1. 데이터시각화 개요

* 시각화

- 인간은 눈 망막 자극이 뇌에 전달될 때 정보의 의미를 파악할 수 있게 진화한 동물이다.

- 인간은 감각 중의 대부분을 **시각**에 의존하는데, 일반적으로 시각이 77%, 청각이 13%, 후각이 7%, 나머지 3%는 촉각과 미각이 나눠가지는 것으로 알려져 있다.

- 인간이 인간답게 활동하게 하는 가장 중요한 감각 정보기관이 **시각**이다.

* 데이터시각화

- 정의 : 많은 양의 데이터를 시각적 요소를 활용하여 요약하여 표현하게 함.

**(눈으로 보는 데이터)**

- 효과 : 수집된 데이터에 대한 **분석 결과를 쉽게 이해**할 수 있도록 여러 시각화 도구를 통해 **효과적으로 정보를 전달하여 의사결정의 기회를 부여**함

- 사람의 눈은 수백 줄의 텍스트 데이터를 읽거나 기초 통계 수치를 계산하는 방법으로는 데이터를 제대로 분석할 수 없기 때문에, **데이터의 숨겨진 패턴을 파악하기 위해 데이터시각화를 사용.**

(데이터의 숨겨진 패턴을 파악하는 것 : 데이터 마이닝)

- 과정

1. 필요한 **데이터를 수집**한 후, 일정 장소에 저장

2. 저장된 데이터를 **전처리** 한 후 분석

3. 다양한 시각화 방법으로 **분석된 데이터를 표현**하여 목적에 부합되도록 활용

( 데이터 분석은 ‘데이터마이닝’으로 수행 )

- 주요 방법

: 데이터시각화의 방법은 목적과 의도, 데이터의 유형에 따라 크게 5가지 형태로 구분

|  |  |  |
| --- | --- | --- |
| 구분 | 주요 시각화 방법 | 의미 |
| 시간 시각화 | 막대 그래프, 누적 막대 그래프, 점 그래프 | 시간에 변화에 따른 데이터를 표현 |
| 관계 시각화 | 히스토그램, 버블차트, 박스 플롯 | 데이터들 사이에 존재하는 관계를 표현 |
| 비교 시각화 | 히트맵, 스타 차트, 평행 좌표계, 다차원 척도법 | 데이터의 값들을 비교하여 표현 |
| 분포 시각화 | 파이 차트, 도우넛 차트, 트리맵, 누적 연속 그래프 | 전체의 관점에서 각 부분 간의 관계 표현 |
| 공간 시각화 | 지도 맵핑 | 지도상에 해당하는 정보를 표현 |

- 시각화 라이브러리

내장 라이브러리 : pandas에 내장된 기본 그래프 라이브러리로 별도의 라이브러리 임포트 없이 사용 가능

Matplotlib : 판다스에서 가장 많이 쓰는 라이브러리로, 데이터 프레임을 시각화 할 때도 내부적으로 사용

Seaborn : matplotlib을 기반으로 다양한 색 테마, 차트 기능을 추가하는 라이브러리로, matplotlib에 의존성을 가지고 있음

* 시각화 단계
  1. 시각화 라이브러리 불러오기 : import matplotlib.pyplot as plt
  2. X, Y축에 어떤 데이터를 표기할 지 결정
  3. Plot()함수에 데이터 입력 : plt.plot([x, y])
  4. 그래프 보여주기 : plt.show()
* 시각화 옵션

1. 제목 : plt.title()
2. 범례 : plt.legend()
3. 색상
4. X, Y축 이름 : plt.xlabel(), plt.ylabel()
5. 그래프 선 모양
6. 그림 범위 지정
7. Matplotlib marker

* 그래프 시각화

- 기본 함수는 plot() 사용

|  |  |
| --- | --- |
| line : 선그래프 | kde : 커널 밀도 그래프 |
| bar : 수직 막대그래프 | area : 면적 그래프 |
| barh : 수평 막대그래프 | pie : 파이 그래프 |
| hist : 히스토그램 | scatter : 산점도 그래프 |
| box : 박스플롯 | hexbin : 고밀도 산점도 그래프 |

* 엔스콤 4분할 그래프

- 데이터를 수치적으로만 확인할 때 발생할 수 있는 함정을 보여주기 위해 만든 그래프

- 데이터 집합은 4개의 그룹으로 구성되어 있음. 4개의 데이터 그룹은 각각 평균, 분산과 같은 수치나 상관관계 회귀선이 모두 같다는 특징이 있다. 결국 4개의 데이터 그룹의 데이터는 모두 같다고 착각할 수 있다.

- 데이터시각화를 수행하면 각 데이터가 다른 분포를 띈다는 것을 알 수 있음.

