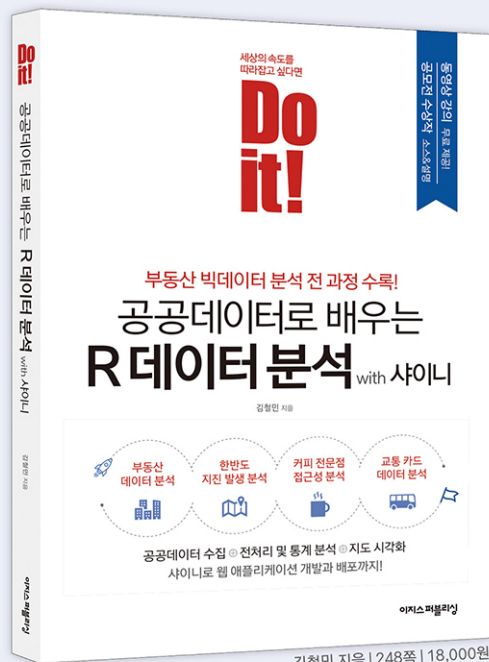


R로 공공데이터를 분석하는 전 과정 실습!
공모전 수상작으로 배우는 R 데이터 분석



공모전
수상작
소스&설명

동영상
강의
무료 제공

김철민 지음 | 248쪽 | 18,000원

이번 장에서는 아파트 실거래 데이터를 대상으로 “어느 지역이 제일 비쌀까?”, “요즘 뜨는 지역은 어디일까?”, “우리 동네가 옆 동네보다 비쌀까?” 등 사람들이 궁금해할 만한 정보를 지도 위에 그려서 분석해 봅니다.

07

분석 주제를 지도로 시각화하기

07-1 어느 지역이 제일 비쌀까?

07-2 요즘 뜨는 지역은 어디일까?

07-3 우리 동네가 옆 동네보다 비쌀까?

07-1 어느 지역이 제일 비쌀까?

앞 장까지 진행하면서 데이터 분석의 기초가 되는 자료를 준비했으니 이제 지도 위에 의미 있는 정보를 표현해 보겠습니다. 첫 번째로는 커널 밀도 추정으로 어느 지역이 제일 비싼지 알아보는 지도를 그려 봅니다.

1단계 지역별 평균 가격 구하기

앞 장까지 진행하면서 데이터 분석의 기초가 되는 자료를 준비했으니 이제 지도 위에 의미 있는 정보를 표현해 보겠습니다. 첫 번째로는 커널 밀도 추정으로 어느 지역이 제일 비싼지 알아보는 지도를 그려 봅니다.

year_ymd	dong_nm	ID	py		ID	avg_price
2021-01-03	시흥동	79520	1420	aggregate()	79520	2277.333
2021-01-06	시흥동	79520	2906			
2021-01-24	시흥동	79520	2506			
2021-01-30	고척동	81479	3402	aggregate()	81479	2433.667
2021-02-03	고척동	81479	1835			
2021-02-08	고척동	81479	2064	aggregate()	82862	4602.000
2021-02-11	망원동	82862	3021			
2021-02-22	망원동	82862	6183			

실거래 자료(apt_price)

ID별 평당 평균가(kde_high)

그림 7-1 그리드 ID별 평균 가격 구하기 예

그리드 ID별 평균가 구하기



07-1 어느 지역이 제일 비쌀까?

1단계 지역별 평균 가격 구하기

Do it! 실거래 + 그리드 데이터 결합

07_지도시각화.R

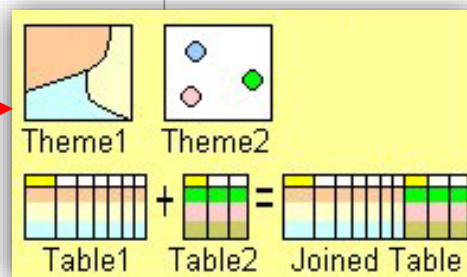
```
08: setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # 작업 폴더 설정
09: load("./06_geodataframe/06_apr_price.rdata") # 실거래 자료 불러오기
10: library(sf) # install.packages("sf")
11: grid <- st_read("./01_code/sigun_grid/seoul.shp") # 서울시 1km 그리드 불러오기
12: apt_price <- st_join(apt_price, grid, join = st_intersects) # 실거래+그리드 결합
13: head(apt_price, 2)
```

👁 실행 결과

	ymd	ym	year	code	addr_1	apt_nm	(...생략...)
1	2021-01-14	2021-01-01	2021	11110	서울_종로	청운현대	(...생략...)
2	2021-01-07	2021-01-01	2021	11110	서울_종로	광화문스페이스본	(...생략...)

속성 테이블 결합하기

특정 APT가 어느 그리드에 속해 있는지 파악하기




07-1 어느 지역이 제일 비쌀까?

1단계 지역별 평균 가격 구하기

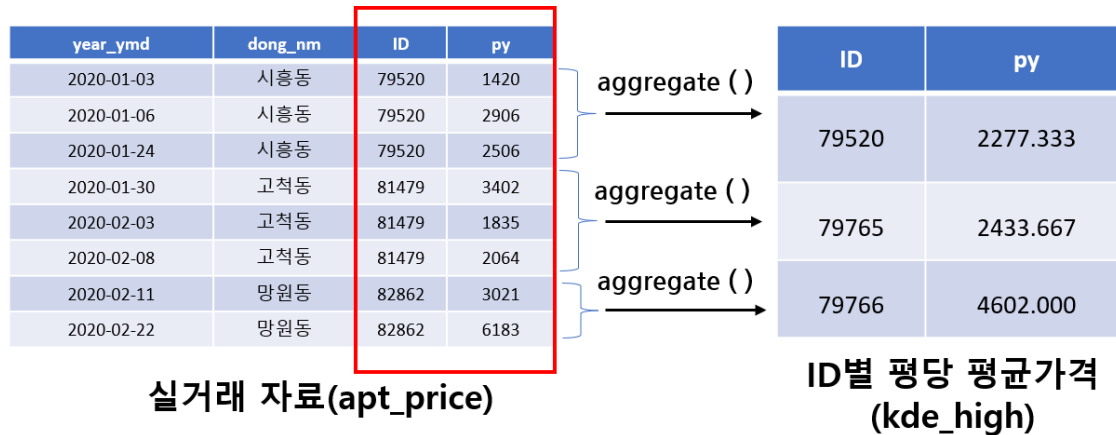
Do it! 그리드별 평균 가격(평당) 계산

07_지도시각화.R

```
15: kde_high <- aggregate(apr_price$py, by=list(apr_price$ID), mean) # 그리드별 평균 가격
16: colnames(kde_high) <- c("ID", "avg_price") # 칼럼명 변경
17: head(kde_high, 2) # 평균가 확인
```

