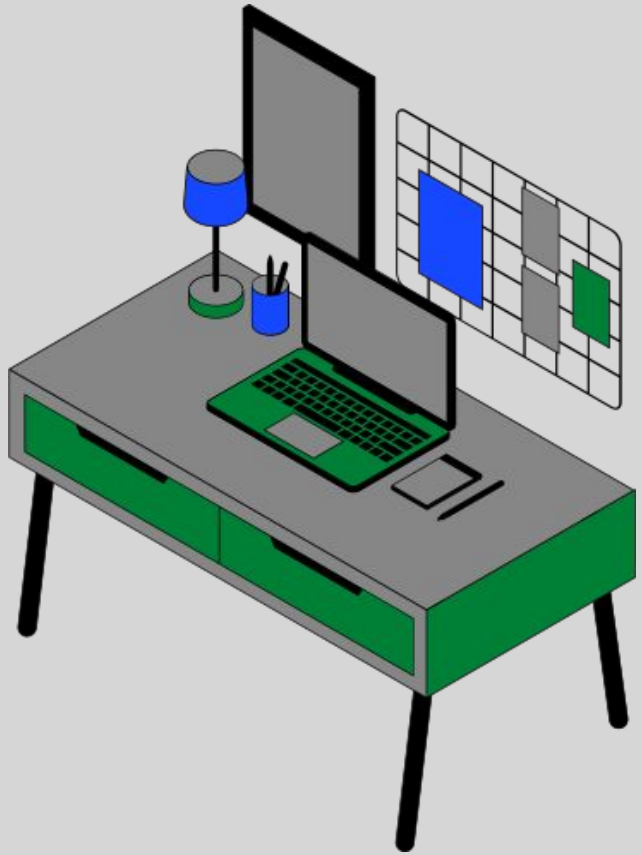
데이터 시각화 교과서

Chapter 13. 독립 변수의 시계열 데이터와 함수 시각화



하나의 변수가 시간 데이터일 때

-데이터 포인트들 사이에 순서 개념 생김

-시간의 흐름에 따라 포인트들을 배열하고, 데이터 포인트마다 이전/이후 값을 정의할 수 있음

-시간적 순서는 보통 선그래프로 나타냄

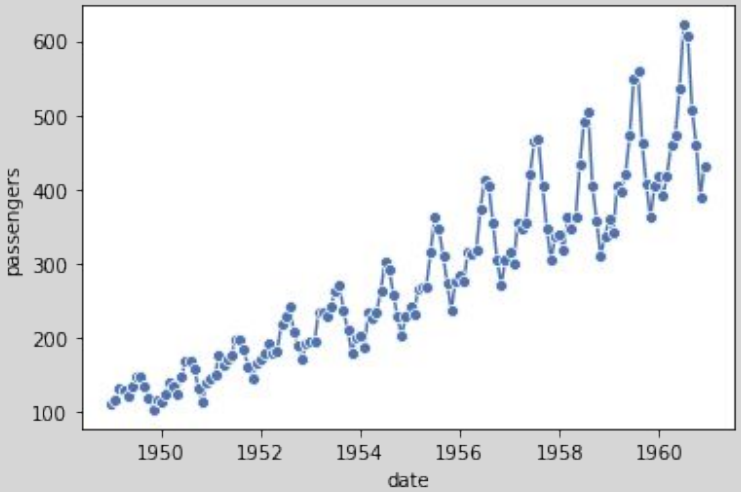
단일 시계열 데이터
-산점도와의 차이

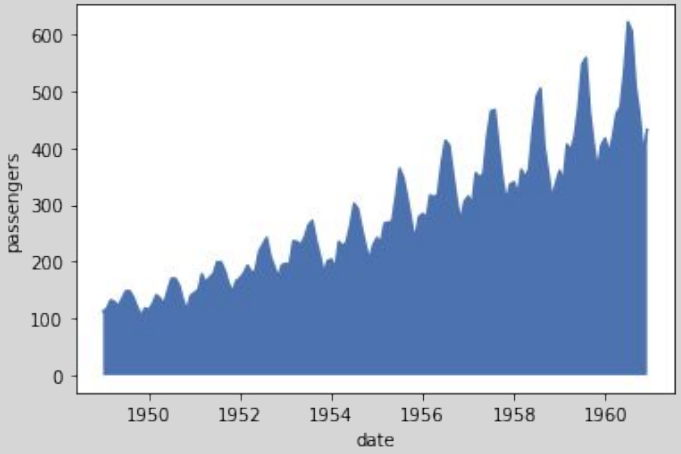
-점들이 x축을 따라 고르게 찍혀있음

-점들 사이에 순서가 있음

