# What is the relationship between professor ratings and grades?

December 20, 2021

## 1 Do the grades students receive influence the ratings they leave on professor reviews?

Among people who design course review websites, there is a common concern that the reviews of the professors paint a misleading picture. The idea is that if a professor is teaching a difficult course, then the average grades they give out will likely be lower. As a result, students will be more likely to negatively rate their experience with that professor, and if someone reads their reviews, they may get the wrong impression that the professor is bad when in reality, it's just that the course they teach is more difficult.

In this analysis, I will use grade and review data from PlanetTerp to look at the relationship between a professor's ratings and the grades they give out. According to the PlanetTerp website, the review data comes in part from OurUMD.com. Professor and course information comes from umd.io. Grade data comes from the UMD Office of Institutional Research, Planning & Assessment (IRPA) and through a request under the state of Maryland's Public Information Act (PIA) (grade data initially from VAgrades.com).

The null hypothesis will be that there is a positive correlation between a professor's ratings and the grades they give out in their courses.

## 2 Average Rating vs. Average GPA

Our first task will be to gather some data. There are two things that we need to start out. The average rating of each professor and their average GPA over all the courses they teach. The first of these is really easy. The PlanetTerp API provides the average rating for a requested professor when you hit a certain endpoint.

First, we need to get all of the professors. We will use this code below to build a dataframe object that contains the following properties: - name: name of professor - slug - type: professor or TA - courses: list of courses taught - average_rating

[205]:
```
import requests
import pandas as pd
import io
import time
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt

cur_offset = 0
```

```
professor_df = None
while cur_offset < 11039:
    time.sleep(3)
    # print(f"Requesting data. Offset is {cur_offset}")
    req = requests.get(f"https://api.planetterp.com/v1/professors?
 ↪limit=1000&offset={cur_offset}").content
    df = pd.read_json(io.StringIO(req.decode('utf-8')))
    cur_offset += 1000
    if professor_df is None:
        professor_df = df
    else:
        professor_df = professor_df.append(df)
```

[206]:
```
professor_df = professor_df.loc[professor_df["type"] == "professor"] # Filter␣
 ↪out all the TA's
professor_df = professor_df[professor_df["average_rating"].notna()] # Drop␣
 ↪anyone who doesn't have any ratings
professor_df
```

[206]:
```
               name            slug       type  \
1         A Anthony         anthony  professor
2      A Kruglanski      kruglanski  professor
4           A Sharma        sharma_a  professor
6      A.U. Shankar    shankar_a.u.  professor
8       Aaron Allen     allen_aaron  professor
..              ...             ...        ...
4     Zhengguo Xiao  xiao_zhengguo  professor
20     Zhongchi Liu    liu_zhongchi  professor
29       Zita Nunes      nunes_zita  professor
32   Zohreh Davoudi         davoudi  professor
34     Zsuzsa Daczo           daczo  professor

                                              courses  average_rating
1                           [AMST202, AMST203, AMST101]            1.00
2    [PSYC743, PSYC748M, PSYC489H, PSYC489T, PSYC78…           2.00
4                                             [ASTR300]            1.80
6        [CMSC412, CMSC414, CMSC712, CMSC216, CMSC798]            2.10
8                                    [HHUM205, HHUM206]            5.00
..                                                  …               …
4    [ANSC460, ANSC214, GEMS296, ANSC212, GEMS297, …           4.00
20             [BSCI378H, BSCI410, CBMG699Y, MOCB899]            4.50
29   [ENGL234, ENGL749B, ENGL448B, ENGL300, ENGL301…           3.75
32                          [PHYS604, PHYS411, PHYS624]            2.50
34               [SOCY105, SOCY227, SOCY325, WMST325]            4.50

[2481 rows x 5 columns]
```

Now that we have a table for our professors, we need to figure out the average GPA they give out for their courses. We'll do this by navigating to their page on PlanetTerp and scraping the content using BeautifulSoup.

[207]:
```python
import re

base_url = "https://planetterp.com/professor/"
matches = []
for idx, row in professor_df.iterrows():
    time.sleep(0.1)
    r = requests.get(base_url + row["slug"])
    root = BeautifulSoup(r.content, "html")
    s = root.find("strong", id="grade-statistics")
    # print(s)
    search_res = re.search(r"\d\.\d{2}", str(s))
    if search_res is None:
        # print(row["slug"])
        matches.append(float("NaN"))
    else:
        matches.append(search_res.group(0))
matches
```

