LAUS 겨울방학 R+GIS 공부하기

R을 이용한 공간정보 분석

3장 R의 공간정보 데이터 구조

2025.01.10.

목 차

00 개요

***패키지란?**
R에서 제공하는 기능(함수, 데이터셋, 문서 등)의 모음. 특정 작업(예: 데이터 시각화, 통계 분석)을 쉽게 수행하도록 도와주는 역할
기본 설치로 제공되지 않는 추가 기능을 제공

```
# 패키지 설치
install.packages("패키지이름")

# 패키지 로드
library(패키지이름)
```

***클래스란?**
객체지향 프로그래밍에서 데이터와 그 데이터를 처리하는 메서드를 포함하는 구조
R에서 객체는 특정 클래스를 가지며, 이는 데이터의 속성과 사용 가능한 메서드를 정의

```
# data.frame: 행렬 형태로 데이터 저장
# matrix: 수치형 데이터를 저장하는 2차원 배열
# Sf: 공간 데이터를 다루는 객체 클래스
# raster: 레스터 데이터를 저장하는 객체 클래스
```

***객체란?**
데이터를 저장하거나 조작하기 위해 생성된 모든 것을 의미
R에서 변수를 생성하거나 값을 계산하거나 데이터를 분석할 때 사용하는 **모든 데이터와 구조**가 객체로 간주됨

ㄴ구조의 종류
벡터 : 동일한 데이터 유형을 가진 1차원 배열 ("numeric", "character", "logical", "integer", "factor" 등)
리스트 : 서로 다른 데이터 유형을 포함할 수 있는 1차원 데이터 구조 (“list”)
매트릭스(행렬) : 동일한 데이터 유형을 가진 2차원 배열 (“matrix”)
데이터프레임 : 열마다 다른 데이터 유형을 가질 수 있는 2차원 구조(“data.frame”)

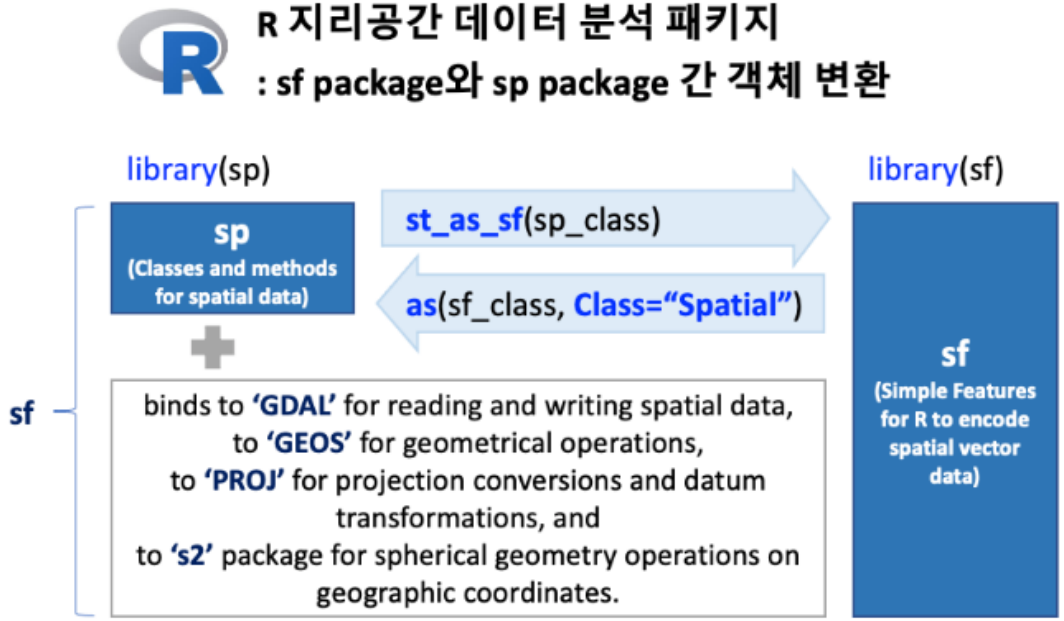
*데이터 요약 및 구조를 알고자 할 때 사용하는 함수

함수명	기능
str()	데이터 구조, 자료형 등을 요약하여 출력
class()	데이터 구조 출력
dim()	차원 출력 (행 및 열의 수)
nrow()	행의 수 출력
ncol()	열의 수 출력
names()	열 이름 출력, colnames()와 동일
head()	데이터셋의 앞부분 일부 출력
tail()	데이터셋의 뒷부분 일부 출력

```
#(1) RTools 패키지 설치 (Windows 사용자만 해당)
https://cran.r-project.org/bin/windows/Rtools/ 에서 운영체제에 맞는 버전 설치

# (2) 공간분석 및 시각화 관련 패키지 설치
# a. 공간분석 기본 패키지
install.packages(c("sp", "rgdal", "rgeos"))

# 참고: rgdal, rgeos는 2023년 10월 이후 CRAN 지원 중단
# 대안 패키지 설치: sf, terra
install.packages("remotes")
remotes::install_github("r-spatial/sf")
install.packages("terra", type = "source")

# b. 시각화 패키지
install.packages(c("ggmap", "tmap"))
# tmap 최신버전 필요시:
remotes::install_github('r-tmap/tmap')

# c. 공간통계 패키지
install.packages(c("spatstat", "spdep"))
install.packages('spDataLarge',
                 repos='https://nowosad.github.io/drat/',
                 type='source')

# d. 래스터 및 공간보간 패키지
install.packages(c("raster", "gstat", "spgwr"))
```

sp 패키지
: 공간정보를 정의하고 공간 분석과 모델링에 필요한 기능을 제공

Spatial 클래스
: 공간정보 데이터 유형 정의하는 클래스
: 공간정보의 형태에 따라 점, 선 면 형태의 벡터 데이터와 화소 형태의 래스터 데이터로 구분



