표지판, 시계이(가) 표시된 사진

자동 생성된 설명

2020년 2학기

Data summary and visualization

|  |
| --- |
| 과목: 데이터사이언스개론 |
| 담당교수: 신효섭 |
| 제출일: 2020년9월20일 (일) |
| 학과: 기술경영학과 |
| 학번: 201512082 |
| 성명: 백 승 호 |

**과제**

엑셀파일 db\_score.xlsx를 입력파일로 받아서 dataframe으로 읽어들이고, 이 데이터들을 토대로 (1) mean, median (2) mode (3) variance, standard deviation, AAD, MAD (4) percentile plot (5) boxplot (6) histogram (7) scatter plot 각각을 python으로 구현

**[ 개발 설명 ]**

Excel파일을 데이터 프레임으로 읽어와서 각각에 필요한 처리들을 진행해준다.

1. Mean, median (grade 제외)

함수: print\_mean(), print\_median()

Grade를 제외한 데이터프레임의 각 열의 리스트에 mean()과 median()을 취해주는 것을 반복하는 함수를 구현한다.

1. Mode (grade만)

함수: print\_mode()

Grade열에 mode()를 취해주고 출력형태를 정제하기 위해 답이 복수개일때 테이블이 아닌 값이 출력되도록 구현한다.

1. Variance, standard deviation, AAD, MAD (grade 제외)

함수: print\_spread()

var(), std()로 각각 분산과 표준편차를 구현한다. AAD의 경우 mean으로 attribute의 평균값을 구한 뒤 각 요소들과 평균값의 차이를 abs()로 절대값으로 변환하고, mean()으로 다시 해당 절대값들의 평균을 구하는것으로 직접 구현했다. MAD의 경우 마찬가지로 mean으로 attribute의 평균값을 구한 뒤 각 요소들과 평균값의 차이를 abs()로 절대값으로 변환하고, 마지막에 median()으로 중간값을 찾는 식으로 직접 구현했다.

1. Percentile plot (grade 제외)

함수: print\_pplot()

Linspace(0,100)으로 100분위를 표현하는 x축을 마련하고 데이터 프레임으로 부터 각 attribute를 y축으로 마련한뒤, np.percentile() 함수를 이용해 interpolation=’linear’옵션으로 그래프를 그릴수있도록 했다. 또한 title과 ylabel각각 attribute에 따라 유동적으로 값이 바뀔수있도록 구현했다.

1. Boxplot (grade 제외)

함수: print\_boxplot()

데이터 프레임으로 부터 타겟인 attribute 리스트를 마련하고 반복문속에서 해당 타겟을 boxplot()의 인자로 넘겨줌으로 구현했다. 각 plot의 제목역시 attribute에 따라 바뀌게끔 값을 할당해줬다.

1. Histogram (모든 attribute)

함수: print\_histogram()

데이터 프레임으로 부터 타겟인 attribute 리스트를 마련하고 .hist()의 인자로 넘겨줌으로 구현했다. 표의 제목은 attribute에 의존적으로 구현했고 ylabel의 이름은 frequency로 고정해서 나타냈다.

1. Scatter plot (grade 제외, 가능한 모든 attribute 조합)

함수: print\_splot()

Grade를 제외한 attribute들의 조합을 토대로 그려야하기에 itertools의 combinations를 import하여 grade를 제외한 두개를 선택하여 조합하는 경우의 리스트를 뽑아냈다. 해당 리스트의 요소를 scatter()의 인자로 전달하고 테이블명, x라벨, y라벨에 각각에 해당하는 attribute의 조합과 attribute들을 할당하여 출력하도록 구현했다.

