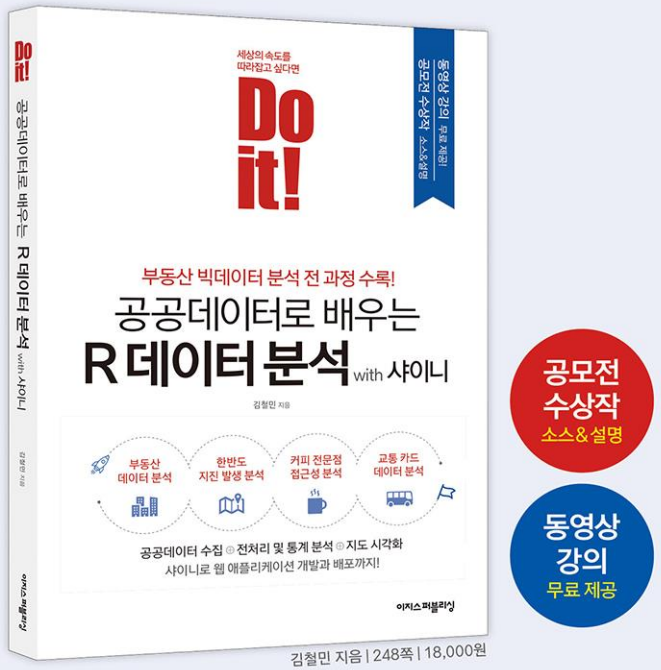
R로 공공데이터를 분석하는 전 과정 실습!

공모전 수상작으로 배우는 **R 데이터 분석**



공모전 수상작 소스&설명

동영상 강의 무료 제공

김철민 지음 | 248쪽 | 18,000원

데이터 과학에서 통계 분석은 중요한 위치를 차지합니다. 그러나 모델링 결과만으로는 정보를 직관적으로 전달하기 어렵습니다. 따라서 차트 등을 활용한 시각화 기법을 함께 사용합니다. 이 장에서는 확률 밀도 함수, 회귀 분석 그리고 주성분 분석 등을 활용하여 데이터에서 의미 있는 정보를 제공하는 통계 분석과 차트로 시각화하는 방법을 살펴봅니다.

08

통계 분석과 시각화

08-1 관심 지역 데이터만 추출하기

이번 장에서는 관심 있는 지역과 서울시 전체 아파트 가격의 차이를 비교하는 차트를 만듭니다. 따라서 관심 지역에서 추출한 데이터가 필요합니다. 관심 지역이란 주목해서 분석하고 싶은 동네를 의미합니다. 여기에서는 관심이 있는 아파트들이 포함된 그리드를 찾아냅니다.

1단계 데이터 준비하기

Do it! 데이터 준비 — 08_통계시각화.R

```
08: library(sf)
09: setwd(dirname(rstudioapi::getSourceEditorContext()$path))
10: load("./06_geodataframe/06_apt_price.rdata")        # 실거래 데이터
11: load("./07_map/07_kde_high.rdata")                  # 최고가 래스터 이미지
12: grid <- st_read("./01_code/sigun_grid/seoul.shp")   # 서울시 그리드
```

