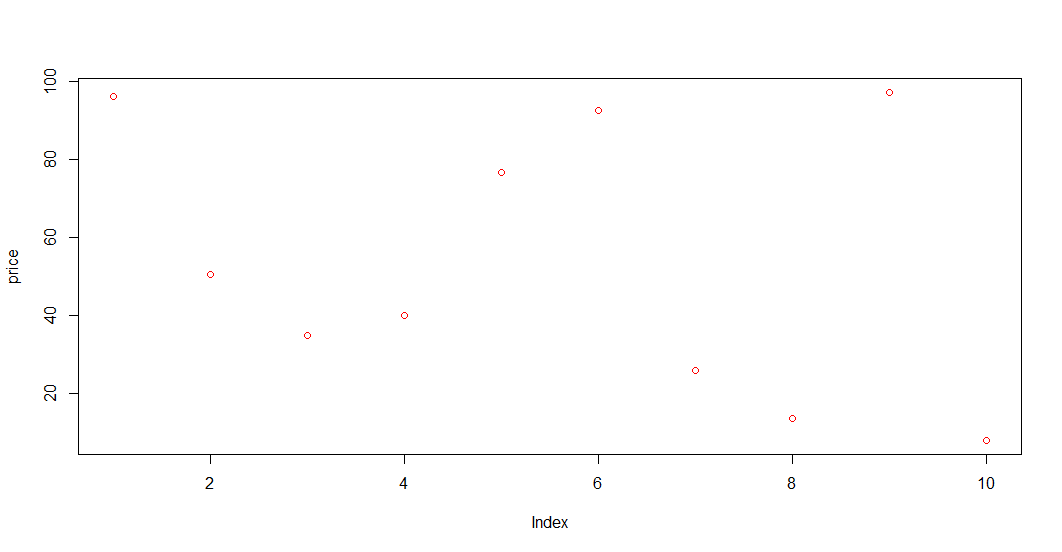
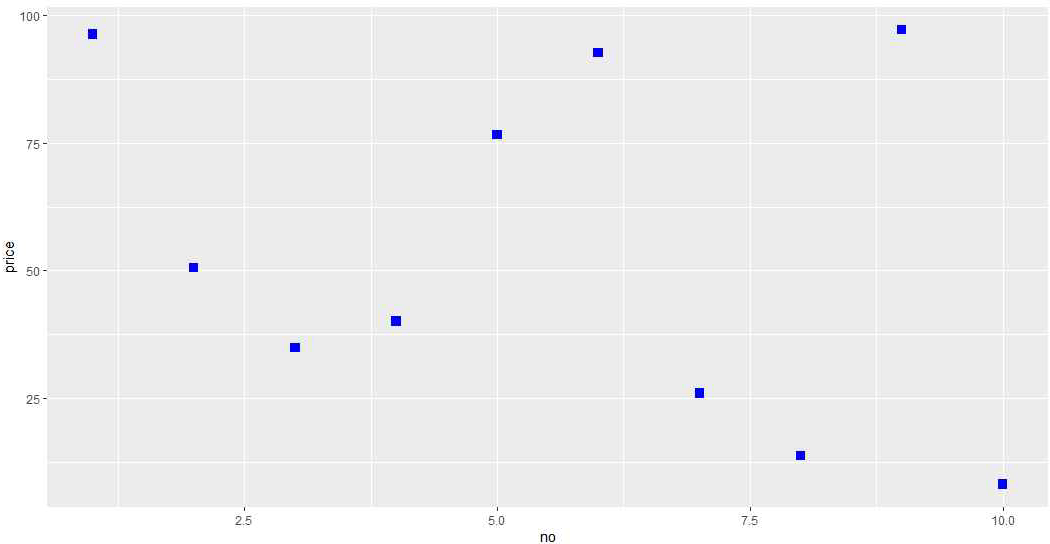
1. 산점도

산점도는 다음 데이터를 기준으로 그렸다.

|  |  |  |
| --- | --- | --- |
|  | pirce | no |
| 1 | 96.241006 | 1 |
| 2 | 50.597622 | 2 |
| 3 | 34.894016 | 3 |
| 4 | 40.117647 | 4 |
| 5 | 76.598465 | 5 |
| 6 | 92.601992 | 6 |
| 7 | 25.984746 | 7 |
| 8 | 13.800023 | 8 |
| 9 | 97.143501 | 9 |
| 10 | 8.141265 | 10 |

이것이 R교재에서 plot()함수를 사용하여 그렸던 그래프다.