 실행 결과

```
      ID avg_price
1 79520 1506.167
2 79765 3600.258
```



07-1 어느 지역이 제일 비쌀까?

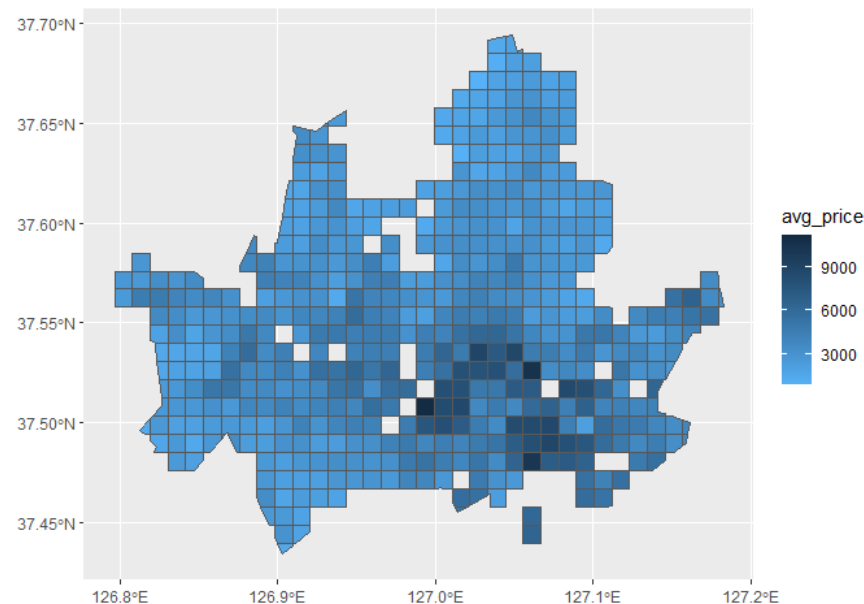
2단계 평균 가격 정보 표시하기

Do it! 그리드 + 평균 가격 결합

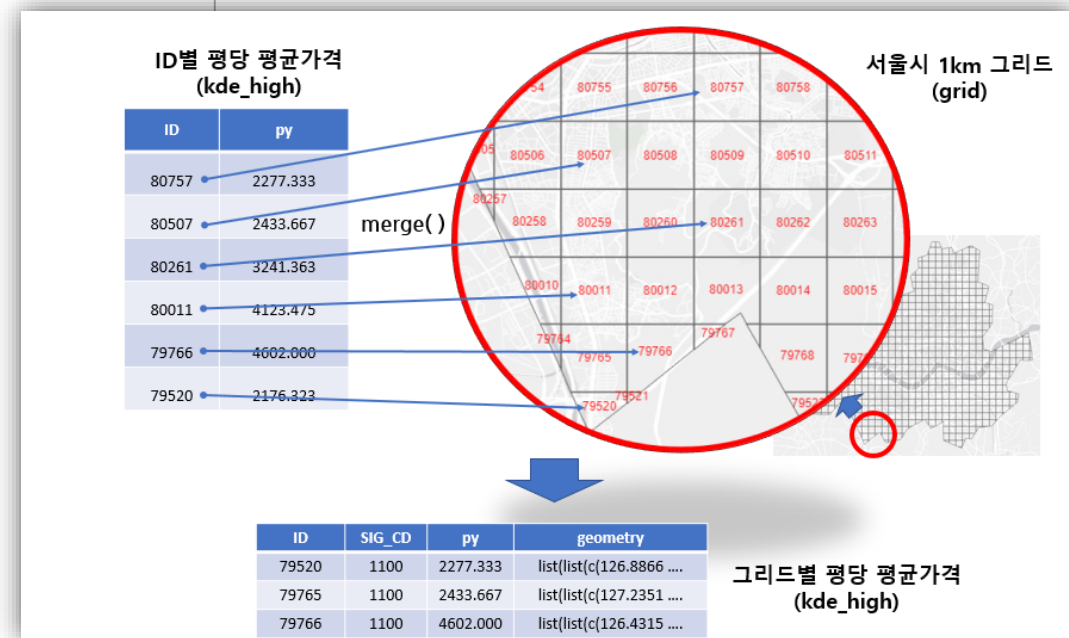
07_지도시각화.R

```
21: kde_high <- merge(grid, kde_high, by="ID") # ID 기준으로 결합
22: library(ggplot2) # install.packages("ggplot2")
23: library(dplyr) # install.packages("dplyr")
24: kde_high %>% ggplot(aes(fill = avg_price)) + # 그래프 시각화
25:     geom_sf() +
26:     scale_fill_gradient(low = "white", high = "red")
```

실행 결과



ID 기준으로 그리드 지도 데이터와 결합



07-1 어느 지역이 제일 비쌀까?

3단계 지도 경계 그리기

어느 지역이 제일 비싼지 알려면 데이터가 집중된 곳을 찾아야 합니다. 이때 커널 밀도 추정*을 이용합니다. 커널 밀도 추정을 하려면 분석의 기초가 되는 대상 영역을 설정해 주어야 하므로 서울시의 경계를 그리는 작업을 수행합니다.