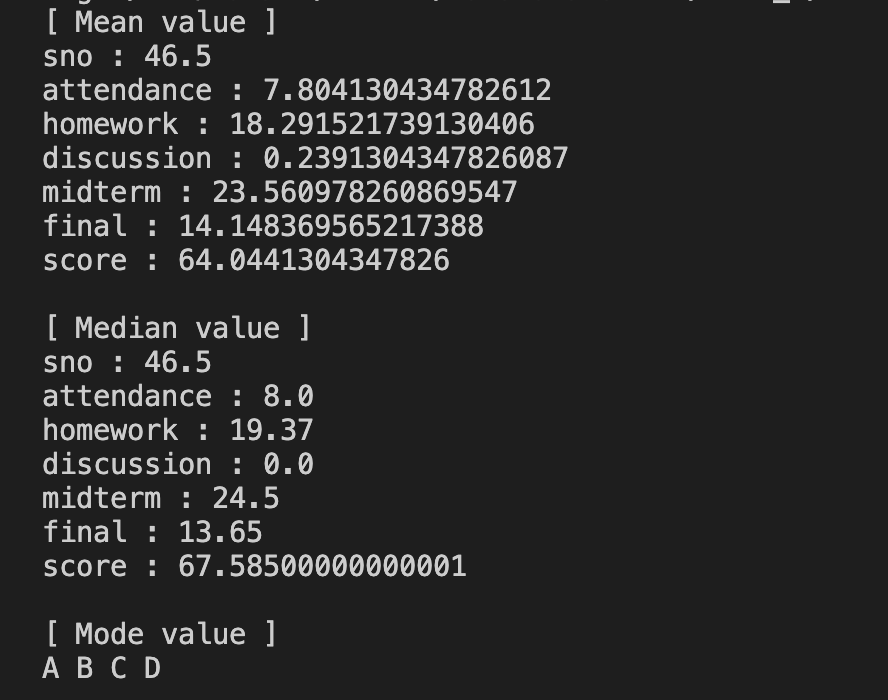
**[ 소스 코드 ]**

파이썬 소스코드

|  |
| --- |
| def print\_mean(df):  col = df.columns  i = 0  print("[ Mean value ]")  while i < len(col) - 1:  print(str(col[i]) + " : " + str(df[col[i]].mean()))  i = i + 1  print()  def print\_median(df):  col = df.columns  i = 0  print("[ Median value ]")  while i < len(col) - 1:  print(str(col[i]) + " : " + str(df[col[i]].median()))  i = i + 1  print()  def print\_mode(df):  col = df.columns  i = 0  print("[ Mode value ]")  result = df[col[7]].mode()  while i < len(result):  print(result[i], end = ' ')  i = i + 1  print()  print()  def print\_spread(df):  col = df.columns  i = 0  print("[ Variance value ]")  while i < len(col) - 1:  print(str(col[i]) + " : " + str(df[col[i]].var()))  i = i + 1  print()  i = 0  print("[ standard deviation value ]")  while i < len(col) - 1:  print(str(col[i]) + " : " + str(df[col[i]].std()))  i = i + 1  print()  i = 0  print("[ AAD value ]")  while i < len(col) - 1:  temp = df[col[i]]  temp\_mean = df[col[i]].mean()  aad = abs(temp - temp\_mean).mean()  print(str(col[i]) + " : " + str(aad))  i = i + 1  print()  i = 0  print("[ MAD value ]")  while i < len(col) - 1:  temp = df[col[i]]  temp\_mean = df[col[i]].mean()  mad = abs(temp - temp\_mean).median()  print(str(col[i]) + " : " + str(mad))  i = i + 1  print()  def print\_pplot(df):  col = df.columns  p = np.linspace(0, 100)  print("[ Percentile plot ]")  i = 0  while i < len(col) - 1:  y = df[col[i]]  ax = plt.gca()  ax.plot(p, np.percentile(y, p, interpolation='linear'))  ax.set(title=col[i] + ' percentile plot',  xlabel='Percentile',  ylabel=col[i])  plt.show()  i = i + 1  print()  def print\_boxplot(df):  col = df.columns  i = 0  print("[ Box plot ]")  while i < len(col) - 1:  target = df[col[i]]  plt.boxplot(target)  plt.title(col[i])  plt.show()  i = i + 1  def print\_histogram(df):  col = df.columns  i = 0  print("[ Histogram ]")  while i < len(col):  target = df[col[i]]  plt.hist(target)  plt.title(col[i], pad=10)  plt.ylabel('frequency', labelpad=10)  plt.show()  i = i + 1  def print\_splot(df):  copy = df  del copy['grade']  col = copy.columns  combination = list(combinations(col, 2))  i = 0  print("[ Scatter plot ]")  while i < len(combination):  target = combination[i]  X = target[0]  Y = target[1]  plt.scatter(df[X], df[Y])  plt.title(X + ' & ' + Y, pad=10)  plt.xlabel(X, labelpad=10)  plt.ylabel(Y, labelpad=10)  plt.show()  i = i + 1  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from itertools import combinations  x\_file = '~/db\_score.xlsx'  df = pd.read\_excel(x\_file)  print\_mean(df)  print\_median(df)  print\_mode(df)  print\_spread(df)  # percentile plot(grade x)  print\_pplot(df)  # box plot (grade x)  print\_boxplot(df)  # histogram (all)  print\_histogram(df)  # scatter plot (every possible combination)  print\_splot(df) |

**[ 실행 결과 캡쳐 ]**

**[ mean, median, mode]**



[ variance, standard deviation, AAD, MAD ]

텍스트이(가) 표시된 사진

자동 생성된 설명

[ percentile plot ]

스크린샷이(가) 표시된 사진

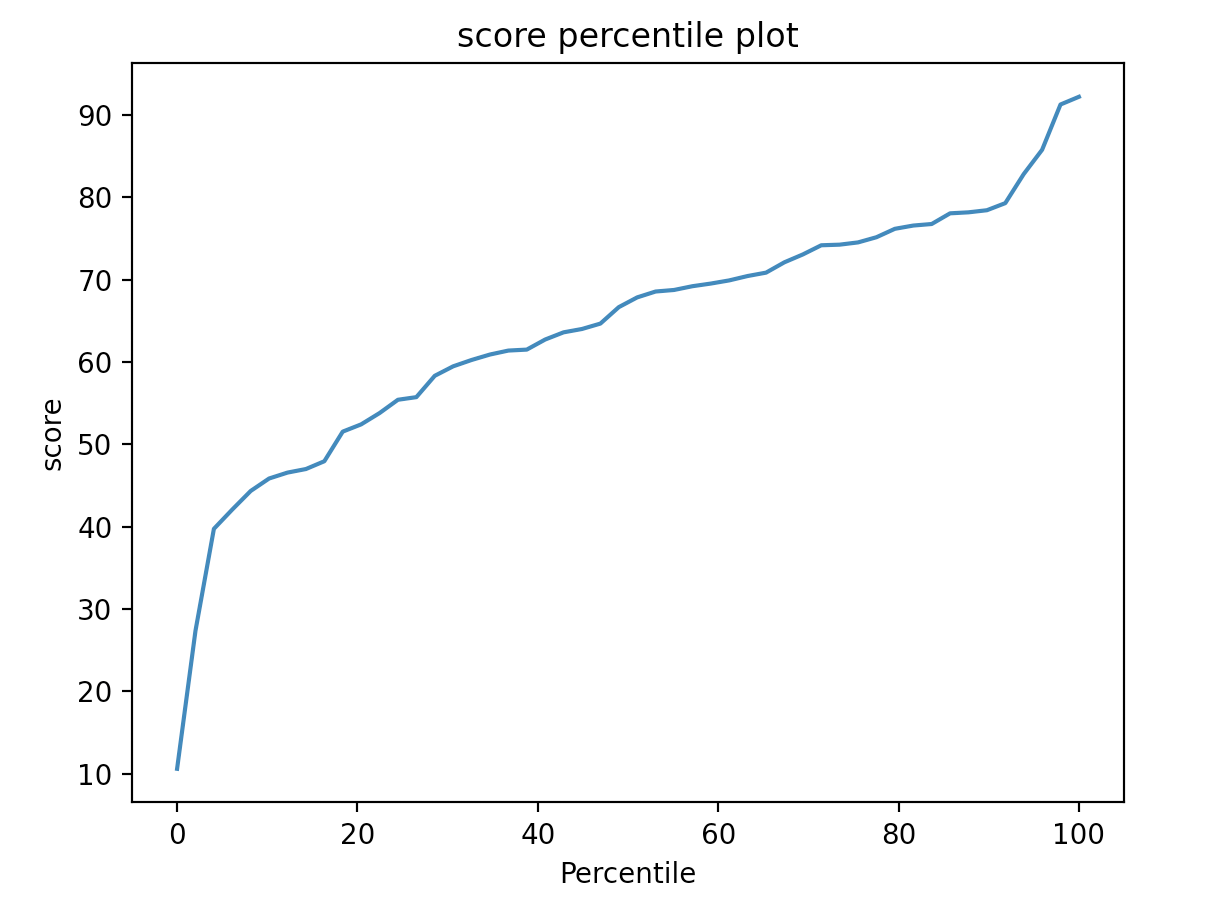
자동 생성된 설명지도, 스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷, 지도이(가) 표시된 사진