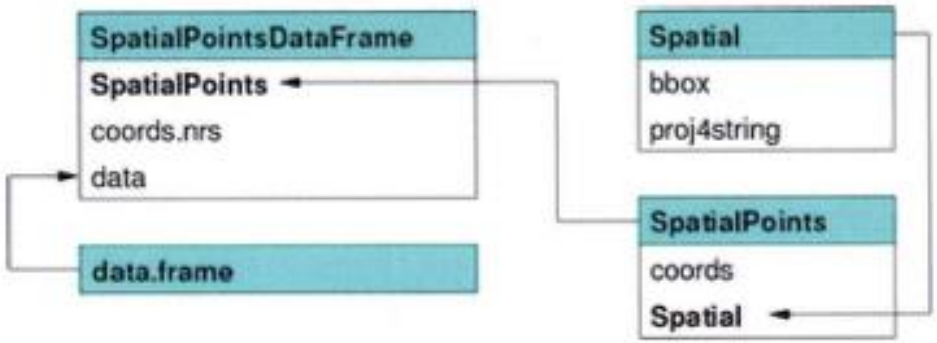
```
#(1) RTools 패키지 설치 (Windows 사용자만 해당)
https://cran.r-project.org/bin/windows/Rtools/ 에서 운영체제에 맞는 버전 설치

# (2) 공간분석 및 시각화 관련 패키지 설치
# a. 공간분석 기본 패키지
install.packages(c("sp", "rgdal", "rgeos"))

# 참고: rgdal, rgeos는 2023년 10월 이후 CRAN 지원 중단
# 대안 패키지 설치: sf, terra
install.packages("remotes")
remotes::install_github("r-spatial/sf")
install.packages("terra", type = "source")

# b. 시각화 패키지
install.packages(c("ggmap", "tmap"))
# tmap 최신버전 필요시:
remotes::install_github('r-tmap/tmap')

# c. 공간통계 패키지
install.packages(c("spatstat", "spdep"))
install.packages('spDataLarge',
                 repos='https://nowosad.github.io/drat/',
                 type='source')

# d. 래스터 및 공간보간 패키지
install.packages(c("raster", "gstat", "spgwr"))
```

	sp/rgdal/rgeos	sf	terra
벡터 데이터 처리	sp+rgeos	직관적인 data.frame 처리	지원 가능하나 벡터보다는 레스터에 강점
벡터 데이터 파일	rgdal::readOGR()	sf::st_read()	terra::vect()
레스터 데이터 처리	rgdal::readGDAL() + raster	제한적 (stars와 함께 사용 가능)	효율적이고 빠름, 대규모 데이터 지원
좌표계 변환	rgdal 및 sp::proj4string	sf::st_crs()	terra::crs()
공간 연산	rgeos	고급 공간 연산	레스터 연산

01 점 데이터 구조 p.37–39

점 데이터의 구조

SpatialPoints	SpatialPointsDataFrame
• 단순히 점 데이터의 위치만을 좌표값으로 표현 • 좌표값은 CRS 클래스 (coordinate referencing system) 에 의해 정의됨 • 공간 범위를 표현하는 bbox + 좌표체계를 나타내는 Spatial 클래스	점 데이터의 위치(SpatialPoints) + 데이터프레임 형태의 속성정보 (data.frame)

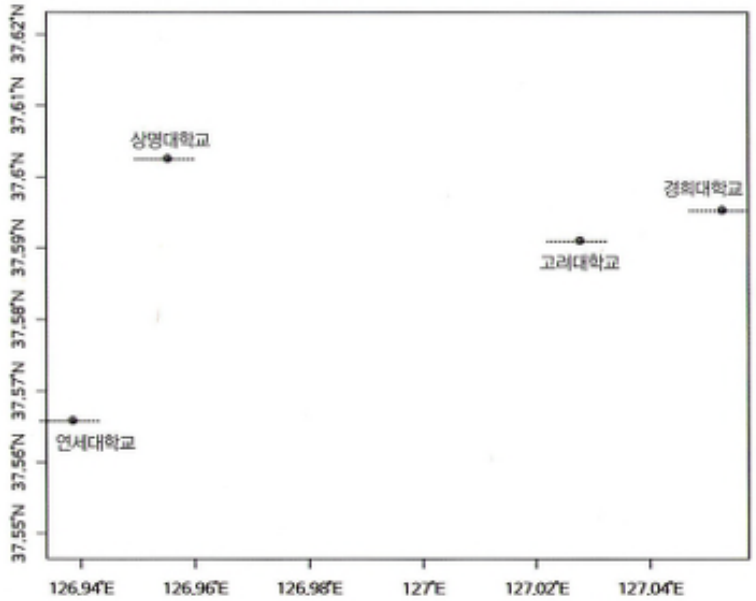


〈그림 3.1〉 점 객체의 클래스 구성도

출처: Bivand et, al,, Applied spatial data analysis with R, p. 35

01 점 데이터 구조 p.37–39

```r
#3.1. 점데이터 구조 : 대학교별 위치 지도 그리기#

#데이터프레임에 x좌표와 y좌표를 담고 있는 필드 구성
x<-c(126.9552, 127.0526, 126.9385, 127.0277)
y<-c(37.6026, 37.5954, 37.5659, 37.5911)
name<-c("상명대학교","경희대학교","연세대학교","고려대학교")
univ<-data.frame(Longtitude=x, Latitude=y, Name = name)

#CRS 함수를 이용한 좌표체계 정보 저장
cs <- CRS("+proj=longlat +datum=WGS84")

# 좌표를 지정하여 SpatialPoints 객체 생성
coordinates(univ) <- ~Longtitude + Latitude

#좌표값을 가진 데이터프레임에 좌표체계 정보를 추가하여 SpatialPoints 객체 생성
sp<- SpatialPoints(univ, proj4string=cs)

#SpatialPoints 객체에 속성 데이터를 결합하여 위치정보와 속성정보가 결합된 공간정보 구성
spdf <- SpatialPointsDataFrame(univ, data.frame(Name = name))

#지금까지 과정을 통해 생성된 공간 객체와 공간 데이터프레임 객체를 지도로 출력
plot(spdf, axes=T, pch=10)
text(spdf,name)
```



