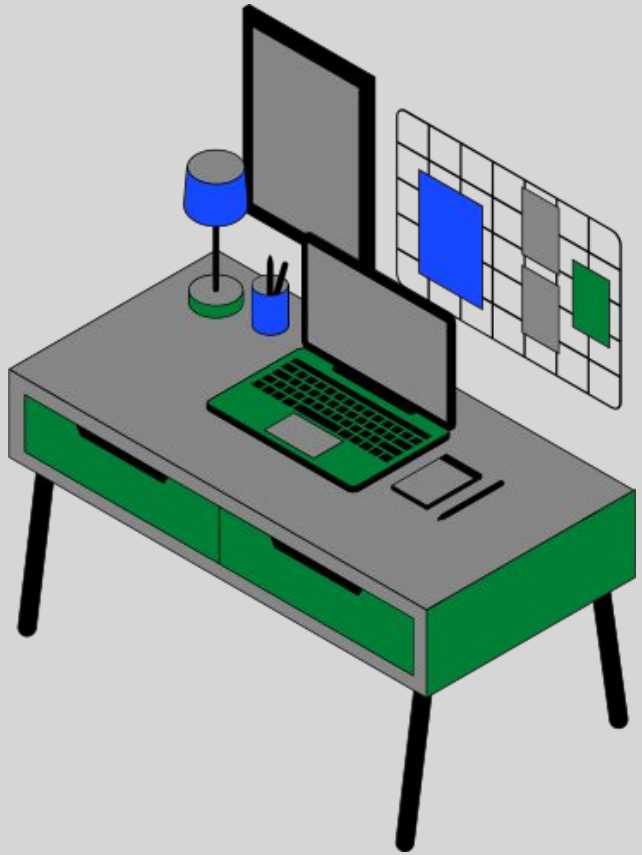
데이터 시각화 교과서

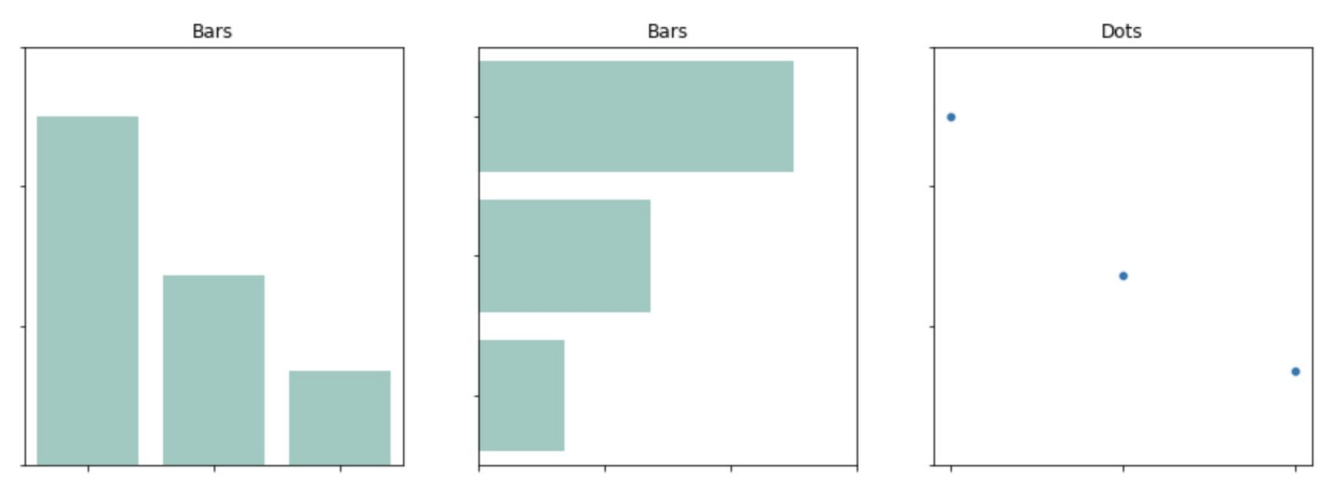
Chapter 5. 다양한 시각화 방식



수량의 시각화

수량을 시각화하는 방법

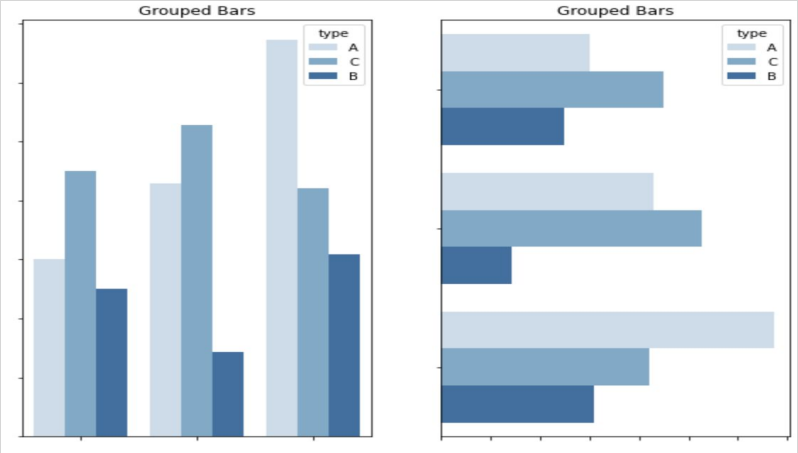
-막대를 가로/세로로 그림
-막대를 그리지 않고 막대 끝에 점을 찍음

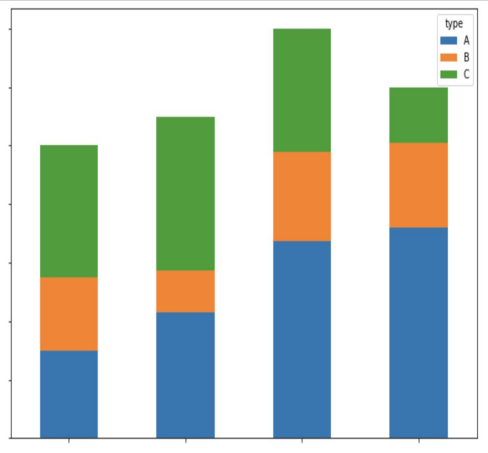


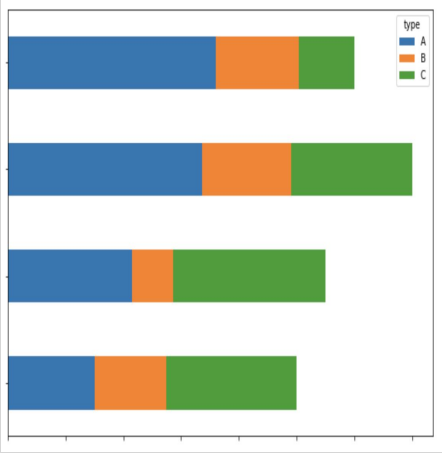
분포의 시각화

수량 값을 지닌 범주가 2개 이상일때

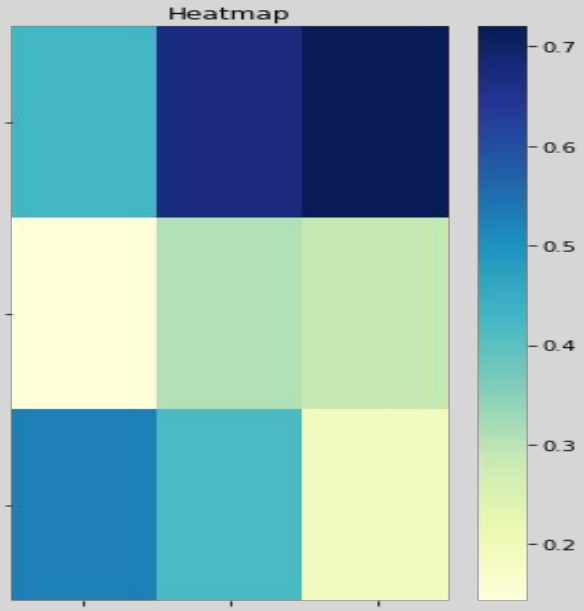