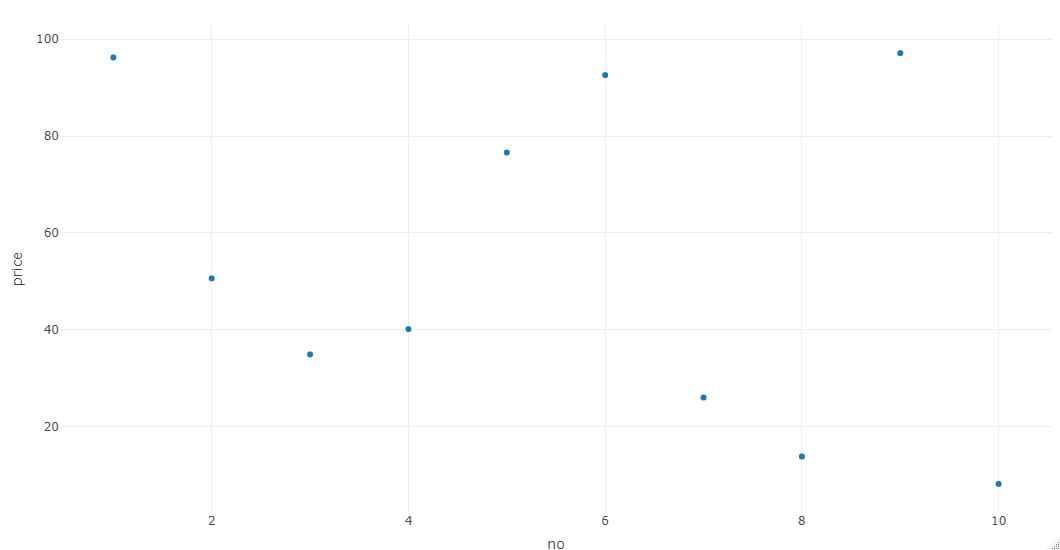
--

ggplot(data2, aes(x=no, y=price)) + geom\_point(shape=15, size=3, colour="blue")

--

위 그래프는 ggplot2 패키지의 geom\_point() 함수를 사용했다.

shape = 15 는 pch 15번 모양으로 점의 모양을 설정하고 size = 3 으로 3의 크기로, colour를 통해 ‘blue’색으로 그래프를 그렸다.

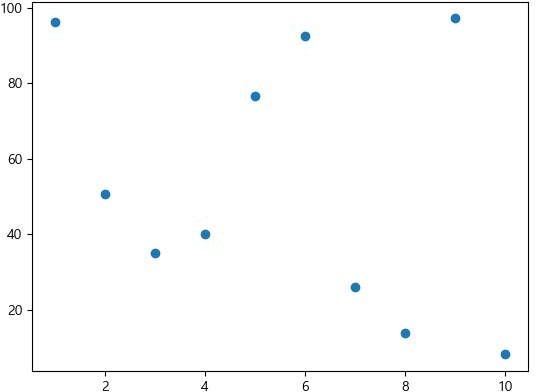


--

plot\_ly(data2 ,type='scatter', x=~no, y=~price)

--

plotly패키지의 plot\_ly()함수를 사용한 그래프는 type=’scatter’ 옵션을 사용하여 그렸다.



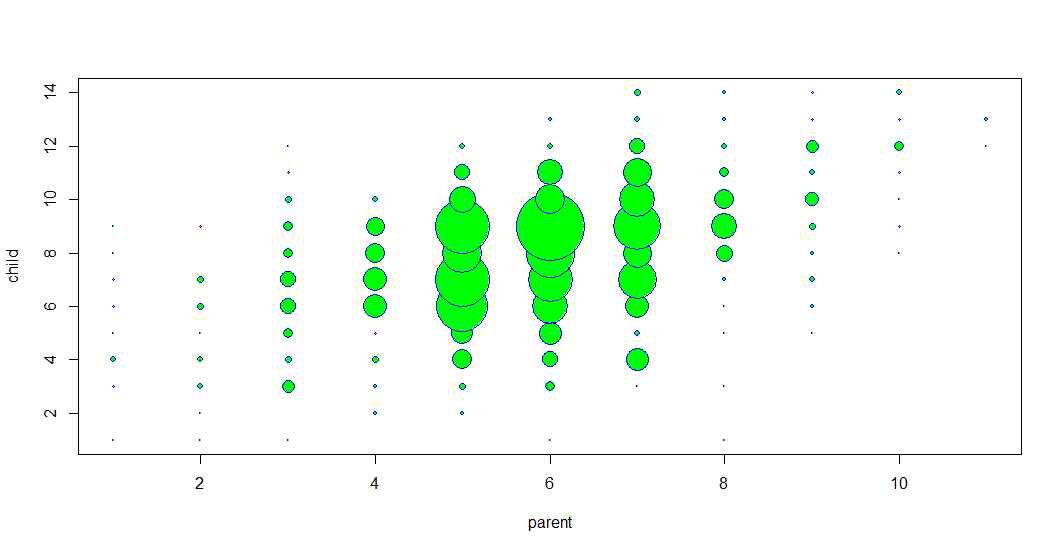
--

plt.scatter(x1, y1) plt.show

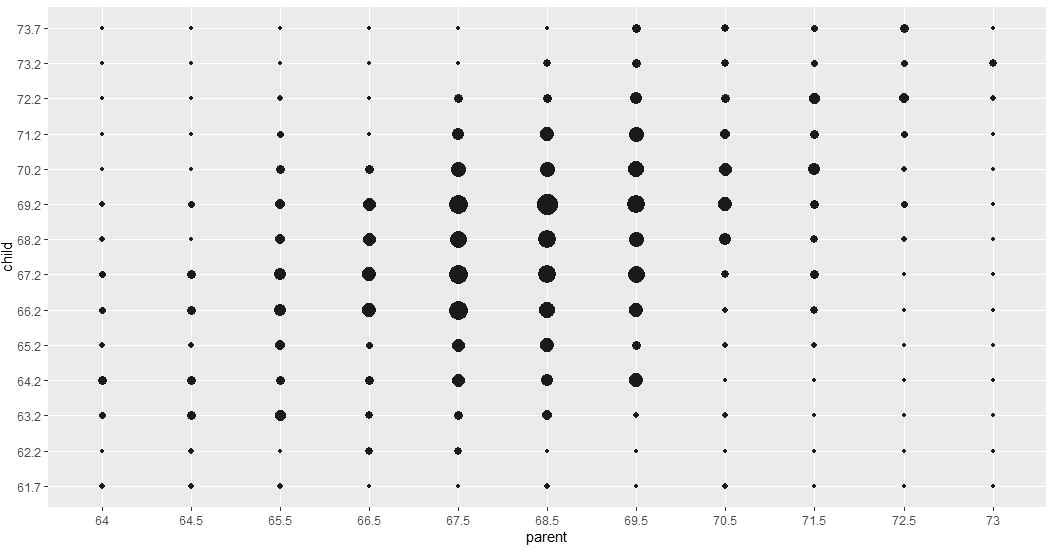
--

python 의 matplotlib 패키지에서는 .scatter 매소드를 사용하여 산점도 그래프를 그렸다.

ggplot2 의 layer 개념과 plotly 의 반응형을 제외하면 산점도에서는 각각의 큰 차이가 없 다.

