '인천' ~ '28',  
 지역 == '전남' ~ '46',  
 지역 == '전북' ~ '45',  
 지역 == '제주' ~ '50',  
 지역 == '충남' ~ '44',  
 지역 == '충북' ~ '43'  
 ))

## df\_spdf와 df\_입학자\_join을 'CTPRVN\_CD'열과 'id'열을 기준으로 조인  
inner\_join(spdf\_shp, df\_입학자\_join, by = c('CTPRVN\_CD' = 'id')) |>  
 ggplot() +   
 ## fill을 일반대학으로 매핑하고 color를 설정한 geom\_sf 레이어 생성  
 geom\_sf(aes(fill = 일반대학), color = 'dodgerblue') +   
 ## fill 스케일을 흰색부터 dodgerblue까지의 그래디언트 색으로 설정  
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue')



**실행결과 10-**Shape 파일과 데이터를 조인한 지도 시각화

### geojson 파일 사용(sp 객체)

이번에는 geomjson 파일의 데이터를 sp 패키지의 SpatialPolygons 객체와 SpatialPolygonsDataFrame 객체로 읽어들여 사용하는 방법을 알아보자.

sp 패키지는 2005년 개발되어 CRAN에 등록되었는데 점, 선, 다각형 및 그리드의 생생과 이를 다루기 위한 다양한 함수를 지원하고 이들을 클래스로 캡슐화(Encapsulation)하는 기능을 제공한다. 이 sp 패키지를 기반으로 350개 이상의 지형 관련 패키지가 생성되었다고 한다. sp 패키지에서 제공하는 모든 공간 객체는 기본적으로 Spatial 클래스에서 파생된 클래스 객체이다. 이 Spatial 클래스는 경계 상저(bounding box)와 CRS(Coordinate Reference System)의 두가지 슬롯으로 구성되어 있다.

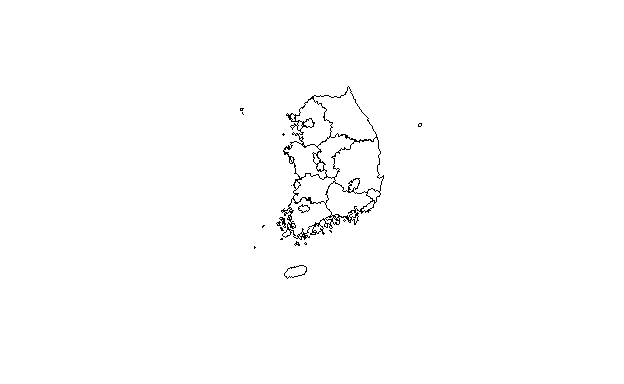
sp 패키지의 기능을 사용하기 위해서는 먼저 sp 패키지를 설치하여야 한다. geojson\_read()를 사용하여 json 파일을 읽어올 때 다음과 같이 what = sp 옵션을 주면 sp 패키지에서 제공하는 클래스(SpatialPolygonsDataFrame) 객체로 사용이 가능하다.

## sp 패키지 설치  
if(!require(sp)) {  
 install.packages('sp')  
 library(sp)  
}  
  
## geojson\_read()을 사용하여 TL\_SCCO\_CTPRVN.json 파일을 읽어옴  
spdf\_geojson\_시도 <- geojson\_read('geojson 파일 저장 경로/TL\_SCCO\_CTPRVN.json', what = "sp")  
  
class(spdf\_geojson\_시도)

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

앞의 실행결과에서 보다시피 geojson 파일의 데이터를 SpatialPolygonsDataFrame 객체로 가져왔다. 이를 ggplot2에서는 바로 시각화할 수 없다. R에서 기본적으로 제공되는 plot()을 이용하면 간단히 볼 수 있다.

plot(spdf\_geojson\_시도)



**실행결과 10-**json을 이용한 지도 시각화

하지만 데이터 시각화를 위해 ggplot2 방식으로 시각화하는 것이 좋겠다. ggplot2에서는 SpatialPolygonsDataFrame객체를 직접적으로 그릴 수 없다. 따라서 ggplot2에서 그릴수 있는 sf 패키지 기반의 객체로 바꾸어 그릴 수 있다. sf 객체로 변환하기 위해서는 sf 패키지의 st\_as\_sf()를 사용한다.