-각 점의 왼편과 오른편 양 옆에는 이웃한 점이 있음

단일 시계열 데이터





-선은 관측 값을 나타내지 않지만 선을 그으면 데이터의 흐름이 두드러짐

-점을 찍지 않은 선 그래프: 보기에 더 간결

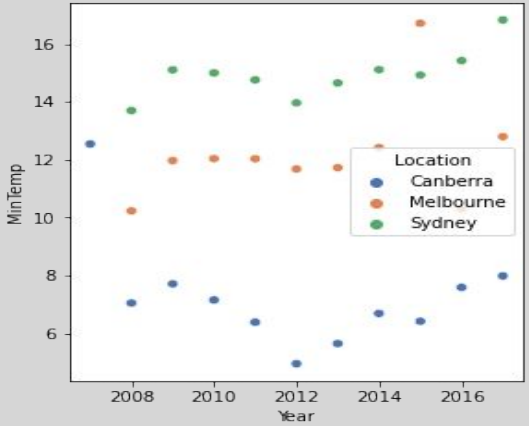
-시계열이 촘촘할수록 각 관측 값을 점으로 나타낼 필요성 감소

점들을 이어서 그은 곡선 아랫부분에 색을 채우기

-곡선 윗쪽과 아래쪽이 분리되어 데이터의 중요한 흐름 돋보임

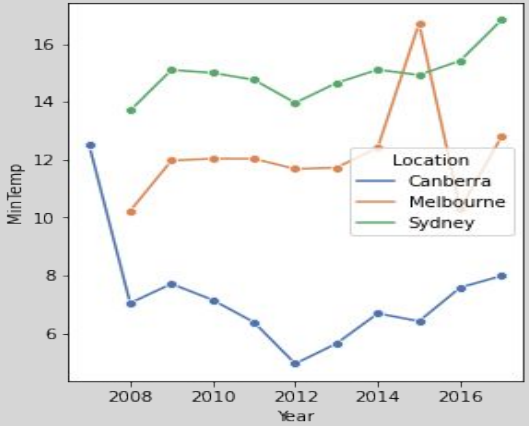
-y축의 시작점이 0일때만 사용해야 (각 시점이 가리키는 높이가 해당시점의 데이터값을 정확하게 반영하기 때문)