〈그림 3.2〉 점 객체(SpatialPointsDataFrame) 출력 결과

02 선 데이터 구조 p.39–42

Lines

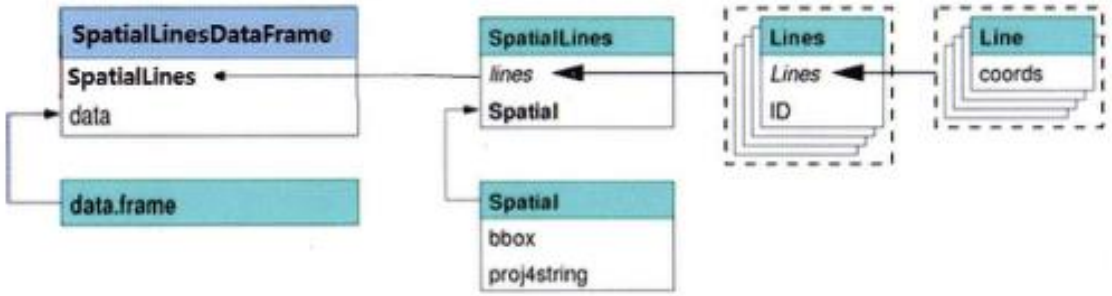
여러 개의 선 데이터가 모여
하나의 선 사상을 표현
GIS 에서는 멀티파트(multi part)라 부름
(선 데이터를 화면에서 선택했는데 다수의 선 데이터가 선택되는 경우)

SpatialLines

생성된 Lines 객체를 다시 합치고
좌표체계 정보를 부여

SpatialLinesDataFrame

SpatialLines 객체와 데이터프레임의
속성정보를 결합하여 선 데이터를 생성

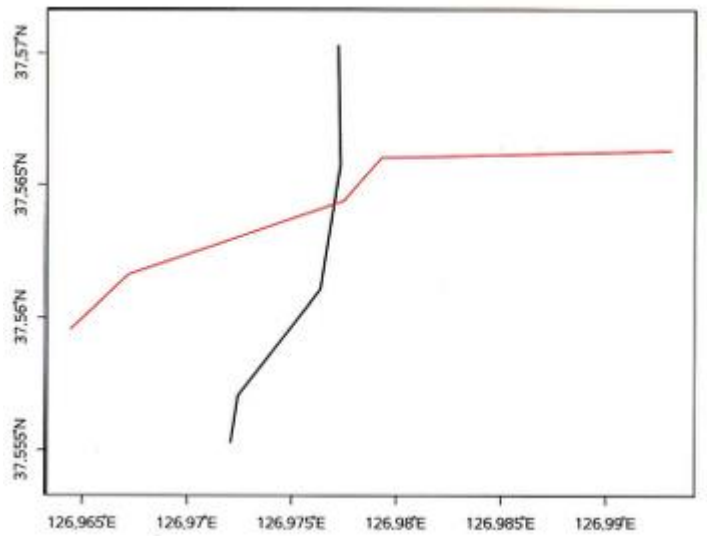


〈그림 3.3〉 선 객체의 클래스 구성도

출처: Bivand et, al,, Applied spatial data analysis with R, p. 40 재구성

02 선 데이터 구조 p.39–42

```r
#3.2.선데이터 구조 : 서울시 지하철 1호선과 2호선 일부 구간을 선 객체로 생성하기#

#선 데이터 x,y좌표를 각각 벡터로 구성하고 cbind 함수로 x,y좌표를 합쳐 선 데이터 좌표값을 메트릭스로 생성
x1<-c(126.9720783,126.9724216, 126.9763698, 126.9773139, 126.9771953)
y1<-c(37.5552612, 37.5570503, 37.5610647, 37.5657592, 37.5702657)
l1<-cbind(x1,y1)
x2<-c(126.9644515, 126.9671981, 126.9774978, 126.9793002, 126.9931189)
y2<-c(37.5595652, 37.5616064, 37.5643959, 37.5660287, 37.5663008)
l2<-cbind(x2,y2)

#Line 함수를 이용하여 각각의 좌표값으로 이루어진 변수를 선 객체로 작성
ln1<-Line(l1)
ln2<-Line(l2)

#Lines 함수를 이용하여 Line 객체를 list함수로 묶어 Lines 객체 생성
#해당 사례의 경우 멀티 파트를 구성하지 않으므로, list 함수에서 하나의 객체만 설정
lns1<-Lines(list(ln1), ID=1)
lns2<-Lines(list(ln2), ID=2)

#CRS 함수를 이용하여 좌표체계 정보 정의, SpatialLines 함수를 이용하여 Lines 객체를 SpatialLines 객체로 작성
cs <- CRS("+proj=longlat +datum=WGS84")
slns<-SpatialLines(list(lns1,lns2), proj4string=cs)

#SpatialLines 객체에 속성 데이터를 결합하여 위치정보와 속성정보가 결합된 공간정보를 구성
subno<-data.frame(ID=c(1,2), name=c("1호선","2호선"))
slnsdf<-SpatialLinesDataFrame(slns, data=subno)

#결과를 지도로 출력
plot(slnsdf,axes=T,col=1:2)
```



〈그림 3.4〉 선 객체(SpatialLines DataFrame) 출력 결과

03 면 데이터 구조 p.42–45

일반적으로 GIS 에서 면 데이터의 모양은 꼭짓점의 좌표값을 가진 다각형의 형태로 표현
단, 첫 번째 꼭짓점과 마지막 꼭짓점의 좌표값은 일치해야 함