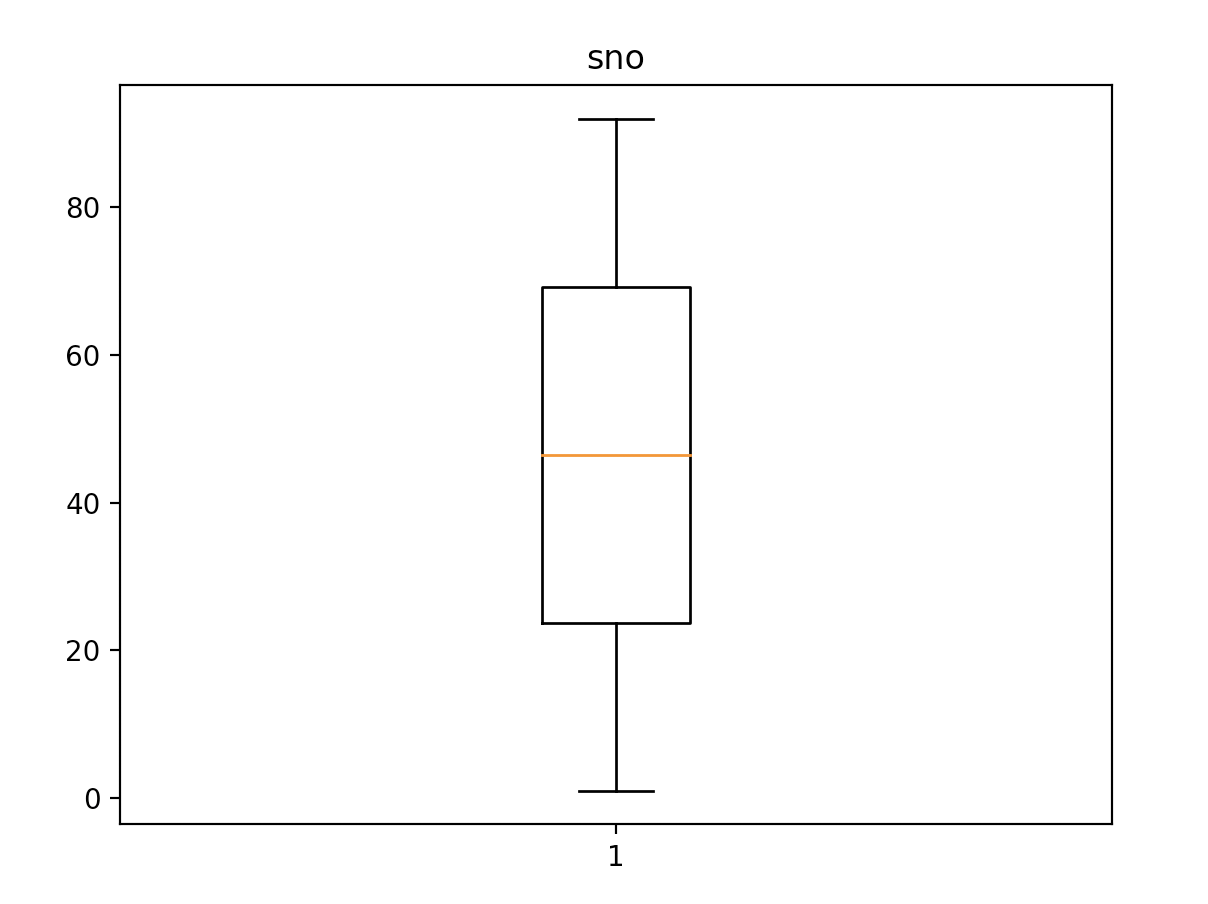
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명지도이(가) 표시된 사진