다중 시계열 데이터와 용량-반응 곡선



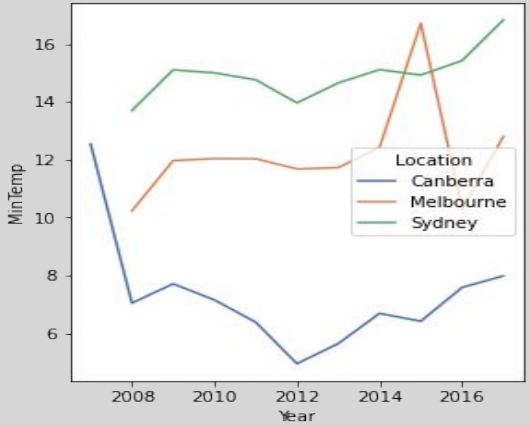
-도표 하나에 여러 시계열 그래프를 함께 담을 때, 산점도는 피하자

-개별 시계열 데이터가 서로 부딪치기 때문



-점들을 선으로 이으면, 문제가 해결

-범례를 따로 만들지 않고, 선 옆에 바로 레이블을 붙이면 도표가 간결해짐

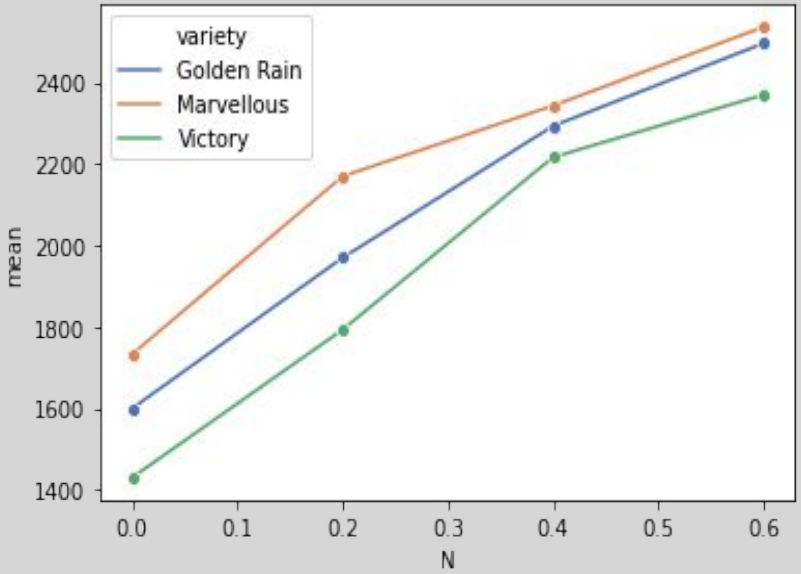


-점을 생략하면, 보기에 더 간결해짐

용량-반응 곡선

시계열 뿐만아니라, 용량-반응 곡선에도

선 그래프 활용 가능



용량-반응 곡선: 실험에서 숫자형 매개변수 일부를 바꾸는 것이

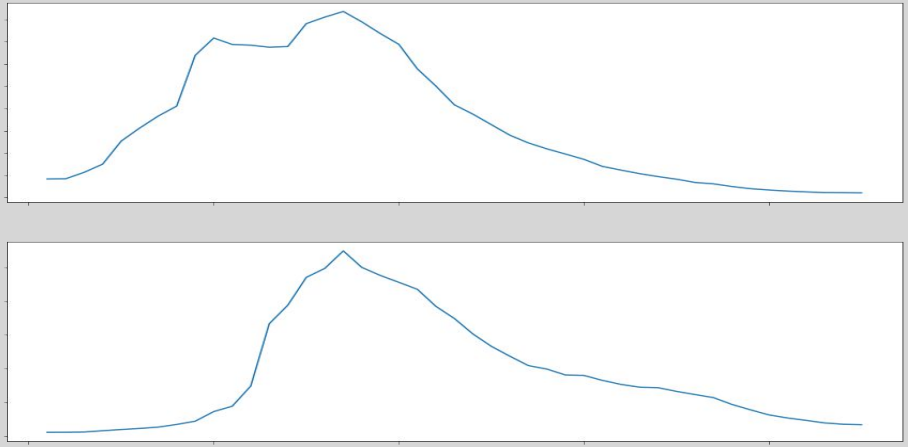
실험결과에 어떤 영향을 미치는지 알아보는 도표

Ex. 거름을 준 후 귀리 품종 별 평균 수확량

3가지 귀리 품종의 용량-반응곡선은 비슷한 형태이지만,