* 커널 밀도 추정(KDE: kernel density estimation)이란 커널 함수로 변수의 밀도를 추정하는 방법의 하나입니다.

Do it! sp형으로 변환과 그리드별 중심 좌표 추출

07_지도시각화.R

```
30: library(sp) # install.packages("sp")
31: kde_high_sp <- as(st_geometry(kde_high), "Spatial") # sf형 => sp형 변환
32: x <- coordinates(kde_high_sp)[,1] # 그리드 중심 x, y 좌표 추출
33: y <- coordinates(kde_high_sp)[,2]
```

Do it! 기준 경계 설정

07_지도시각화.R

```
35: l1 <- bbox(kde_high_sp)[1,1] - (bbox(kde_high_sp)[1,1] * 0.0001)
36: l2 <- bbox(kde_high_sp)[1,2] + (bbox(kde_high_sp)[1,2] * 0.0001)
37: l3 <- bbox(kde_high_sp)[2,1] - (bbox(kde_high_sp)[2,1] * 0.0001)
38: l4 <- bbox(kde_high_sp)[2,2] + (bbox(kde_high_sp)[1,1] * 0.0001)
```

07-1 어느 지역이 제일 비쌀까?

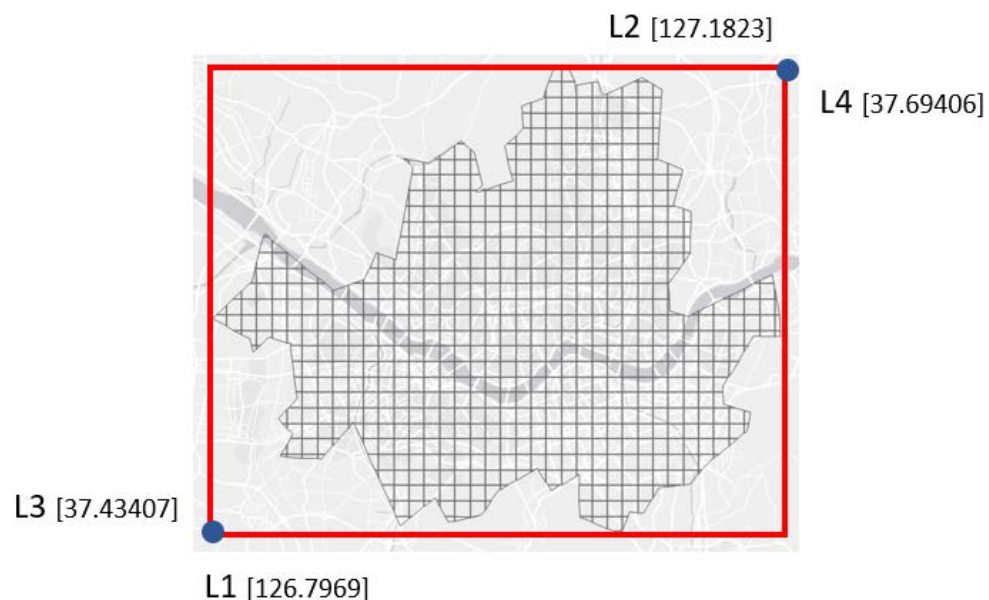
3단계 지도 경계 그리기

Do it! 지도 경계선 그리기

07_지도시각화.R

```
40: library(spatstat) # install.packages("spatstat")
41: win <- owin(xrange=c(l1,l2), yrange=c(l3,l4))
42: plot(win) # 지도 경계선 확인
43: rm(list = c("kde_high_sp", "apt_price", "l1", "l2", "l3", "l4")) # 변수 정리
```

실행 결과



07-1 어느 지역이 제일 비쌀까?

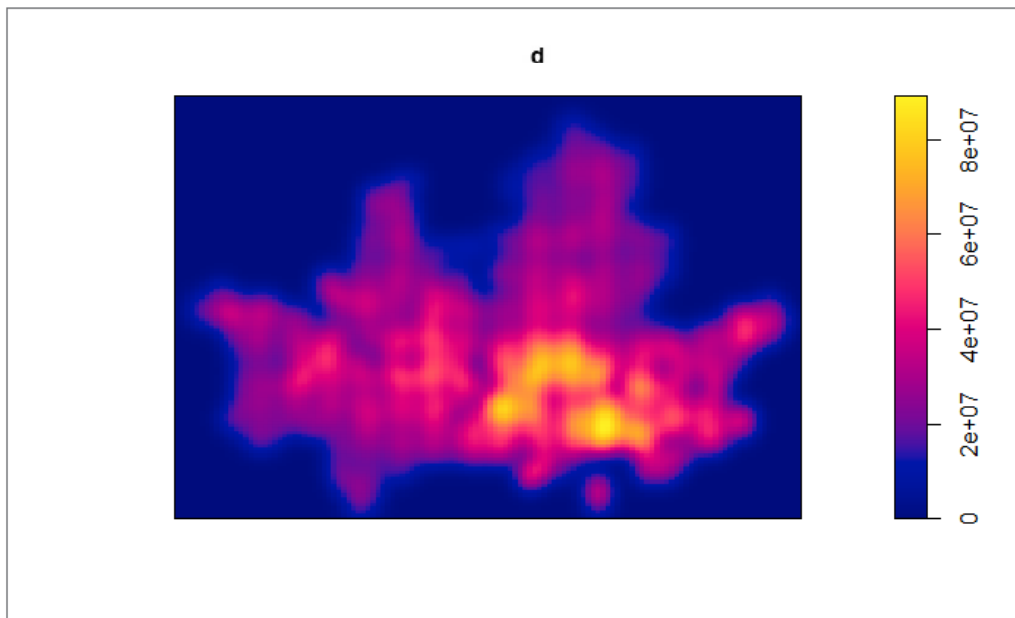
4단계 밀도 그래프 표시하기

Do it! 밀도 그래프 표시하기

07_지도시각화.R

```
47: p <- ppp(x, y, window = win) # 경계선 위에 좌푯값 포인트 생성
48: d <- density.ppp(p, weights = kde_high$avg_price, # 커널 밀도 함수로 변환
49:                 sigma = bw.diggle(p),
50:                 kernel = 'gaussian')
51: plot(d) # 밀도 그래프 확인
52: rm(list = c("x", "y", "win", "p")) # 변수 정리
```