막대들을 그룹으로 묶거나 쌓을 수 있음







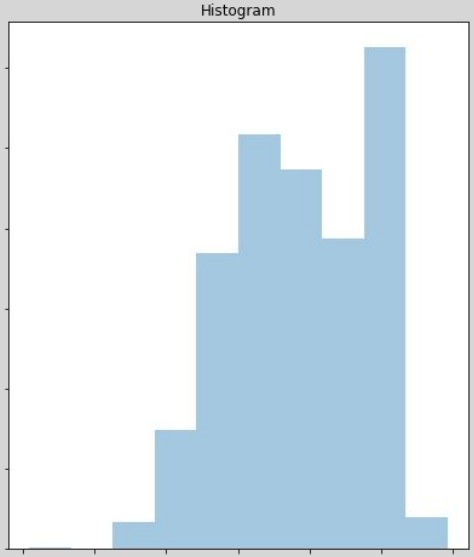
분포의 시각화

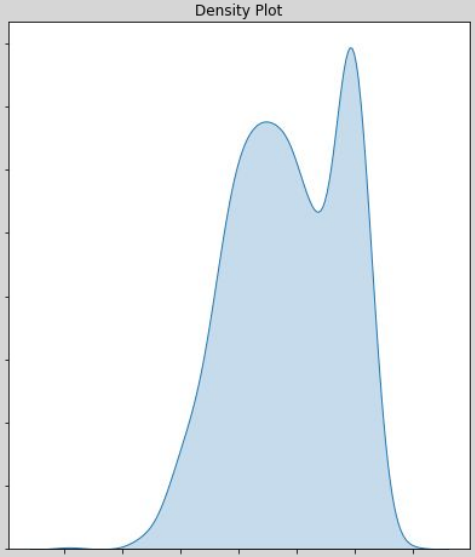


수량 값을 지닌 범주가 2개 이상일때

히트맵의 x축과 y축에 범주를 표시하고
색으로 수량을 나타낼 수 있음

분포의 시각화





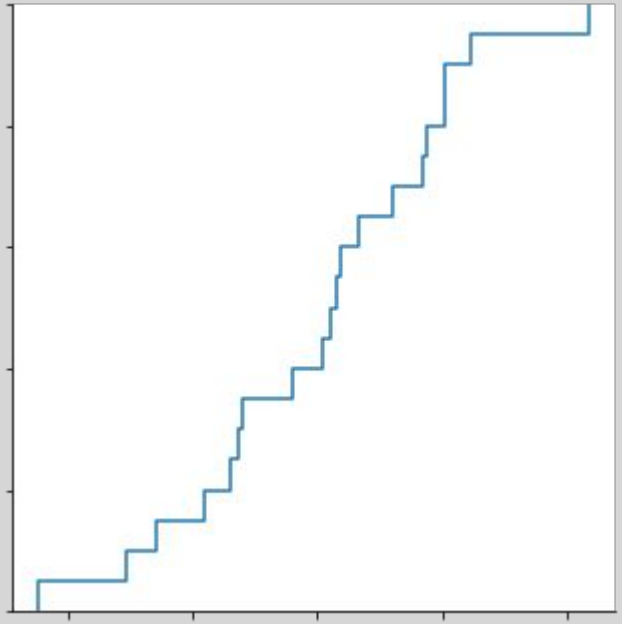
히스토그램과 밀도 도표

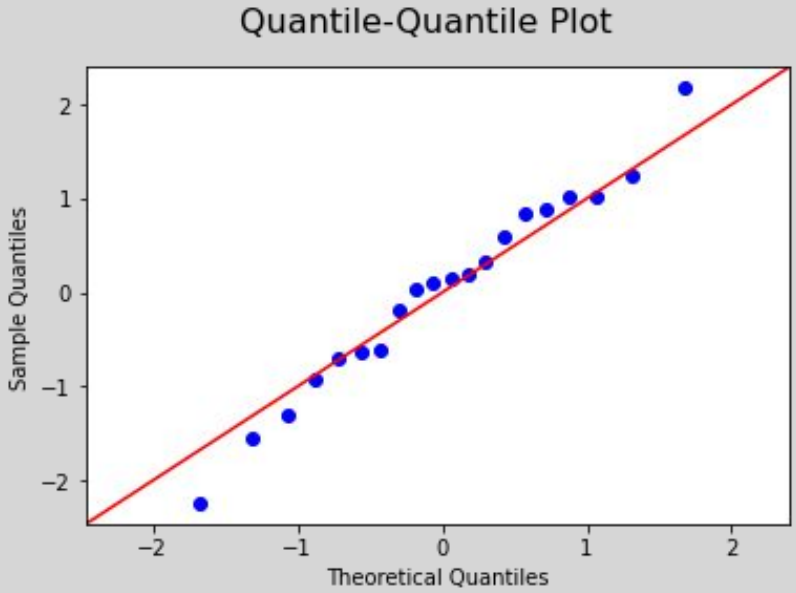
-분포 데이터를 가장 직관적으로 보여줌
-But, 두가지 모두 파라미터를 임의로 정해야 해서 데이터가 잘못 전달될 수 있음