08-1 관심 지역 데이터만 추출하기

2단계 서울에서 가장 비싼 지역 찾기

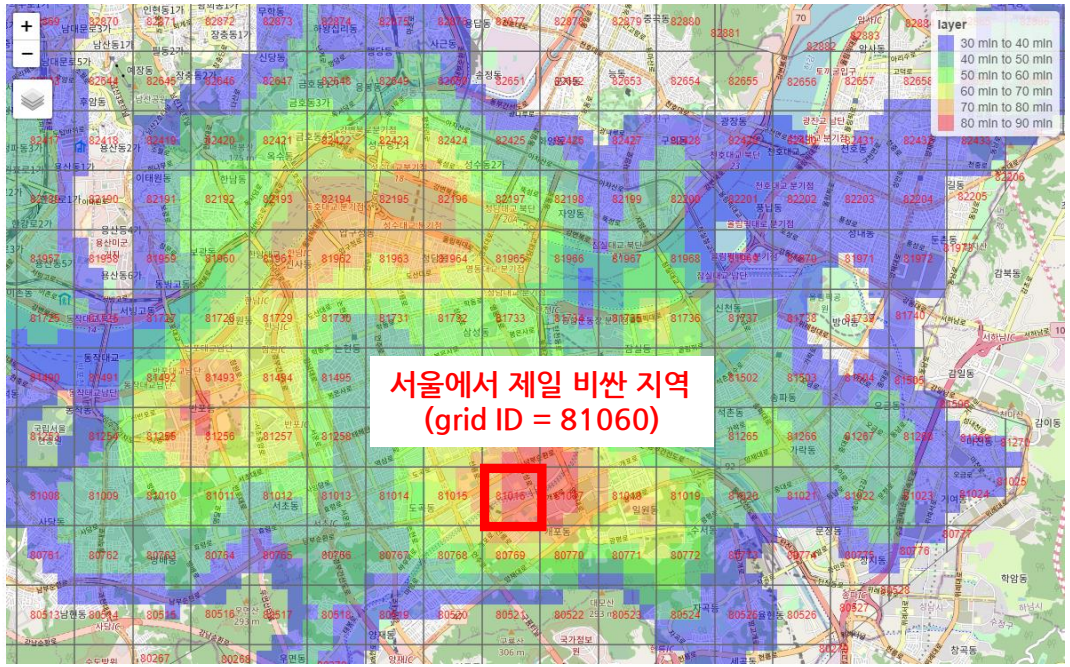
지도 시각화 결과 2021년도 실거래가 기준*으로 서울에서 평당 아파트 가격이 가장 비싼 지역은 개포동 일대로서 ID가 81016 그리드로 나타났습니다.

* 여기에서 사용된 아파트 실거래 데이터는 2021년 1월에서 12월까지 1년 동안의 데이터입니다. 또한 그리드별 평균 아파트 가격은 아파트 크기에 관계없이 해당 그리드에 속하는 모든 아파트의 평당 거래가 평균입니다. 기간과 아파트 크기에 따라 가격이 높은 지역이 달라질 수 있습니다.

Do it! 관심 지역 그리드 찾기 — 08_통계시각화.R

```
16: library(tmap)       # install.packages("tmap")
17: tmap_mode('view')
18: #---# 그리드 그리기
19: tm_shape(grid) + tm_borders() + tm_text("ID", col = "red") +
20:   #---# 래스터 이미지 그리기
21:   tm_shape(raster_high) +
22:   #---# 래스터 이미지 색상 패턴 설정
23:   tm_raster(palette = c("blue", "green", "yellow","red"), alpha =.4) +
24:   #---# 기본 지도 설정
25:   tm_basemap(server = c('OpenStreetMap'))
```

☞ 실행 결과



Thematic maps 사용

참고: https://cran.r-project.org/web/packages/tmap/vignettes/tmap-getstarted.html

08-1 관심 지역 데이터만 추출하기

3단계 전체 지역/관심 지역 저장하기

통계 차트를 편리하게 분석하고자 all과 sel로 전체 지역과 관심 지역을 구분하여 저장

서울시 전체

Do it! 전체 지역 / 관심 지역 저장 — 08_통계시각화.R

```
29: library(dplyr)
30: apt_price <-st_join(apt_price, grid, join = st_intersects)   # 실거래 + 그리드 결합
31: apt_price <- apt_price %>% st_drop_geometry()   # 실거래에서 공간 속성 지우기
32: all <- apt_price                                # 전체 지역(all) 추출
33: sel <- apt_price %>% filter(ID == 81016)        # 관심 지역(sel) 추출
34: dir.create("08_chart")   # 새로운 폴더 생성
35: save(all, file="./08_chart/all.rdata")   # 저장
36: save(sel, file="./08_chart/sel.rdata")
37: rm(list = ls())   # 정리하기
```

선택 지역(ID=81060)