st\_as\_sf(x, …, agr = NA\_agr\_, coords, wkt, dim = “XYZ”, remove = TRUE, na.fail = TRUE, sf\_column\_name = NULL)  
- x : sf 클래스 객체로 변환할 객체  
- … : st\_sf() 함수에서 사용하는 매개변수  
- agr : attribute-geometry-relationship의 준말로 지형 기하요소와 속성값의 관계를 설정  
- coords : 좌표계 설정을 위한 좌표계 이름이나 번호  
- wkt : Well- Known Text(WKT, 벡터 기하학 객체 를 표현하기 위한 텍스트 마크업 언어)으로 기술된 열의 이름이나 번호  
- dim : 포인트의 설정(st\_point)에 필요한 설정  
- remove : coords나 wkt가 주어질때 해당 열을 제거할지를 설정하는 논리값  
- na.fail : 결측치 발생시 에러를 낼지에 대한 논리값  
- sf\_column\_name : sf 객체로 표현되는 리스트 열의 이름

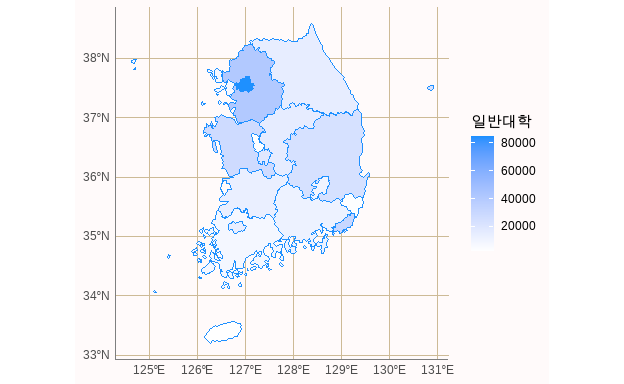
다음과 같이 st\_as\_sf()를 사용하여 SpartialPolygonDataFrame 객체를 Simple Feature 객체로 변환해준다.

sf\_geojson\_시도 <- st\_as\_sf(spdf\_geojson\_시도)   
  
class(sf\_geojson\_시도)

## [1] "sf" "data.frame"

지도의 표현에 필요한 sf 객체가 완성되었다. 이제 지도에 표현할 데이터를 조인시켜야 한다. 이후는 앞선 방법과 동일하다.

inner\_join(sf\_geojson\_시도, df\_입학자\_join, by = c('CTPRVN\_CD' = 'id')) |>  
ggplot() +   
 ## simple feature 객체를 사용하여 geom\_sf 레이어를 생성  
 geom\_sf(aes(fill = 일반대학), color = 'dodgerblue') +   
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue')



**실행결과 10-**sf 객체와 데이터를 조인한 지도 시각화

## 지도 위에 점 표기(Symbol Map)

지도를 사용해 데이터를 시각화할때 사용하는 또 하나의 방법은 지도위에 특정 위치를 표현하는 것이다. 이 과정에서 많이 겪는 문제가 좌표계에 관련된 문제이다. 우리는 보통 위도와 경도의 좌표계에 익숙하지만 세상에는 매우 많은 좌표계가 존재한다. 이 좌표계만 신경쓰면 지도위에 점을 표기하는 것은 생각보다 쉽다.

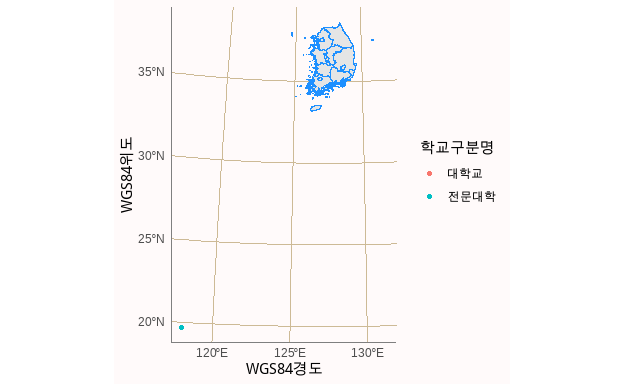
여기서는 경기도에 위치하는 대학의 위치를 점으로 표기해보도록 하겠다. 이를 위해 먼저 경기도에 위치하는 대학의 위도와 경도 데이터가 필요하다. 이 데이터는 ‘경기 데이터 드림’ 사이트에서 다운받았다.[[5]](#footnote-34)

다음과 같이 다운로드 받은 파일을 불러들이고 일부 결측치를 제거하고 사용할 데이터만 필터링 한다.