분포의 시각화

누적 밀도 도표와 QQ 도표

- 데이터를 정확하게 보여줌
-But, 의미를 파악하기 까다로움

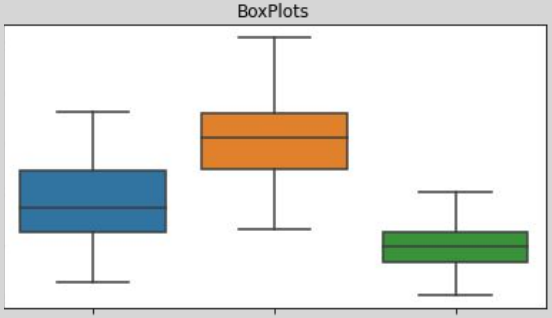


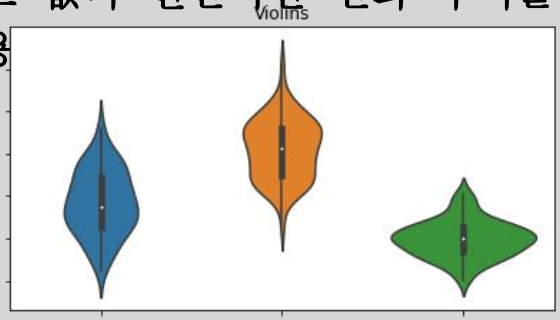


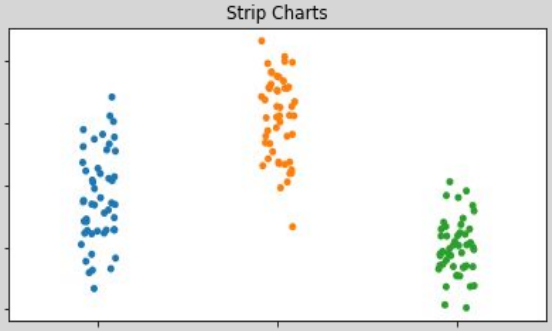
분포의 시각화

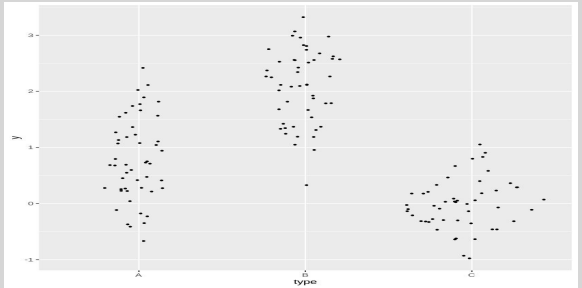
박스플롯, 바이올린 도표, 스트립 차트, 시나도표

여러 분포 값을 한번에 나타내려 하거나, 분포 값의 전반적인 변화 추이를 보려 할 때 유용





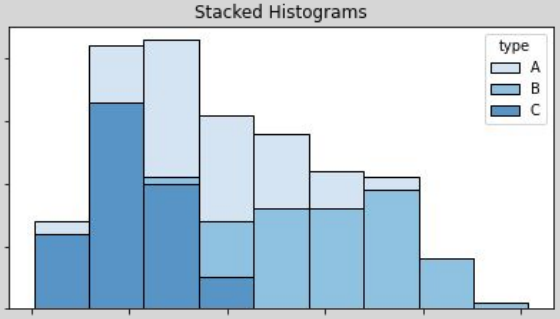


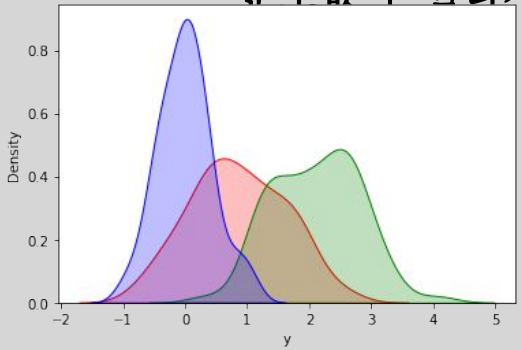


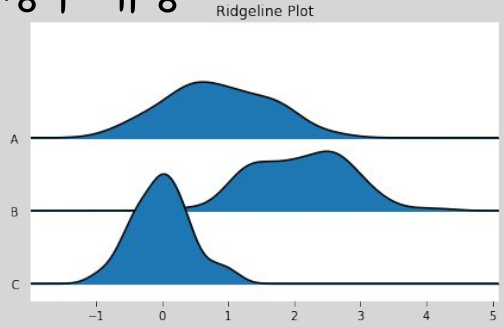
분포의 시각화

누적 히스토그램&중첩 밀도 도표
-분포 값의 개수가 적을 때 심층 비교 분석 가능
-But, 누적 히스토그램: 데이터를 해석하기 어려움

융기선 도표
-바이올린 도표 대신 사용하기 좋음
-분포값이 아주 많은 경우/시간의 흐름에 따라 분포값이 달라지는 경우 유용





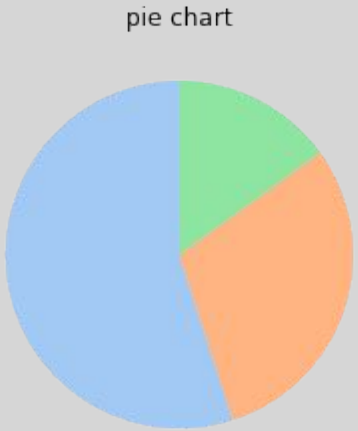


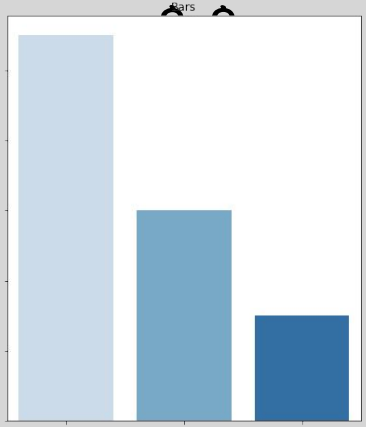
비율의 시각화