* Pandas

- 고수준의 자료구조와 파이썬을 통한 빠르고 쉬운 데이터 분석을 위한 여러가지 도구들을 포함

- numpy를 기반으로 개발되어 numpy를 사용하는 애플리케이션에서 쉽게 사용 가능

- 행렬/선형대수/통계 : numpy ( : 수치를 다루기 위한 패키지)

- 자료구조 : pandas ( : series, data frame)

- 시각화 : Matpotlib, Seaborn

* Numpy

- 과학 계산을 위한 파이썬의 외부 라이브러리

- **다차원 배열**을 처리하는데 필요한 여러 기능을 제공

- 딥러닝을 구현할 때 배열이나 행렬 계산이 많으므로, 계산을 보다 간단하고 빠르게 해결하기 위해 사용

* Seaborn

- matplotlib을 기반으로 하는 파이썬 시각화 라이브러리

- 조금 더 동적인 시각화 -> 색 표현력이 우수

- 시각적 우수 ->

- 간결한 구문을 제공

- 판다스 데이터프레임에 최적화

- 데이터프레임을 집계해서 쉽게 차트로 요약 가능

1. 파이썬 설치

* 파이썬의 코딩 구조

- 기존의 구조적 프로그래밍 언어에서는 중괄호를 이용하여 반복문의 시작과 끝을 알려주어야 했지만,

파이썬에서는 기호 대신에 들여쓰기를 사용

=> 코드의 가독성을 높이고, 코드 내에서 기호 사용을 줄임

* Jupyter notebook

- 오픈 소스로써 파이썬 코드에 대한 수식, 코멘트 등을 꾸며주는 웹 애플리케이션

- 웹 브라우저에서 수행 가능한 대화형 파이썬 환경

- 코드 및 수식과 시각화 된 데이터를 한꺼번에 저장하고, 공유할 수 있도록 해줌

- 수행한 코드들을 인라인 형식으로 보여줌.

* 통합 개발 도구(IDE)를 사용한 python 실행

- ‘통합개발환경’이란 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어

1. Pandas

* 판다스의 **적합 분야**

텍스트이(가) 표시된 사진

자동 생성된 설명

* 판다스의 중요한 **데이터 구조**

텍스트이(가) 표시된 사진

자동 생성된 설명

* 판다스의 **주요 기능**

텍스트이(가) 표시된 사진

자동 생성된 설명

* Series

- **동일한 데이터 타입을 저장하기 위한 일련의 객체를 담을 수 있는 1차원 배열 같은 자료 구조**

- 데이터 색인을 지정하지 않으면 0부터 시작하는 정수의 값으로 자동 설정

- 인덱스를 이용하여 순서대로 배열, 인덱스의 이름으로 값을 읽음

Series( data = None, index = None, dtype = None, copy = False, fastpath = False )

- 콜론(:)을 사용하면 연속적인 데이터를 추출

숫자형 : 시작 ~ 끝-1, 문자형 : 시작 ~ 끝

- 콤마(,)를 사용하면 서로 분리되어 있는 데이터를 추출

* DataFrame

- **2차원 형태의 표(행/열) 구조를 가지는 자료 구조**

- 행과 열에 대한 인덱스를 가지고 순서대로 배열

- 열 1개가 하나의 Series 구조가 됨

- DataFrame은 **행 구분자를 의미하는 색인(Index)**과 **열 구분자를 의미하는 라벨 및 자료** 등으로 구성

DataFrame( data=None, index=None, columns=None, dtype=None, copy=False)

` 테이블이(가) 표시된 사진

자동 생성된 설명

- 데이터 프레임을 생성하기 위해 Index, 컬럼, 데이터에 대한 변수들에 값을 설정

Index = [‘A’, ‘B’, ‘C’, ‘D’, ‘E]

Columns = [range(1, 6)]

Values = list(10 \* data for data in range(1, 26))

Df = pd.DataFrame(np.reshape(Values, (5, 5)), index = Index, columns=Columns)

- **iloc**함수 : **행 인덱스 번호를 기준**으로 **행을 추출**하는 함수

- **loc**함수 : **라벨을 이용**하여 **행을 추출**하는 함수

-> 대괄호를 2개 사용하면 DataFrame을 반환

-> [[행목록], [열목록]] 형식으로 데이터를 읽음

- **np.random.choice()**함수 : 랜덤 추출

- 콜론(:)을 사용하면 연속적인 데이터를 추출

숫자형 : 시작 ~ 끝-1, 문자형 : 시작 ~ 끝

- 콤마(,)를 사용하면 서로 분리되어 있는 데이터를 추출

- Boolean 값으로 데이터를 읽어올 수 있음 : True인 행만 불러옴(가로)

Ex) Result = df.loc[df[‘A’]<=100] # ‘A’값이 100이하인 항목들만 조회(가로)

- **all**()함수 : and연산자와 동일한 개념 (모두 True여야 True)

- **any**()함수 : or연산자와 동일한 개념 (모두 False여야 False)

- 람다함수 : lambda df : df[‘A’]>130 # ‘A’의 값이 130인 것들만 True

Ex) ‘경주’의 값이 150 이하인 행에 대하여, ‘경주’와 ‘광주’의 컬럼의 값을 0으로 변경

=> df.loc[df[‘경주’] <=150, [‘경주’, ‘광주’]] = 0

* 함수 적용과 매핑 (apply 함수)