실행 결과



커널 밀도 추정 시 기억해야 할 2가지 옵션

커널 함수(kernel function)의 종류

gaussian, epanechnikov, quartic

시그마(sigma)

대역폭 파라미터(bandwidth parameter)

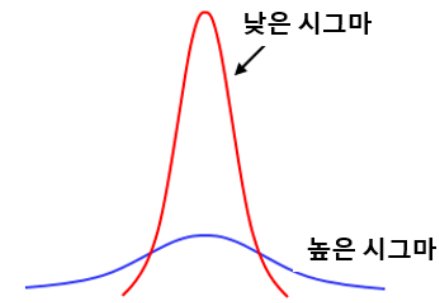


그림 7-4 시그마 변화에 따른 커널 밀도 함수 형태

07-1 어느 지역이 제일 비쌀까?

5단계 래스터 이미지로 변환하기

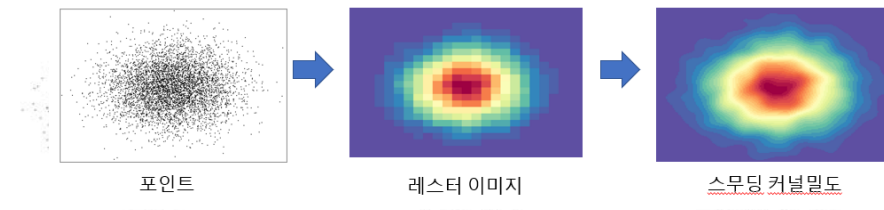
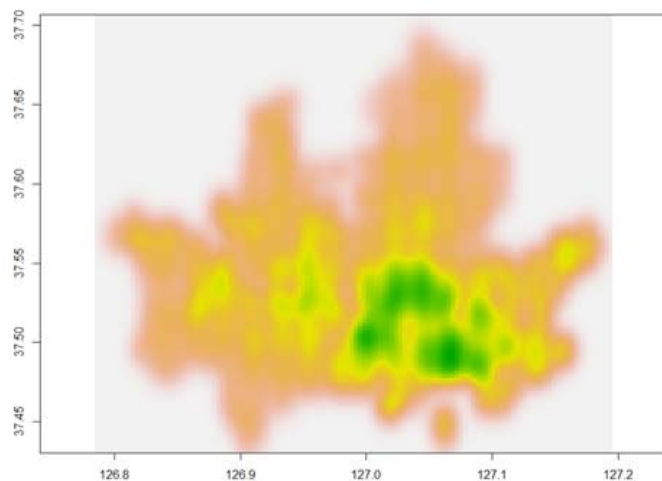


그림 7-5 포인트 데이터를 래스터 이미지로 변환하는 과정

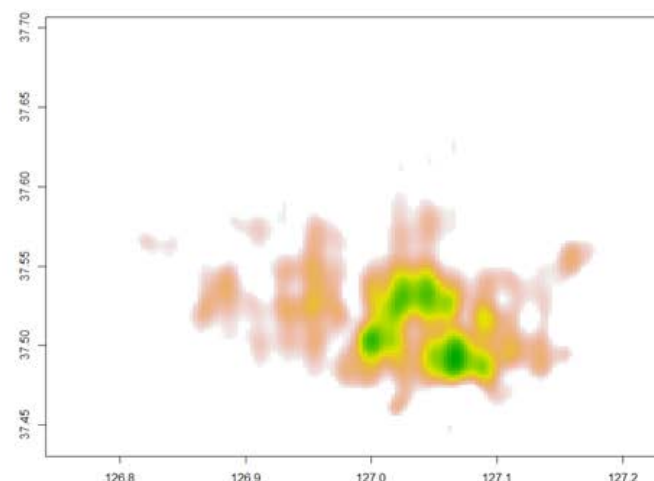
Do it! 노이즈 제거와 래스터 이미지로 변환

07_지도시각화.R

```
56: d[d < quantile(d)[4] + (quantile(d)[4] * 0.1)] <- NA # 노이즈 제거
57: library(raster) # install.packages("raster")
58: raster_high <- raster(d) # 래스터 변환
59: plot(raster_high)
```



노이즈 제거 이전



노이즈 제거 이후

그림 7-6 노이즈 제거에 따른 결과 차이

07-1 어느 지역이 제일 비쌀까?