[207]:
```
['2.48',
 '3.49',
 '2.82',
 '2.41',
 '3.61',
 '2.52',
 '2.53',
 '3.70',
 '3.82',
 '3.10',
 '3.14',
 '3.37',
 '3.19',
 '3.01',
 '3.18',
 '2.88',
 '3.08',
 nan,
 '3.13',
 '3.23',
 '3.21',
 '3.18',
 '2.53',
 '2.49',
 '3.17',
nan,
'2.91',
'3.99',
'3.51',
'2.87',
'3.20',
'2.91',
'3.31',
'2.89',
'2.64',
'2.94',
'3.17',
'3.57',
'3.20',
'3.12',
'3.42',
'2.81',
'2.95',
'2.55',
'3.15',
'3.29',
'3.42',
'2.98',
'3.52',
'3.29',
'2.73',
'2.62',
nan,
'3.59',
'3.07',
'2.24',
'2.49',
nan,
'3.25',
'3.31',
'3.71',
'3.39',
'2.98',
'3.33',
'3.20',
'3.80',
'2.91',
'2.94',
'3.41',
nan,
'3.50',
'3.42',
'3.70',
'3.40',
'3.85',
'2.80',
'3.18',
'2.59',
'2.40',
'2.48',
'3.59',
'3.48',
'3.83',
'3.71',
'3.56',
'2.14',
'2.90',
'3.41',
'2.72',
'2.58',
'3.65',
'3.48',
'2.79',
'2.81',
'3.44',
'3.49',
'3.33',
'3.58',
'2.34',
'3.21',
'3.40',
'2.76',
'3.37',
'3.54',
'2.61',
'3.27',
'3.55',
'3.42',
'3.60',
'3.51',
'2.44',
'3.00',
'3.11',
'3.64',
'3.01',
'3.56',
'3.21',
'3.51',
'3.04',
'3.80',
'3.03',
'3.10',
'3.71',
'3.13',
'3.36',
'3.46',
nan,
'3.18',
'3.62',
'3.42',
'3.38',
'2.48',
'3.38',
'3.13',
'3.32',
'3.71',
'2.75',
nan,
'3.31',
'2.89',
'3.55',
'3.44',
'3.17',
'3.86',
'2.81',
nan,
'3.48',
'3.04',
'3.46',
'3.29',
'3.31',
'2.79',
'3.23',
'3.50',
'3.41',
'3.75',
'2.63',
'3.28',
'3.14',
'2.78',
'3.70',
'3.35',
'3.99',
'2.27',
'3.43',
'3.41',
'3.50',
'3.35',
'2.88',
'3.17',
'3.69',
'3.70',
'0.00',
'3.33',
'3.48',
nan,
'3.06',
'2.29',
'3.70',
'3.45',
'2.25',
'2.54',
'3.20',
'2.69',
'2.99',
'2.68',
'2.81',
'3.07',
'1.71',
'2.90',
'3.29',
'3.73',
'2.26',
'3.28',
'2.04',
'2.85',
nan,
'3.37',
'3.72',
'3.78',
'3.38',
'3.22',
'2.94',
'3.38',
'3.94',
'3.62',
'3.47',
'3.16',
'3.07',
'3.54',
'3.63',
'3.30',
'3.35',
'3.40',
'3.83',
'3.30',
'3.54',
'3.82',
'3.10',
'3.43',
'3.29',
'3.54',
'3.76',
nan,
'3.35',
'3.56',
'2.95',
'0.71',
'3.38',
'3.15',
'3.52',
'3.04',
'2.65',
'3.05',
'3.32',
'2.77',
'2.04',
'2.99',
'3.75',
'3.20',
'2.89',
'3.68',
nan,
'3.08',
'3.14',
'3.64',
'3.34',
'3.18',
'3.29',
'2.96',
'3.39',
'3.51',
nan,
'3.25',
'3.39',
'3.25',
'2.98',
'3.04',
'3.17',
'2.35',
'3.48',
'3.67',
'3.04',
'2.28',
nan,
'2.71',
nan,
'2.37',
'3.28',
'2.22',
'3.73',
'3.45',
'3.46',
'2.93',
'3.17',
'3.58',
'3.44',
'3.16',
'3.50',
'3.57',
'3.06',
'3.87',
'3.65',
'3.19',
nan,
'3.22',
'3.43',
'3.52',
'3.33',
'3.27',
'2.23',
nan,
'3.58',
'3.88',
'3.06',
'3.46',
'3.10',
'2.29',
'3.16',
'2.95',
'3.85',
'3.64',
'2.81',
'3.31',
'3.33',
'3.84',
'3.62',
nan,
'3.36',
'3.09',
'2.84',
'3.36',
'3.53',
'3.63',
'3.54',
'2.15',
'3.51',
'3.42',
nan,
'3.55',
'3.51',
'3.25',
'2.37',
'3.21',
'3.44',
'3.45',
'2.75',
'2.75',
'2.42',
'3.03',
'3.59',
'3.04',
'3.56',
'3.18',
'2.93',
'3.24',
'3.56',
'2.98',
'3.82',
'3.09',
'3.46',
'3.62',
'3.73',
'2.93',
'2.85',
'3.45',
'2.84',
'3.47',
'3.70',
'2.86',
'2.12',
'3.32',
'3.37',
'2.49',
'2.52',
'2.95',
'3.48',
'2.93',
'2.57',
'3.43',
'3.75',
'3.68',
'2.57',
'3.79',
'2.40',
'3.41',
'2.24',
'3.27',
'3.25',
'3.39',
'3.60',
'3.38',
'2.47',
'2.72',
'2.73',
'3.34',
'3.46',
'3.42',
'3.10',
'2.85',
'3.57',
'2.88',
'3.35',
'3.08',
'3.23',
'3.21',
'2.68',
'2.72',
'2.85',
'3.39',
'3.45',
'3.46',
'3.27',
'3.59',
'2.84',
'2.83',
'3.09',
'2.80',
'2.91',
'2.91',
'3.78',
'3.11',
'3.47',
'3.02',
'3.39',
'3.22',
'3.01',
'3.82',
'3.78',
'3.67',
'2.96',
'3.70',
'3.49',
'3.82',
'3.68',
nan,
'2.32',
'2.51',
'3.93',
'2.51',
'3.29',
'0.00',
'3.25',
'3.03',
'3.78',
'3.37',
'3.02',
nan,
'3.14',
'3.27',
'2.87',
'3.94',
'2.45',
'3.81',
'3.49',
'3.60',
'3.53',
'3.51',
'3.31',
'2.96',
'2.95',
'2.64',
'3.14',
'3.15',
'3.33',
'3.21',
'2.78',
'3.70',
'2.64',
'2.84',
'3.12',
'3.24',
'2.68',
'2.72',
'3.35',
'3.24',
'3.14',
'3.02',
nan,
'3.19',
'2.66',
'3.19',
'3.81',
'3.52',
'3.35',
'3.41',
'3.27',
'3.09',
'2.53',
'3.90',
'3.18',
'2.40',
'3.52',
'3.33',
'3.84',
'3.25',
nan,
'3.10',
'3.57',
'2.78',
'3.59',
'3.02',
'3.49',
'3.16',
'3.56',
'3.26',
'3.33',
'2.74',
'3.24',
'2.70',
nan,
'1.86',
'3.50',
'3.20',
'3.23',
'3.47',
'4.00',
'3.40',
'2.59',
'3.15',