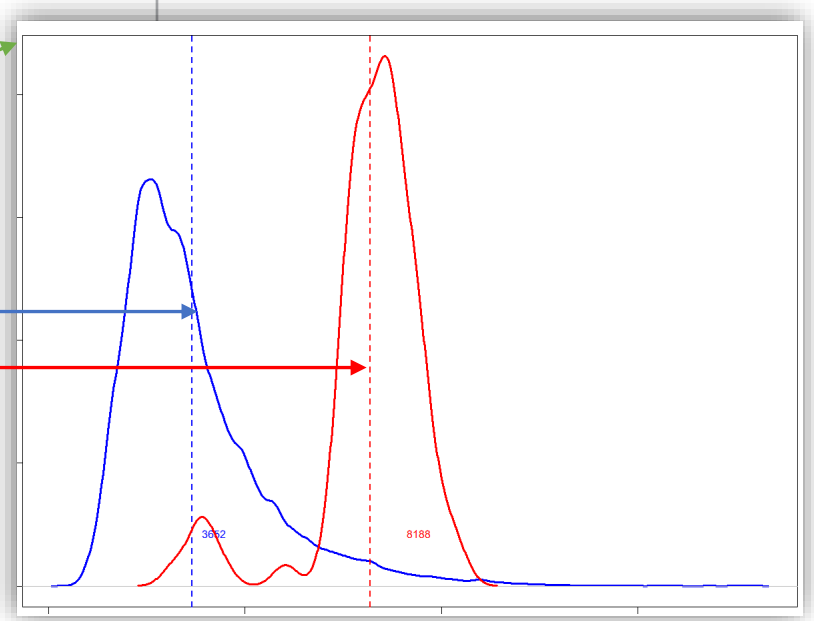
08-2 확률 밀도 함수: 이 지역 아파트는 비싼 편일까?

1단계 확률 밀도 분포로 변환하기

최종으로 얻은 값은 서울시 전체 아파트의 평당 평균가(avg_all)와 관심 지역 아파트의 평당 평균가(avg_sel), 그리고 그래프의 y축 최댓값(plot_high)입니다.

Do it! 그래프 준비하기 08_통계시각화.R

```
46: setwd(dirname(rstudioapi::getSourceEditorContext()$path))
47: load("./08_chart/all.rdata")    # 전체 지역
48: load("./08_chart/sel.rdata")    # 관심 지역
49: max_all <- density(all$py) ; max_all <- max(max_all$y)
50: max_sel <- density(sel$py) ; max_sel <- max(max_sel$y)
51: plot_high <- max(max_all, max_sel)   # y축 최댓값 찾기
52: rm(list = c("max_all", "max_sel"))
53: avg_all <- mean(all$py)   # 전체 지역 평당 평균가 계산
54: avg_sel <- mean(sel$py)   # 관심 지역 평당 평균가 계산
55: avg_all ; avg_sel ; plot_high   # 전체/관심 평균 가격과 y축 최댓값 확인
```



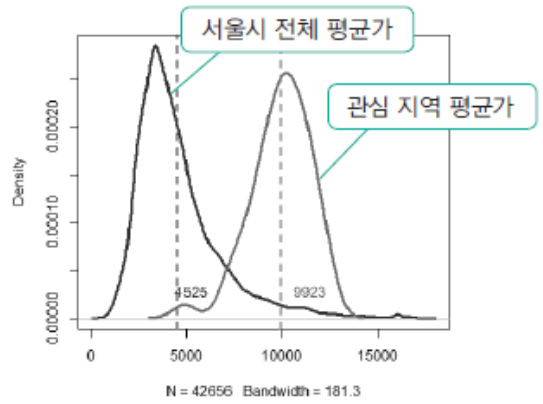
08-2 확률 밀도 함수: 이 지역 아파트는 비싼 편일까?