df\_univ\_loc <- read\_excel('C:/R/git/datavisualization/chap10/전문및대학교현황.xls',   
 ## 'data' 시트의 데이터를 불러오는데,  
 sheet = '전문 및 대학교 현황',  
 ## 첫번째 행은 열 이름을 설정  
 col\_names = TRUE,   
 ## 열의 타입을 설정, 처음 8개는 문자형으로 다음 56개는 수치형으로 설정  
 col\_types = c(rep('text', 8), rep('numeric', 2)))  
  
df\_univ\_loc <- df\_univ\_loc |> filter(!is.na(WGS84위도), 학교구분명 %in% c('전문대학', '대학교'))

이제 전처리된 데이터의 대학 위치를 표시하기 위한 기본 지도를 그린다. 경기도 대학을 대상으로 그릴것이기 때문에 앞서 만든 spdf\_shp 객체에 앞서 전처리한 위경도에 점을 그린다. 이 점은 다시 학교구분별로 색을 달리하도록 color를 매핑한다.

spdf\_shp |>  
 ggplot() +   
 ## simple feature 객체를 사용하여 geom\_sf 레이어를 생성  
 geom\_sf(color = 'dodgerblue') +   
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue') +  
 geom\_point(data = df\_univ\_loc, aes(y = WGS84위도, x = WGS84경도, color = 학교구분명))



**실행결과 10-**좌표계가 어긋난 지도의 시각화

# ggrepel::geom\_text\_repel(data = df\_univ\_loc, aes(y = WGS84위도, x = WGS84경도, label = 시설명), max.overlaps =50, size = 2) +   
## coord\_sf(xlim = c(126, 128), ylim = c(37, 38))

그런데 지도와 점이 따로 논다. 이 부분이 앞서 말한 좌표계가 맞지 않아서 생기는 문제이다. 지도를 그리기 위해 Shape 파일을 읽어 생성한 spdf\_shp 객체에 CRS(Coordinate Reference System) 정보와 전처리한 데이터의 내용을 보면 다음과 같다.

## spdf\_shp의 좌표계 확인  
st\_geometry(spdf\_shp)

## Geometry set for 17 features   
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: 746110.3 ymin: 1458754 xmax: 1387950 ymax: 2068444  
## Projected CRS: PCS\_ITRF2000\_TM  
## First 5 geometries:

앞의 결과를 보면 ‘Projected CRS: PCS\_ITRF2000\_TM’이라는 문구가 보인다. 이게 무슨 말인지 알 수 있다면 아마도 지리학 전공자일 것이다. 하지만 아래의 ’MULTIPOLYGON’ 부분에 있는 두개씩의 수치들, 매우 90만에서 110만을 왔다갔다 하는 이 수치들을 보면 뭔가 감이 온다. 다각형을 이루는 X, Y값이 우리가 흔히 알고 있는 위도, 경도 값이 아님을 알 수 있다. 반면 대학의 위치를 찍기 위해 전처리한 데이터 중 위도와 경도의 데이터를 살펴보자.

## df\_univ\_loc의 위도, 경도 데이터 확인  
df\_univ\_loc |> select(ends\_with(c('위도', '경도'))) |> head()

## # A tibble: 6 x 2  
## WGS84위도 WGS84경도  
## <dbl> <dbl>  
## 1 37.7 127.  
## 2 37.6 127.  
## 3 37.7 127.  
## 4 37.4 127.  
## 5 37.3 127.  
## 6 37.3 127.

앞의 결과에서 보면 열 이름에 ‘WGS84’라는 문구가 들어가 있다. 또 위도와 경도가 우리가 아는 대충의 값과 비슷하다. 여기서 나오는 ’WGS84’가 흔히 우리가 사용하는 위도와 경도 좌표 시스템을 의미한다. 결국 앞선 geojson으로 저장된 객체에서는 ’WGS84’ 좌표계가 아닌 다른 좌표계를 쓰고 그 값이 ’WSG84’에 비해 매우 크기 떄문에 두 좌표계를 같이 쓰면 앞에서 보여진 것과 같이 좌측 하단 귀퉁이에 점들이 찍히게 된다.