'3.41',
'3.59',
'3.32',
'2.79',
'3.19',
'3.15',
'2.87',
'2.84',
'2.48',
'2.38',
'3.06',
'3.35',
'3.01',
'3.55',
'3.43',
'3.78',
'2.68',
'3.37',
'3.75',
'3.28',
'2.48',
'2.80',
'3.92',
'3.86',
'3.39',
'3.70',
'3.84',
'3.46',
'3.49',
'2.59',
'3.09',
'3.02',
'3.64',
'3.70',
'3.78',
'3.12',
'3.32',
'2.91',
'3.48',
'3.57',
'3.51',
'2.31',
'3.43',
'2.98',
'3.17',
'3.90',
'2.72',
'3.38',
'3.15',
'3.42',
'3.22',
'3.49',
'3.38',
'3.11',
'2.82',
'3.88',
'2.70',
'1.60',
'3.16',
nan,
'2.73',
'3.61',
'2.00',
'3.54',
'3.26',
'3.12',
'3.09',
'3.56',
'3.53',
'3.73',
'2.84',
'3.20',
'2.63',
'2.54',
'3.15',
'3.57',
'3.31',
'2.82',
'2.63',
'3.59',
'3.06',
'2.58',
'3.97',
'2.63',
'3.02',
'2.65',
'3.36',
'3.25',
'3.30',
'3.42',
'3.77',
'2.86',
'3.51',
'3.80',
'2.40',
nan,
'3.22',
'3.32',
'2.63',
'3.14',
'3.30',
'3.65',
'3.38',
'2.52',
'2.84',
'3.49',
'3.48',
'3.15',
'3.43',
'3.02',
'2.64',
'3.71',
'3.48',
nan,
'3.72',
'3.74',
nan,
'3.54',
'2.58',
'2.14',
'3.41',
'3.91',
'3.50',
'2.90',
'3.65',
'2.92',
'3.16',
'3.17',
'3.85',
nan,
'3.03',
'3.51',
'3.47',
'3.57',
'2.84',
'3.94',
nan,
'3.01',
'2.57',
nan,
'4.00',
'3.06',
'3.38',
'3.27',
nan,
'3.09',
'3.89',
'3.88',
'2.88',
'3.73',
'3.38',
'2.83',
'3.61',
'3.83',
'3.58',
'2.93',
'3.63',
'3.04',
'2.96',
'3.56',
'2.99',
'3.26',
'3.81',
nan,
'2.97',
'3.61',
'3.78',
'3.11',
'2.82',
'3.68',
'2.76',
'3.01',
'2.94',
'3.58',
'3.24',
'2.65',
'3.70',
'3.23',
'2.98',
'3.03',
'3.10',
'3.11',
'2.18',
'3.61',
'3.06',
'3.60',
'3.78',
'3.33',
'3.78',
'2.58',
'2.52',
'3.81',
nan,
'3.34',
'2.73',
'3.01',
'3.70',
'3.28',
'2.91',
'2.91',
'3.30',
'3.12',
'3.39',
'2.76',
'2.49',
'3.45',
nan,
'2.98',
'3.49',
'2.85',
'3.10',
'3.54',
'3.26',
nan,
'3.30',
nan,
'3.59',
'2.41',
'3.07',
'2.74',
'3.60',
'3.15',
'3.44',
'3.31',
'1.68',
'3.39',
'3.85',
'3.17',
'3.11',
'3.07',
'3.05',
'3.34',
'3.50',
'2.81',
'3.36',
'3.73',
'3.49',
'3.61',
'3.43',
'3.43',
'3.23',
'3.05',
'2.65',
'3.58',
'3.32',
'3.32',
'2.82',
'3.42',
'2.97',
'3.08',
'3.51',
'3.69',
'3.53',
nan,
'3.12',
'3.34',
'3.13',
'3.51',
'3.74',
'3.49',
'2.74',
'3.67',
'3.57',
'2.22',
'3.01',
'3.50',
'2.72',
'3.63',
'3.38',
'3.65',
'3.07',
'3.63',
'3.01',
'3.69',
'3.61',
nan,
'3.09',
'3.15',
'2.73',
'3.04',
'3.72',
'2.89',
'2.97',
'2.82',
'3.23',
'3.22',
'3.40',
'3.23',
'4.00',
'2.73',
'2.82',
'2.92',
'2.95',
'3.14',
'3.14',
'3.58',
'3.25',
'3.14',
'3.39',
'3.50',
'3.92',
'3.09',
'3.65',
'3.75',
'2.99',
'2.72',
'3.36',
'3.90',
'3.48',
'3.27',
'3.61',
nan,
'2.65',
'3.48',
nan,
'3.12',
'2.92',
'3.45',
'2.31',
'3.25',
'3.73',
'3.50',
'3.54',
nan,
'3.36',
'3.64',
'2.50',
'3.40',
'3.64',
'2.55',
'3.44',
'2.81',
'2.49',
'3.07',
'2.70',
nan,
'3.37',
'0.00',
nan,
'3.65',
'3.24',
'3.07',
'2.74',
'3.63',
'3.27',
'3.13',
'3.66',
'2.97',
'3.49',
'3.06',
'2.70',
'2.83',
'3.75',
'3.21',
'3.95',
'3.06',
'3.54',
'3.67',
'3.66',
'2.57',
'3.30',
'2.78',
'2.59',
'2.72',
'3.25',
'3.97',
'3.16',
'2.43',
'3.43',
'3.35',
'3.16',
'3.80',
'2.42',
'3.49',
'3.41',
'3.26',
'2.80',
nan,
'3.13',
'3.62',
'3.09',
'3.60',
'3.56',
'3.36',
'2.99',
'2.42',
'3.58',
'3.37',
'2.05',
'3.18',
'3.64',
'3.13',
'3.01',
'3.44',
'3.14',
'3.39',
nan,
'3.27',
'1.19',
'3.43',
'3.67',
nan,
'2.10',
'3.23',
'2.88',
'3.11',
'2.32',
'3.07',
'3.15',
'3.38',
'3.52',
'2.65',
'3.46',
'3.12',
'3.66',
'3.62',
'3.17',
'3.44',
'3.85',
'3.14',
'2.56',
'3.35',
'3.80',
'3.06',
'3.31',
'3.61',
'3.46',
'2.75',
'3.52',
'2.82',
'3.11',
'2.64',
'3.68',
'3.31',
'3.47',
'3.01',
'3.12',
'3.33',
'2.61',
'3.47',
'3.10',
'3.57',
'3.73',
'2.84',
'3.58',
'3.53',
'3.01',
'3.07',
'3.57',
'3.17',
'2.95',
'3.38',
'2.65',
'2.81',
'3.46',
'3.24',
'3.51',
'3.25',
nan,
'3.33',
'3.71',
'2.70',
'2.66',
'3.35',
'3.06',
 '3.30',
 '3.60',
 '3.04',
 nan,
 '3.19',
 '3.95',
 '2.81',
 '3.02',
 '3.60',
 '3.02',
 '2.28',
 '3.83',
 '2.90',
 '2.81',
 '2.86',
 '2.85',
 '3.28',
 '3.17',
 '3.08',
 '3.18',
 '3.50',
 '2.94',
 '3.51',
 '3.59',
 '3.37',
 '3.33',
 '2.88',
 '3.54',
 '3.52',
 nan,
 '3.58',
 '3.53',
 '3.61',
 '3.67',
 '2.59',
 …]
```