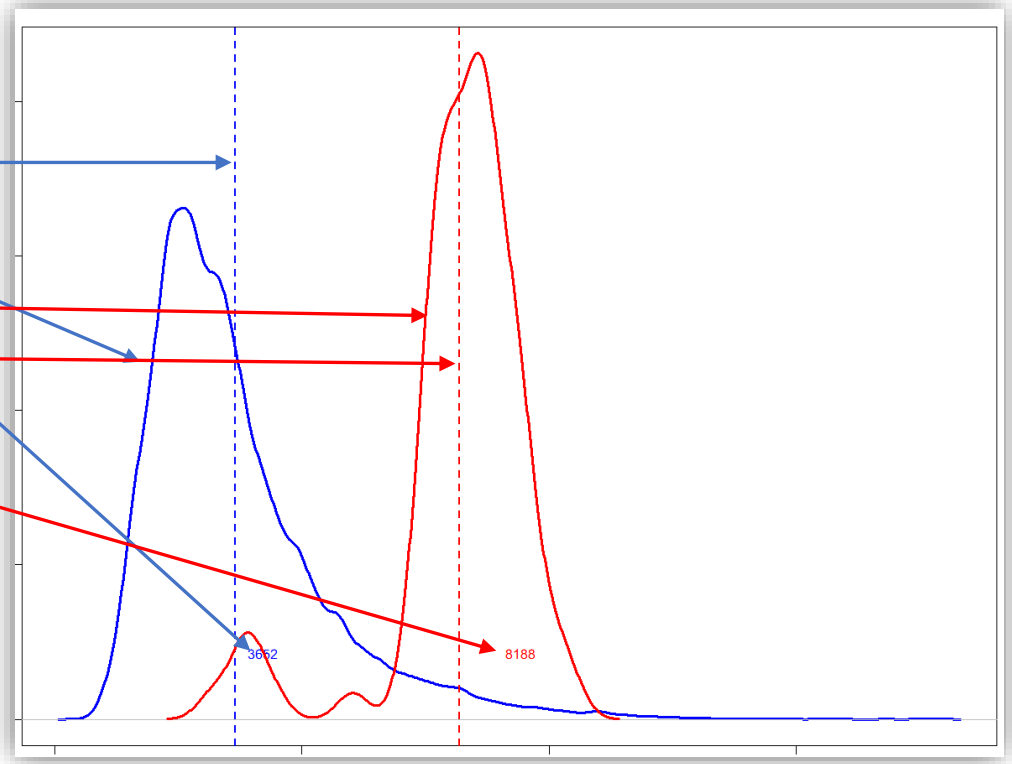
2단계 그래프 그리기

Do it! 확률 밀도 함수 그리기 — 08_통계시각화.R

```
59: plot(stats::density(all$py), ylim=c(0, plot_high),
60:  col="blue", lwd=3, main= NA)   # 전체 지역 밀도 함수 띄우기
61: abline(v = mean(all$py), lwd = 2, col = "blue", lty=2)  # 전체 지역 평균 수직선 그리기
62: text(avg_all + (avg_all) * 0.15, plot_high * 0.1,
63:  sprintf("%.0f",avg_all), srt=0.2, col = "blue")   # 전체 지역 평균 텍스트 입력
64: lines(stats::density(sel$py), col="red", lwd=3)    # 관심 지역 확률 밀도 함수 띄우기
65: abline(v = avg_sel, lwd = 2, col = "red", lty=2)   # 관심 지역 평균 수직선 그리기
66: text(avg_sel + avg_sel * 0.15 , plot_high * 0.1,
67:  sprintf("%.0f", avg_sel), srt=0.2, col = "red")   # 관심 지역 평균 텍스트 입력
```

☞ 실행 결과





08-3 회귀 분석: 이 지역은 일년에 얼마나 오를까?