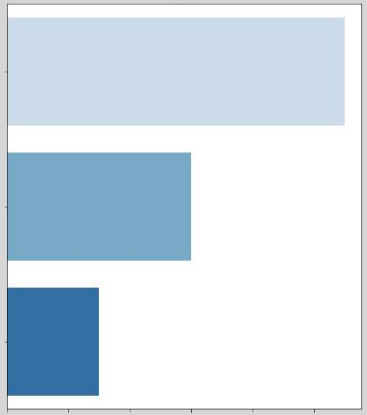
파이 차트: 부분이 모여 전체를 이룬다는 사실 강조

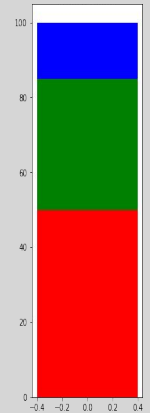
막대: 각 조각 쉽게 비교 가능

누적막대: 다중 데이터셋의 비율을 비교할 때





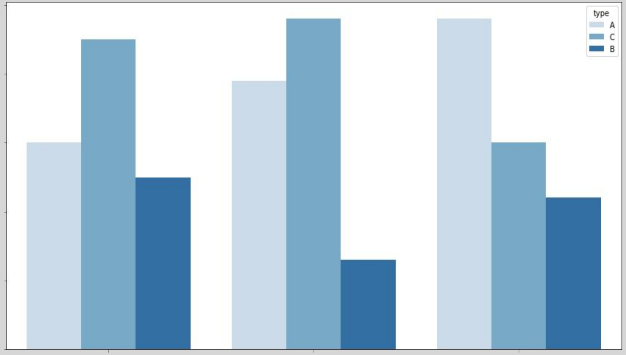


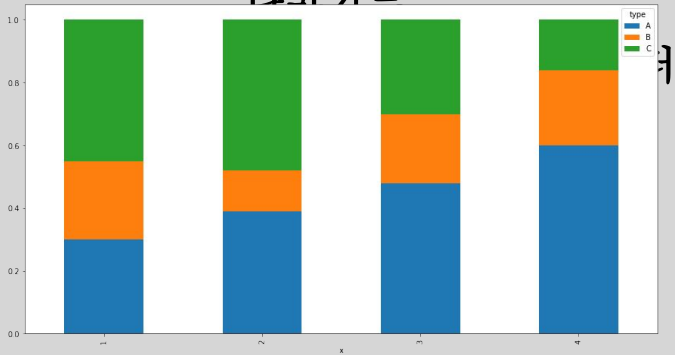


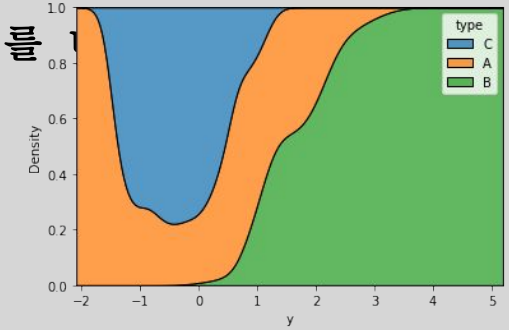
비율의 시각화

여러 개의 비율 데이터셋을 다루거나 조건에 따라 변동되는 비율 값을 나타내는 경우
=> 파이 차트를 사용하게 되면 공간 낭비+집합 간 관계 나타나지 X

묶은 막대 차트: 비교할 조건의 개수가 적당할 때
누적 막대 도표: 비교할 조건의 수가 많을 때
누적 밀도 도표: 연속형 변수에 따라 비율이 [illegible]





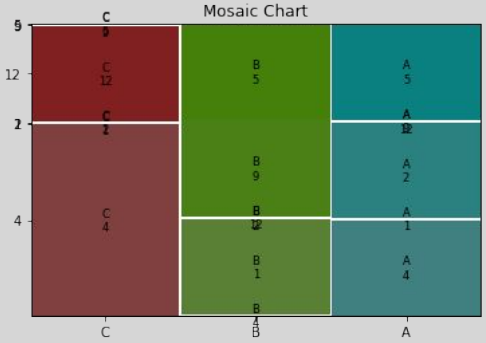


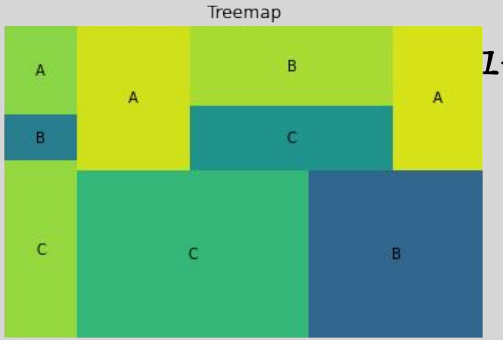
비율의 시각화

여러 그룹화 변수에 따라 비율값이 정해지는 경우
=>모자이크 도표, 트리맵, 평행집합이 유용

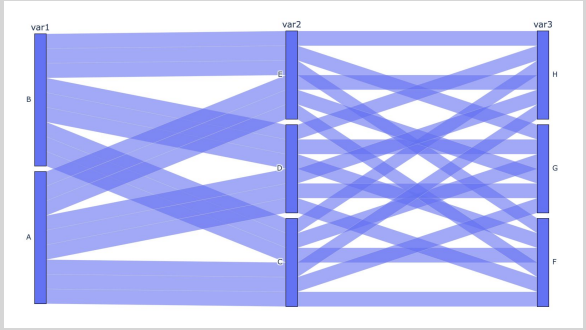
모자이크 도표: 한 그룹화 변수의 모든 수준을 다른 그룹화 변수의 모든 수준과 결합할 수 있다고 가정