[208]:

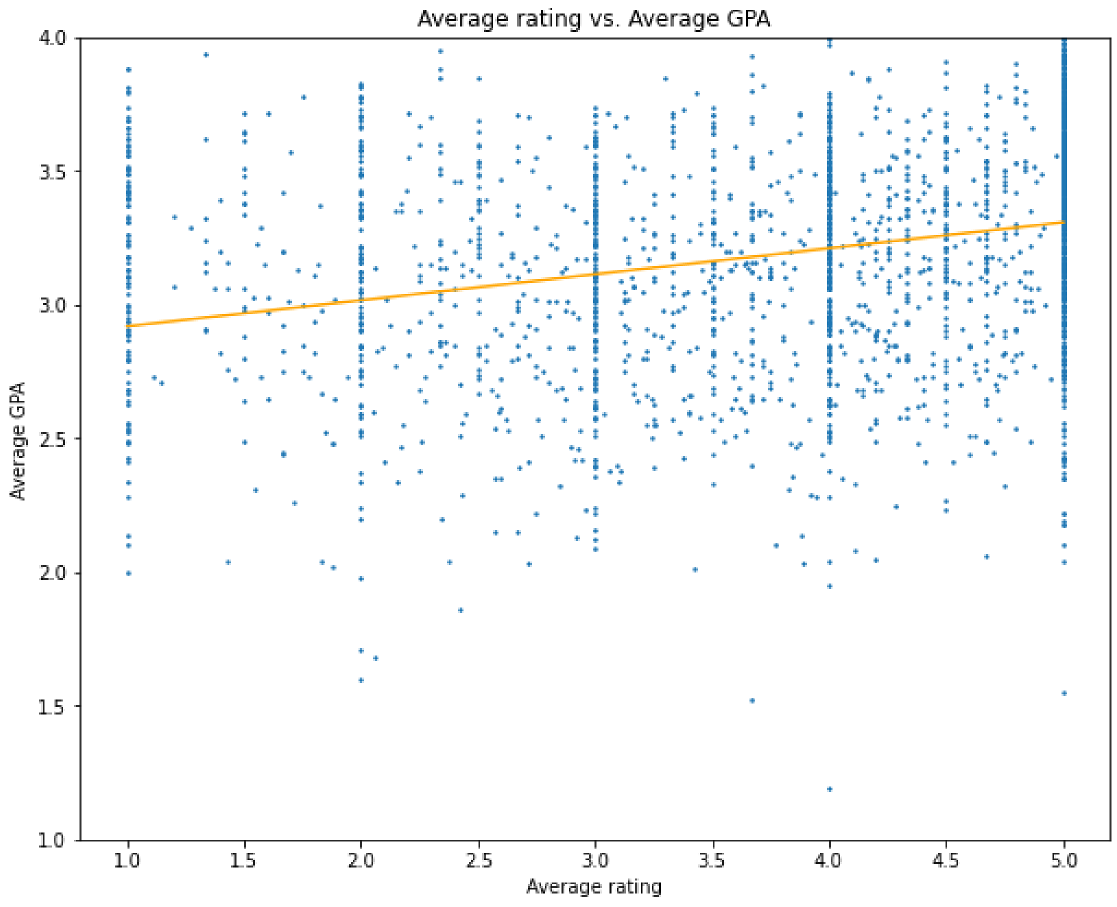
```
professor_df["average_gpa"] = matches
old_professor_df = professor_df
professor_df = professor_df[professor_df["average_gpa"].notna()] # Drop anyone␣
↪who doesn't have an average gpa
professor_df
```