- **apply()함수**는 **사용자가 직접 작성한 함수를 한번에 데이터 프레임의 각 행과 열에 적용하여 실행**할 수 있게 해주는 메소드

- **함수를 브로드캐스팅 해야하는 경우**에 사용

- 반복문 보다는 속도가 빠르기 때문에 대용량의 데이터 처리에 유용

텍스트이(가) 표시된 사진

자동 생성된 설명

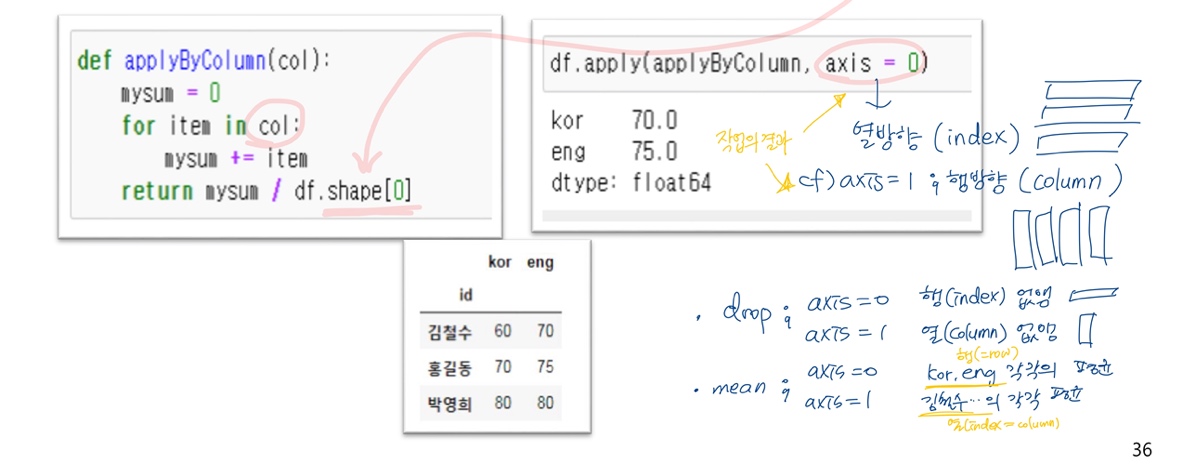
Ex) 어떤 수의 n배를 곱해주는 함수 정의

def multi(x, n):

return n\*x

ex = df[‘kor’].apply(multi, n=2) # apply(함수, 인자)

ex) 컬럼 방향으로 평균값을 구해주는 함수



Ex) 행 방향으로 평균값을 구해주는 함수

테이블이(가) 표시된 사진

자동 생성된 설명

* 데이터 병합

- 관계형 데이터 베이스에는 행을 합치거나, 양쪽 테이블의 공통된 컬럼을 이용하여 join 연산을 수행

- **merge()**함수 : pandas.merge는 **하나 이상의 키를 기준**으로 **DataFrame의 column(열)을 합치**는 기능

키의 이름을 지정하지 않으면 동일한 이름의 컬럼을 이용하여 merge

pd.merge(left, right, how=’inner’, on=None, left\_on = None, right\_one’None, left\_index=False,

right\_index=False, sort=True, suffixes(‘\_x’, ‘\_y’), copy=True, indicator=False)

테이블이(가) 표시된 사진

자동 생성된 설명

Ex) pd.merge(df1, df2, on=’name’)

Ex) pd.merge(df1, df2, how=’outer’)

-> 양쪽 DataFrame에 **동일 이름의 컬럼이 존재하지 않는 경우**에는 **on 매개변수 사용 불가**

=> **left\_on, right\_on 매개변수 사용**

Ex) pd.merge(df1, df2, left\_on = ‘column이름’, right\_on=’column이름’)

-> 여러 개의 키나 column을 병합할 때는 컬럼 이름이 들어간 리스트 적으면 됨

-> **컬럼 2개의 이름이 겹치는 경우** : **suffixes를 이용**하면 **이름이 겹치는 컬럼에 대하여 접미사 지정**

Ex) pd.merge(df1, df2, on=’key1’, suffixes = (‘\_left’, ‘\_right’))

-> **index로 이동**할 때 : **set\_index()함수** 이용

Ex) df1.set\_index(‘name’)

-> index를 이용하여 병합할 때 : left\_index, right\_index 매개변수 사용

Ex) pd.merge(newdf1, newdf2, left\_index = True, right\_index = True, indicator = True)

# indicator : 두 데이터프레임의 행이 같이 합쳐졌으면 both, 아니면 Only라고 출력

* 데이터 추출 방법

- read\_csv() : DataFrame으로 데이터 불러옴

- shape : df.shape[0] # 행, df.shape[1] # 열

- columns : 컬럼 읽어오기

- **dtypes** : **각 컬럼들의 데이터 타입 정보** 확인

- **Info()** : **DataFrame이라는 데이터 구조**를 알려줌

- 대괄호를 사용 => DataFrame에서 특정 컬럼에 대한 정보를 별도로 추출 ( 반환 : **Sereis 타입** )

- **2개**의 대괄호 사용 => 2개 이상의 컬럼 추출 ( 반환 : **DataFrame 타입** )

- head() : 상위 5개 보여줌 ( 반환 : **DataFrame** )

- tail() : 하위 5개 보여줌 ( 반환 : **DataFrame** )

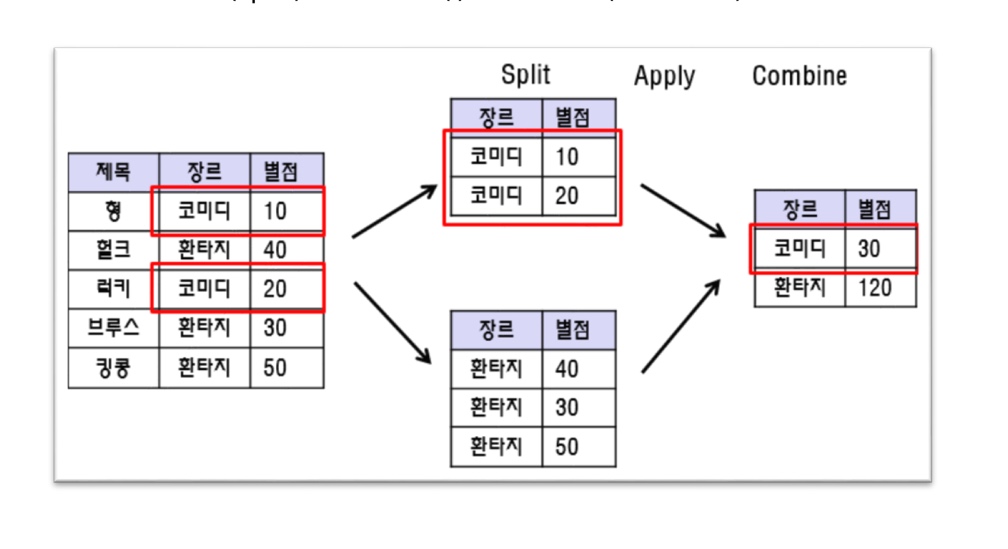
- **loc**속성 : **라벨의 이름**으로 행 또는 열의 위치를 찾기 ( 반환 : **Series 타입** )