1. 중첩자료 시각화

R교재에서 plot() 함수를 사용하여 중첩자료를 표현한 그래프는 위와 같다.



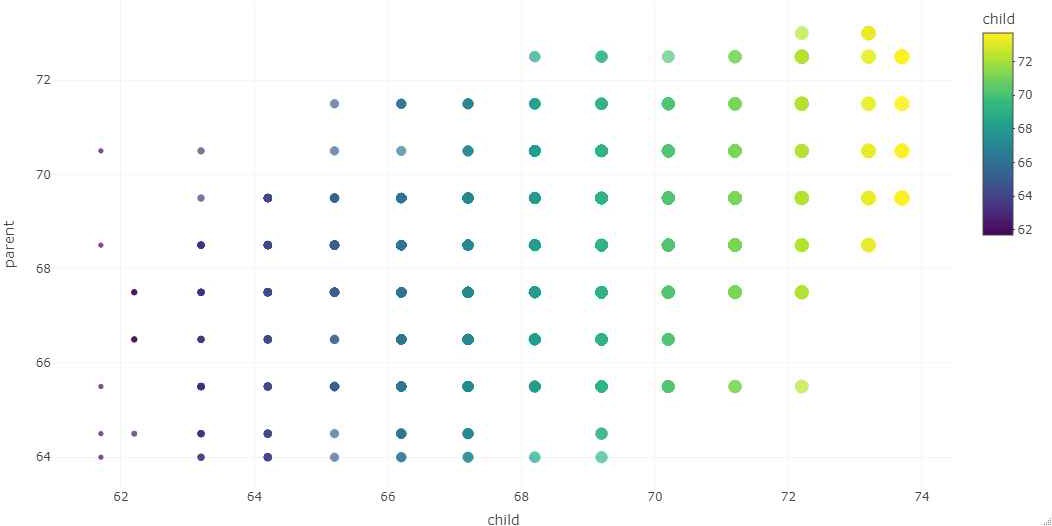
--

g <- ggplot(data=freqData, aes(x = parent, y = child))+ scale\_size(range = c(1,7), guide = 'none') + geom\_point(colour="grey10", aes(size=freq)) ;g

--

ggplot2 패키지에서 scale\_size()함수와 geom\_point()함수를 사용하여 그렸다.

range = c(1,7)를 통해 점 크기의 최소&최대를 설정하고 guide = ‘none’ 은 모르겠다. colour=“grey10”으로 점의 색을 설정하고 aes(size=freq)으로 빈도수에 따른 점 크기를 분배했다.



--

ga\_pl <- plot\_ly(galton,

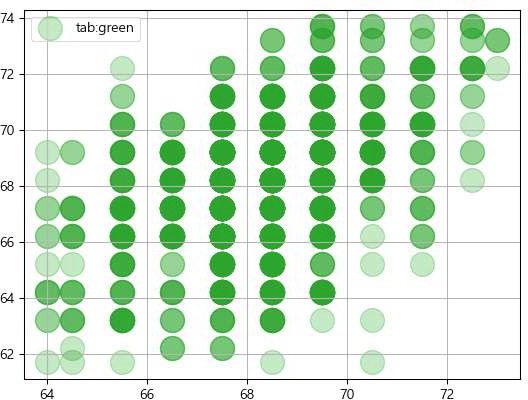
x = ~child , y = ~parent, color = ~child , size = ~child)

ga\_pl

--

plotly 패키지의 plot\_ly()함수를 사용하여 그린 그래프다.

왜 color 값과 size 값에 child 데이터를 넣었는지 모르겠다.



--

fig, ax = plt.subplots() for color in ['tab:green']:

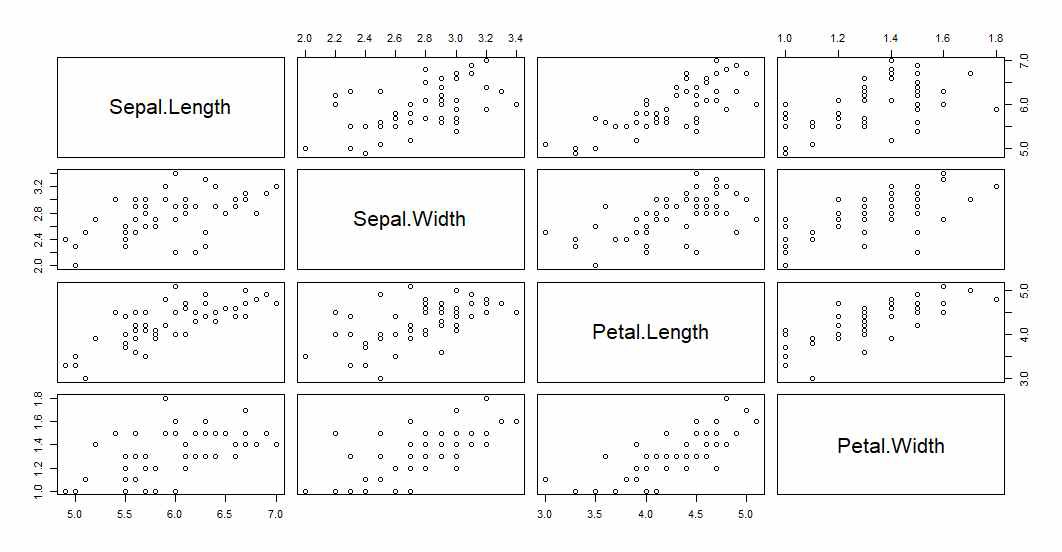
x = data["parent"] y = data["child"] scale = 300