자동 생성된 설명지도이(가) 표시된 사진

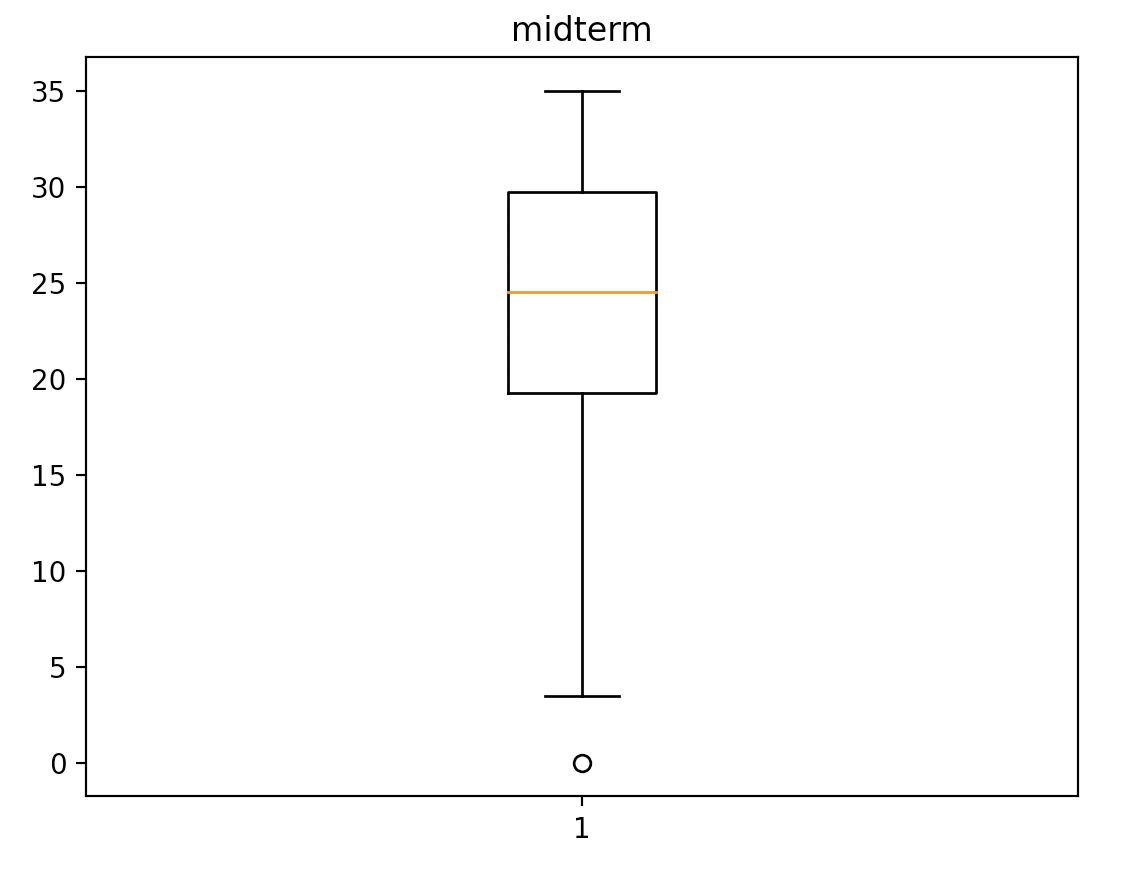
자동 생성된 설명

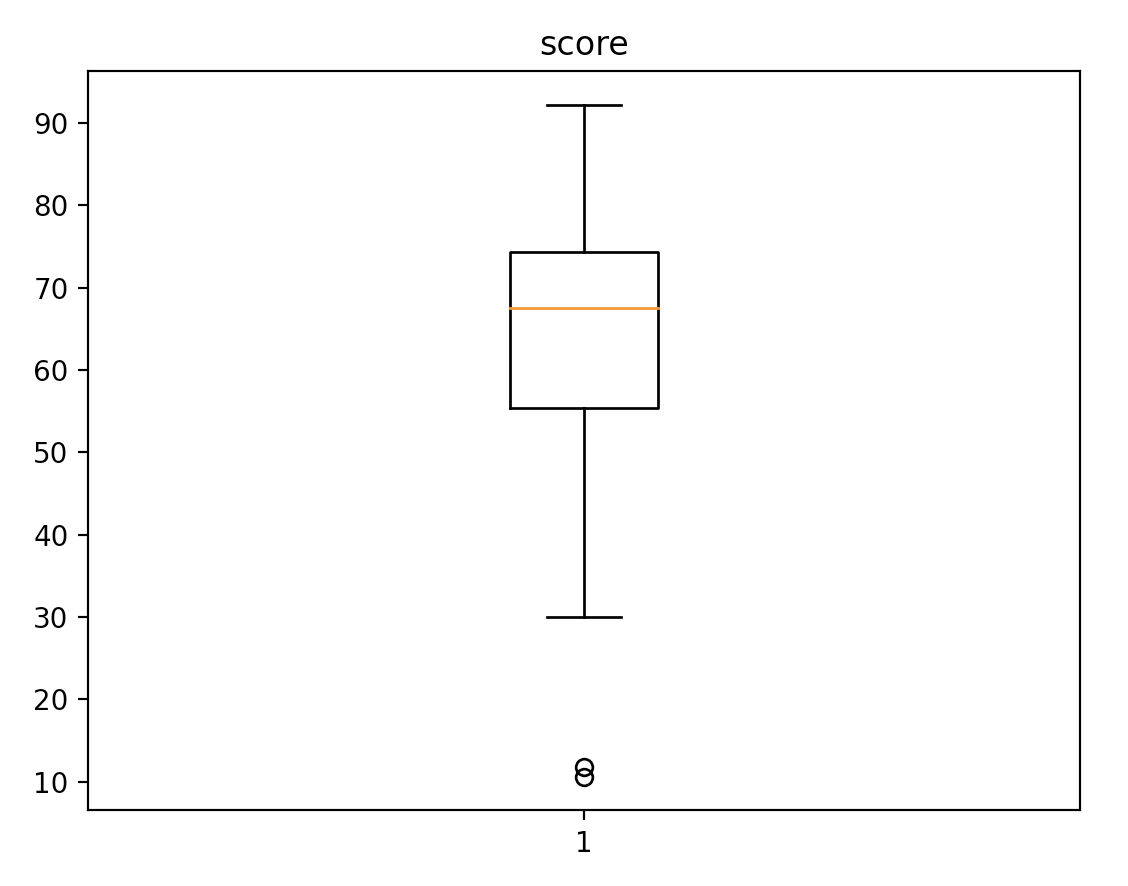
[ box plot ]

스크린샷이(가) 표시된 사진

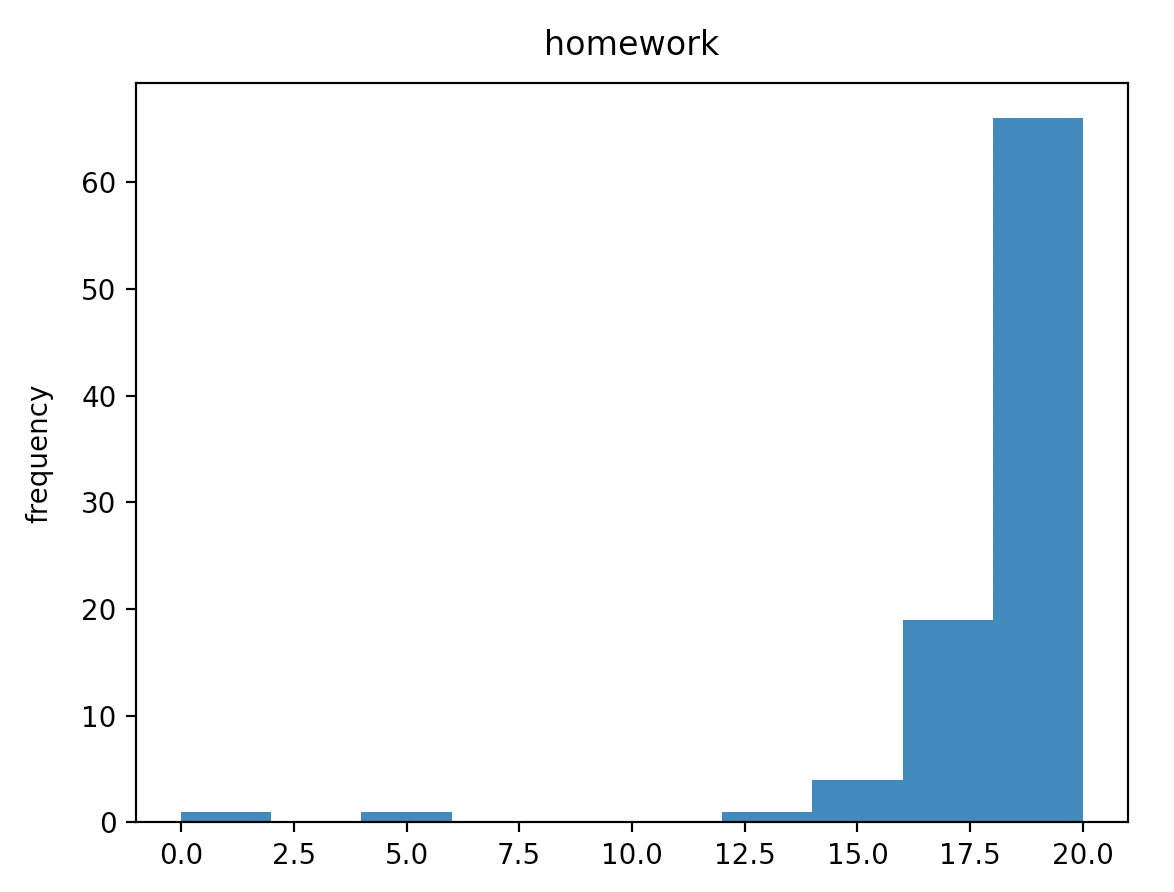
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