따라서 sf 객체에 저장된 CRS를 ’WSG84’로 변환하야 한다. sf 객체의 좌표계를 바꾸는 데에는 st\_transform()을 사용한다.

st\_transform(x, crs = st\_crs(x))  
- x : 변환의 대상이 되는 sf 객체  
- crs : 목표 CRS 시스템의 값 설정

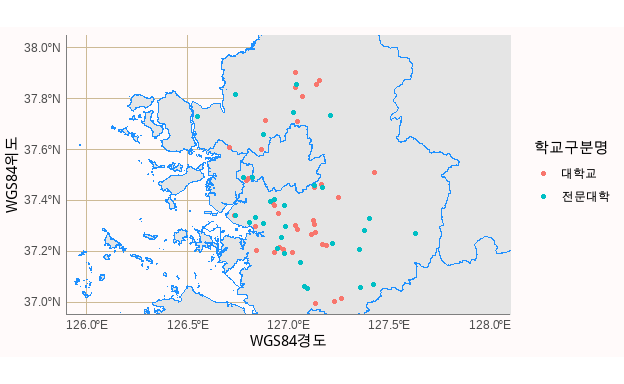
’WSG84’를 의미하는 CRS 코드 값은 ’EPSG:4326’이다. 이 코드 값으로 sf 객체의 좌표계를 바꾸는 코드는 다음과 같다.

## 좌표계를 'EPSG:4326'으로 바꾼 sf객체 생성  
spdf\_shp1 <- sf::st\_transform(spdf\_shp, crs = 'EPSG:4326')  
  
## 좌표계 확인  
st\_geometry(spdf\_shp1)

## Geometry set for 17 features   
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: 124.6097 ymin: 33.11371 xmax: 131.8728 ymax: 38.61371  
## Geodetic CRS: WGS 84  
## First 5 geometries:

앞의 결과를 보면 CRS가 변경되었음을 확인할 수 있고 MULTIPOLYGON의 X, Y 값도 위도, 경도에 적절한 값으로 바뀌었음을 알 수 있다. 이제 ggplot2로 다음과 같이 지도를 그리고 대학의 위치를 찍어볼 수 있다. 만약 데이터에 대학의 추가적 특성값(학생수, 교수수 등)의 데이터를 가지고 있다면 size를 매핑시켜 추가적인 데이터를 표현하는 지도(Bubble Map)로 시각화할 수 있다.

## 좌표계가 변환된 sf 객체를 사용  
spdf\_shp1 |>  
ggplot() +   
 ## simple feature 객체를 사용하여 geom\_sf 레이어를 생성  
 geom\_sf(color = 'dodgerblue') +   
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue') +  
 ## 대학의 좌표가 찍힌 데이터에서 X축을 경도, Y축을 위도, color를 학교구분명으로 매핑한 geom\_point 레이어 추가  
 geom\_point(data = df\_univ\_loc, aes(x = WGS84경도, y = WGS84위도, color = 학교구분명)) +   
 ## 경기도가 확대되도록 X축과 Y축의 범위를 설정  
 coord\_sf(xlim = c(126, 128), ylim = c(37, 38))



**실행결과 10-**대학 위치가 표현된 심볼 지도

# 육각 지도(Hexbin Map)

앞에서 그린 지도들은 매우 섬세한 지형의 지도이다. 우리나라처럼 섬이 많은 나라는 지도에 매우 많은 점들이 나타나서 지도가 다소 지저분하게 보일수도 있고 지도의 섬세함에 눈길을 뺏겨 데이터에 집중하지 못하는 경우도 있다. 이렇게 데이터에 집중하는 지도의 시각화를 위해 지도를 단순화시켜서 사용할 수 있다. 이럴때 사용하는 방법이 사각형이나 육각형으로 지도를 단순화하여 데이터를 표현하는 방법이다.

보통 육각지도는 행정구역나 일정 면적을 육각형으로 단순화하고 이들간의 상대적 위치에 육각형을 위치시킴으로써 지도를 단순화한다. 각 행정구역의 크기는 무시되고 모두 같은 크기의 육각형으로 표현된다. 이렇게 지도를 단순화하여 표기할 경우에는 지형이 가지는 면적이나 지역적 특성이 무시되기 떄문에 반드시 라벨을 붙여 주어야 한다는 점을 주의해야 한다.