Polygons

다각형의 꼭짓점을 좌표값으로 구성하여 Polygon 객체를 구성 후, 이를 합쳐 Polygons 객체 생성

←

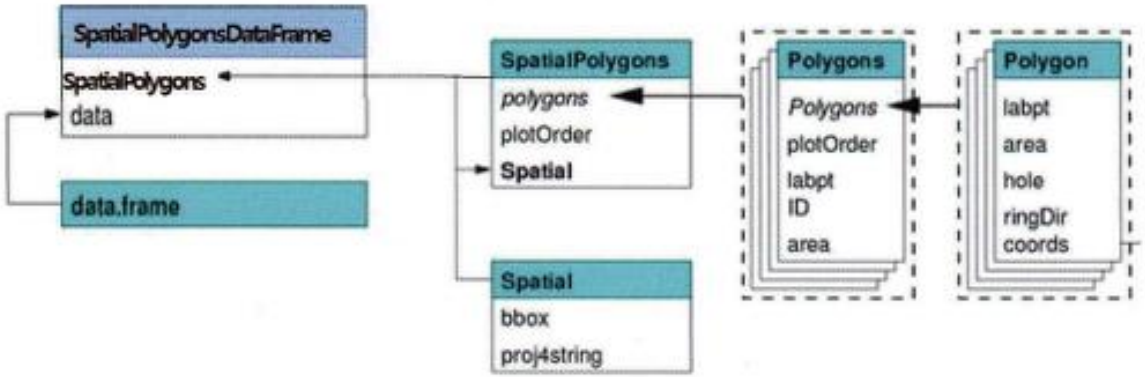
SpatialPolygons

생성된 Polygons 객체를 다시 합치고 좌표체계 정보를 부여

←

SpatialPolygonsDataFrame

SpatialPolygons 객체와 데이터프레임의 속성정보를 결합하여 면 데이터를 생성

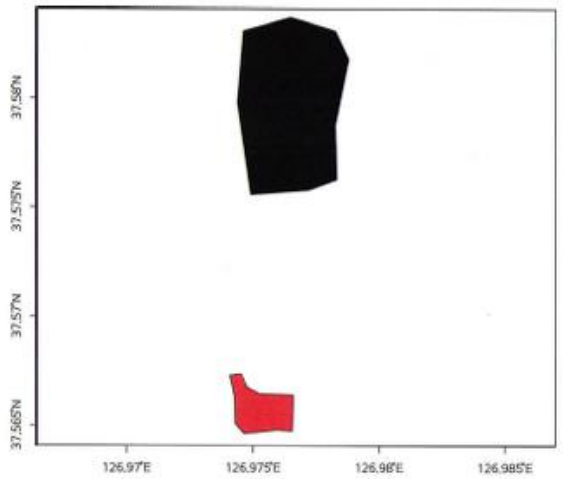


〈그림 3.5〉 면 객체의 클래스 구성도

출처: Bivand et al., Applied spatial data analysis with R, p.40 재구성

03 면 데이터 구조 p.42–45

```r
#3.3.면데이터 구조 : 서울시 경복궁과 덕수궁 모양을 면 객체로 표현하기#

#x,y좌표 벡터로 구성, 다각형 꼭지점 좌표값을 매트릭스로 작성
x1=c(126.9744937, 126.9737212,126.9740645,126.9768111,126.979386,126.9801585,126.979386,126.9794719,126.9778411)
y1=c(37.5756889, 37.5799063, 37.5831712, 37.5837834, 37.5831712, 37.5818789, 37.578886, 37.5763691, 37.5758929)
p1=cbind(x1,y1)

x2=c(126.9769001,126.9769538,126.9750011,126.9742823, 126.973939, 126.9732845, 126.9735527, 126.9736064, 126.9741106, 126.9
y2=c(37.5648948, 37.5665361, 37.5665956, 37.5668933, 37.5675056, 37.5674545, 37.5664851, 37.5652264, 37.5647757, 37.5649458
p2<-cbind(x2,y2)

#polygon 함수를 이용하여 각각의 좌표값으로 이루어진 변수를 면 객체로 작성
pl1<-Polygon(p1) #p1 변수 좌표값 면 객체 설정
pl2<-Polygon(p2) #p2 변수 좌표값 면 객체 설정

#Polygons 함수를 이용하여 Polygon 객체를 list 함수로 묶어 생성
polys1<-Polygons(list(pl1),ID=1)
polys2<-Polygons(list(pl2),ID=2)

#CRS 함수를 이용하여 좌표체계 정보 정의, SpatialPolygons 함수를 이용하여 Polygons 객체를 SpatialPolygons 객체로 작성
cs <- CRS("+proj=longlat +datum=WGS84")
spolys <- SpatialPolygons(list(polys1,polys2), proj4string=cs)

#SpatialPolygons 객체에 속성 데이터 결합, 위치정보와 속성정보가 결합된 공간정보 구성
palace<-data.frame(ID=c(1,2), name=c("경복궁","덕수궁"))
spolysdf <- SpatialPolygonsDataFrame(spolys, data=palace)

#지도로 출력
plot(spolysdf, axes=T, col=1:2)
```



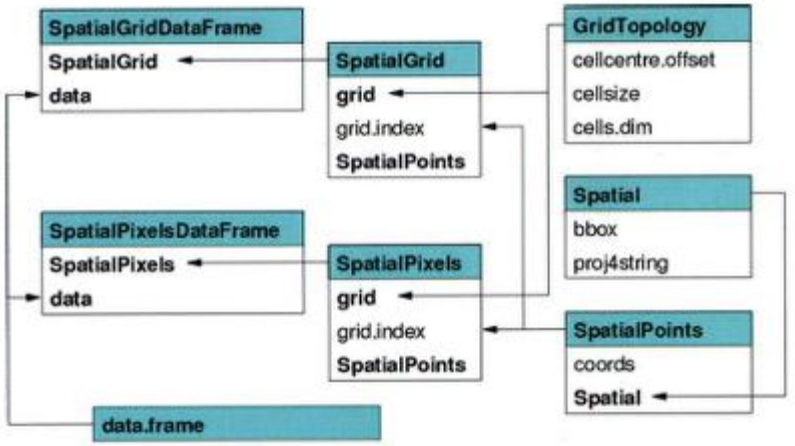
〈그림 3.6〉 면 객체(SpatialPolygonsDataFrame) 출력 결과

04 레스터 데이터 구조 p.45–48

래스터 데이터는 대상공간을 일정 크기의 격자(래스터)로 분할하고 각 격자의 값을 표현하는 구조
래스터를 표현하기 위해서는 각 격자의 값을 표현하는 데이터와 함께 격자의 위치와 해상도에 대한 정보가 필요

*래스터 데이터를 표현하는 방법
1) Spatial 클래스에서 레스터 객체를 표현하는 경우 〈참고만〉
- grid 객체가 격자 데이터를 표현하는 부분이며, GridTopology 클래스가 격자의 위치와 해상도를 표현
- 래스터 클래스는 SpatialGrid와 SpatialPixels 클래스로 구분