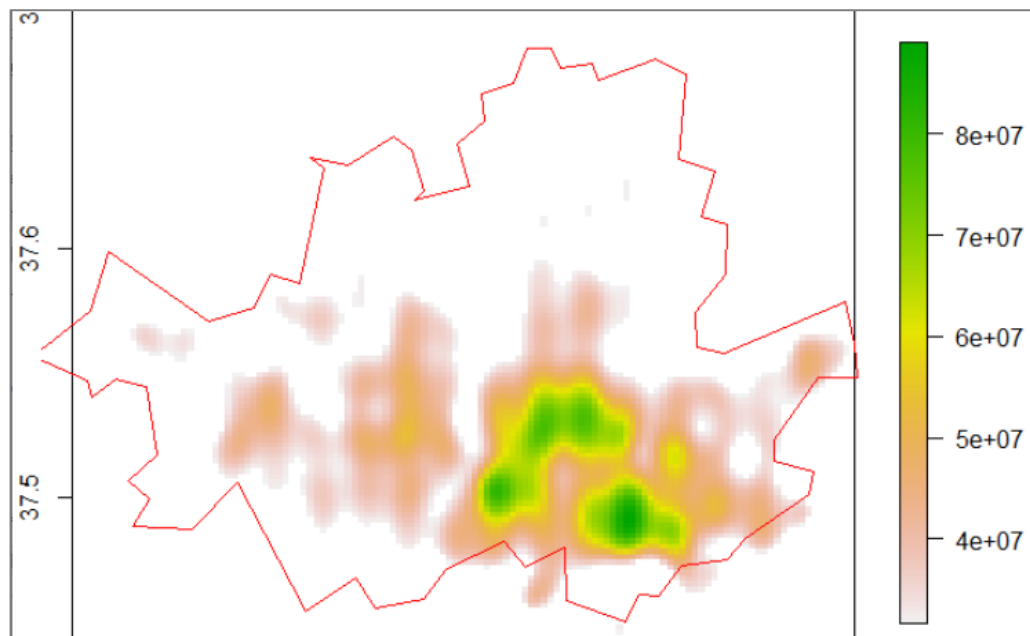
6단계 불필요한 부분 자르기

Do it! 서울시 외곽선 자르기

07_지도시각화.R

```
63: bnd <- st_read("./01_code/sigun_bnd/seoul.shp") # 서울시 경계선 불러오기
64: raster_high <- crop(raster_high, extent(bnd)) # 외곽선 자르기
65: crs(raster_high) <- sp::CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
    +towgs84=0,0,0") # 좌표계 정의
66: plot(raster_high) # 지도 확인
67: plot(bnd, col=NA, border = "red", add=TRUE)
```

☞ 실행 결과



07-1 어느 지역이 제일 비쌀까?

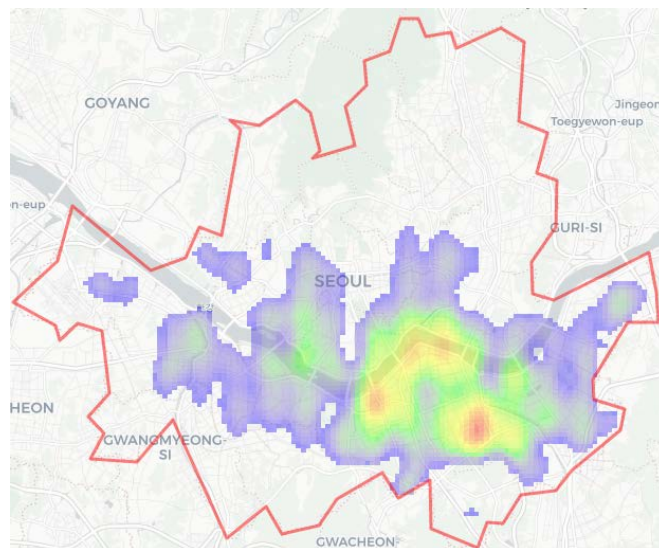
7단계 지도 그리기

Do it! 지도 위에 래스터 이미지 올리기

07_지도시각화.R

```
71: library(leaflet) # install.packages("leaflet")
72: leaflet() %>%
73:   #---# 기본 지도 불러오기
74:   addProviderTiles(providers$CartoDB.Positron) %>%
75:   #---# 서울시 경계선 불러오기
76:   addPolygons(data = bnd, weight = 3, color = "red", fill = NA) %>%
77:   #---# 래스터 이미지 불러오기
78:   addRasterImage(raster_high,
79:     colors = colorNumeric(c("blue", "green", "yellow", "red"),
80:       values(raster_high), na.color = "transparent"), opacity = 0.4)
```

실행 결과



07-1 어느 지역이 제일 비쌀까?

8단계 평균 가격 정보 저장하기

Do it! 8단계: 저장하기

07_지도시각화.R

```
84: dir.create("07_map") # 새로운 폴더 생성
85: save(raster_high, file="./07_map/07_kde_high.rdata") # 최고가 래스터 저장
86: rm(list = ls()) # 메모리 정리
```

07-2 요즘 뜨는 지역은 어디일까?

1단계 데이터 준비하기

두 번째로 분석할 주제는 일정 기간 동안 가장 많이 오른 지역을 특정하는 것입니다. 이는 똑같은 그리드를 대상으로 두 시점 사이의 가격 변화를 비교하는 방식으로 단순히 특정 그리드의 평균 가격을 측정하는 방식보다 조금 더 복잡합니다. 그럼 두 번째 지도를 만들어 보겠습니다.

Do it! 데이터 준비

07_지도시각화.R

```
94: setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # 작업 폴더 설정
95: load("./06_geo_df/06_geo_df.rdata") # 실거래 불러오기
96: grid <- st_read("./01_code/sigun_grid/seoul.shp") # 서울시 1km 그리드 불러오기
97: apt_price <- st_join(apt_price, grid, join = st_intersects) # 실거래 + 그리드 결합
98: head(apt_price, 2)
```

실행 결과

	ymd	ym	year	code	addr_1	apt_nm	(...생략...)
1	2021-01-14	2021-01-01	2021	11110	서울_종로	청운현대	(...생략...)
2	2021-01-07	2021-01-01	2021	11110	서울_종로	광화문스페이스본	(...생략...)

07-2 요즘 뜨는 지역은 어디일까?

2단계 이전/이후 데이터 세트 만들기

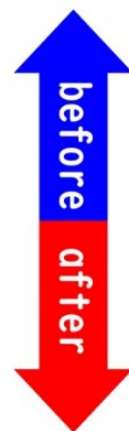
Do it! 이전/이후 데이터 세트 만들기

07_지도시각화.R

```
102: kde_before <- subset(apt_price, ymd < "2021-07-01") # 이전 데이터 필터링
103: kde_before <- aggregate(kde_before$py, by=list(kde_before$ID),mean) # 평균 가격
104: colnames(kde_before) <- c("ID", "before") # 컬럼명 변경
105:
106: kde_after <- subset(apt_price, ymd > "2021-07-01") # 이후 데이터 필터링
107: kde_after <- aggregate(kde_after$py, by=list(kde_after$ID),mean) # 평균 가격
108: colnames(kde_after) <- c("ID", "after") # 컬럼명 변경
109:
110: kde_diff <- merge(kde_before, kde_after, by="ID") # 이전 + 이후 데이터 결합
111: kde_diff$diff <- round((((kde_diff$after-kde_diff$before) /
112:                           kde_diff$before) * 100), 0) # 변화율 계산
113:
114: head(kde_diff, 2) # 변화율 확인
```

☞ 실행 결과

	ID	before	after	diff
1	79520	1451.000	1517.200	5
2	79765	3545.216	3681.720	4



07-2 요즘 뜨는 지역은 어디일까?