트리맵: 한 그룹의 하위부가 다른 그룹의 하위부와 아예 다르더라도 구애 받지 않음





그룹화

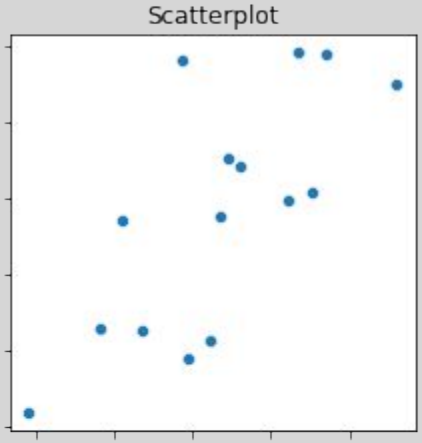


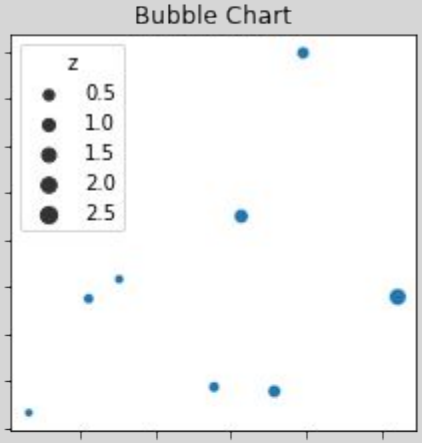
x-y 관계로 나타내는 시각화

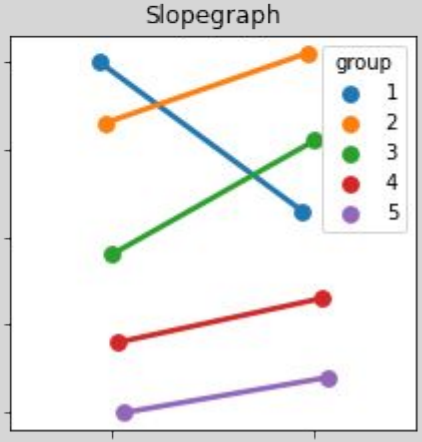
산점도: 하나의 변수를 다른 변수와 비교해 나타낼 때

버블 차트: 정량적 변수가 세개일때 하나의 변수를 점 크기로 나타내 산점도를 변형한 것

경사 차트: 쌍을 이루는 점들을 직선으로 연결



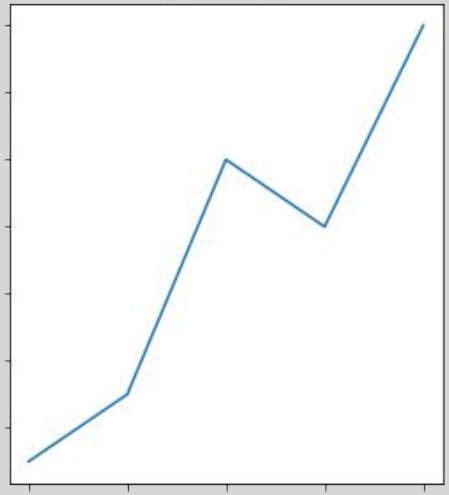


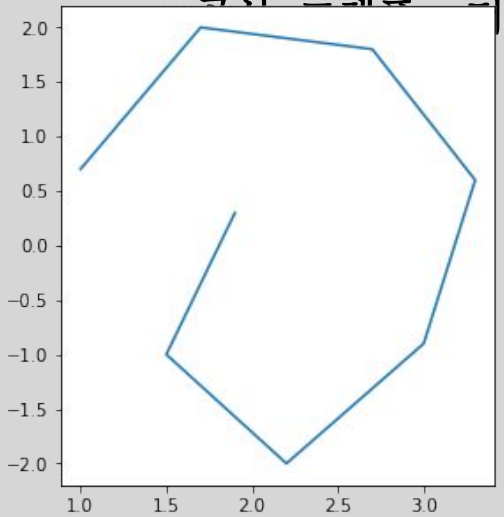


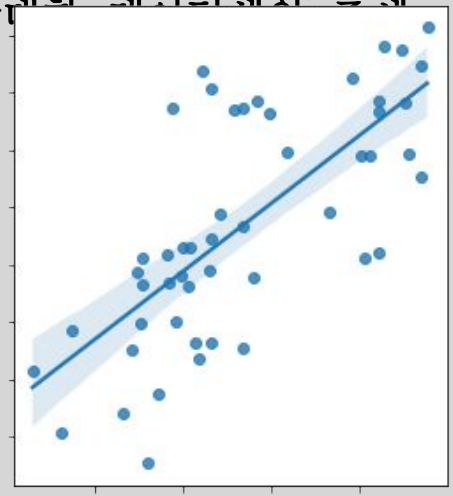
x-y 관계로 나타내는 시각화

x축 데이터가 엄밀하게 증가하는 수치(ex.시간)
=> 꺾은선 그래프

연결된 산점도: 두 반응 변수의 시간적 순서 나타냄





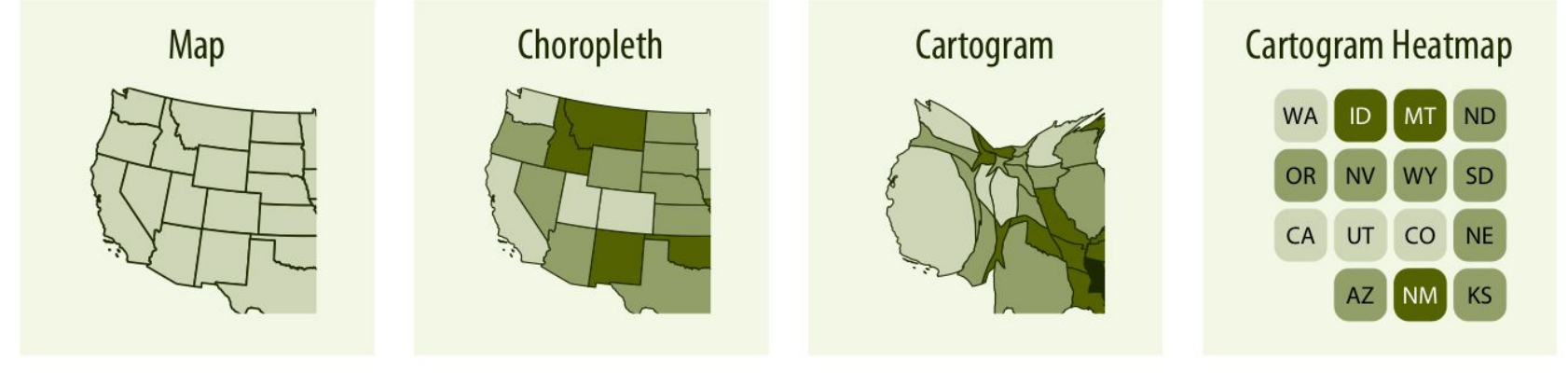


지리공간 데이터의 시각화

지리공간 데이터: 지도 위에 표시

단계구분도: 지도에 데이터를 기준으로 다른 책을 칠함

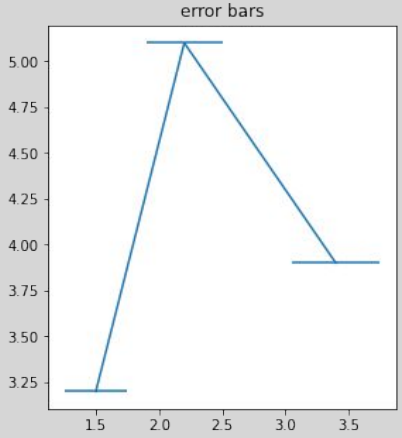
카토그램: 데이터 값에 따라 지역의 형태 왜곡 등

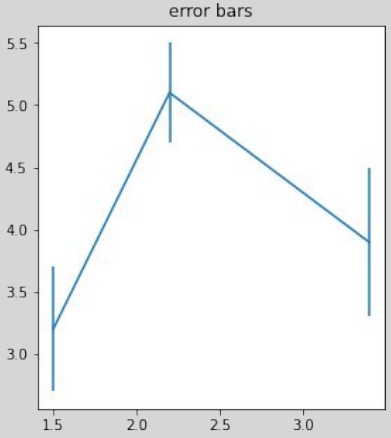


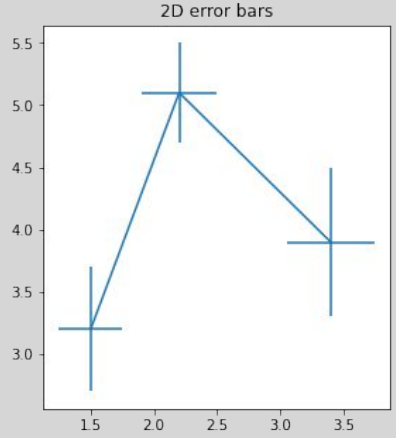
불확실성의 시각화

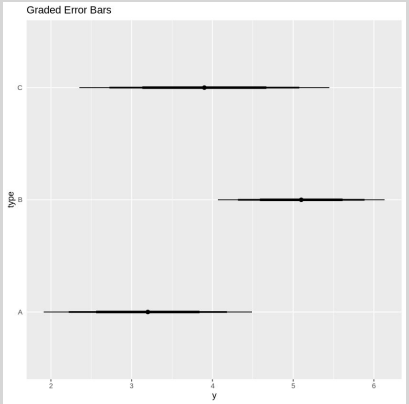
오차 막대
-데이터의 예상 값 범위를 나타냄
-기준점에서 가로/세로 방향으로 확장