: 두 클래스 모두 래스터 데이터를 구현하기 위하여 만들어진 클래스이며 , 각 클래스마다 속성 정보가 추가되면 각각 SpatialGridDataFrame과 SpatialPixelDataFrame 클래스로 생성
:SpatialGrid 클래스와 SpatialPixels 클래스의 차이는 각 격자마다 좌표값이 주어지는가에 있음



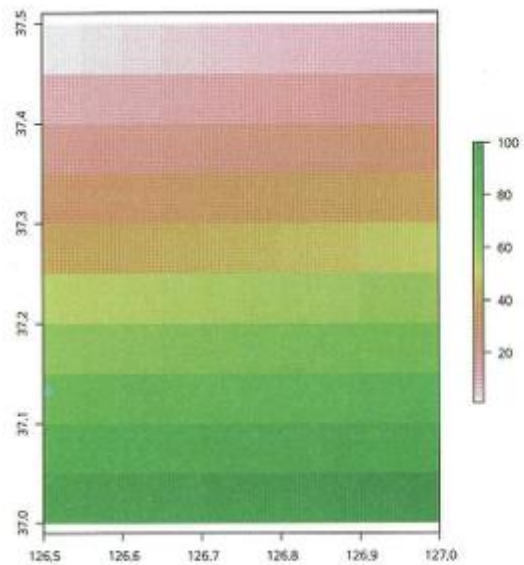
〈그림 3.7〉 래스터 객체의 클래스 구성도
출처: Bivand et al., Applied spatial data analysis with R, p.52

2) raster 패키지로 레스터 데이터를 처리하는 경우 〈실습에서 다룰 예정〉
raster 패키지에서는 3가지 클래스로 레스터 데이터를 표현
① RasterLayer : 단일 레스터 레이어(하나의 주제만을 담은 레스터 데이터)를 표현,
각 격자의 값과 함께 레스터 데이터를 구성하는 행과 열의 수, 공간적 범위와 좌표체계 등을 담고 있음
② RasterStack : 단일 레스터인 RasterLayer 를 합쳐 여러 주제를 가진 다중 레이어의 레스터 데이터를 표현
공간 범위와 해상도가 동일한 RasterLayer 의 집합
③ RasterBrick : 하나의 파일만을 이용해 레스터 데이터를 불러오는 방식으로 앞선 두 방식보다 자료 처리가 효율적

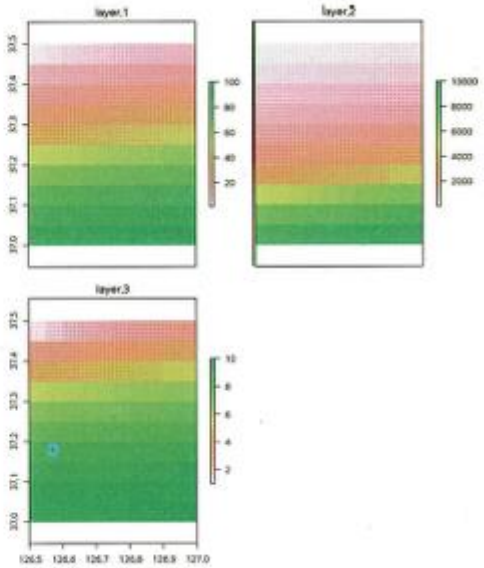
R에서는 래스터 데이터의 크기와 해상도, 영역을 지정 →각 격자에 해당하는 속성값을 부여하여 래스터 데이터를 생성

04 레스터 데이터 구조 p.45–48

```
#3.4. 레스터 데이터 구조 : RasterLayer, RasterBrick 비교#
#RasterLayer 객체(r) 생성
r<-raster(ncol=10, nrow=10, xmn=126.5, xmx=127, ymn=37, ymx=37.5)

#values함수를 이용하여 레스터 데이터의 격자에 값을 부여(1~100까지 일련번호)
values(r)<-1:100

#단일 레이어의 레스터 파일을 이용하여 다중 레이어의 레스터 파일 작성
r2<-r*r
r3<-sqrt(r)
rs<-stack(r,r2,r3)

#brick 함수를 이용하여 RasterStack 객체(rs)로부터 RasterBrick 객체(rb) 생성
rb<-brick(rs)

#RasterLayer, RasterBrick 출력
plot(r)
plot(rb)
```



〈그림 3.8〉 RasterLayer 출력 결과



〈그림 3.9〉 RasterBrick 출력 결과

05 좌표체계의 정의와 변환 p.48–52

〈표 3.1〉 우리나라에서 주로 사용되는 좌표계와 EPSG 코드

좌표계 이름		EPSG 코드	매개변수	비고
WGS84 좌표계		4326	+proj=longlat +ellps=WGS84 +datum=WGS84	경위도 좌표계
초기 TM 좌표계	서부원점	2098	+proj=tmerc +lat_0=38 +lon_0=125 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	2000년 이전 국가기본도에서 사용하던 좌표계. bessel 1841 타원체 이용
	중부원점	2097	+proj=tmerc +lat_0=38 +lon_0=127 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	
	동부원점	2096	+proj=tmerc +lat_0=38 +lon_0=129 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	
2002년 이전 TM 좌표계	서부원점	5173	+proj=tmerc +lat_0=38 +lon_0=125.0028902777778 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	경도 원점의 오류를 보정한 TM 좌표계. bessel 1841 타원체 이용
	중부원점	5174	+proj=tmerc +lat_0=38 +lon_0=127.0028902777778 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	
	동부원점	5176	+proj=tmerc +lat_0=38 +lon_0=129.0028902777778 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	
	동해원점	5177	+proj=tmerc +lat_0=38 +lon_0=131.0028902777778 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m	
현재 TM 좌표계	서부원점	5185	+proj=tmerc +lat_0=38 +lon_0=125 +k=1 +x_0=200000 +y_0=600000 +ellps=GRS80 +units=m	2002년 이후 국가기본도에서 사용하는 좌표계. 타원체를 GRS80으로, 북쪽방향 가산값을 600,000으로 변경
	중부원점	5186	+proj=tmerc +lat_0=38 +lon_0=127 +k=1 +x_0=200000 +y_0=600000 +ellps=GRS80 +units=m	
	동부원점	5187	+proj=tmerc +lat_0=38 +lon_0=129 +k=1 +x_0=200000 +y_0=600000 +ellps=GRS80 +units=m	
	동해원점	5188	+proj=tmerc +lat_0=38 +lon_0=131 +k=1 +x_0=200000 +y_0=600000 +ellps=GRS80 +units=m	
UTM-K(bessel)		5178	+proj=tmerc +lat_0=38 +lon_0=127.5 +k=0.9996 +x_0=1000000 +y_0=2000000 +ellps=bessel +units=m	베셀 타원체를 이용한 단일 원점 체계
UTM-K(GRS80)		5179	+proj=tmerc +lat_0=38 +lon_0=127.5 +k=0.9996 +x_0=1000000 +y_0=2000000 +ellps=GRS80 +units=m	GRS80 타원체를 이용한 단일 원점 체계
KATEC			+proj=tmerc +lat_0=38 +lon_0=128 +k=0.9999 +x_0=400000 +y_0=600000 +ellps=bessel +units=m	자동차 네비게이션용 비공식 좌표계