사실 육각 지도는 우리나라 전체 지도에 적용하는 것에는 다소 무리가 따른다. 육각 지도는 각각의 지역이 포함되지 않고 위치할때 구현되기도 좋고 지역을 인지하기도 쉽다. 하지만 우리나라의 시도 지역은 포함되는 지역들이 있다. 서울과 인천은 경기도에 , 부산과 울산은 경상남도에 포함되어 있다. 또 대구, 광주, 대전, 세종도 각각 여러 도에 포함되어 있다. 육각 지도로 표현은 가능하지만 효과적이지는 않다. 하지만 서울의 각 구는 서로 포함되는 구가 없기 때문에 비교적 표현에 효과적이다. 따라서 여기서는 서을의 각 구를 육각 지도로 표현해보도록 하겠다.

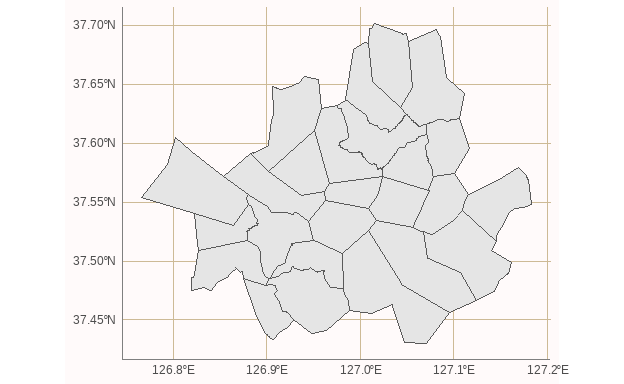
먼저 서울 지도에 표시할 데이터가 필요하다. 이 데이터는 행정구역별 대학 학과수 및 학년별 재학생수 데이터를 사용한다.[[6]](#footnote-38) 이 데이터 중 합계, 소계를 제거하고 서울지역 데이터만 필터링한다. 행정구역 열에 있는 문자열 중 필요없는 문자열을 제거(‘서울’)하고 사용할 열(시군구 행정구역 이름, 대학수)만 남긴다. 전처리 코드는 다음과 같다 .

## 다운로드 받은 엑셀파일에서 데이터를 읽어옴  
df\_행정구역 <- read\_excel('파일 저장경로/2021\_행정구역별 학과수 및 학년별 재적학생수.xlsx',   
 ## 'sheet0' 시트의 데이터를 불러오는데,  
 sheet = 'Sheet0',  
 ## 앞의 3행을 제외하고  
 skip = 3,   
 ## 첫번째 행은 열 이름이 아님을 설정  
 col\_names = FALSE,   
 ## 열의 타입을 설정, 처음 4개는 문자형으로 다음 39개는 수치형으로 설정  
 col\_types = c(rep('text', 4), rep('numeric', 39)))  
  
## 읽어온 데이터에서 '소계', '전체'에 해당하는 데이터는 제거하고 '서울'데이터만 필터링해서 필요한 열만 남김  
df\_행정구역 <- df\_행정구역 |> filter(...1 == '서울', ...2 != '소계', ...3 == '전체', ...4 == '소계') |>  
 select(2, 5)  
  
## 열 이름을 설정  
names(df\_행정구역) <- c('구이름', '대학수')  
  
## 구이름에서 필요없는 문자열 제거(gsub()는 특정 문자열을 바꾸어 주는 함수)  
df\_행정구역$구이름 <- gsub('서울 ', '', df\_행정구역$구이름)

다음으로 서울시 지도를 불러와야 한다. 여기서는 앞에서 geojson으로 읽어들인 sp 데이터 중에서 서울 데이터만 필터링한다. 이를 위해 sp 객체를 sf객체로 먼저 변환한다. 그래야 filter와 같은 tidyverse에서 제공하는 함수들을 사용할 수 있다. st\_as\_sf()를 사용하여 sf객체로 변환한다.

## st\_as\_sf()를 사용해 sp객체를 sf객체로 변환  
sf\_spdf\_geojson <- st\_as\_sf(spdf\_geojson)  
  
## 변환된 sf객체 중 서울 데이터(SIG\_CD < 12000)만 필터링  
sf\_spdf\_seoul <- sf\_spdf\_geojson |> filter(SIG\_CD < 12000)

sf\_spdf\_seoul |> ggplot() +  
 geom\_sf()

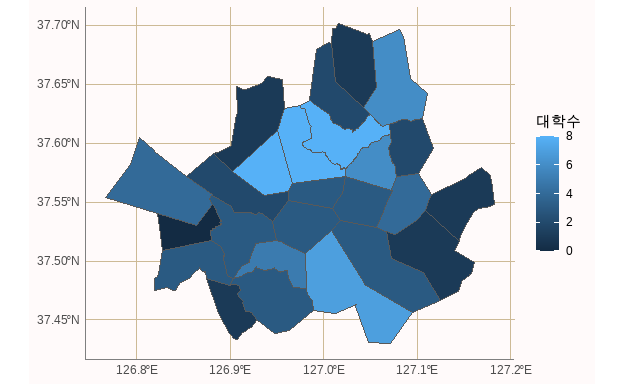


**실행결과 10-**서울시 구별 기본 지도

다음으로 전처리한 서울 데이터와 지형 데이터를 조인한다.