1단계 월별 거래가 요약하기

Do it! 월별 평당 거래가 요약 — 08_통계시각화.R

```
76: setwd(dirname(rstudioapi::getSourceEditorContext()$path))
77: load("./08_chart/all.rdata")   # 전체 지역
78: load("./08_chart/sel.rdata")   # 관심 지역
79: library(dplyr)       # install.packages("dplyr")
80: library(lubridate)   # install.packages("lubridate")
81: all <- all %>% group_by(month=floor_date(ymd, "month")) %>%
82:  summarize(all_py = mean(py))   # 전체 지역 카운팅
83: sel <- sel %>% group_by(month=floor_date(ymd, "month")) %>%
84:  summarize(sel_py = mean(py))   # 관심 지역 카운팅
```

08-2 확률 밀도 함수: 이 지역 아파트는 비싼 편일까?

2단계 회귀식 모델링하기

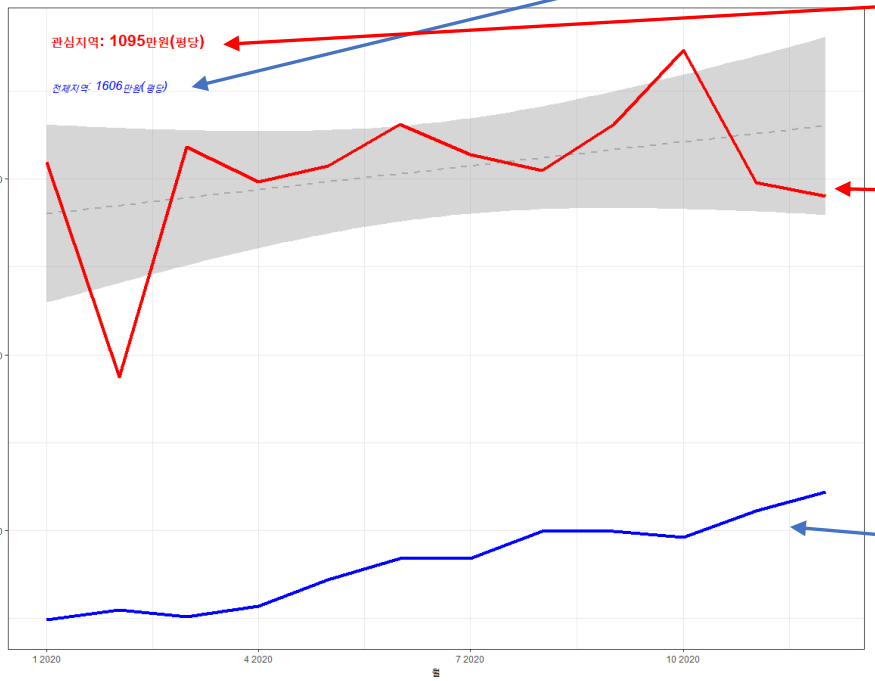
Do it! 회귀식 모델링 08_통계시각화.R

```
88: fit_all <- lm(all$all_py ~ all$month)   # 전체 지역 회귀식
89: fit_sel <- lm(sel$sel_py ~ sel$month)   # 관심 지역 회귀식
90: coef_all <- round(summary(fit_all)$coefficients[2], 1) * 365   # 전체 회귀 계수
91: coef_sel <- round(summary(fit_sel)$coefficients[2], 1) * 365   # 관심 회귀 계수
```

08-2 확률 밀도 함수: 이 지역 아파트는 비싼 편일까?