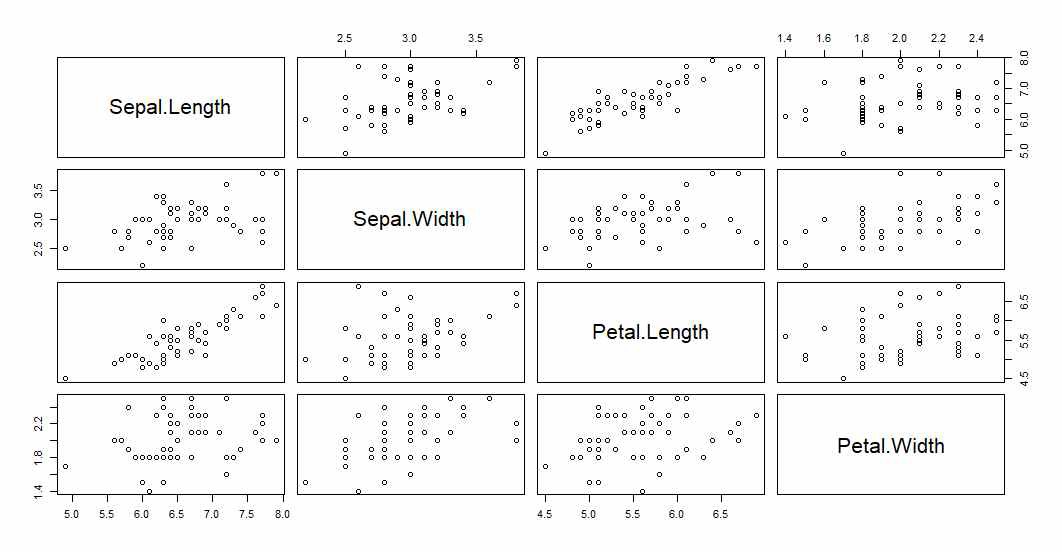
ax.scatter(x, y, c=color, s=scale, label=color, alpha=0.3)

ax.legend() ax.grid(True)

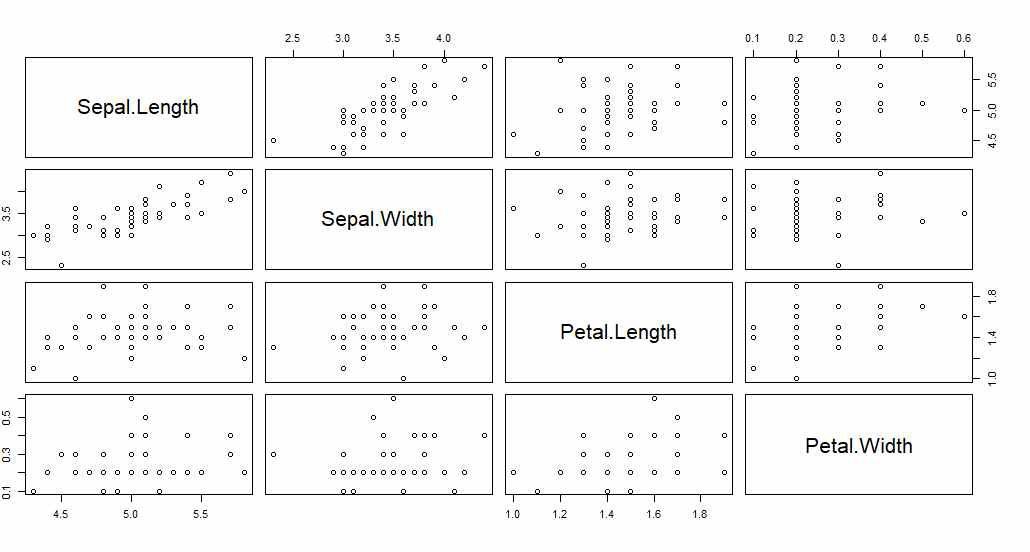
--

python에서 matplotlib 패키지의 .subplots() 매소드와 .scatter() 매소드를 활용했다. 그리고 for문을 사용하여 중첩자료를 시각화했다.