05 좌표체계의 정의와 변환 p.48–52

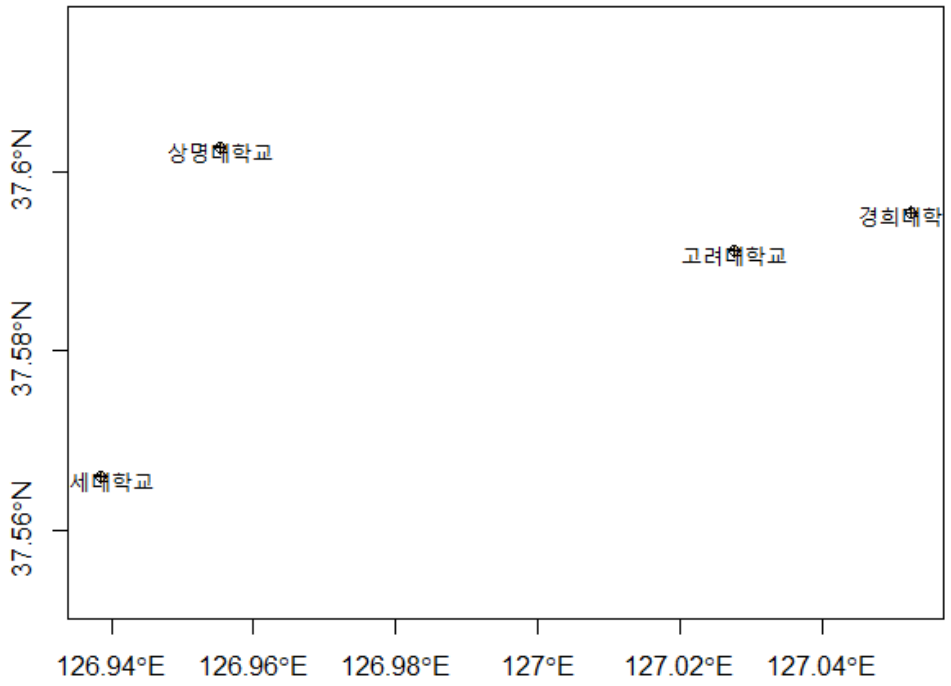
```r
#3.5.좌표체계의 정의와 변환#
#CRS("+매개변수=값") 형태로 작성 : CRS함수는 앞에서 정의한 EPSG 코드에서 정의한 매개변수 값을 정의함으로써 수행됨
CRS("+proj=longlat +datum=WGS84") #WGS84 좌표계
CRS("+proj=utm +zone=51 +datum=WGS84") # UTM 좌표계 (우리나라는 51번, 52번 구역을 사용하며 해당 코드는 51번 구역)
CRS("+proj=tmerc +lat_0=38 +lon_0=127 +k=1 +x_0=200000 +y_0=600000 +ellps=GRS80 +units=m") #우리나라 중부원점의 TM좌표계
CRS("+proj=tmerc +lat_0=38 +lon_0=127.5 +k=0.9996 +x_0=1000000 +y_0=2000000 +ellps=GRS80 +units=m") #단일원점 좌표체계인 UTM-K의 좌표체계

cs = CRS("+proj=longlat +datum:WGS84")
sp<-SpatialPoints(univ, proj4string=cs) #univ 라는 데이터프레임에 sp 라는 공간객체를 정의하고자 CRS에서 정의한 좌표체계 cs를 공간객체의 좌표체계로 부여

cs = CRS("+proj=longlat +datum=WGS84")
proj4string(spdf) = cs #proj4string 으로 기존 공간 객체(spdf)에 새로이 좌표체계 정보(cs) 부여

cs2 = CRS("+proj=tmerc +lat_0=38 +lon_0=127.5 +k=0.9996 +x_0=1000000 +y_0=2000000 +ellps=GRS80 +units=m")
spdf2 = spTransform(spdf, cs2) #좌표체계 변환을 통한 공간객체 좌표값 변경(spTransform 함수 사용)
```