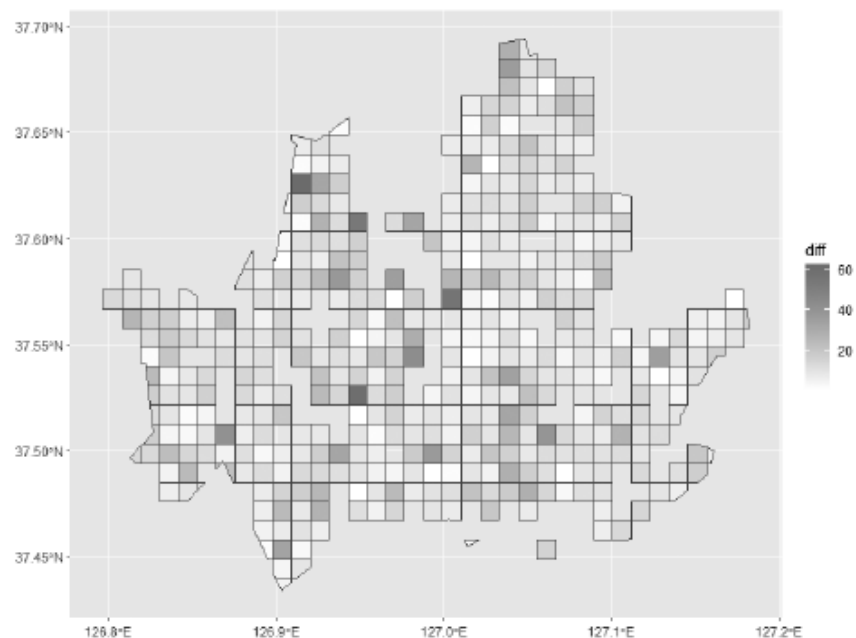
3단계 가격이 오른 지역 찾기

Do it! 가격이 오른 지역 찾기

07_지도시각화.R

```
118: library(sf)          # install.packages("sf")
119: kde_diff <- kde_diff[kde_diff$diff > 0,]  # 상승 지역만 추출
120: kde_hot <- merge(grid, kde_diff, by="ID") # 그리드에 상승 지역 결합
121: library(ggplot2)      # install.packages("ggplot2")
122: library(dplyr)        # install.packages("dplyr")
123: kde_hot %>%           # 그래프 시각화
124:   ggplot(aes(fill = diff)) +
125:   geom_sf() +
126:   scale_fill_gradient(low = "white", high = "red")
```

실행 결과



07-2 요즘 뜨는 지역은 어디일까?

4~7단계 기타 지도 작업

이제 지도 경계, 밀도 그래프, 래스터 이미지, 불필요한 부분 자르기 등의 작업을 해야 하는데, 모두 「07-1」절 3~7단계에서 이미 해보았습니다. 코드가 거의 중복되므로 책에서는 생략하고 필자가 제공한 실습 파일을 참고 바랍니다.

07-2 요즘 뜨는 지역은 어디일까?

8단계 지도 그리기

Do it! 지도 그리기

07_지도시각화.R

```
172: library(leaflet) # install.packages("leaflet")
173: leaflet() %>%
174:   #---# 기본 지도 불러오기
175:   addProviderTiles(providers$CartoDB.Positron) %>%
176:   #---# 서울시 경계선 불러오기
177:   addPolygons(data = bnd, weight = 3, color= "red", fill = NA) %>%
178:   #---# 래스터 이미지 불러오기
179:   addRasterImage(raster_hot,
180:     colors = colorNumeric(c("blue", "green", "yellow","red"),
181:       values(raster_hot), na.color = "transparent"), opacity = 0.4)
```

실행 결과



07-2 요즘 뜨는 지역은 어디일까?

9단계 평균 가격 변화율 정보 저장하기

Do it! 분석 결과 저장

07_지도시각화.R

```
185: save(raster_hot, file="./07_map/07_kde_hot.rdata") # 급등지 래스터 저장  
186: rm(list = ls()) # 메모리 정리
```

07-3 우리 동네가 옆 동네보다 비쌀까?

1단계 데이터 준비하기

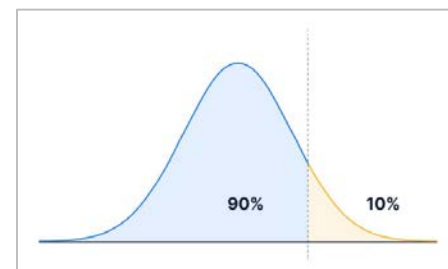
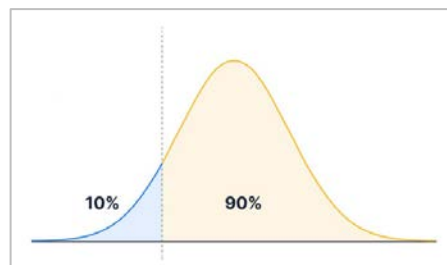
Do it! 1단계: 데이터 준비

07_지도시각화.R

```
194: setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # 작업 폴더 설정
195: load("./06_geodataframe/06_apr_price.rdata") # 실거래 자료 불러오기
196: load("./07_map/07_kde_high.rdata") # 최고가 래스터 이미지
197: load("./07_map/07_kde_hot.rdata") # 급등지 래스터 이미지
198:
199: library(sf) # install.packages("sf")
200: bnd <- st_read("./01_code/sigun_bnd/seoul.shp") # 서울시 경계선
201: grid <- st_read("./01_code/sigun_grid/seoul.shp") # 서울시 그리드 파일
```

07-3 우리 동네가 옆 동네보다 비싸까?