- **iloc**속성 : **행 인덱스 번호**를 기준으로 행을 추출

* GroupBy 메카닉

- **그룹 연산** : **데이터를 집계하거나, 변환하는 등의 작업을 한 번에 처리할 수 있도록** 하는 기능

- 데이터를 **분할(split)**하고 **반영(apply)**하고, 결합(combine)시키는 과정을 거침



- 그룹 연산은 **데이터 집합을 분류하고 그룹별로 집계나 어떤 변형 같은 함수를 적용**

그룹 연산을 이용하면 큰 용량의 데이터를 손쉽게 처리

- groupby()함수를 이용하여 구한 객체를 그룹 오브젝트라고 한다.

테이블이(가) 표시된 사진

자동 생성된 설명

- unique()함수 : 중복되는 값을 제거하고 **유일 값**만 저장 ( 반환 : **numpy 객체 – class ‘numpy.ndarray’** )

Ex) 컬럼 ‘marriage’에 대하여 그룹핑을 수행한 다음, ‘code\_religion’컬럼에 대한 평균을 구함

df.groupby(‘marriage’)[‘code\_religion’].mean()

ex) 컬럼 ‘marriage’, ‘birth’을 이용하여 그룹화를 수행 후, 컬럼 ‘code\_religion’, ‘income’에 대한 평균

df.groupby([‘marriage’, ‘birth’])[[‘code\_religion’, ‘income’]].mean()

# 반환 타입 : DataFrameGroupBy 객체

1. Matplotlib

* Matplotlib

- 파이썬에서 사용 가능한 그래프를 그려주는 라이브러리

- line plot, bar chart, pie chart, histogram, box plot, scatter plot 등 다양한 차트와 플롯 스타일 지원

- numpy 및 pandas가 제공하는 자료와 잘 연동

- 옵션

테이블이(가) 표시된 사진

자동 생성된 설명

* 변수 개수 및 형태별 그래프 종류

- **그래프란**? 서로 연관성이 있는 1개 또는 그 이상의 양에 대한 상대 값을 도형으로 나타낸 것

- 순서 : 데이터의 구조 파악 -> 전처리 수행 -> 시각화 수행  
- 종류

테이블이(가) 표시된 사진

자동 생성된 설명

* 꺾은 선 그래프 ( **plot** )

- **연속적으로 변화하는 데이터**를 보고자 할 때

- 일반적으로 꺾은 선 그래프는 **시간에 따른 데이터의 연속적인 변화량**을 관찰할 때 사용

- 수량을 점으로 우선 표시한 후 해당 점들을 선분으로 이어서 그리기 : 증가와 감소의 상태 쉽게 찾을 수 있음

Ex) 단일 축 꺾은 선 그래프 – 변수 한 개

텍스트이(가) 표시된 사진

자동 생성된 설명

Ex) 다변량 꺾은 선 그래프

텍스트이(가) 표시된 사진

자동 생성된 설명

+ 이중축 꺾은 선 그래프

- 특정 항목의 값이 다른 항목의 값과 현저히 차이가 나게 되면 다른 데이터들은 상대적으로 크거나 작아지기 때문에, 비교 판단하기 쉽게 **보조 y축**을 만들어서 이중 축을 만듦.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 산점도 그래프 ( **scatter** )

- **2개의 연속형 변수 간의 관계**를 보기 위해 직교 좌표의 x, y축에 관측점을 찍어서 만든 그래프

- 산포도에 표시되는 각 점들은 자료의 관측 값을 의미



****

* 막대 그래프

- 여러 가지 통계 데이터나 양을 막대 모양의 길이로 나타낸 그래프

- **수량의 많고 적음을 비교**하고자 할 때 사용 : 크고 작음을 한 눈에 이해할 수 있기 때문에 많이 사용됨

- 시간의 흐름에 따라 변화를 표현하는 것이라면 꺾은 선 그래프를 사용하는 것이 좋음



(누적 막대 그래프 – stacked 옵션 True)



Ex) Y값의 그래프 상 범위 찍을 때

텍스트이(가) 표시된 사진

자동 생성된 설명

* 파이 그래프

- **전체에 대한 각 부분의 비율**을 부채꼴 모양으로 나타낸 그래프

- 전체를 100%으로 놓고, 그 중에서 무엇이 얼마나 많은 비율을 차지하고 있는지 확인할 수 있는 도표

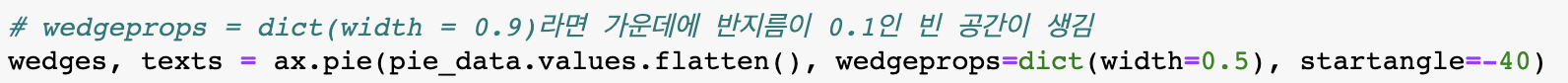
- 각 **부채꼴의 중심각이 전체에서 차지하는 비율**을 나타냄 : 비율을 한 눈에 볼 수 있다 (장점)