[208]:

```
             name          slug       type  \
1       A Anthony       anthony  professor
2    A Kruglanski    kruglanski  professor
4        A Sharma      sharma_a  professor
```

```
6     A.U. Shankar   shankar_a.u.  professor
8      Aaron Allen    allen_aaron  professor
..             …             …        …
4    Zhengguo Xiao  xiao_zhengguo  professor
20    Zhongchi Liu   liu_zhongchi  professor
29      Zita Nunes     nunes_zita  professor
32  Zohreh Davoudi        davoudi  professor
34    Zsuzsa Daczo          daczo  professor

                                                  courses  average_rating  \
1                            [AMST202, AMST203, AMST101]            1.00
2   [PSYC743, PSYC748M, PSYC489H, PSYC489T, PSYC78…           2.00
4                                              [ASTR300]            1.80
6       [CMSC412, CMSC414, CMSC712, CMSC216, CMSC798]            2.10
8                                     [HHUM205, HHUM206]            5.00
..                                                    …              …
4   [ANSC460, ANSC214, GEMS296, ANSC212, GEMS297, …           4.00
20             [BSCI378H, BSCI410, CBMG699Y, MOCB899]            4.50
29  [ENGL234, ENGL749B, ENGL448B, ENGL300, ENGL301…           3.75
32                           [PHYS604, PHYS411, PHYS624]            2.50
34                 [SOCY105, SOCY227, SOCY325, WMST325]            4.50

   average_gpa
1         2.48
2         3.49
4         2.82
6         2.41
8         3.61
..          …
4         2.95
20        3.37
29        2.75
32        3.38
34        3.62

[2363 rows x 6 columns]
```