거름을 아예 사용하지 않은 시작점의 각 값이 품종마다 다름을 강조

2개 이상의 반응 변수를 포함한 시계열 데이터



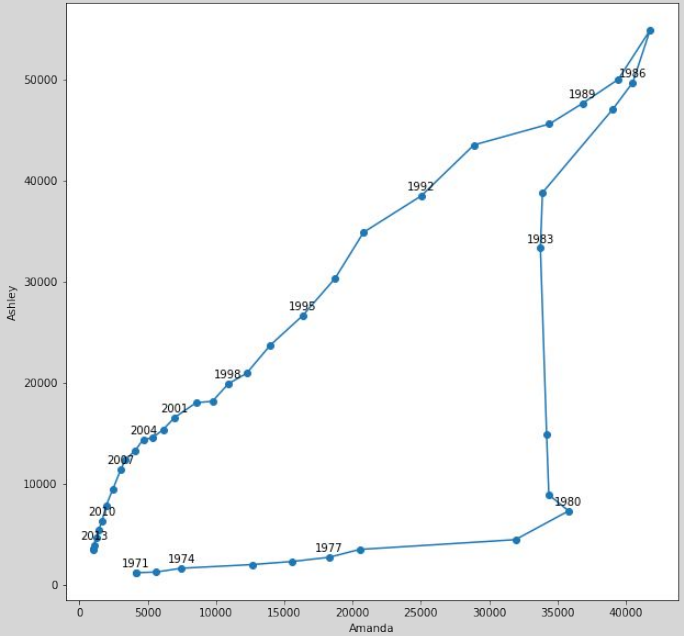
별개의 선 그래프 두개를 위아래로 나란히 그림

=>그래프 각각을 해석하기엔 쉽지만,

두 변수가 따로 있어서 두 데이터를 비교하기엔 어려움

연결 산점도 (위상도)

두 변수를 한 도표에 그린 다음 이웃한 점들을 연결해서 완성



왼쪽 아래에서 오른쪽 위방향: 양의 상관관계

왼쪽 위에서 오른쪽 아래방향: 음의 상관관계

두 변수에 순환관계가 있다면, 연결 산점도는 동그라미 모양을 띰

연결산점도 그릴때

=>데이터의 방향과 시간 범위를 모두 표시해야

연결 산점도

Vs

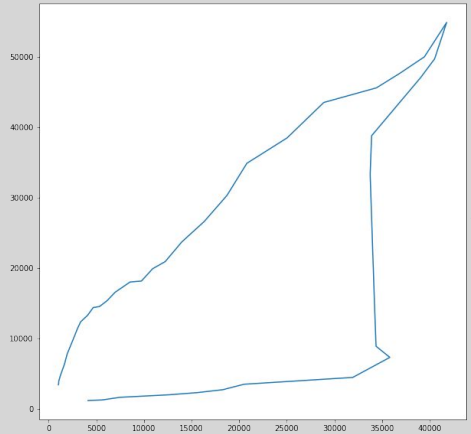
선그래프 2개

선 그래프에서 발견하기 어려운 패턴을 찾아낼 수 있음(ex. 순환관계)