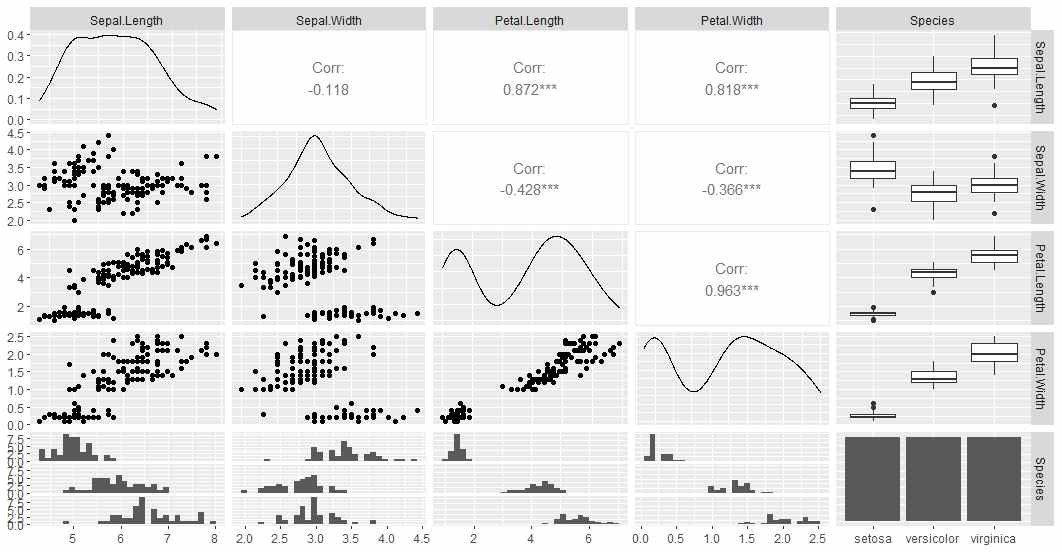
1. 변수간 비교 시각화

vesicolor 의 변수간 비교 자료이다.

virginica 의 변수간 비교 자료이다.



setosa 의 변수간 비교 자료이다.

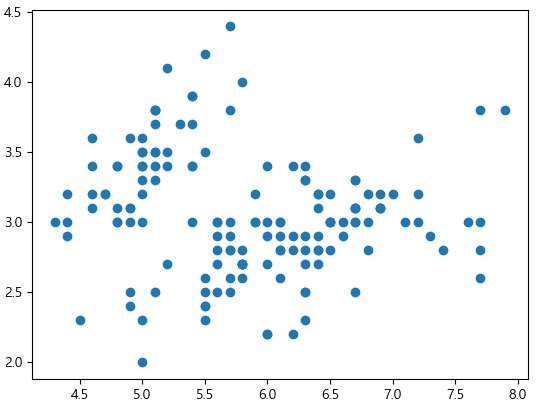


--

ggpairs(iris, columns = colnames(iris))

--

ggplot2 패키지에서는 ggpairs()함수를 사용하여 한 눈에 변수간 비교를 시각화 할 수 있 다.



위 그래프는 python에서 matplotlib 패키지에 포함되어 있는 .scatter 매소드를 사용하여 sepal.length와 sepal.width를 비교한 그래프이다.

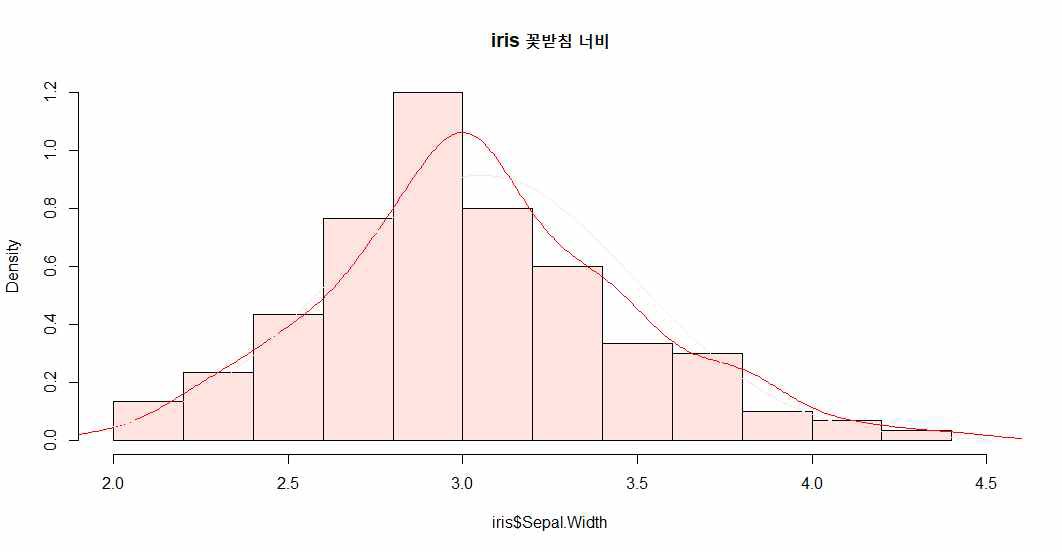
--

plt.scatter(iris.data[:,0],iris.data[:,1])

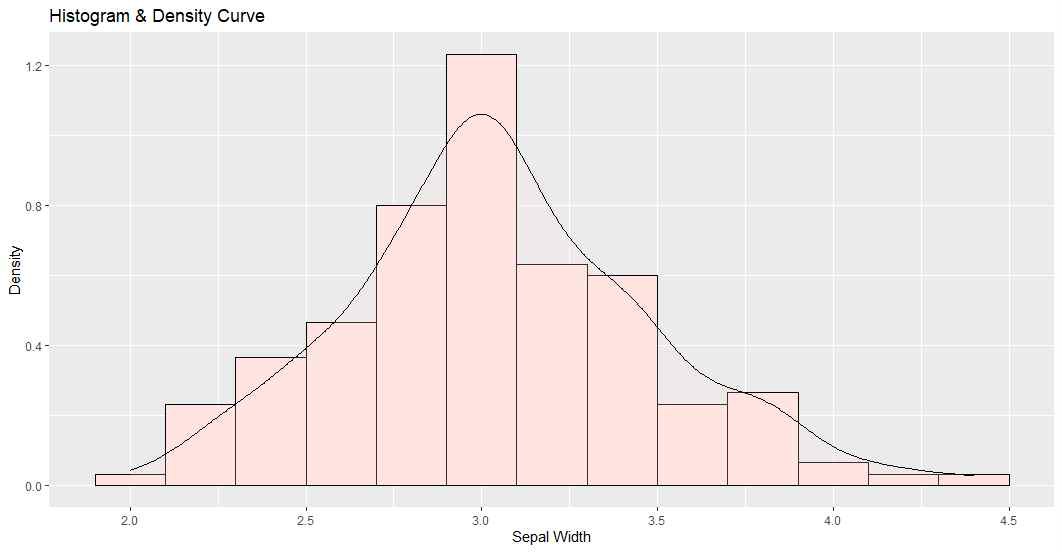
plt.scatter(iris.data[:,2],iris.data[:,3])