3단계 그래프 그리기

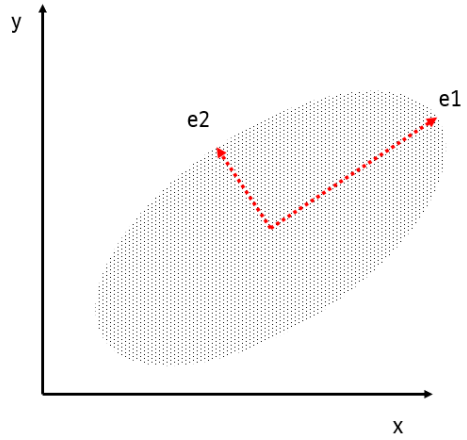
☞ 실행 결과



Do it! 회귀 분석 그리기 — 08_통계시각화.R

```
95: #---# 분기별 평당 가격 변화 주석 만들기
96: library(grid)   # install.packages("grid")
97: grob_1 <- grobTree(textGrob(paste0("전체 지역: ", coef_all, "만원(평당)"), x=0.05,
98:  y=0.88, hjust=0, gp=gpar(col="blue", fontsize=13, fontface="italic")))
99: grob_2 <- grobTree(textGrob(paste0("관심 지역: ", coef_sel, "만원(평당)"), x=0.05,
100: y=0.95, hjust=0, gp=gpar(col="red", fontsize=16, fontface="bold")))
101: #---# 관심 지역 회귀선 그리기
102: library(ggpmisc)   # install.packages("ggpmisc")
103: gg <- ggplot(sel, aes(x=month, y=sel_py)) +
104:  geom_line() + xlab("월")+ ylab("가격") +
105:  theme(axis.text.x=element_text(angle=90)) +
106:  stat_smooth(method='lm', colour="dark grey", linetype = "dashed") +
107:  theme_bw()
108: #---# 전체 지역 회귀선 그리기
109: gg + geom_line(color= "red", size=1.5) +
110:  geom_line(data=all, aes(x=month, y=all_py), color="blue", size=1.5) +
111:  #---#  주석 추가하기
112:  annotation_custom(grob_1) +
113:  annotation_custom(grob_2)
114: rm(list = ls())   # 메모리 정리하기
```

08-4 주성분 분석: 이 동네 단지별 특징은 무엇일까?



1단계 주성분 분석하기

Do it! 주성분 분석 — 08_통계시각화.R

```
122: setwd(dirname(rstudioapi::getSourceEditorContext()$path))
123: load("./08_chart/sel.rdata")   # 관심 지역 데이터 불러오기
124: pca_01 <- aggregate(list(sel$con_year, sel$floor, sel$py, sel$area),
125:                     by=list(sel$apt_nm), mean)   # 아파트별 평균값 구하가기
126: colnames(pca_01) <- c("apt_nm", "신축", "층수","가격", "면적")
127: m <- prcomp(~ 신축 + 층수 + 가격 + 면적, data=pca_01, scale=T)   # 주성분 분석
128: summary(m)
```

☞ 실행 결과

주성분(PC) 1-2 만으로 전체 데이터의 90.9%를 차지함

```
Importance of components:
                          PC1    PC2    PC3     PC4
Standard deviation     1.4836 1.1979 0.56678 0.2069
Proportion of Variance 0.5503 0.3587 0.08031 0.0107
Cumulative Proportion  0.5503 0.9090 0.98930 1.0000
```