-소용돌이를 통해 두 변수 간 관계를 파악할 수 있음

-순서와 방향을 헷갈리기 쉽고, 상관관계를 파악하기 어려움

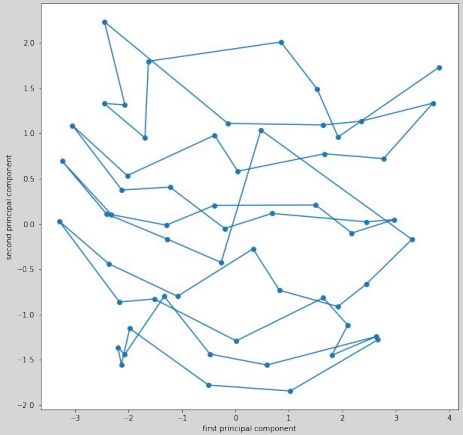
-주목도가 높아서 보는 이의 관심을 끌 수 있음

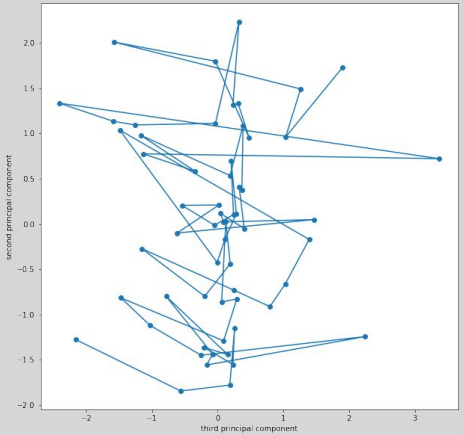


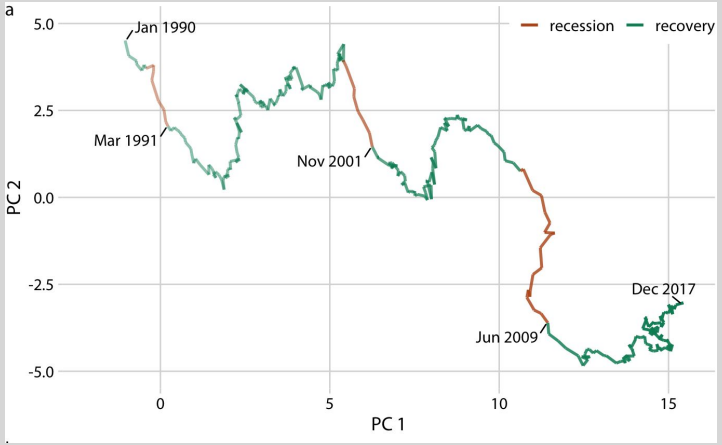
날짜 표시/색의 음영
등이 없어서 모호함

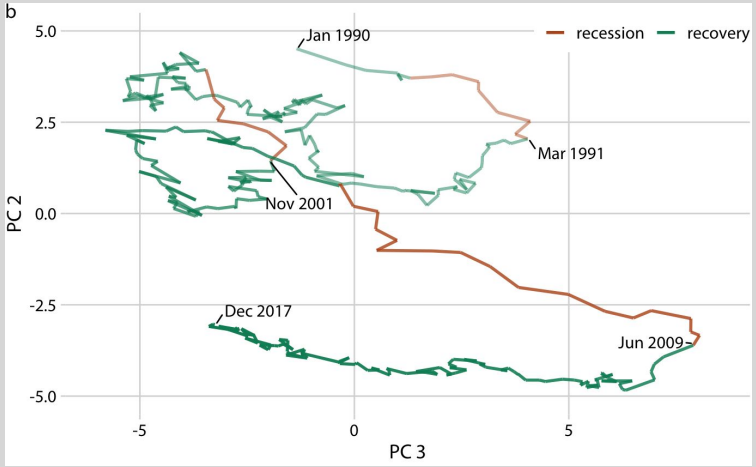
연결 산점도를 활용한 3차원 이상의 데이터셋 시각화

- 차원 축소 후 그 공간에 연결 산점도를 그려 더 높은 차원의 데이터셋을 시각화할 수 있음









거시경제지표 100개를 매월 관찰한 데이터베이스

-불황기와 회복기를 색으로 나타냄(불황: red, 호황: green)

-위 그래프: 왼쪽에서 오른쪽으로 시간이 흐르는 일반적인 선 그래프로 보임

=> 주성분 분석 시 첫번째 성분이 시스템의 전체적인 규모를 나타내기 때문

-불황: 주성분 2 감소와 관련이 있음

-선이 시계방향으로 소용돌이: 불황기와 회복기가 반복되며 경제가 순환하는 특성을 강조

```
#my code
from matplotlib import pyplot as plt
import seaborn as sns
import random
import pandas as pd
from pandas import Series, DataFrame
import numpy as np
df =
pd.read_csv('https://github.com/mwaskom/seaborn-data/raw/master/fl
ights.csv')
month2int = {'January': 1,'February': 2,'March': 3,'April':
4,'May': 5,'June': 6,'July': 7,'August': 8,'September':
9,'October': 10,'November': 11,'December': 12}
df['month'] = df['month'].map(month2int)
df['day'] = 1
df['date'] = pd.to_datetime(df[['year', 'month', 'day']])
df['1y'] = df['passengers'].rolling(window=12).mean()
sns.set_palette("deep")
sns.scatterplot(x='date',y='passengers',data=df)
plt.show()
sns.lineplot(x='date',y='passengers',data=df,marker='o')
plt.show()
x=df['date']
y=df['passengers']
sns.lineplot(x,y)
plt.fill_between(x,y)
plt.show()
```