06 R을 이용한 공간정보 생성 실습 p.48–52

〈그림 3.11〉 구글 맵에서 좌표값의 확인

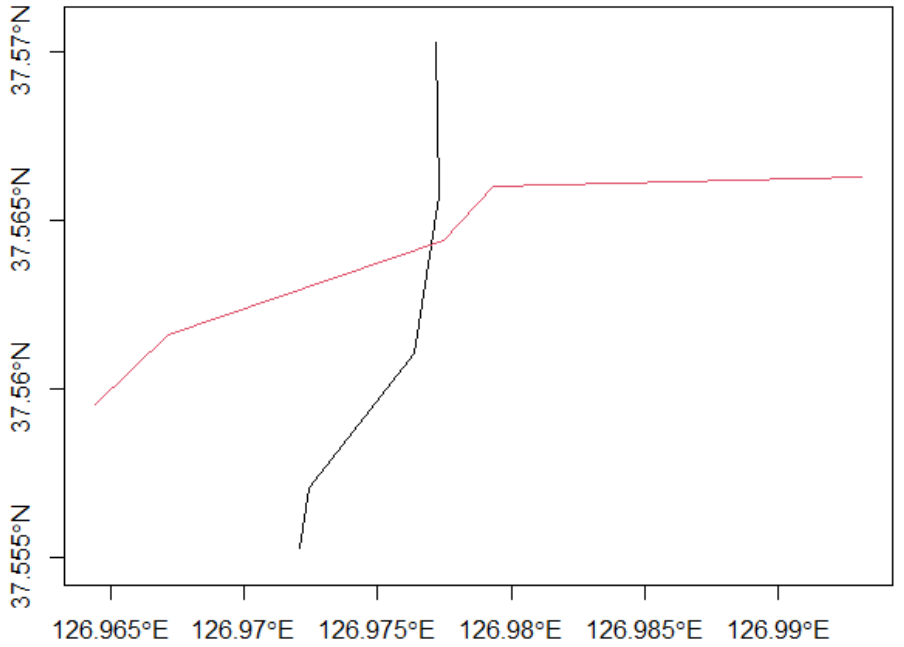
구글 맵에서 좌표값을 확인

```r
#3.6.1. 포인트 객체 만들기#

#4개 지점을 선택해 위도와 경도 설정
library(sp)
x <- c(126.9552,127.0526,126.9385,127.0277) #벡터x작성
y <- c(37.6026,37.5954,37.5659,37.5911) #벡터y작성
name <- c("상명대학교","경희대학교","연세대학교","고려대학교") #벡터 name 작성

#data.frame 함수를 이용해 벡터를 데이터프레임으로 변환하고 포인트 객체로 만들기
univ <- data.frame(Longitude=x, Latitude=y) #데이터프레임 작성
cs <- CRS("+proj=longlat +datum=WGS84") #좌표계 정의
sp <- SpatialPoints(univ,proj4string=cs) #SpatialPoints 생성

#완성된 포인트 객체에 위치의 이름을 속성정보로 부여하여 포인트 객체 데이터프레임 작성, 출력
spdf <- SpatialPointsDataFrame(sp, data=data.frame(Name=name)) #속성정보 부여
plot(spdf, axes=T, pch=10) #그래프 출력
text(spdf,name)
```



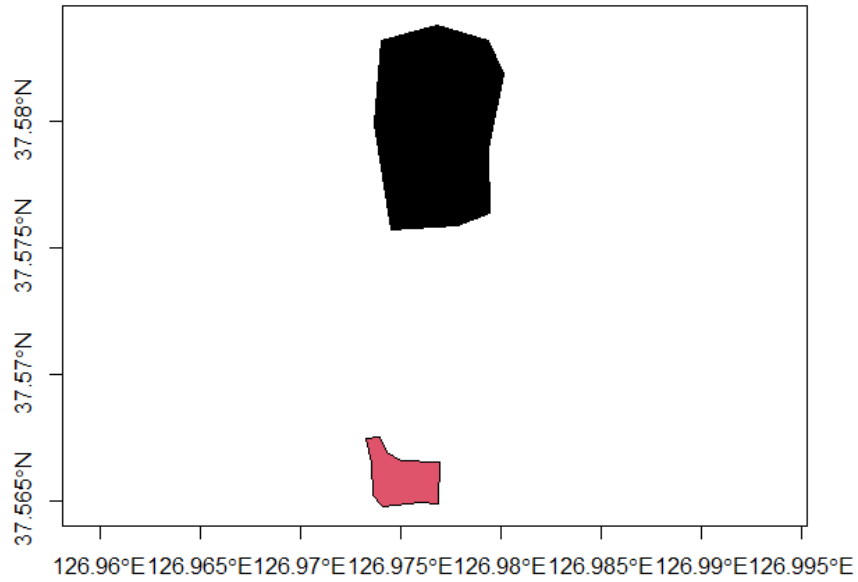
06 R을 이용한 공간정보 생성 실습 p.48–52

```r
#3.6.2. 선 객체 만들기#

#5개 꼭짓점으로 구성된 2개의 선으로부터 각각 위도, 경도 값을 작성(꼭짓점 좌표 취득)
library(sp)
x1<-c(126.9720783,126.9724216, 126.9763698, 126.9773139, 126.9771953)
y1<-c(37.5552612, 37.5570503, 37.5610647, 37.5657592, 37.5702657)
x2<-c(126.9644515, 126.9671981, 126.9774978, 126.9793002, 126.9931189)
y2<-c(37.5595652, 37.5616064, 37.5643959, 37.5660287, 37.5663008)

#x,y벡터를 하나의 벡터(Line 객체)로 합치기(l1, l2)
l1 <- cbind(x1,y1) #x1,y1좌표로 구성된 매트릭스 작성
ln1 <- Line(l1)
l2 <- cbind(x2,y2) #x2,y2 좌표로 구성된 매트릭스 작성
ln2 <- Line(l2)

#각각의 Line 객체를 Lines 객체로 변환한 후 공간객체로 변환
lns1 <- Lines(list(ln1), ID=1)
lns2 <- Lines(list(ln2), ID=2) #Lines객체 작성
cs <- CRS("+proj=longlat +datum=WGS84") #좌표계 정의
slns <- SpatialLines(list(lns1,lns2), proj4string=cs) #SpatialLines 작성

#각 선의 이름을 속성정보로 부여, 선 객체 데이터프레임 작성 및 출력
subno <- data.frame(ID= c(1,2), name=c("1호선","2호선")) #데이터프레임 작성
slnsdf <- SpatialLinesDataFrame(slns, data=subno) #SpatialLinesDataFrame 작성
plot(slnsdf, axes=T, col=1:2) #그래프 출력
```