누적 비율

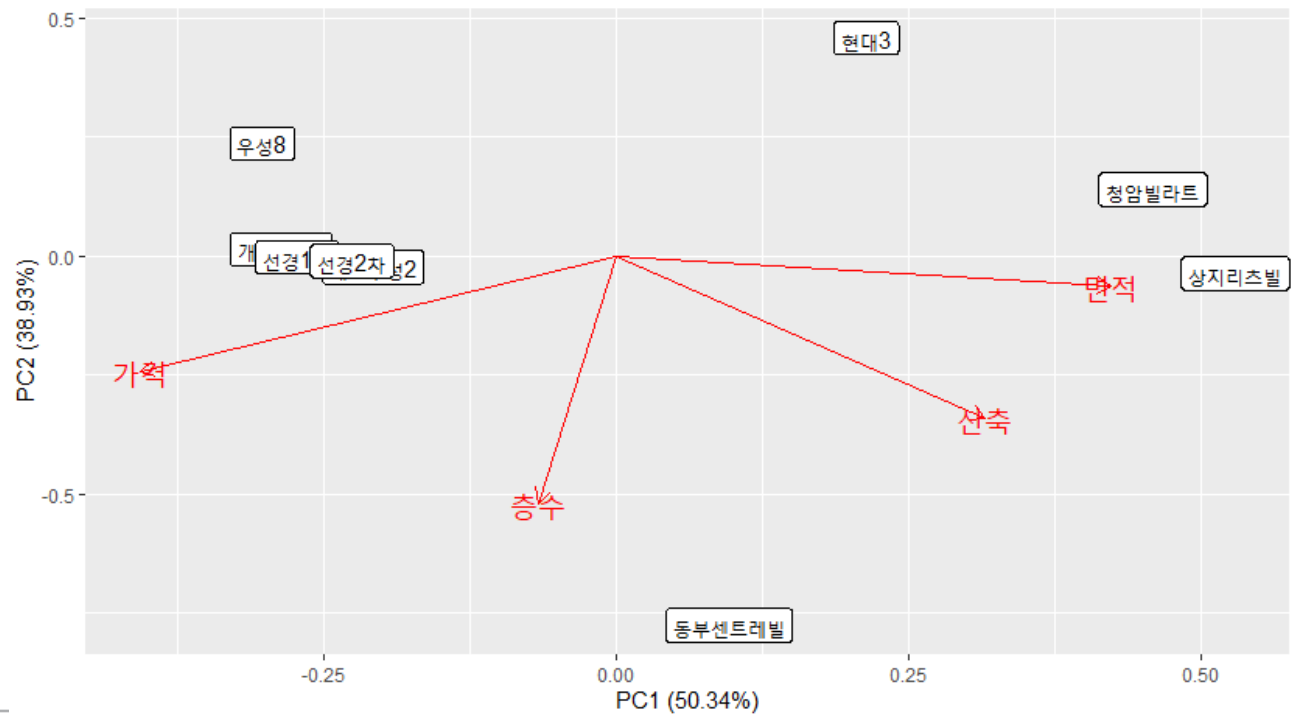
08-4 주성분 분석: 이 동네 단지별 특징은 무엇일까?

2단계 그래프 그리기

Do it! 그래프 그리기 08_통계시각화.R

```
132: library(ggfortify)
133: autoplot(m, loadings.label=T, loadings.label.size=6)+
134:  geom_label(aes(label=pca_01$apt_nm), size=4)
```

☞ 실행 결과



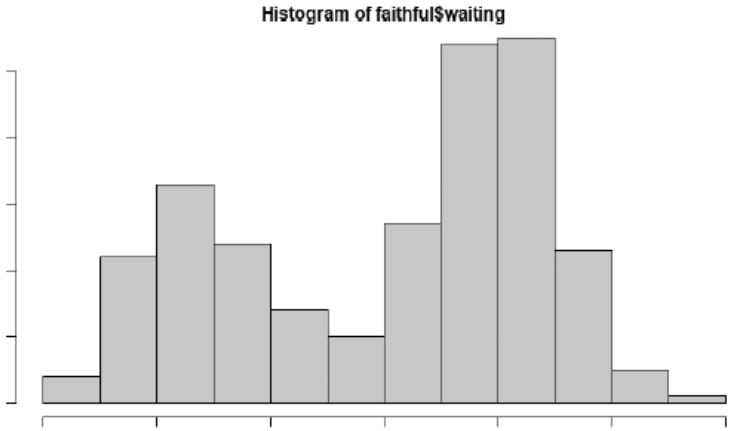
단골 코드 정리하기

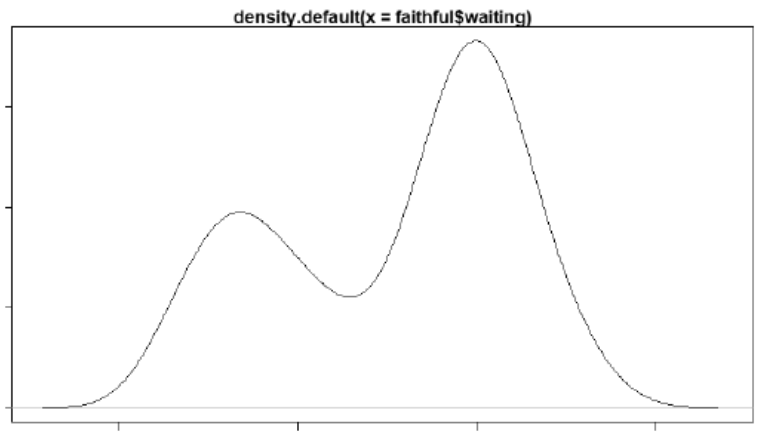
이번 장에서는 통계 분석을 기반으로 하는 차트 시각화 과정을 살펴보았습니다. 통계 차트 시각화에서 다루었던 주요 기능인 연속 확률 밀도, 회귀 분석 등을 요약해 보겠습니다.