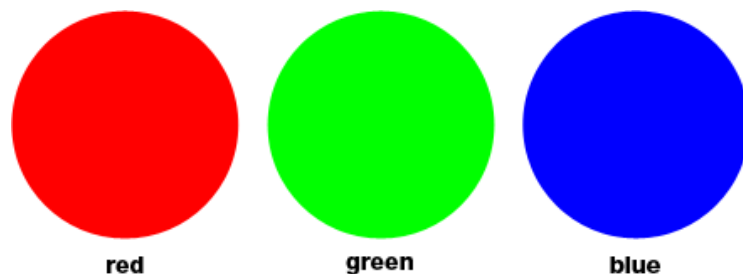
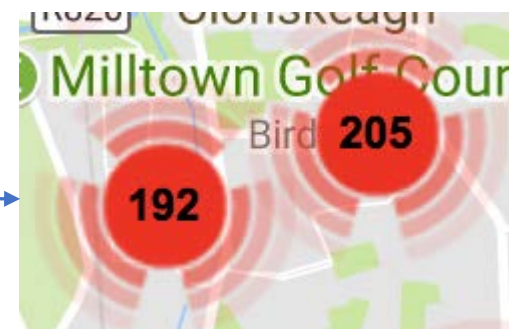
2단계 마커 클러스터링 옵션 설정하기



Do it! 마커 클러스터링 옵션 설정

07_지도시각화.R

```
205: #---# 이상치 설정 (하위 10%, 상위 90% 지점)
206: pcnt_10 <- as.numeric(quantile(apr_price$py, probs = seq(.1, .9, by = .1))[1])
207: pcnt_90 <- as.numeric(quantile(apr_price$py, probs = seq(.1, .9, by = .1))[9])
208: #---# 마커 클러스터링 함수 등록
209: load("./01_code/circle_marker/circle_marker.rdata")
210: #---# 마커 클러스터링 색상 설정: 상, 중, 하
211: circle.colors <- sample(x=c("red", "green", "blue"), size=1000, replace=TRUE)
```



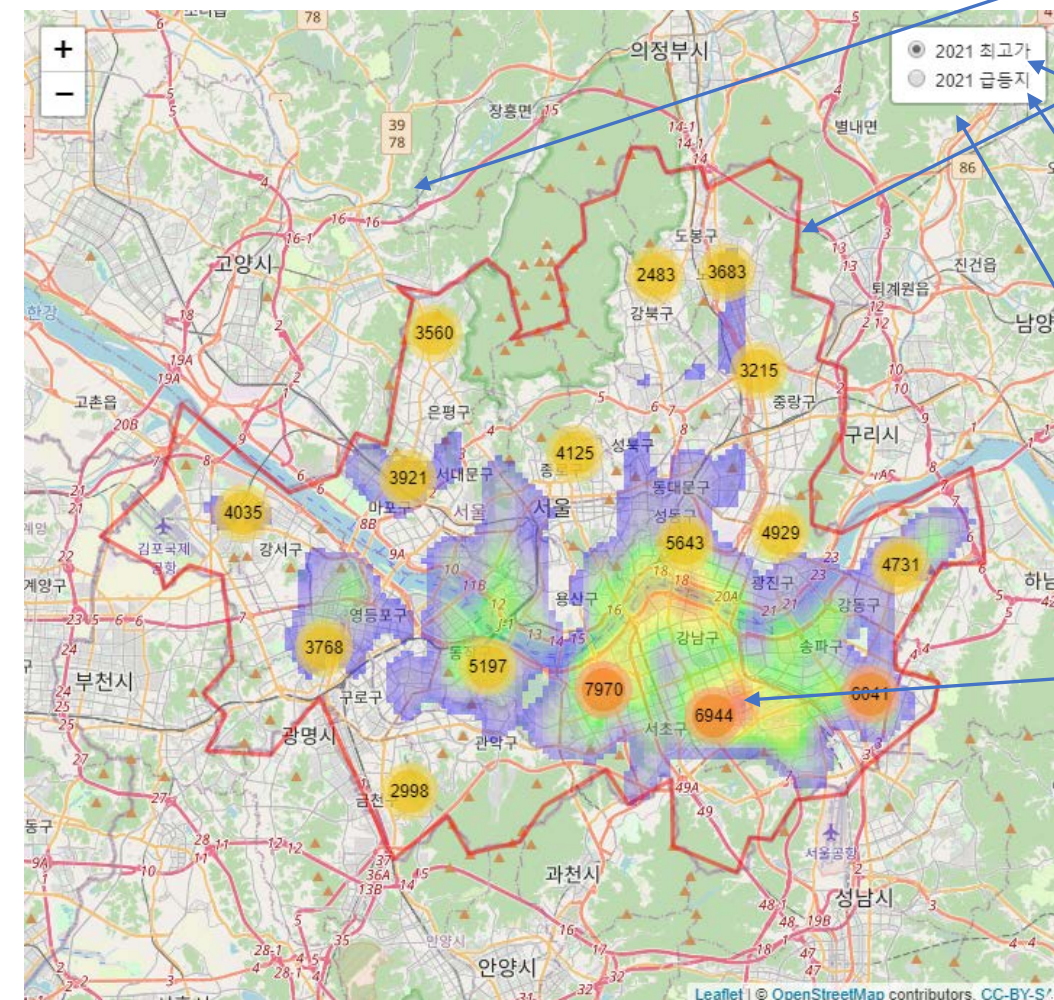
red

green

blue

07-3 우리 동네가 옆 동네보다 비싸까?