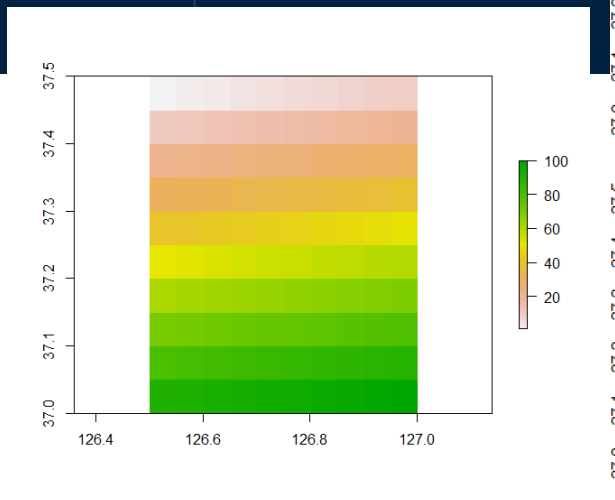
06 R을 이용한 공간정보 생성 실습 p.48–52

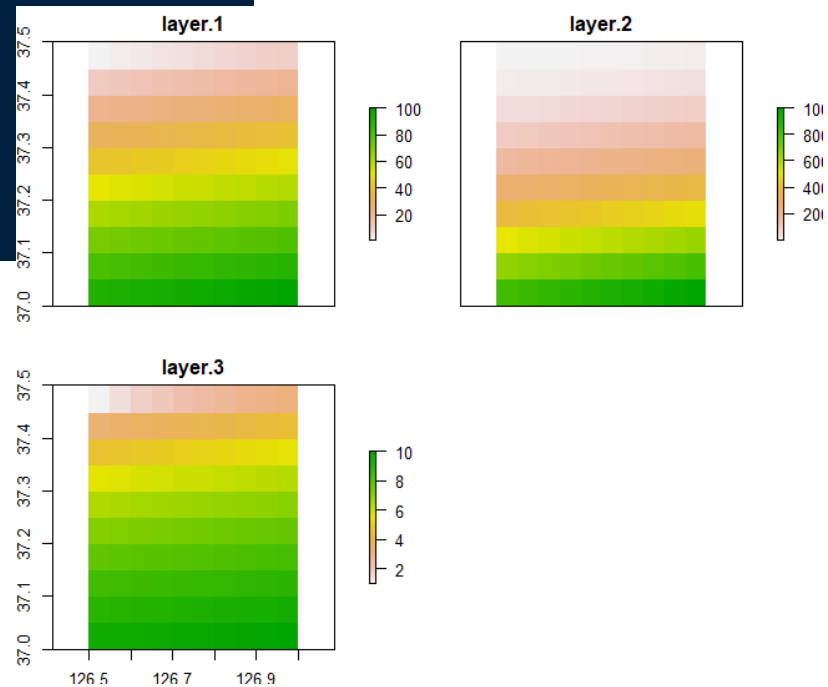
```r
#3.6.3. 면 객체 만들기#
library(sp)
x1 = c(126.9744937,126.9737212,126.9740645,126.9768111,126.979386, 126.9801585,126.979386,126.9794719,126.9778411)
y1 = c(37.5756889, 37.5799063, 37.5831712, 37.5837834, 37.5831712, 37.5818789, 37.578886, 37.5763691, 37.5758929)
x2 = c(126.9769001, 126.9769538, 126.9750011, 126.9742823, 126.973939, 126.9732845, 126.9735527, 126.9736064, 126.9741106, 126.9759989)
y2 = c(37.5648948, 37.5665361, 37.5665956, 37.5668933, 37.5675056, 37.5674545, 37.5664851, 37.5652264, 37.5647757, 37.5649458)

p1<-cbind(x1,y1)
pl1<-Polygon(p1) #x1,y1좌표로 구성된 메트릭스 작성
p2 <-cbind(x2,y2)
pl2<-Polygon(p2) #x2,y2 좌표로 구성된 메트릭스 작성

polys1 <- Polygons(list(pl1), ID=1)
polys2 <- Polygons(list(pl2), ID=2) # Polygons 객체 생성
cs <- CRS("+proj=longlat +datum=WGS84") #좌표계 정의
spolys <- SpatialPolygons(list(polys1,polys2),proj4string=cs) #SpatialPolygons 생성

palace <- data.frame(ID=c(1,2), name=c("경복궁","덕수궁"))
spolysdf <- SpatialPolygonsDataFrame(spolys, data=palace) #SpatialPolygonsDataFrame 작성
plot(spolysdf, axes=T, col=1:2) #그래프로 출력
```



06 R을 이용한 공간정보 생성 실습 p.48–52

```r
#3.6.4. 레스터 객체 만들기#

#레스터 행렬 생성
library(raster)
r <- raster(ncol=10,nrow=10, xmn=126.5, xmx=127, ymn=37, ymx=37.5) #RasterLayer 생성(책에서는 ymxplo 로 나오는데, 해당 ra

#Raster Layer 생성
values(r) <- 1:100 #레스터 레이어의 셀 값을 1부터 100까지의 숫자로 채우기(이 데이터가 하나의 Raster Layer 에 해당)
r2 <- r*r #각 셀의 값을 제곱한 데이터 생성
r3 <- sqrt(r) #각 셀의 값을 제곱근한 데이터 생성

#Raster Layer를 다중 레스터, Raster Brick 으로 생성
rs <- stack(r,r2,r3) # RasterStack 생성
rb <- brick(rs) #RasterBrick 생성
plot(r) #RasterLayer 출력
plot(rb) #RasterBrick 출력
```





06 R을 이용한 공간정보 생성 실습 p.48–52

```r
#3.6.5. 공간 데이터 좌표 변환#
cs2 = CRS("+proj=tmerc +lat_0=38 +lon_0=127.5 +k=0.9996 +x_0=1000000 +y_0=2000000 +ellps=GRS80 +units=m") #UTM-K로 좌표체계 정의

spdf2 = spTransform(spdf, cs2)
slnsdf2=spTransform(slnsdf,cs2)
spolysdf2=spTransform(spolysdf,cs2) #spTransform함수로 점 객체(spdf),선 객체(slnsdf),폴리곤 객체(spolysdf) 를 UTM-K로 변환

plot(spdf2,axes=T,pch=10)
plot(slnsdf2,axes=T,col=1:2)
plot(spolysdf2,axes=T,col=1:2) #변환된 결과 출력
```

slnsdf –〉 spolysdf