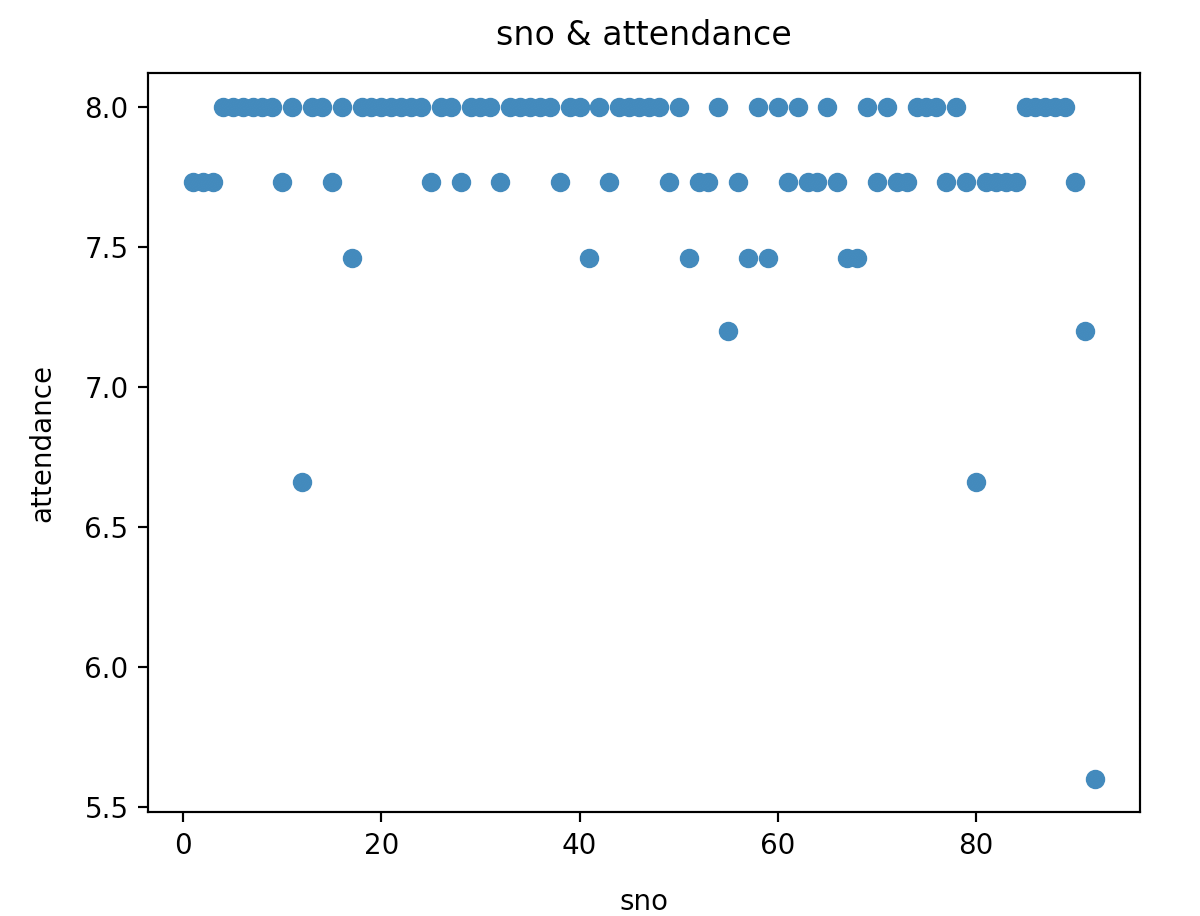
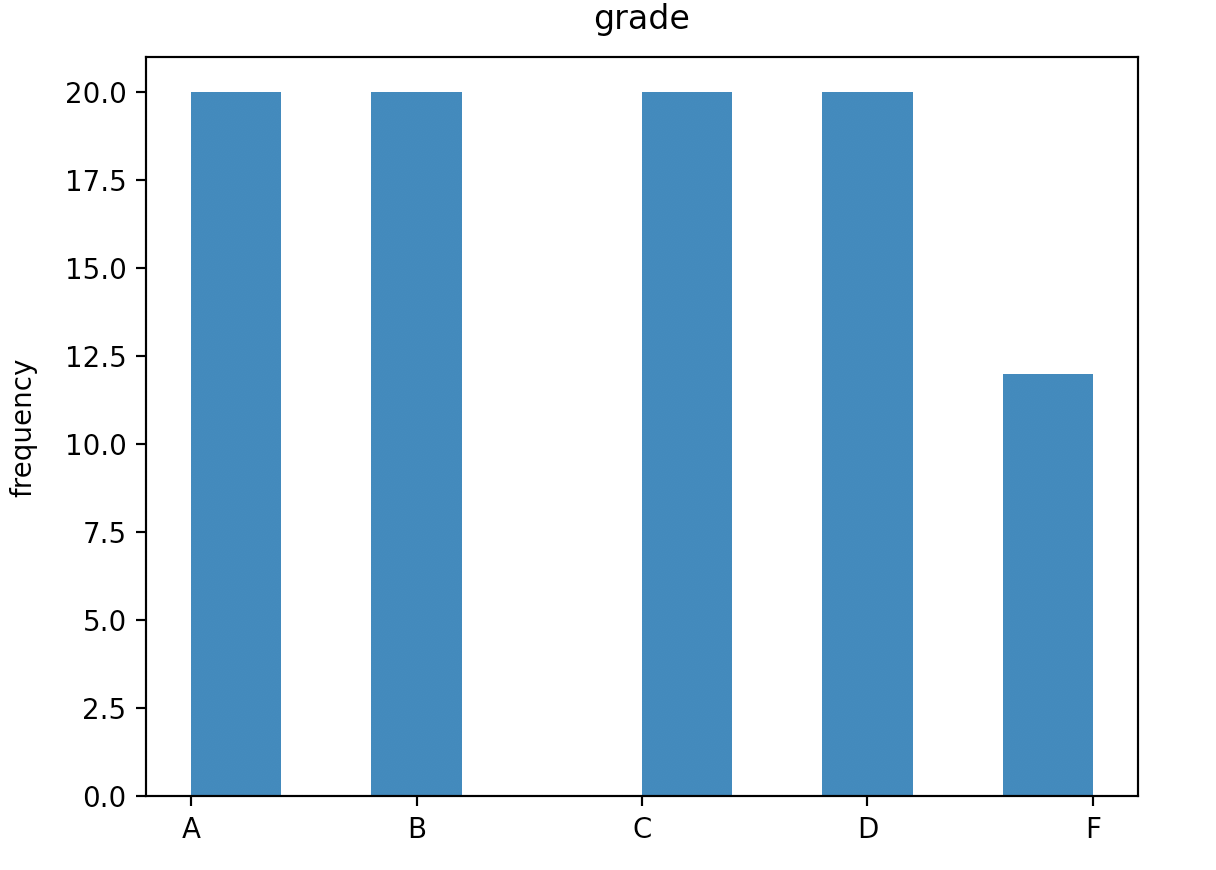
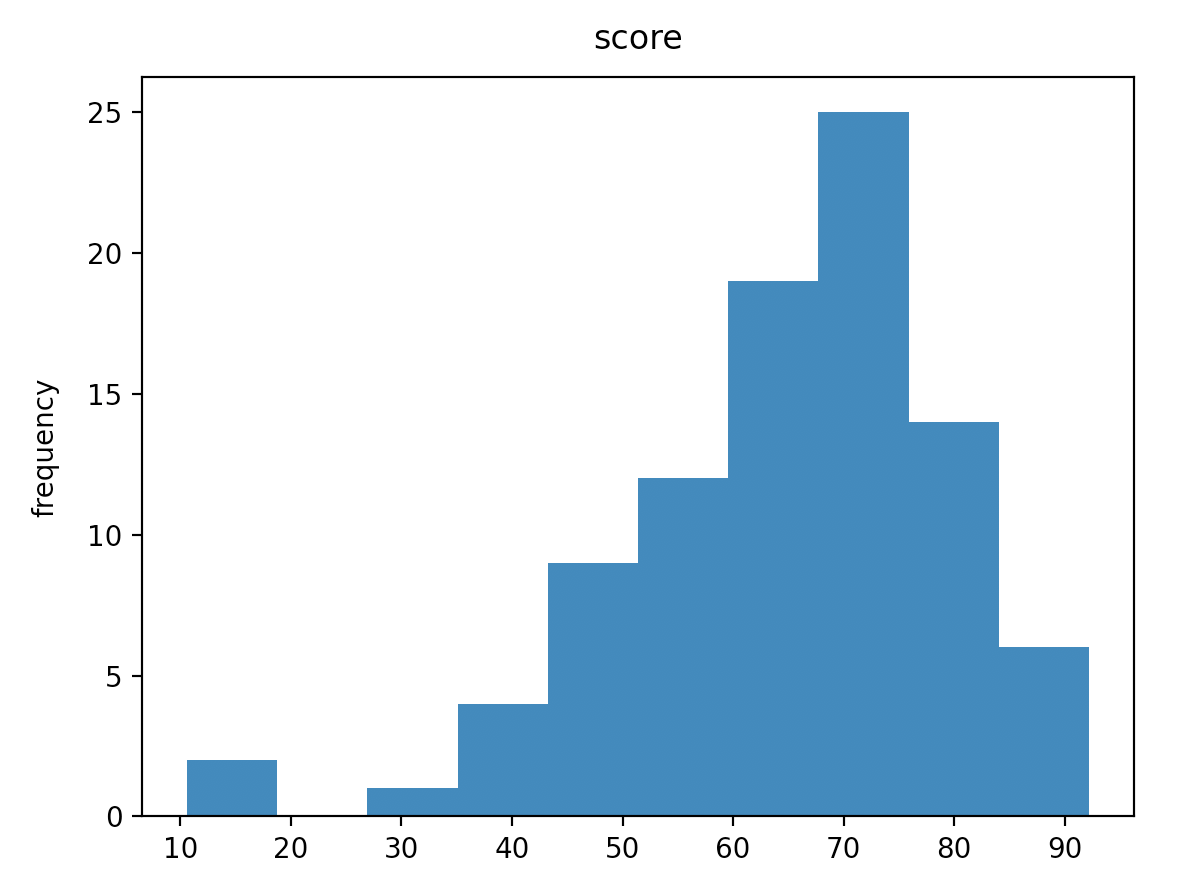
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