단계별 오차 막대: 범위 마다 신뢰도가 다름







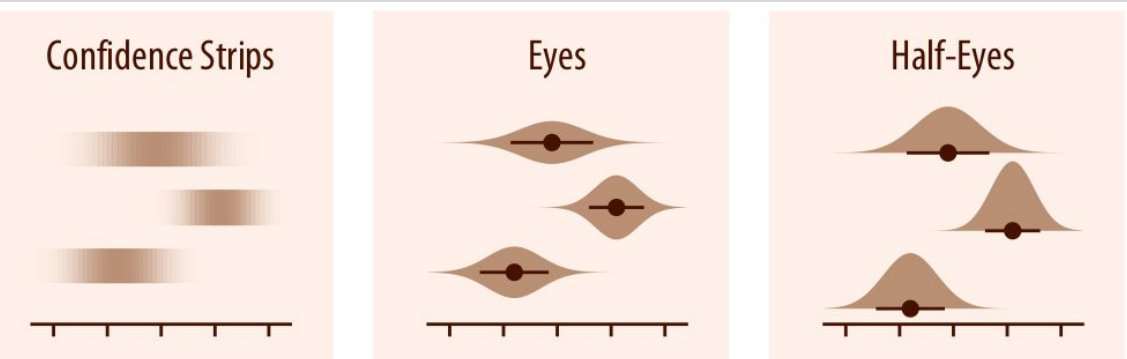


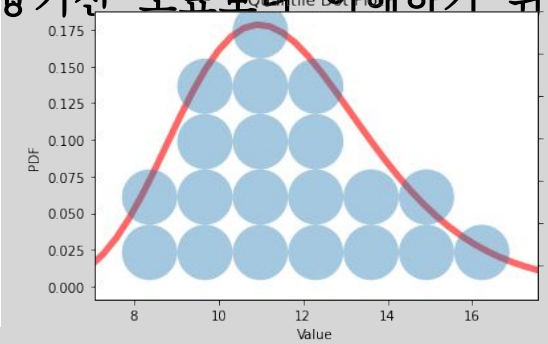
불확실성의 시각화

신뢰 스트립
-불확실성은 쉽게 알아볼 수 있음
-정확한 수치 읽기 어려움

눈/감은 눈 모양 도표
-오차 막대에 분포도를 시각화해 결합
-정확한 신뢰수준과 전반적인 불확실성 분포 상태 보여줌

분위수 점 도표
-아주 정확하진 않음
-바이올린 도표보다 이해하기 쉬움

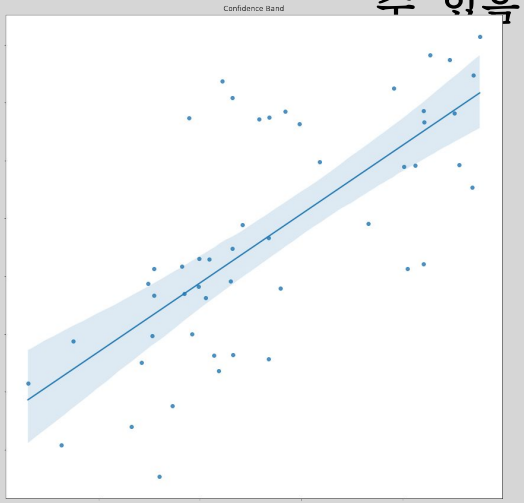


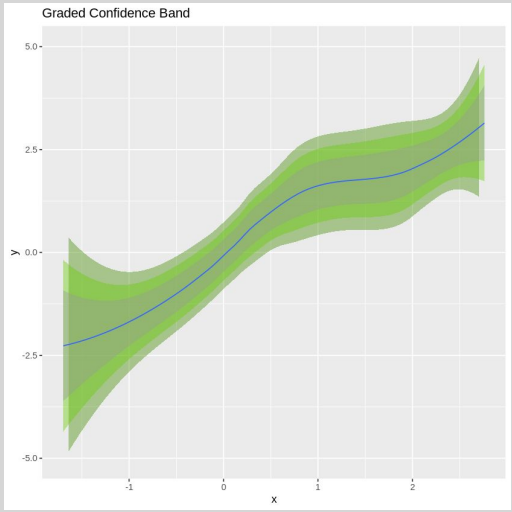


불확실성의 시각화

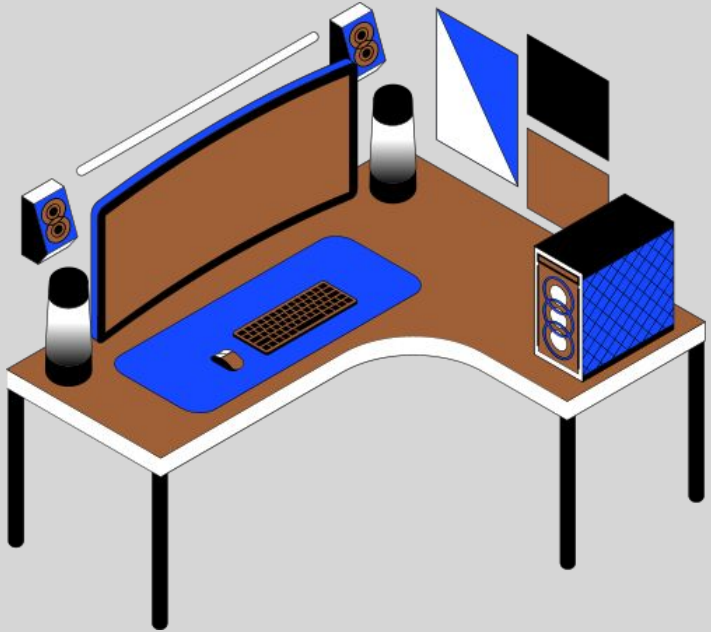
신뢰 대역: 곡선 그래프에서 오차 막대의 역할을 함

단계별 신뢰 대역: 다수의 신뢰수준을 한번에 볼 수 있음





```python
from matplotlib import pyplot as plt
import seaborn as sns
import random
import pandas as pd
from pandas import Series, DataFrame
from google.colab import drive
drive.mount('/content/drive')
fig = plt.figure(figsize=(15,5))
area1=fig.add_subplot(1,3,1)
area2=fig.add_subplot(1,3,2)
area3=fig.add_subplot(1,3,3)
area1.set_title('Bars')
area2.set_title('Bars')
area3.set_title('Dots')
df_props = DataFrame({"value" : [55, 30, 15],"group" : ["A", "B", "C"]})
a1 = sns.barplot(data =df_props,x='group', y='value',ax=area1,palette=sns.color_palette("Set3",1))
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a1.set_yticks([0,22,44,66])
a2 = sns.barplot(data =df_props,x='value', y='group',ax=area2,palette=sns.color_palette("Set3",1))
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2.set_xticks([0,22,44,66])
a3 = sns.scatterplot(data =df_props,x='group', y='value',ax=area3,palette=sns.color_palette("Set3",1))
a3.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a3.set_yticks([0,22,44,66])
plt.show()
fig = plt.figure(figsize=(15,8))
title=['Grouped Bars','Grouped Bars','Heatmap']
for i in range(1,4):
 globals()['area{}'.format(i)]=fig.add_subplot(1,3,i)
 globals()['area{}'.format(i)].set_title(title[i-1])
a1 = sns.barplot(data =df_multi_amounts[df_multi_amounts.x!=4],x='x', y='y',hue='type',hue_order=["A","C","B"],ax=area1,palette=sns.color_palette("Blues",3))
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2 = sns.barplot(data =df_multi_amounts[df_multi_amounts.x!=4],x='y', y='x',hue='type',hue_order=["A","C","B"],ax=area2,palette=sns.color_palette("Blues",3),orient='h')
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a3 = sns.heatmap(data =a3_df,ax=area3,cmap='YlGnBu')
a3.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
plt.show()
#stacked bar는 seaborn에서 지원 안함
stacked_df = df_multi_amounts.pivot('x','type','y')
stacked_bar =stacked_df.plot.bar(stacked=True, figsize=(10,7))
stacked_bar.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
stacked_bar_h = stacked_df.plot.barh(stacked=True, figsize=(10,7))
stacked_bar_h.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
df_one_dist=pd.read_csv('df_one_dist.csv')
df_one_normal=pd.read_csv('df_one_normal.csv')
df_multi_dist=pd.read_csv('df_multi_dist.csv')
df_multi_dist_small=pd.read_csv('df_multi_dist_small.csv')

fig = plt.figure(figsize=(15,8))
title=['Histogram','Density Plot','Cumulative Density','Quantile-Quantile Plot']
for i in range(1,3):
 globals()['area{}'.format(i)]=fig.add_subplot(1,2,i)
 globals()['area{}'.format(i)].set_title(title[i-1])
a1 = sns.distplot(df_one_dist['x'],bins=10,ax=area1,hist=True,kde=False,rug=False)
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2 = sns.distplot(df_one_dist['x'],bins=10,ax=area2,hist=False,kde=True,rug=False,kde_kws={"shade":True})
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a3 = sns.displot(df_one_normal['x'],kind="ecdf",ax=area3)