[209]:
```python
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['figure.figsize'] = [10, 8]
average_rating = [float(x) for x in professor_df["average_rating"]]
average_gpa = [float(x) for x in professor_df["average_gpa"]]
# print(average_gpa)
plt.scatter(x=average_rating, y=average_gpa, s=2)
plt.title("Average rating vs. Average GPA")
plt.xlabel("Average rating")
plt.ylabel("Average GPA")
plt.ylim([1,4])
plt.plot(np.unique(average_rating), np.poly1d(np.polyfit(average_rating,␣
↪average_gpa, 1))(np.unique(average_rating)), c = "orange")
plt.show()
```



Due to the volume of data, I've made each point rather small so that we can better distinguish between the different data points. As expected, there are a lot of points concentrated on ratings that are whole numbers (1.0, 2.0, 3.0, 4.0, 5.0). This is likely because not every professor has multiple reviews and even the ones who do may have just a few reviews that are all of the same rating. However, despite the volume of data points, we can see that there seems to be a large density of points around the 5.0 rating and they seem to be skewed towards high GPA. Furthermore, the line of best fit seems to show that there is some positive relationship between the two measures. Let's calculate the R squared value to be sure.

[210]:
```
import statsmodels.formula.api as smf
res = smf.ols(formula="average_rating ~ average_gpa", data=professor_df).fit()
print('R squared: {}'.format(res.rsquared))
```

```
R squared: 0.15051243926885027
```

While an R squared value of 0.15 is not huge, it does seem appropriate given what we are measuring. There are a lot of factors that students take into consideration when rating a professor, such as how organized the class was, the professor's lecture style, how good the TA's were, etc. Furthermore, even if a student receives a low grade, it could be the case that they received that grade just because the class was difficult, and it had nothing to do with the quality of the professor. When you consider the multitude of factors that influence the reviews students give, anything more than a modest relationship between the professor's rating and their average GPA would seem suspicious.

## 3 Another Approach

There may be a better, more direct way to go about measuring this relationship. On PlanetTerp, when you leave a review, there is an option to leave what grade you are expecting in the class. Instead of just measuring the average GPA of a professor across all the courses that he/she teaches, we could instead look at the individual reviews for each professor and see if there is a relationship between the grade that the reviewer is expecting to get and the rating they give the professor.

In the code below, I've marked down the grades as A, B, C, D, F, Pass (P), Withdrawn (W), and Not Reported (NR). The expectation is that those who received high grades (A's and B's) are likely going to leave higher ratings than those who recieved low grades. We also expect that people who withdrew from the class, passed the class (which was likely a lot of people during the COVID semesters), or simply didn't report the grade they got will give lower ratings to the professors.