--

이처럼 변수를 비교한 자료를 시각화 할 때 ggplot2를 사용하면 가장 간편하게 시각화할 수 있고 python 에서는 더 많은 코드를 사용해야 한다.

1. 밀도 그래프

위는 R교재에서 hist(), lines(), curve() 함수를 사용하여 그린 밀도그래프이다.



--

density <- ggplot(data=iris, aes(x=Sepal.Width))

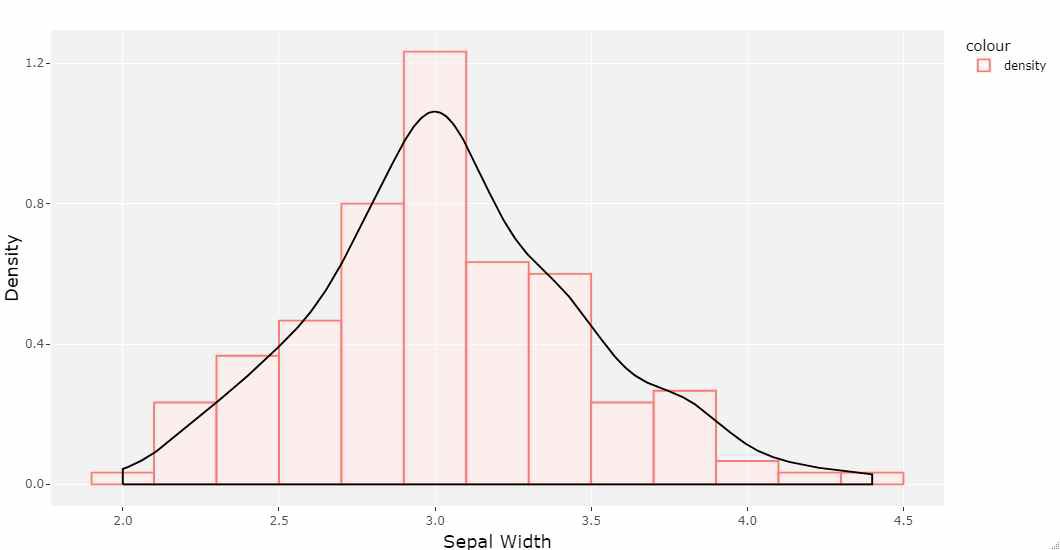
density + geom\_histogram(binwidth=0.2, color="black", fill="mistyrose", aes(y=..density..)) +

geom\_density(stat="density", alpha=I(0.2), fill="mistyrose") +

xlab("Sepal Width") + ylab("Density") + ggtitle("Histogram & Density Curve")

--

ggplot2 패키지에선 geom\_histogram()함수에서 aes(y=..density..)를 써서 밀도값으로 히 스토그램을 그렸고 geom\_density()함수를 써서 밀도선을 그렸다.



--

gg <- ggplot(iris,aes(x = Sepal.Width, color = 'density')) +

geom\_histogram(aes(y = ..density..), binwidth = 0.2, fill = 'mistyrose', alpha =

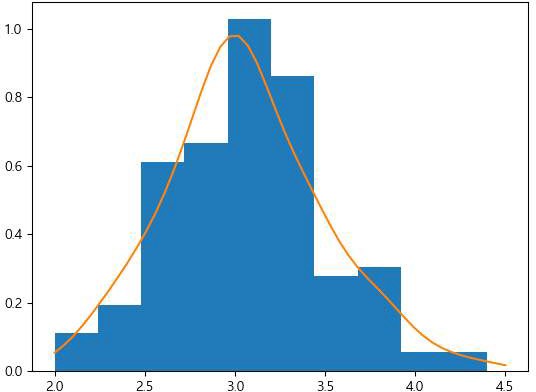
0.5) +

geom\_density(color = 'black') ggplotly(gg)%>%

layout(xaxis = list(title='Sepal Width'), yaxis = list(title='Density'))

--

plotly 패키지에서는 ggplotly를 사용하였는데 ggplot2패키지와 유사하다.