a3.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
import statsmodels.api as sm
fig = sm.qqplot(df_one_normal['x'], line='45')
fig.suptitle('Quantile-Quantile Plot', fontsize=16)
plt.show()

df_multi_dist_a = df_multi_dist[df_multi_dist.type=='A']
df_multi_dist_b = df_multi_dist[df_multi_dist.type=='B']
df_multi_dist_c = df_multi_dist[df_multi_dist.type=='C']
fig = sns.kdeplot(df_multi_dist_a['y'], shade=True, color="r")
fig = sns.kdeplot(df_multi_dist_b['y'], shade=True, color="g")
fig = sns.kdeplot(df_multi_dist_c['y'], shade=True, color="b")
plt.show()
joyplot(df_multi_dist, by = 'type', column = 'y',title='Ridgeline Plot')
plt.show()
plt.pie(df_props['value'],colors=sns.color_palette('pastel'),startangle=90)
plt.title('pie chart')
plt.show()
fig = plt.figure(figsize=(15,8))
title=['Bars','Bars']
for i in range(1,3):
 globals()['area{}'.format(i)]=fig.add_subplot(1,2,i)
 globals()['area{}'.format(i)].set_title(title[i-1])
a1 = sns.barplot(data =df_props,x='group', y='value',ax=area1,palette=sns.color_palette("Blues",3))
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2 = sns.barplot(data =df_props,x='value', y='group',ax=area2,palette=sns.color_palette("Blues",3))
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
plt.show()

plt.figure(figsize=(3,8))
c=plt.bar(0,100,color='b')
b=plt.bar(0,85,color='g')
a=plt.bar(0,50,color='r')
plt.title('stacked bars')
plt.show()
plt.subplot(1,3,1)
plt.pie(df_fractions[df_fractions.x==1]['y'],colors=sns.color_palette('pastel'),startangle=90)
plt.subplot(1,3,2)
plt.pie(df_fractions[df_fractions.x==2]['y'],colors=sns.color_palette('pastel'),startangle=90)
plt.subplot(1,3,3)
plt.pie(df_fractions[df_fractions.x==3]['y'],colors=sns.color_palette('pastel'),startangle=90)
plt.show()
fig = plt.figure(figsize=(15,8))
title=['Grouped Bars','Stacked Bars']
a1 = sns.barplot(data =df_fractions[df_fractions.x!=4],x='x',
y='y',hue='type',hue_order=["A","C","B"],palette=sns.color_palette("Blues",3))
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
pivot_df = df_fractions.pivot(index='x', columns='type', values='y')
pivot_df.plot.bar(stacked=True, figsize=(15,8))
plt.show()
sns.kdeplot(data=df_multi_dist, x="y", hue="type",hue_order=["C","A","B"], multiple="fill")
plt.show()
import squarify
fig =squarify.plot(sizes=df_multi_props['count'],value=df_multi_props['var1'])
plt.axis('off')
plt.title('Treemap')
plt.show()

from statsmodels.graphics.mosaicplot import mosaic
labelizer=["","",""]
fig = mosaic(df_multi_props, ['var1', 'count'],title='Mosaic Chart')
plt.show()
import plotly.express as px
fig=px.parallel_categories(df_sets,dimensions=['var1', 'var2', 'var3'])
fig.show()
ㄴㅇㄹdf_dense_scatter=pd.read_csv('df_dense_scatter.csv')
df_paired = pd.read_csv('df_paired.csv')
df_scatter_xy=pd.read_csv('df_scatter_xy.csv')
df_scatter_xyz=pd.read_csv('df_scatter_xyz.csv')
fig = plt.figure(figsize=(8,8))
title=['Scatterplot','Bubble Chart','Paired Scatterplot','Slopegraph']

for i in range(1,5):
 globals()['area{}'.format(i)]=fig.add_subplot(2,2,i)
 globals()['area{}'.format(i)].set_title(title[i-1])
a1 = sns.scatterplot(x = "x", y = "y", data = df_scatter_xy,ax=area1)
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2 = sns.scatterplot(data=df_scatter_xyz,x = "x", y = "y",size='z',ax=area2)
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
pt = df_paired.pivot('group','x','y')
pt = pt.T.values.tolist()