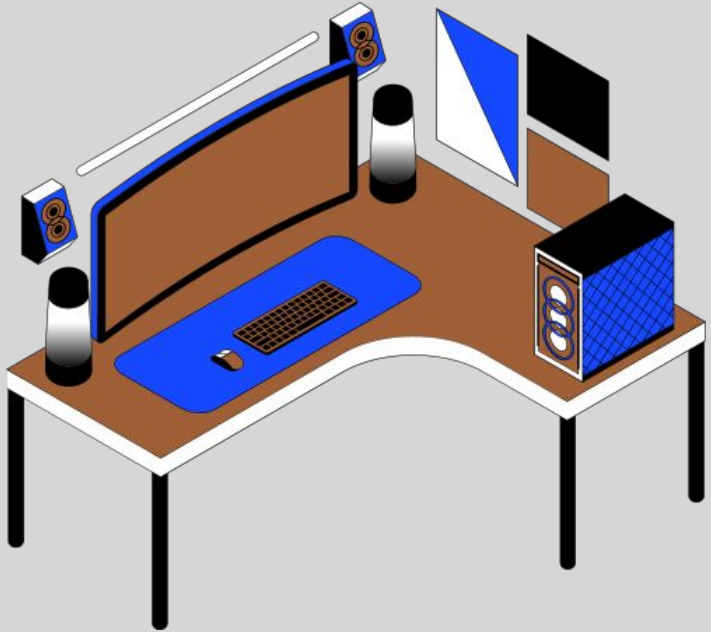
```
from seaborn import palettes
weather_df = pd.read_csv('weatherAUS.csv', usecols=['Date',
'Location', 'MinTemp', 'MaxTemp'])
location = weather_df.query("Location == 'Sydney'|Location ==
'Canberra'|Location == 'Melbourne'")
location['Date'] = location['Date'].astype('datetime64')
location['Year'] = location['Date'].dt.year
location = location.groupby(['Year','Location']).mean()
fig = plt.figure(figsize=(15,5))
for i in range(1,4):
 globals()['area{}'.format(i)]=fig.add_subplot(1,3,i)

a1 =
sns.scatterplot(x='Year',y='MinTemp',hue='Location',data=locati
on,ax=area1)
a2 =
sns.lineplot(x='Year',y='MinTemp',hue='Location',data=location,
marker='o',ax=area2)
a3 =
sns.lineplot(x='Year',y='MinTemp',hue='Location',data=location,
ax=area3)
plt.show()
```

```
#my code
df =
pd.read_csv("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/5_OneCatSevNumOrdered.csv")
# filter data
df = df.loc[(df.name=="Ashley") | (df.name=="Amanda")]
df = df.loc[(df.sex=="F") & (df.year>1970)]
df = pd.pivot_table(df, values='n', index=['year'], columns=['name'])
# set the figure size
plt.figure(figsize=(10, 10))
# plot the connected scatterplot
plt.plot(df.Amanda, df.Ashley, '-', marker='o')
# add annotations in every 3 data points with a loop
for line in range(0, df.shape[0], 3):
    plt.annotate(
         df.index[line],
         (df.Amanda.iloc[line], df.Ashley.iloc[line]+300 ) ,
         va='bottom',
         ha='center'
    )
# x axis label
plt.xlabel('Amanda')
# y axis label
plt.ylabel('Ashley')
# show the graph
plt.show()
```

```
fig = plt.figure(figsize=(15,10))
for i in range(1,3):
 globals()['area{}'.format(i)]=fig.add_subplot(2,1,i)

a1 = sns.lineplot(x='year',y='Amanda',data=df,ax=area1)
a1.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
a2 = sns.lineplot(x='year',y='Ashley',data=df,ax=area2)
a2.set(xticklabels=[],yticklabels=[],xlabel=None,ylabel=None)
plt.show()
#30차원을 3차원으로 줄이기(PCA)
X = df.values
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
from sklearn.decomposition import PCA
pca_3 = PCA(n_components= 3, random_state=2020)
pca_3.fit(X_scaled)
X_pca_3 = pca_3.transform(X_scaled)
plt.figure(figsize = (10,7))
sns.scatterplot(x=X_pca_3[:,0], y = X_pca_3[:,1])
plt.show()
```



Do you have any questions?

Thank you!