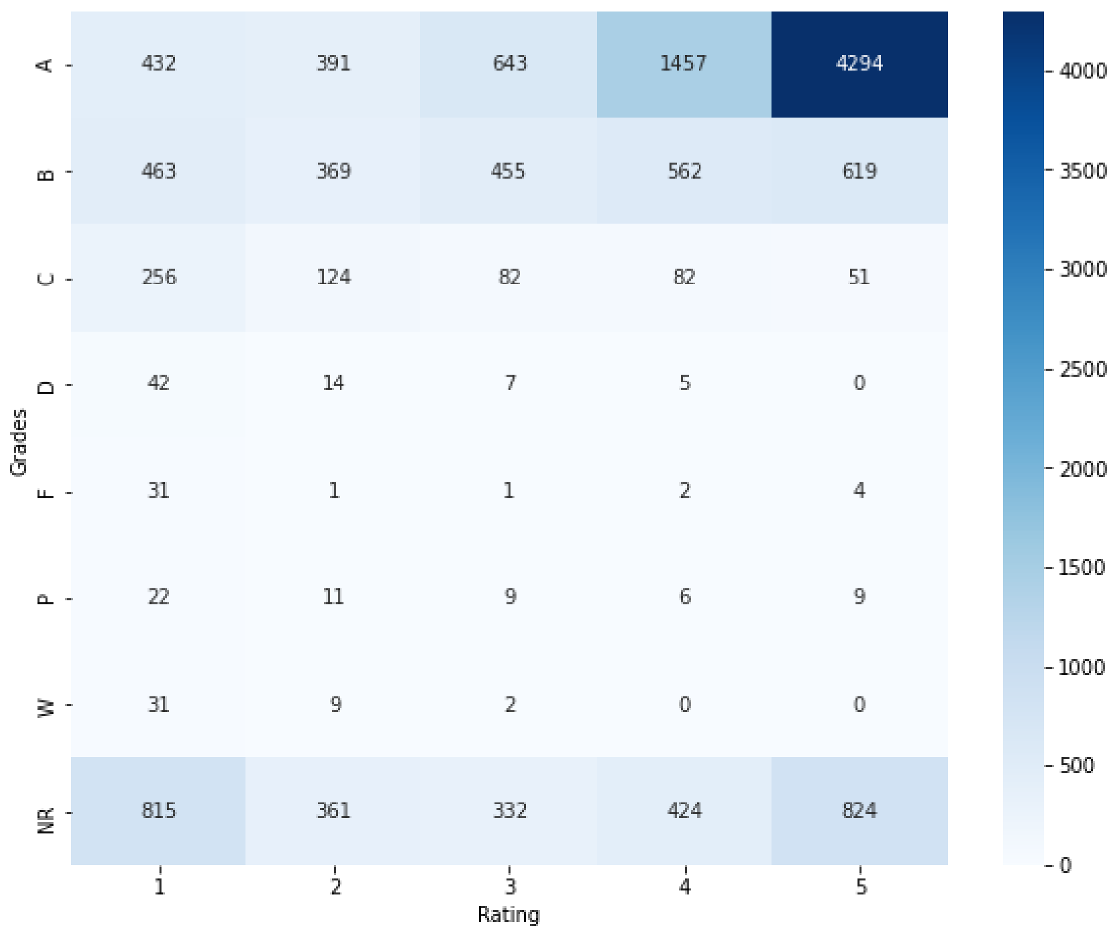
This next block of code takes a long time to run, so you might want to go for a walk or put on a TV show if you are going to try and run this snippet yourself.

[211]:
```
grades = {
    "A": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "B": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "C": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "D": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "F": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "P": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "W": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0},
    "NR": {"1": 0, "2": 0, "3": 0, "4": 0, "5": 0}
}
g = ["A", "B", "C", "D", "F", "P", "W"]
for idx, row in professor_df.iterrows():
    # print(idx)
    reviews = requests.get(f"https://api.planetterp.com/v1/professor?
 ↪name={row['name']}&reviews=true").json()
    if reviews is None or "reviews" not in reviews:
        # print(reviews)
        continue
    reviews = reviews["reviews"]
    for rev in reviews:
        if rev["expected_grade"] == "" or rev["expected_grade"] is None:
            # print(rev["course"])
            grades["NR"][str(rev["rating"])] += 1
            continue
        grade = rev["expected_grade"][0].upper()
        if grade not in g:
            # print(f"Continuing, {grade}")
            continue
        # print(grade)
        grades[grade][str(rev["rating"])] += 1
print(grades)
```

```
{'A': {'1': 432, '2': 391, '3': 643, '4': 1457, '5': 4294}, 'B': {'1': 463, '2':
369, '3': 455, '4': 562, '5': 619}, 'C': {'1': 256, '2': 124, '3': 82, '4': 82,
'5': 51}, 'D': {'1': 42, '2': 14, '3': 7, '4': 5, '5': 0}, 'F': {'1': 31, '2':
1, '3': 1, '4': 2, '5': 4}, 'P': {'1': 22, '2': 11, '3': 9, '4': 6, '5': 9},
'W': {'1': 31, '2': 9, '3': 2, '4': 0, '5': 0}, 'NR': {'1': 815, '2': 361, '3':
332, '4': 424, '5': 824}}
```