a3 = sns.scatterplot(x = pt[0], y = pt[1],ax=area3)
a3.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a4 = sns.pointplot(x="x", y="y", hue="group",data=df_paired, dodge=True,ax=area4)
a4.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
plt.show()
df_one_line=pd.read_csv('df_one_line.csv')
fig = plt.figure(figsize=(15,5))
title=['Line Graph','Connected Scatter Plot','Smooth Line Graph']
area1=fig.add_subplot(1,3,1)
sns.lineplot(x=df_one_line.x, y=df_one_line.y,ax=area1)
area1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None,title=title[0])
df_connected_scatter = pd.read_csv("df_connected_scatter.csv")
area2=fig.add_subplot(1,3,2)
plt.plot(df_connected_scatter.x,df_connected_scatter.y)
plt.title(title[1])
area3=fig.add_subplot(1,3,3)
sns.regplot(x=df_dense_scatter_sample.x, y=df_dense_scatter_sample.y,ci=95,ax=area3)
area3.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None,title=title[2])

plt.show()
fig = plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.errorbar(data=df_uncertain,x='x',y='y',xerr='dx')
plt.title('error bars')
plt.subplot(1,3,2)
plt.errorbar(data=df_uncertain,x='x',y='y',yerr='dy')
plt.title('error bars')
plt.subplot(1,3,3)
plt.errorbar(data=df_uncertain,x='x',y='y',xerr='dx',yerr='dy')
plt.title('2D error bars')
plt.show()
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
from matplotlib.collections import PatchCollection
import matplotlib.ticker as ticker
import numpy as np
from scipy.stats import lognorm

# Parameters
sample = 20
n_bins = 7
args = {'s': 0.2, 'scale': 11.4}
data = lognorm.rvs(size=10000, **args)
pdf = lognorm.pdf

# Evenly sample the CDF and do the inverse transformation (quantile
function) to have x.
# probability of drawing a value less than x (i.e. P(X ( x)) and the
corresponding
# value of x to achieve that probability on the underlying distribution
p_less_than_x = np.linspace(1 / sample / 2, 1 - (1 / sample / 2), sample)
x = np.percentile(data, p_less_than_x * 100)  # Inverce CDF (ppf)

# Create bins
hist = np.histogram(x, bins=n_bins)
bins, edges = hist
radius = (edges[1] - edges[0]) / 2

# Plot
fig, ax = plt.subplots()

# Real PDF
x_ = np.linspace(0, 30, 100)
ax.plot(x_, pdf(x_, **args), 'r-', lw=5, alpha=0.6, label='lognorm pdf')
ax.set_ylabel('PDF')
ax.set_xlabel('Value')

# Dotplot
ax2 = ax.twinx()
patches = []
max_y = 0
for i in range(n_bins):
   x_bin = (edges[i + 1] + edges[i]) / 2
   y_bins = [(i + 1) * (radius * 2) for i in range(bins[i])]

   max_y = max(y_bins) if max(y_bins) ) max_y else max_y

   for _, y_bin in enumerate(y_bins):
      circle = Circle((x_bin, y_bin), radius)
      patches.append(circle)

p = PatchCollection(patches, alpha=0.4)
ax2.add_collection(p)

# Axis tweek
y_scale = (max_y + radius) / max(pdf(x, **args))
#ticks_y = ticker.FuncFormatter(lambda x, pos: '{0:g}'.format(x / y_scale))
#ax2.yaxis.set_major_formatter(ticks_y)
ax2.set_yticklabels([])
ax2.set_xlim([min(x) - radius, max(x) + radius])
ax2.set_ylim([0, max_y + radius])
ax2.set_aspect(1)
ax2.set_title('Quantile Dot Plot')
plt.show()
df_dense_scatter_sample=pd.read_csv('df_dense_scatter_sample.csv')
fig = plt.figure(figsize=(15,15))
area1=fig.add_subplot(1,1,1)
area1.set_title('Confidence Band')
a1 = sns.regplot(x=df_dense_scatter_sample.x,
y=df_dense_scatter_sample.y,ci=95,ax=area1)
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
plt.show()
```



Do you have any questions?

Thank you!