• 히스토그램 → 연속 확률 밀도 그래프 변환

```
faithful <- faithful
hist(faithful$waiting)            # 히스토그램 그리기
plot(density(faithful$waiting))   # 연속 확률 밀도 변환
```

☞ 실행 결과





단골 코드 정리하기

• 회귀 분석 → 배기량(displ)과 시내 연비(cty) 관계 분석

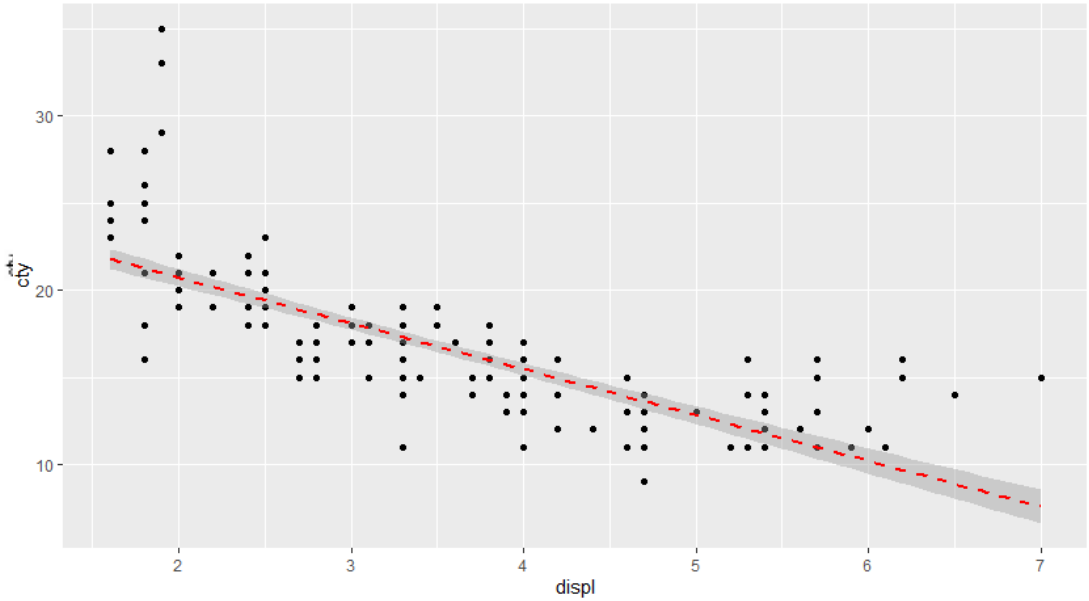
```
mpg_lm <- lm(mpg$cty ~ mpg$displ)  # 회귀식 모델링
mpg_lm
```

☞ 실행 결과

```
Call:
  lm(formula = mpg$cty ~ mpg$displ)
Coefficients:
(Intercept)    mpg$displ
 25.99        -2.63
```

```
ggplot(mpg, aes(x=displ, y=cty)) +  # 회귀 모델 차트
   geom_point() +
   stat_smooth(method='lm', colour="red", linetype = "dashed")
```

☞ 실행 결과



단골 코드 정리하기

• 주성분 분석 → iris 데이터 PCA 분석

```
library(ggfortify)
df <- iris[1:4]   # 데이터 추출
pca <- prcomp(df, scale. = TRUE) # 주성분 분석
pca
```

☞ 실행 결과

```
                PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
autoplot(pca, data = iris, colour = 'Species', loadings.label=T, loadings.label.size=3)
# 바이플롯 시각화
```

☞ 실행 결과