--

df = pd.read\_csv('iris.csv')

kde = gaussian\_kde(df["Sepal.Width"]) X = np.linspace(2, 4.5)

Y = kde(X)

fig, ax = plt.subplots()

hists = ax.hist(df["Sepal.Width"], density=True) ax.plot(X, Y)

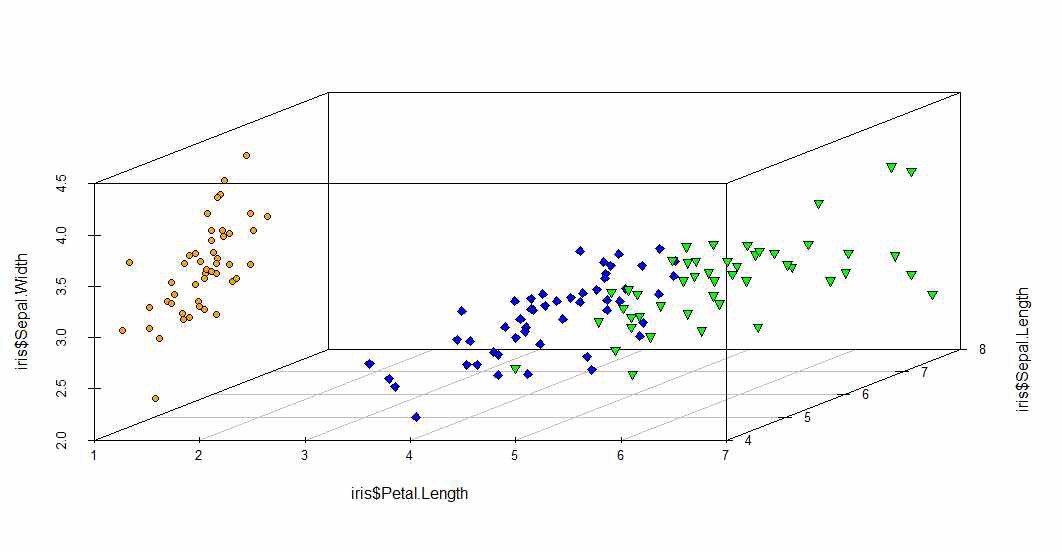
plt.show()

--

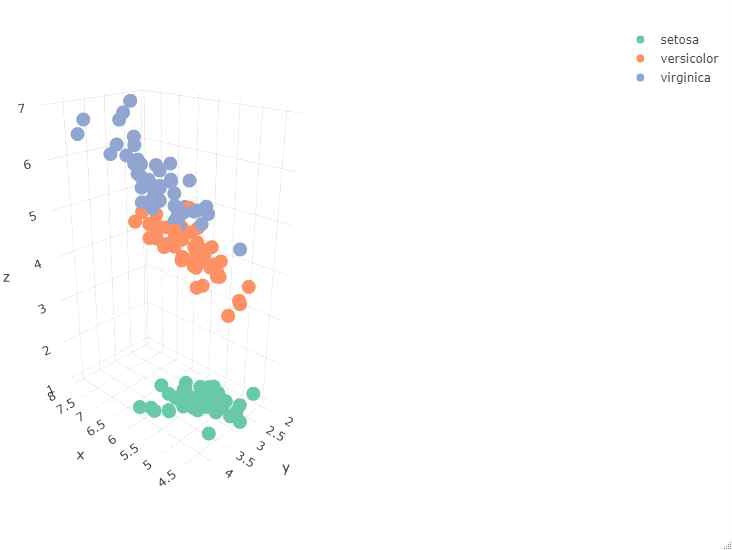
python에서 matplotlib 패키지와 scipy 패키지를 사용했다.

scipy.stats에서 gaussian\_kde를 통해 밀도를 계산하여 .hist 매소드와 .plot 매소드를 사 용하여 밀도 그래프를 그렸다.

plotly 패키지와 matplotlib 패키지는 결국 외부 패키지를 사용해야 하는 번거로움이 있지 만, ggplot2는 자체적으로 밀도를 표현할 수 있다.

1. 3차원 산점도 그래프

R교재에서 scatterplot3d 패키지를 사용하여 그렸던 그래프는 위와 같다.



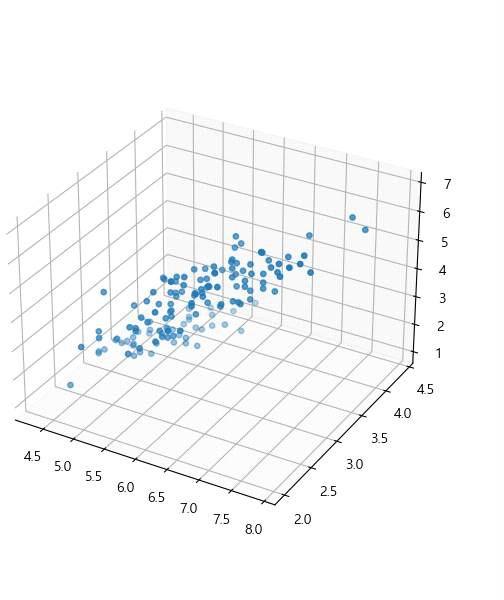
--

p1 <- plot\_ly(iris, x = iris$Sepal.Length,

y = iris$Sepal.Width, z = iris$Petal.Length, color = iris$Species)

--

plotly 패키지에서 plot\_ly() 함수를 사용하여 3d 산점도를 그린 그래프는 위와 같다.



--

iris = sns.load\_dataset('iris')

xs = iris[["sepal\_length"]] ys = iris[["sepal\_width"]] zs = iris[["petal\_length"]]