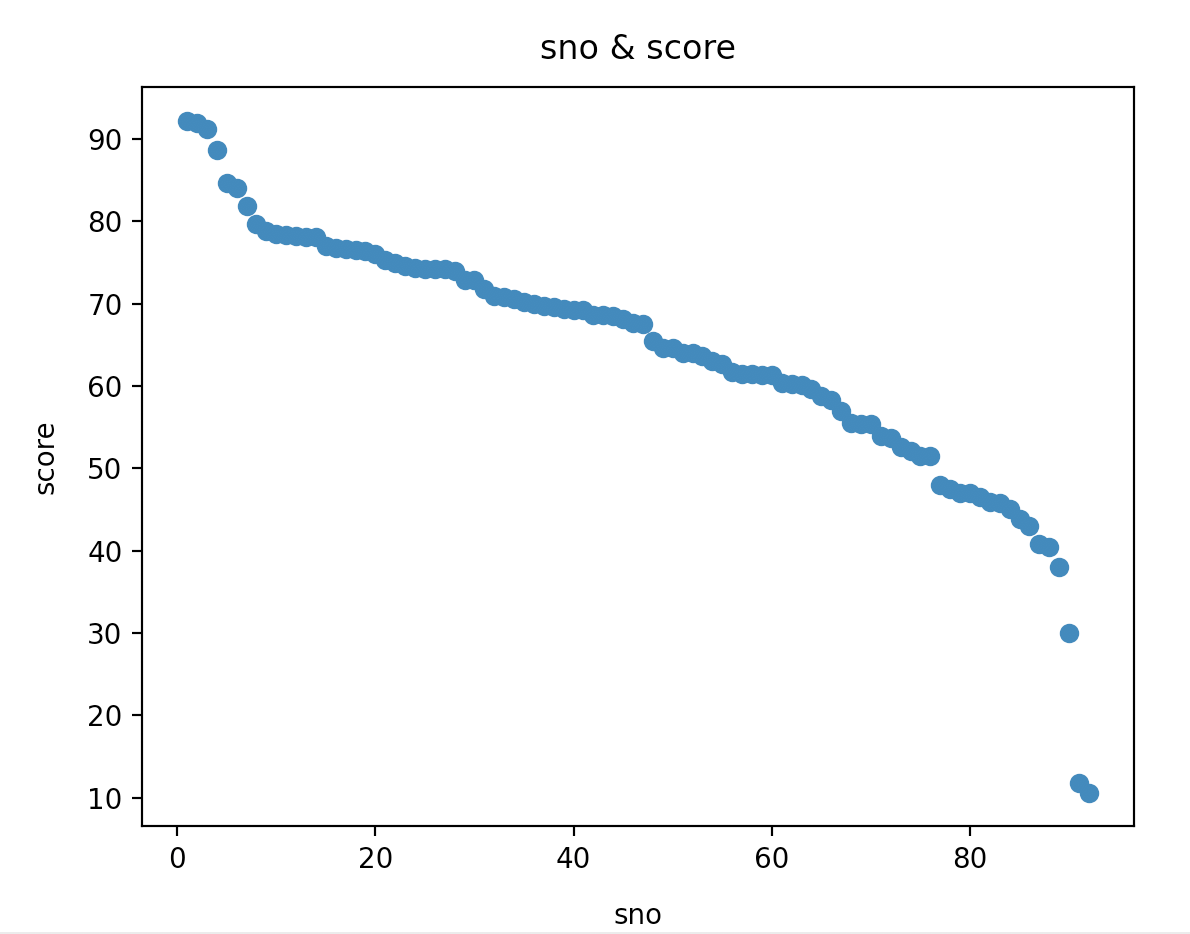
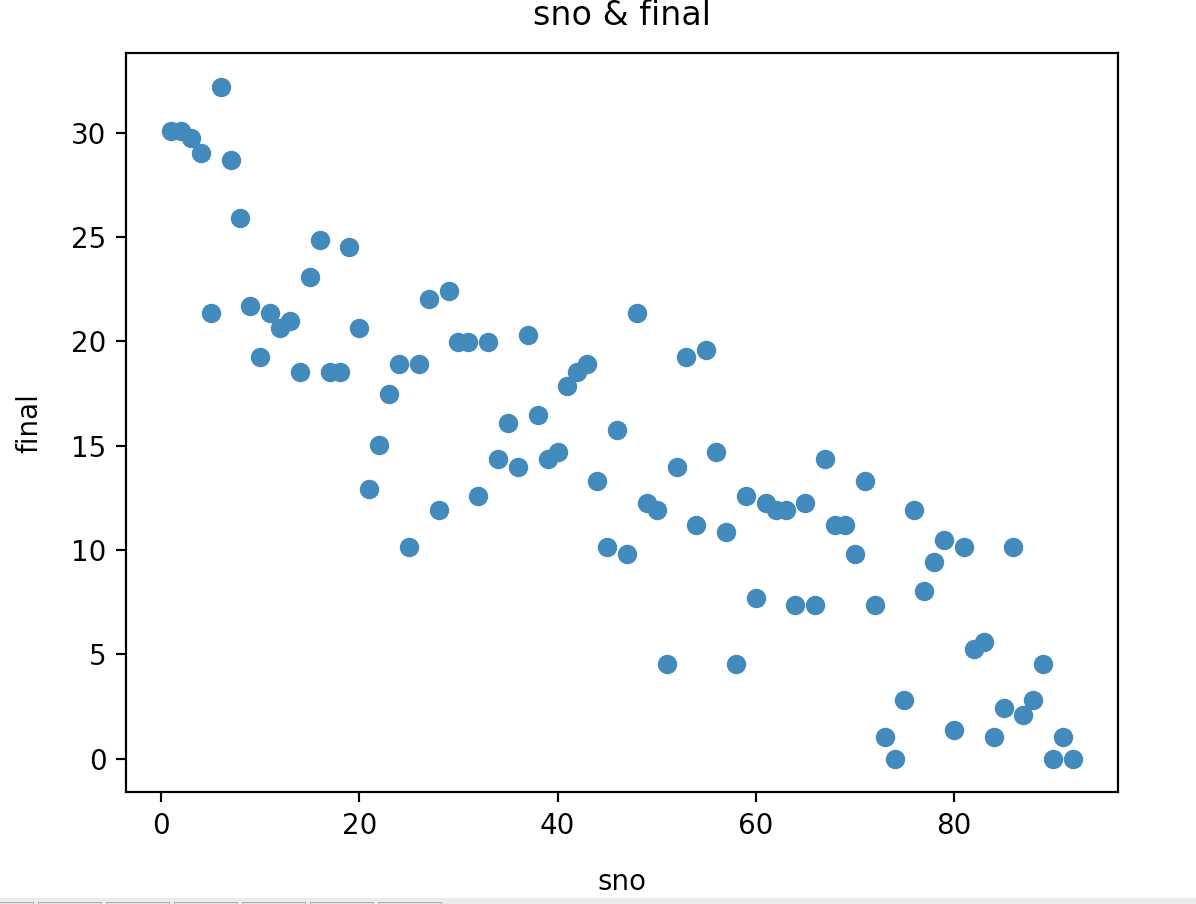
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