3단계 마커 클러스터링 시각화하기



Do it! 마커 클러스터링 시각화

07_지도시각화.R

```
215: library(purrr) # install.packages("purrr")
216: leaflet() %>%
217:   #---# 오픈스트리트맵 불러오기
218:   addTiles() %>%
219:   #---# 서울시 경계선 불러오기
220:   addPolygons(data = bnd, weight = 3, color= "red", fill = NA) %>%
221:   #---# 최고가 래스터 이미지 불러오기
222:   addRasterImage(raster_high,
223:     colors = colorNumeric(c("blue","green","yellow","red"), values(raster_high),
224:     na.color = "transparent"), opacity = 0.4, group = "2021 최고가") %>%
225:   #---# 급등지 래스터 이미지 불러오기
226:   addRasterImage(raster_hot,
227:     colors = colorNumeric(c("blue","green","yellow","red"), values(raster_hot),
228:     na.color = "transparent"), opacity = 0.4, group = "2021 급등지") %>%
229:   #---# 최고가/급등지 선택 옵션 추가하기
230:   addLayersControl(baseGroups = c("2021 최고가", "2021 급등지"),
231:     options = layersControlOptions(collapsed = FALSE)) %>%
232:   #---# 마커 클러스터링 불러오기
233:   addCircleMarkers(data = apt_price, lng =unlist(map(apt_price$geometry,1)),
234:     lat = unlist(map(apt_price$geometry,2)), radius = 10, stroke = FALSE,
235:     fillOpacity = 0.6, fillColor = circle.colors, weight = apt_price$py,
236:     clusterOptions = markerClusterOptions(iconCreateFunction=JS(avg.formula)))
237:   #---# 메모리 정리하기
238:   rm(list = ls())
```



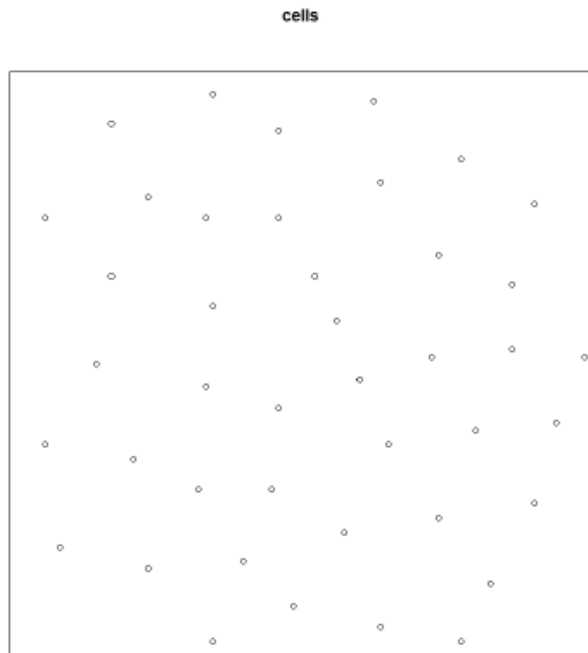
단골 코드 정리하기

이번 장에서는 지오 데이터프레임을 지도로 시각화하는 과정을 살펴보았습니다. 지도 시각화에서 다루었던 주요 기능인 포인트 데이터를 밀도 데이터로 변환한 다음 래스터 이미지로 저장하는 방법을 정리해 보겠습니다.

• 포인트 데이터 불러오기

```
library(spatstat) # install.packages("spatstat")  
data(cells)  
density(cells, 0.05, at="points")  
plot(cells)
```

실행 결과





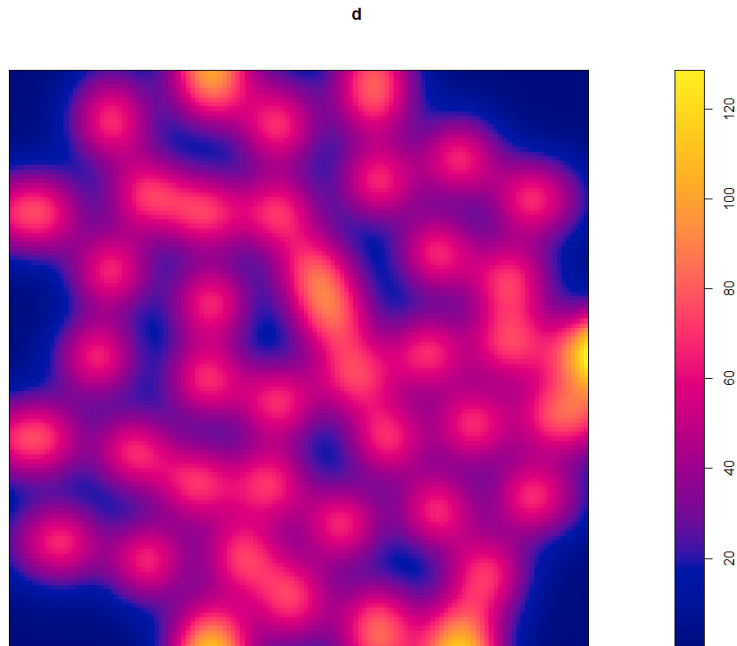
단골 코드 정리하기

이번 장에서는 지오 데이터프레임을 지도로 시각화하는 과정을 살펴보았습니다. 지도 시각화에서 다루었던 주요 기능인 포인트 데이터를 밀도 데이터로 변환한 다음 래스터 이미지로 저장하는 방법을 정리해 보겠습니다.

• density.ppp()로 밀도 데이터로 변환하기

```
d <- density.ppp(cells, 0.05) # 밀도 데이터로 변환  
plot(d)
```

실행 결과





단골 코드 정리하기

이번 장에서는 지오 데이터프레임을 지도로 시각화하는 과정을 살펴보았습니다. 지도 시각화에서 다루었던 주요 기능인 포인트 데이터를 밀도 데이터로 변환한 다음 래스터 이미지로 저장하는 방법을 정리해 보겠습니다.

• density.ppp()로 밀도 데이터로 변환하기

```
library(raster)
raster <- raster(d) # 래스터 변환
plot(raster) # 확인
```

실행 결과