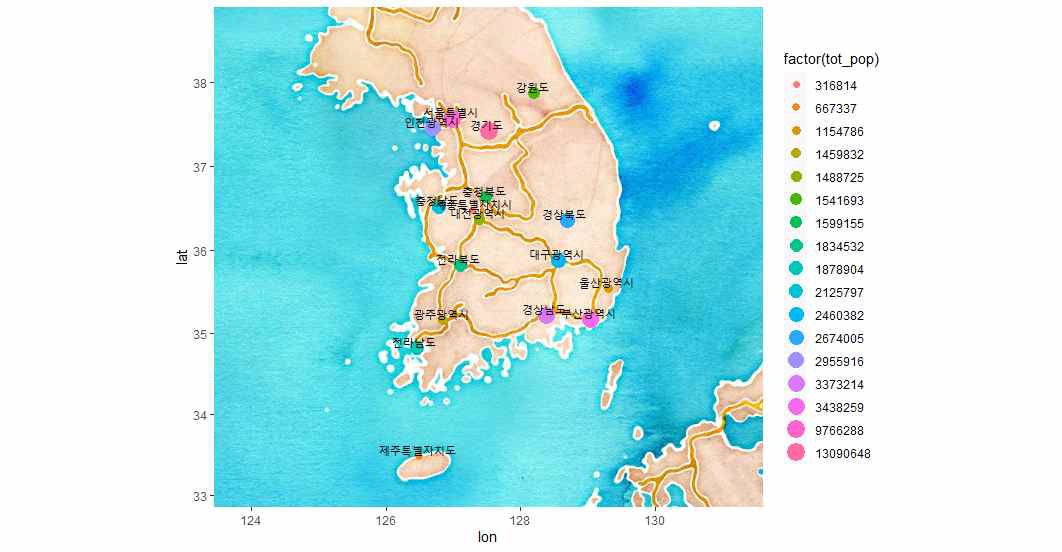
fig = plt.figure(figsize=(6, 6))

ax = fig.add\_subplot(111, projection='3d')

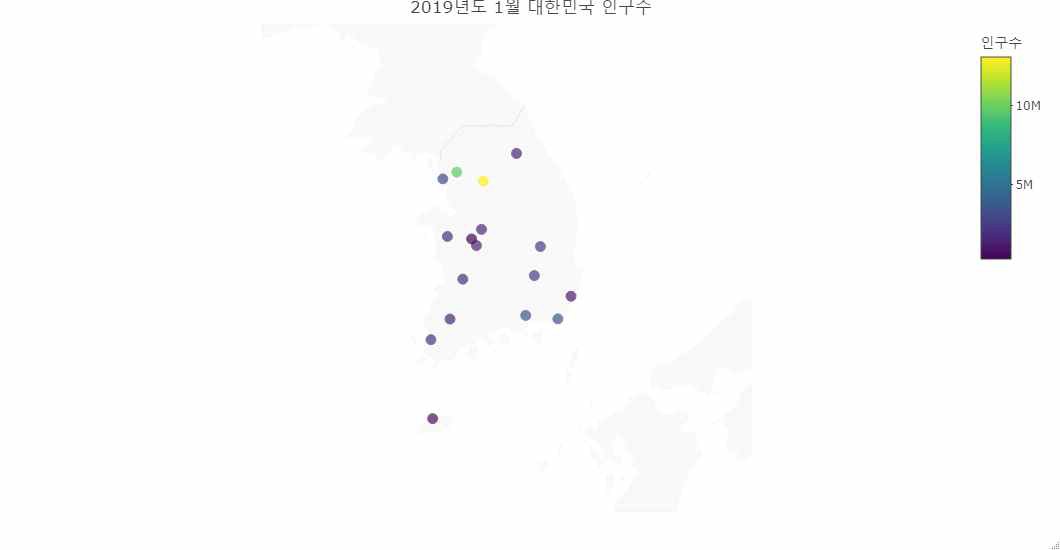
ax.scatter(xs, ys, zs, marker='o', s=15, cmap='Greens')

--

python에서 matplotlib 패키지를 이용하여 3d산점도를 그렸다.

1. 지도 시각화

R교재에서 ggplot2 패키지를 사용하여 그려본 지도 시각화는 위와 같다.



--

gmap <- list( scope = 'asia',

showland = TRUE, landcolor = toRGB("gray95"),

countrycolor = toRGB("gray85"), countrywidth = 0.5

)

gmap2 <- c( gmap, resolution = 50,

list(lonaxis = list(range = c(123, 133))),

list(lataxis = list(range = c(32, 40)))

)

fig3 <- plot\_geo(df, lat = ~lat, lon = ~lon) fig3 <- fig3 %>% add\_markers(

text = ~paste(region, paste(tot\_pop, "명"), sep = "<br />"),

color = ~tot\_pop, symbol = I("circle"), size = I(50), hoverinfo = "text"

)

fig3 <- fig3 %>% colorbar(title = "인구수") fig3 <- fig3 %>% layout(

title = "2019년도 1월 대한민국 인구수", geo = gmap2

)

fig3

--

먼저 plotly 패키지의 layout.geo에 들어갈 옵션을 list형식으로 나열한 gmap 객체와 gmap2 에 대한 설명이다.

gmap에선 scope = ‘asia’ 를 사용해 아시아지역의 지도를 지정하고, showland = TRUE 는 지도에서 땅의 색상을 설정하여 채울지 말지의 설정이다. landcolor = toRGB("gray95")를 통해 땅의 색상을 지정하고, countrycolor = toRGB("gray85"), countrywidth = 0.5를 써서 국경선의 색상과 두께를 지정한다.

gmap2에선 gmap을 포함하여, resolution = 50를 통해 지도의 해상도를 설정한다. 값의 단위는 km/mm이다다. 예를 들어 50은 1:50,000,000의 축척 비율에 해당한다. 설정하지 않는 다면 기본값은 110이다. 그리고 list(lonaxis = list(range = c(123, 133))), list(lataxis = list(range = c(32, 40)))를 넣어 우리나라 위치의 경도와 위도 범위를 지정 해준다.

plot\_geo()함수로 데이터값의 경도, 위도를 지정하고

add\_markers()함수로 지정된 위치에 symbol의 모양, 크기, 설명text를 넣는다.