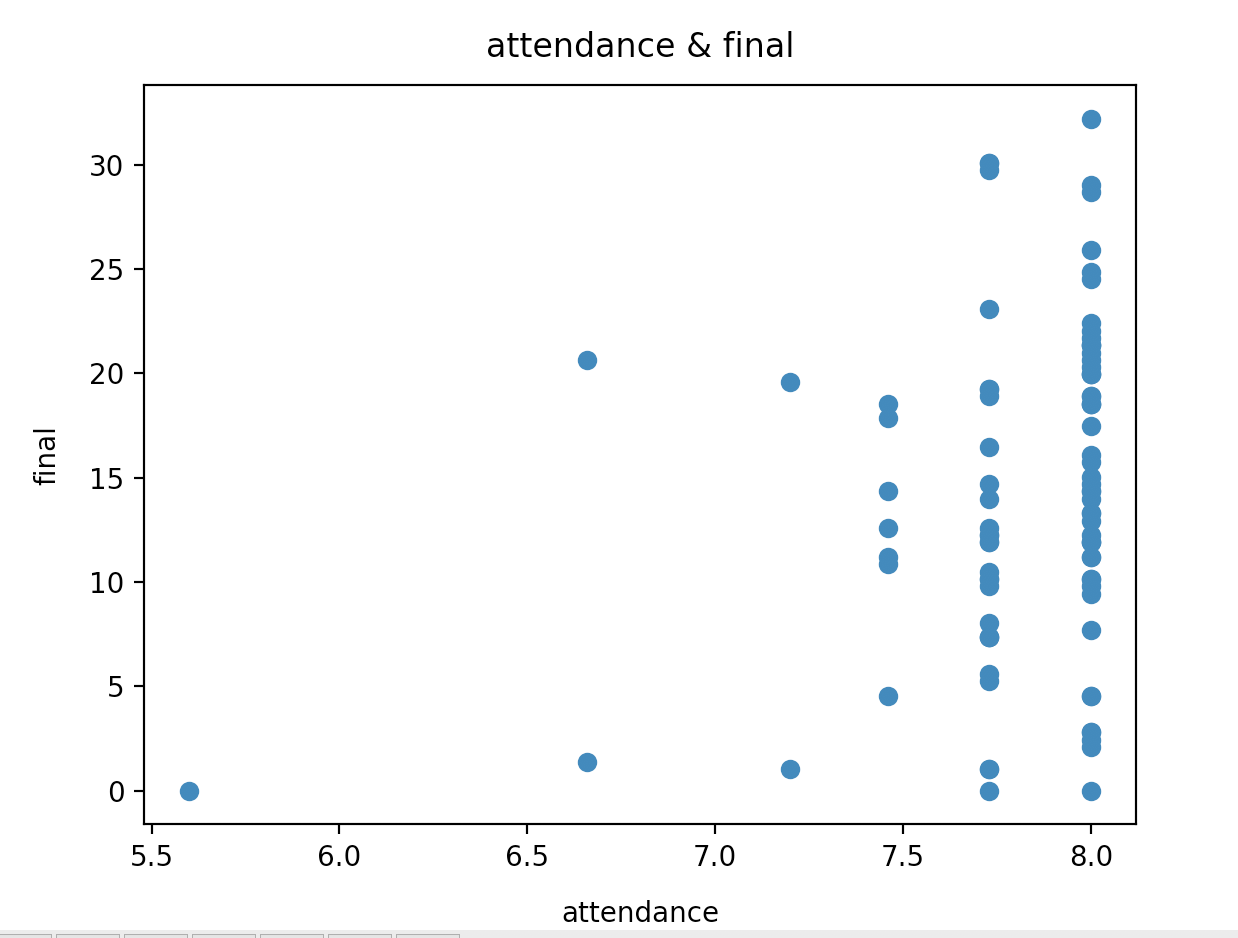
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