## sf 객체의 SIG\_KOR\_NM열과 전처리한 데이터프레임의 구이름 열을 기준으로 inner\_join()으로 조인  
sf\_spdf\_seoul\_joined <- inner\_join(sf\_spdf\_seoul, df\_행정구역, by = c('SIG\_KOR\_NM' = '구이름'))

sf\_spdf\_seoul\_joined |> ggplot() +  
 ## fill을 대학수로 매핑한 geom\_sf 레이어 생성  
 geom\_sf(aes(fill = 대학수))



**실행결과 10-**대학수가 표현된 서울시 구별 지도

이제 데이터가 표현된 서울 지도가 준비되었다. 이 맵을 육각 지도로 바꾸어 보겠다. 먼저 육각 지도의 표현에 필요한 geogrid 패키지를 설치해야 한다. geogrid 패키지는 지형과 같이 불규칙한 다각형(polygon)을 사각형 또는 육각형 그리드로 전환하는데 사용되는 함수들을 제공한다. 여기서는 calculate\_grid()와 assign\_polygons()를 사용한다. calculate\_grid()는 지형 다각형을 육각형이나 사각형 그리드로 변환하데 사용하고 assign\_polygons()은 calculate\_grid()에서 변환된 다각형의 위치를 지형 데이터에 매핑한 위치로 할당하는데 사용한다.

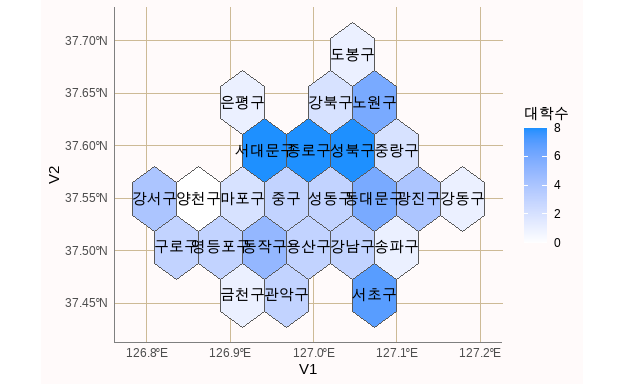
calculate\_grid(shape, learning\_rate = 0.03, grid\_type = c(“hexagonal”, “regular”), seed = NULL, verbose = FALSE)  
- shape : 다각형으로 표현할 SpatialPolygonsDataFrame나 sf 객체  
- learning\_rate : 최적의 셀 크기를 찾는 비율  
- grid\_type : 육각형 또는 사각형의 다각형 형태 설정  
- seed : 그리드 계산 알고리즘에 사용될 랜덤 시드값  
- verbose : 알고리즘 반복시 메세지 출력을 설정하는 논리값

assign\_polygons(shape, new\_polygons)  
- shape : 다각형으로 표현할 SpatialPolygonsDataFrame나 sf 객체  
- new\_polygons : calculate\_grid에서 생성된 geogrid 객체

calculate\_grid()와 assign\_polygons()을 사용하여 서울 지형을 육각형으로 변환하는 코드는 다음과 같다.

## geogrid 패키지 설치  
if(!require(geogrid)) {  
 install.packages('geogrid')  
 library(geogrid)  
}  
  
## calculate\_grid()를 사용하여 sf\_spdf\_seoul\_joined를 육각형 그리드형태로 변환  
new\_cells\_hex <- calculate\_grid(shape = sf\_spdf\_seoul\_joined, grid\_type = "hexagonal")  
  
## assign\_polygons()을 사용하여 sf\_spdf\_seoul\_joined위에 new\_cells\_hex을 할당  
resulthex <- assign\_polygons(sf\_spdf\_seoul\_joined, new\_cells\_hex)