- **도넛 형태**로 표기되기도 함

텍스트이(가) 표시된 사진

자동 생성된 설명

( 도넛 모양의 파이 그래프 )



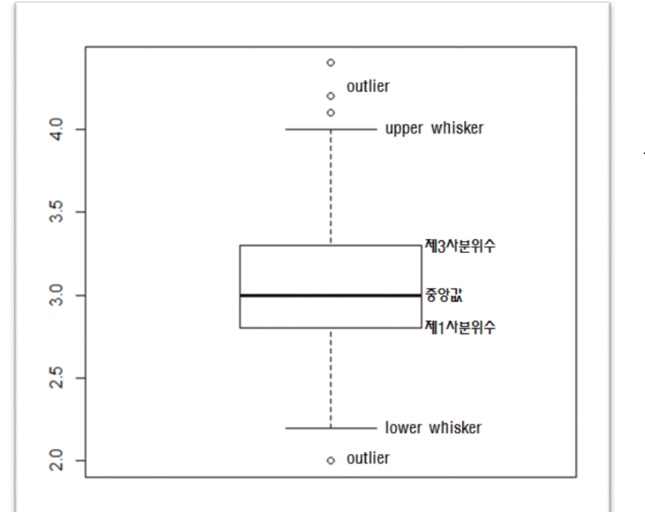
* 상자 수염 그래프

- **자료의 특성을 요약**하는 그래프

- 상자 그림의 각 요소

: Q1에서 Q3 사이인 사분위간 범위(IQR)로 몸통을 구성하고, 근접 값들로 꼬리를 구성

테이블이(가) 표시된 사진

자동 생성된 설명 

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 히스토그램

- **특정 데이터의 빈도수**를 막대 모양으로 표시한 그래프

- 도수 분포표와 히스토그램은 가장 많이 사용되는 통계 분석 도구로써, **데이터의 특성/분포를 파악**하는 역할

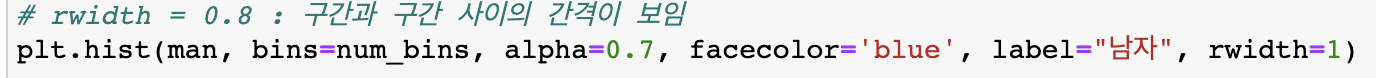
- 가로 축에는 계급, 세로 축에는 해당 도수 또는 비율을 지정

(계급 : 변수의 구간, 서로 겹치지 x, 계급끼리 서로 붙어 있어야 함)

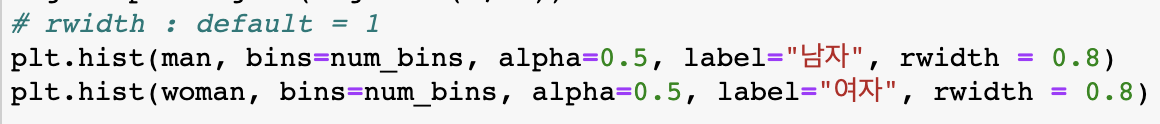
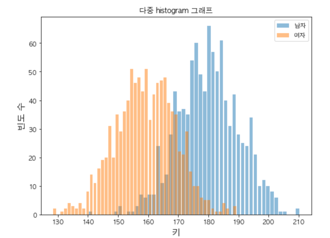
테이블이(가) 표시된 사진

자동 생성된 설명

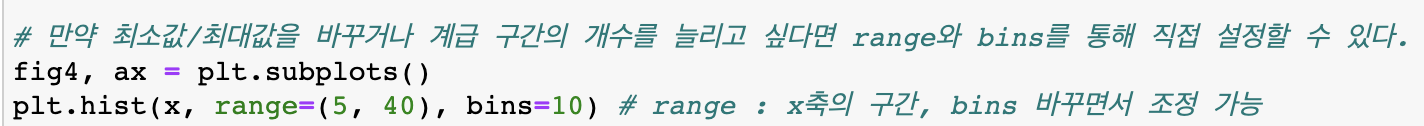
Ex) 단일 히스토그램



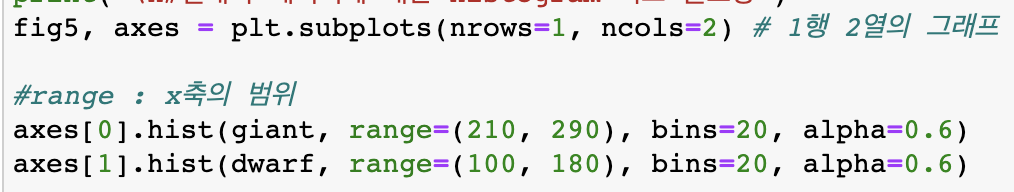
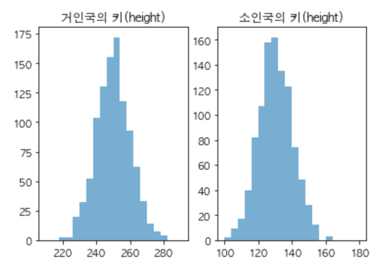
Ex) 다중 히스토그램