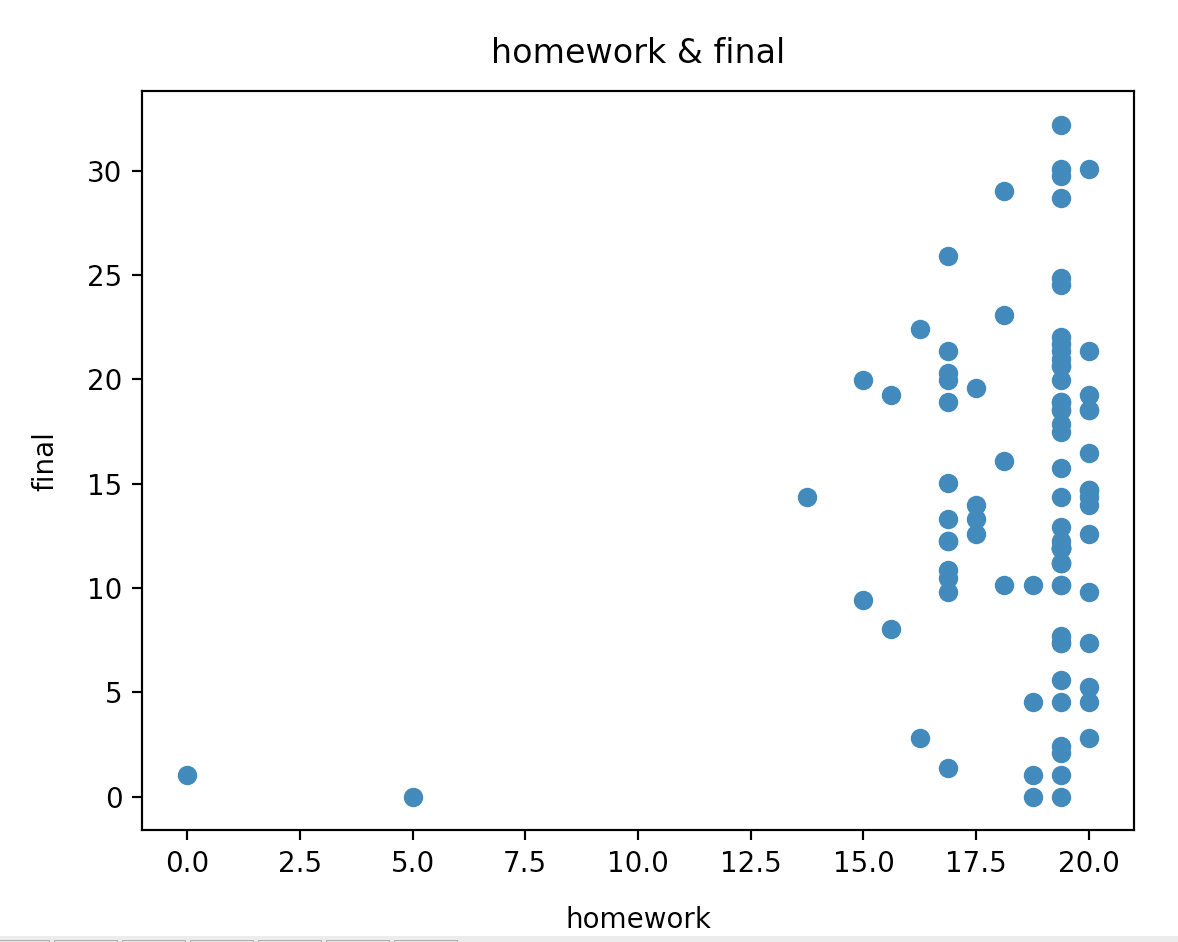
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

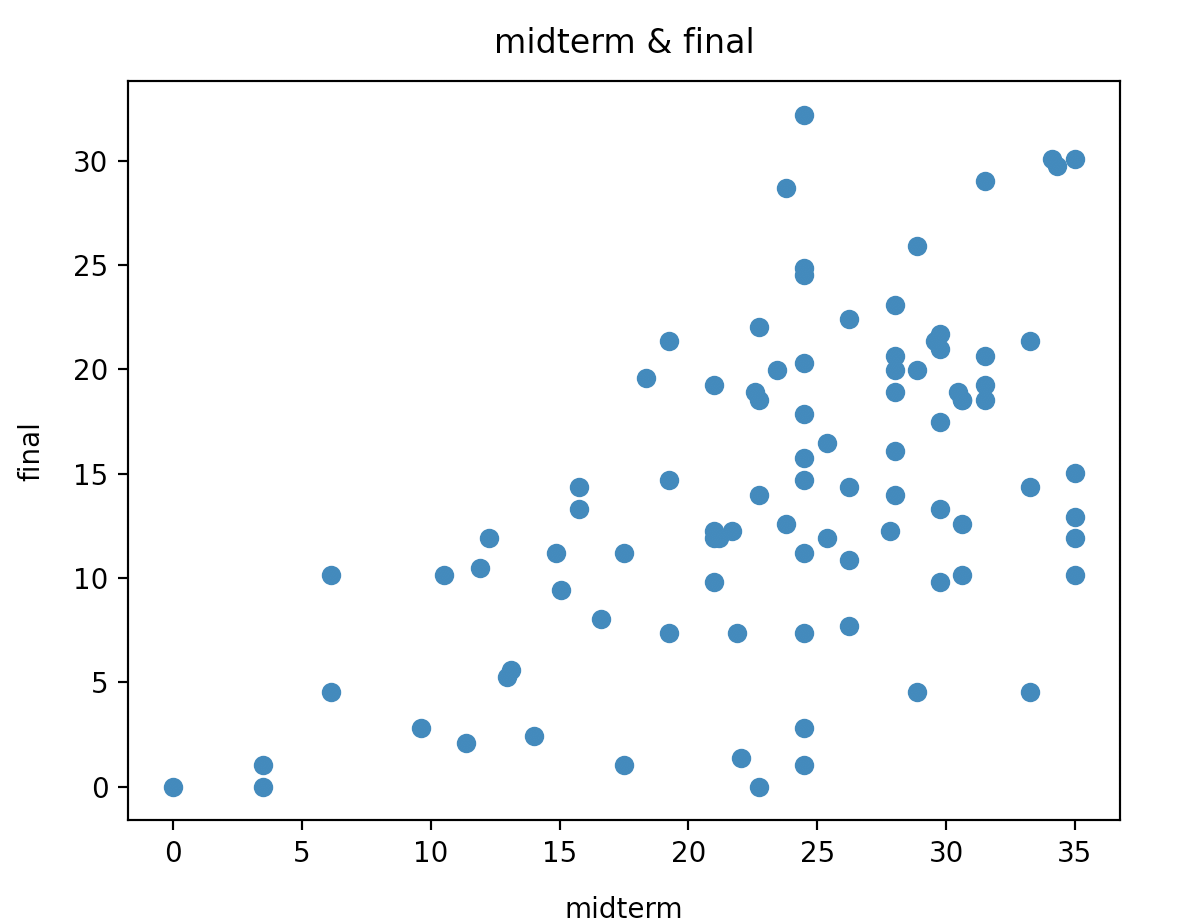
자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명