## assign\_polygons()의 결과를 ggplot 객체 데이터로 사용  
resulthex |>  
 ggplot() +  
 ## fill을 대학수로 매핑한 geom\_sf 레이어 생성  
 geom\_sf(aes(fill = 대학수)) +   
 ## X축을 V1, Y축을 V2, label을 SIG\_KOR\_NM을 매핑한 geom\_text 레이어 추가  
 geom\_text(aes(x = V1, y = V2, label = SIG\_KOR\_NM)) +   
 ## fill 스케일을 흰색부터 dodgerblue까지 가지는 그래디언트로 설정  
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue')

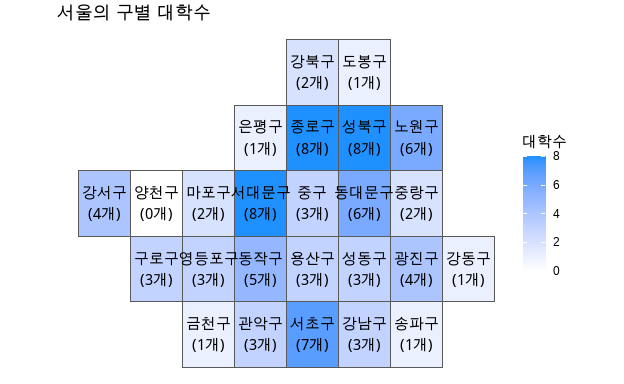


**실행결과 10-**대학수가 표현된 서울시 육각 지도

위의 육각 지도를 사각지도로 바꾸는 코드는 다음과 같다. 지도를 좀 더 알아보기 쉽게 그래프 테마요소를 모두 지우는 theme\_void()를 설정하고 적절한 라벨과 제목도 설정하였다.

## calculate\_grid()를 사용하여 sf\_spdf\_seoul\_joined를 사각형 그리드형태로 변환  
new\_cells\_reg <- calculate\_grid(shape = sf\_spdf\_seoul\_joined, grid\_type = "regular")  
  
## assign\_polygons()을 사용하여 sf\_spdf\_seoul\_joined위에 new\_cells\_reg을 할당  
resultreg <- assign\_polygons(sf\_spdf\_seoul\_joined, new\_cells\_reg)

## assign\_polygons()의 결과를 ggplot 객체 데이터로 사용  
resultreg |>  
 ggplot() +  
 ## fill을 대학수로 매핑한 geom\_sf 레이어 생성  
 geom\_sf(aes(fill = 대학수)) +   
 ## X축을 V1, Y축을 V2, label을 SIG\_KOR\_NM을 매핑한 geom\_text 레이어 추가  
 geom\_text(aes(x = V1, y = V2, label = paste0(SIG\_KOR\_NM, '\n(', 대학수, '개)'))) +   
 ## fill 스케일을 흰색부터 dodgerblue까지 가지는 그래디언트로 설정  
 scale\_fill\_gradient(low = 'white', high = 'dodgerblue') +   
 theme\_void() +   
 labs(title = '서울의 구별 대학수')



**실행결과 10-**대학수가 표현된 서울시 사각 지도

1. 여기서 사용하는 우리나라 지형의 Shape 파일은 <http://www.gisdeveloper.co.kr/?p=2332> 에서 다운로드 받았다. [↑](#footnote-ref-20)
2. <https://ko.wikipedia.org/wiki/GeoJSON> [↑](#footnote-ref-23)
3. <https://neurowhai.tistory.com/350> [↑](#footnote-ref-25)
4. S4 클래스는 R이 객체지향(Object Oriented) 언어의 컨셉을 구현하기 위해 사용되는 클래스이다. S4 클래스에 대한 설명은 이 책의 범위를 벗어나므로 더 이상의 설명은 생략하나 S4클래스를 더 자세히 알고 싶다면 Hadley Wickham의 ‘Advanced R’(<http://adv-r.had.co.nz/>)을 참조하라. [↑](#footnote-ref-27)
5. <https://data.gg.go.kr/portal/data/service/selectServicePage.do?page=1&rows=10&sortColumn=&sortDirection=&infId=V8VGMZPEPZH4XN27NXS021454377&infSeq=1&order=&loc=&SCHOOL_DIV_NM=&FACLT_NM=&REFINE_ROADNM_ADDR=> [↑](#footnote-ref-34)
6. <https://kess.kedi.re.kr/stats/school?menuCd=0102&cd=5511&survSeq=2021&itemCode=01&menuId=m_010207&uppCd1=010207&uppCd2=010207&flag=A> [↑](#footnote-ref-38)