Ex) x축의 최소/최대값 바꾸거나, 구간의 개수 조정할 때



Ex) 2개의 histogram 따로 그리기

Ex) 같은 축을 공유하는 histogram

텍스트이(가) 표시된 사진

자동 생성된 설명 

1. seaborn

* 한국 복지 패널 데이터

- 한국 보건 사회 연구원에서 가구의 경제 활동을 연구하여 정책 지원에 반영할 목적으로 발간하는 조사 자료

- 이 데이터는 전처리가 되어 있지 않아서, **데이터 전처리 작업**부터 진행

* Seaborn 라이브러리

- matplotlib의 상위 호환 데이터시각화를 위한 라이브러리로써, **matplotlib을 기반**으로 하는 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지

- 판다스 데이터프레임으로 다양한 통계 지표를 낼 수 있는 시각화 차트를 제공하기 때문에 데이터 분석에 활발히 사용되고 있는 라이브러리

- 기본적인 **시각화 기능은 matplotlib 패키지에 의존**하며, 통계 기능은 Statsmodels 패키지에 의존

- seaborn은 matplotlib보다 기본 색상표가 더 뛰어나기 때문에 **색 표현력이 우수**

**- 장점**

1. 시각적 우수

2. 간결한 구문을 제공

3. 판다스 데이터프레임에 최적화

4. 데이터프레임을 집계해서 쉽게 차트로 요약 가능

* 한국 복지 패널 데이터 전처리 과정

- ‘gender’ – 1이면 ‘남성’, 2면 ‘여성’으로 변경

- ‘marriage – 1이면 ‘결혼’, 2면 ‘이혼’, 나머지는 ‘무응답’으로 변경

- ‘code\_job’ : 직업 코드와 관련된 컬럼 (welfare\_job.csv 파일과 결합하여 직업 이름 정보를 추출)

- ‘code\_religion’ : 7개 권역별 지역을 문자열로 변경

- ‘religion’ : 1이면 ‘있음’, 2이면 ‘없음’으로 변경

- ‘income’ : 소득이 없는 데이터에 대하여 다른 월급들의 평균값을 적용

- ‘age’ : 현재 년도에서 태어난 년도를 뺄셈 (신규 추가)

- ‘ageg’ : 30대 미만은 ‘청년’, 30세 이상은 ‘중년’, 60세 이상은 ‘노년’으로 지정 (신규 추가)

* 척도 (scale)

- 척도를 나누는 **기준** : **수량화 가능 여부**

|  |  |  |
| --- | --- | --- |
| 범주형 | 성별, 직업 | 크기를 비교할 필요가 없음 |
| 연속형 | 만족도, 키, 몸무게 | 만족도는 1-5 점수로 평가가 가능 |

테이블이(가) 표시된 사진

자동 생성된 설명

-> 섭씨 눈금의 온도는 두 개의 정의된 점이 정의된 후 100개의 간격으로 분리됨. 20도가 10도보다 두 배 더 뜨겁다고 말할 수 없음

* 막대그래프 (countplot())

- **countplot()** : **범주형 데이터**에 대하여 항목별 개수를 세어서(**빈도 수**) 막대 그래프를 그림

x : 데이터프레임의 열 이름 (문자열)

data : 데이터프레임 이름

hue : 특성을 구분할 열 지정

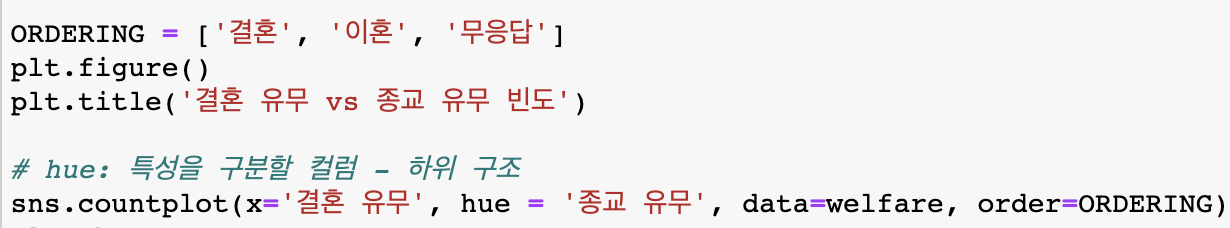
order : 열거할 데이터의 순서를 사용자가 정의하고자 할 때

linewidth : 테두리 선의 두께

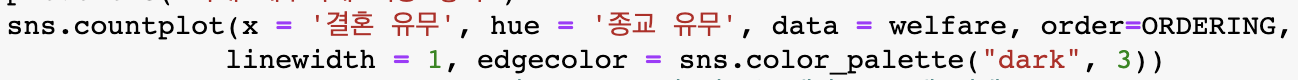
edgecolor : bar의 테두리 색상을 지정

palette : 색상 팔레트 지정

Ex) ‘결혼 유무’ 및 ‘종교 유무’ 컬럼과 관련된 빈도수를 표현 – **수직막대형식**



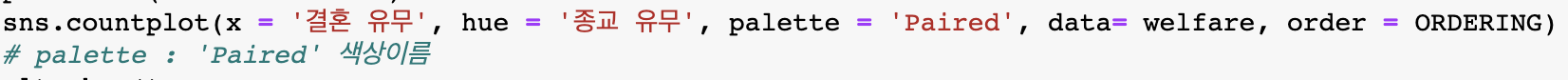
Ex) 테두리 선의 두께나 색상을 지정할 때



Ex) **수평막대형식 –** y 사용



ex) 지정된 팔레트를 사용하여 색상을 지정



* 히스토그램 (distplot())

- **distplot()** : 데이터의 **분포**와 **밀도**를 확인, 러그와 커널 밀도 표시 기능, **1차원 데이터**에 사용

x : series, 1d-array, list 사용

bins : 계급의 개수를 지정

kde : True면 histogram보다 부드러운 형태의 분포 곡선을 보여줌

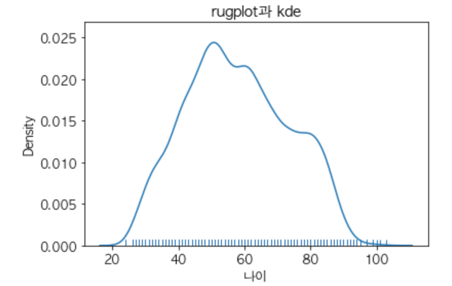
rug : True면 rugplot을 그림

vertical : True면 가로로 그래프를 그림

color : 색상 지정

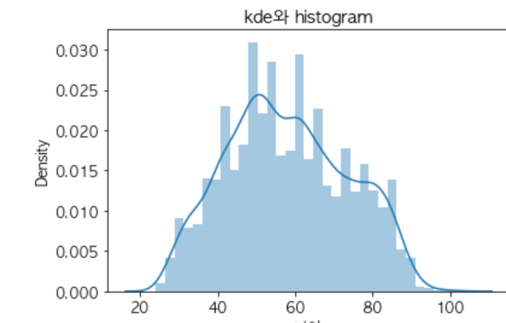
ex) rug, kde만 그림





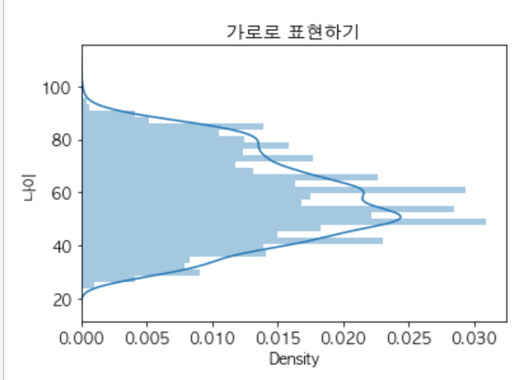
Ex) hist, kde만 그림





Ex) 가로 방향으로 distplot 그리기





Ex) color 바꾸기



* 히트맵 (heatmap)

- **heatmap()** : **2차원 데이터** & 모든 값이 **카테고리 값**일 경우

data : 2차원 형식의 데이터 셋

cmap : colormap 이름이나, 색상을 담고 있는 List 자료구조

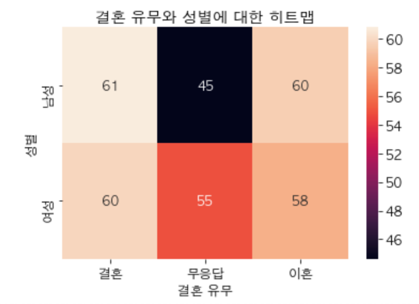
annot : True면 cell에 해당하는 수치 값

- 색상으로 표현된 행렬 정보를 이용하여 사각형의 그래프를 그림.

- 색상으로 표현할 수 있는 다양한 정보를 일정한 이미지 위에 열 분포 형태의 visual graphic으로 출력

Ex) ‘결혼 유무’를 x축으로 ‘성별’을 y축으로 히트맵 그림

텍스트이(가) 표시된 사진

자동 생성된 설명 

Ex) 상관계수를 이용한 히트맵

텍스트이(가) 표시된 사진

자동 생성된 설명 

* 짝그래프(pairplot)

- **pariplot()** : **3차원 이상**의 **다차원 실수형** 데이터

data : 적용할 데이터 프레임

hue : 특성을 구분할 컬럼

palette : 팔레트

height : 각 facet의 높이를 inches로 지정 (default = 2.5)

markers : 마킹될 도형의 모양을 지정

- dataset 중에서 2개 컬럼 간의 관계를 그래프로 그려줌

Ex) 결혼 유무는 범주형이므로 색깔로 구분 컬러, 높이 지정





* 바이올린 그래프(violinplot)

- **violinplot()** : KDE plot과 Box plot을 서로 조합해서 그린 그래프, 커널 밀도 추정치를 이용하여 그래프 그림

x : 그리고자 하는 컬럼을 명시

data : 대상이 되는 데이터프레임

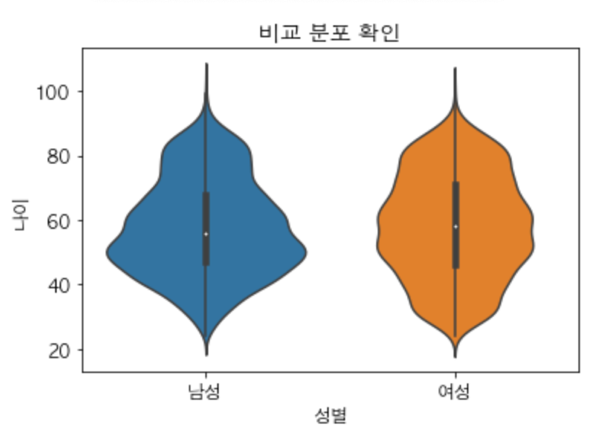
hue : 특성을 구분할 컬럼

palette : 색상 팔레트

ex) 나이에 대한 바이올린 그래프



Ex) x, y축을 지정해줌으로써 바이올린을 분할하여 비교 분포 가능



Ex) 가로 형태의 바이올린 그래프



Ex) Hue 옵션으로 비교 – 분리 : 단일 column에 대한 바이올린 모양의 비교

종교 유무에 따른 바이올린 그래프를 그리되, 성별로 분리함





* 선형 회귀 모델 그래프 (Lmplot)

- **lmplot()** : 선형회귀 모델과 연관이 있는 함수, column간의 **선형적인 관계** 확인, 이상치데이터 짐작 가능

x, y : x축과 y축에 그려질 데이터

data : 사용할 dataframe

hue : 특성을 구분할 컬럼

height : 각 항목의 면에 대한 높이는 inch 단위로 지정

col = ‘ ’ : 항목별 그래프

col\_wrap = 숫자 : 한 줄에 표기할 column의 개수

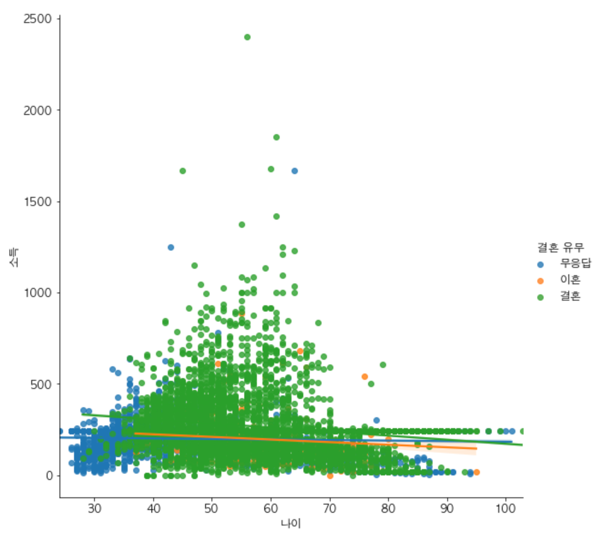
scatter\_kws : scatter 관련 keyword를 사전 형식으로 지정

ex) 나이와 소득에 대한 산점도 그래프



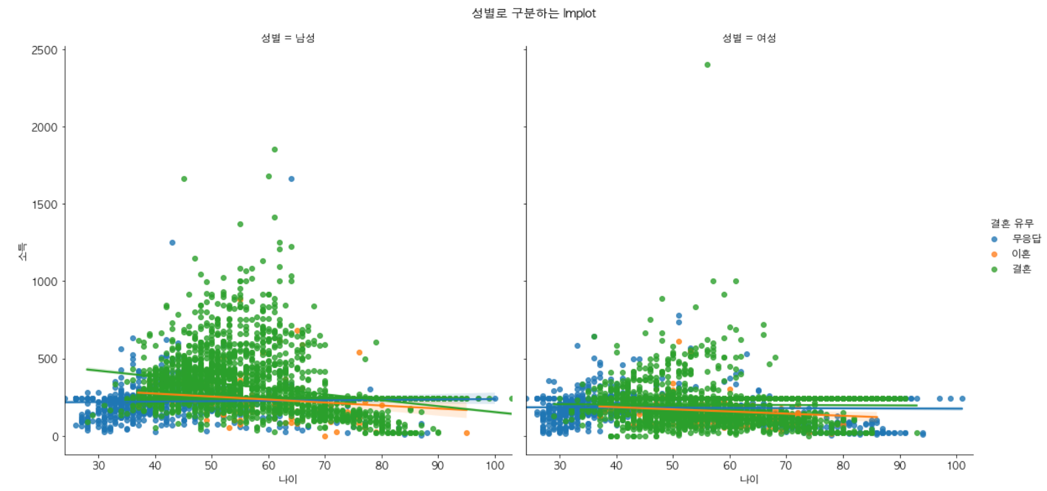
Ex) 데이터 항목별 색상 표기 – Hue 옵션





Ex) 항목별 그래프 별도로 그리기 – col 매개변수





* 산점도그래프 (replot)

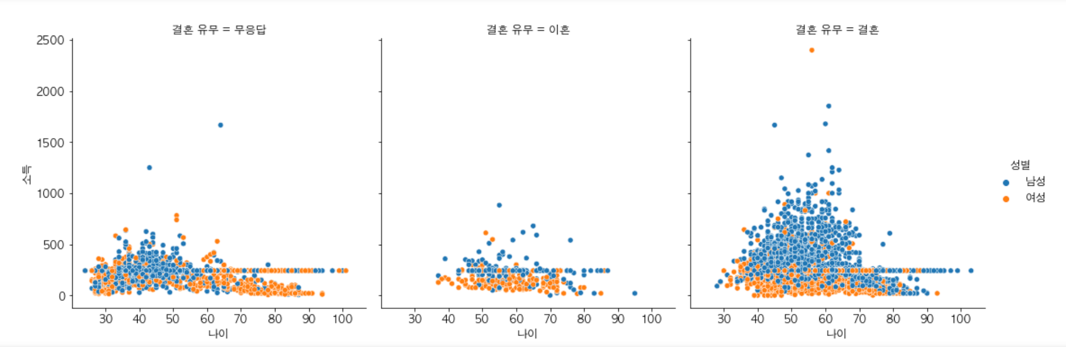
- **replot()** : 두 column간의 **상관관계**를 보여주지만, lmplot처럼 선형 관계를 그려주지 않음

Ex) 나이와 소득에 따른 산점도, 성별은 색으로 나타남



Ex) 결혼 유무 범주형 데이터 개수 만큼 데이터를 별도로 그림 – col 사용





Ex) 위 그래프처럼 그려지지만, 연령대가 행, 결혼 유무가 열로 나타남 (3x3 차트 영역)



Ex) 색상 지정