[212]:
```
data = [np.array(list(d.values())) for d in grades.values()]
import seaborn as sb
sb.heatmap(data, xticklabels="12345", yticklabels=["A", "B", "C", "D", "F",␣
 ↪"P", "W", "NR"], cmap="Blues", annot=True, fmt="g");
plt.xlabel("Rating")
plt.ylabel("Grades")
```

[212]:
```
Text(69.0, 0.5, 'Grades')
```



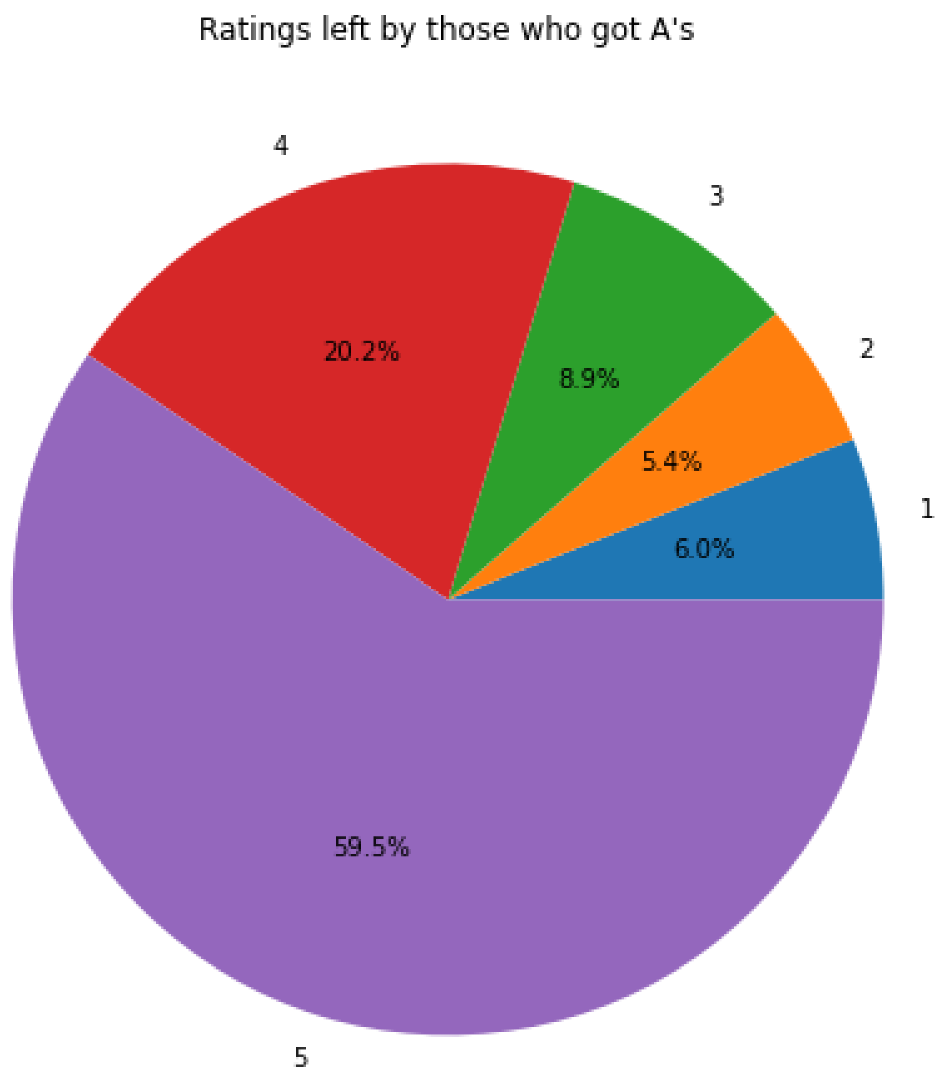
Wow. That certainly tells us something about the kinds of people who are leaving reviews on PlanetTerp. It's not even a contest. People who gave their professor 5-star ratings and also got A's in the class make up the grand majority of the reviews on the site. The group that comes in second place (people who got A's and left 4-star reviews) has less than half the number of people in this majority group. All other groups look absolutely tiny by comparison, as evidenced by the fact that they are all colored a very pale blue.

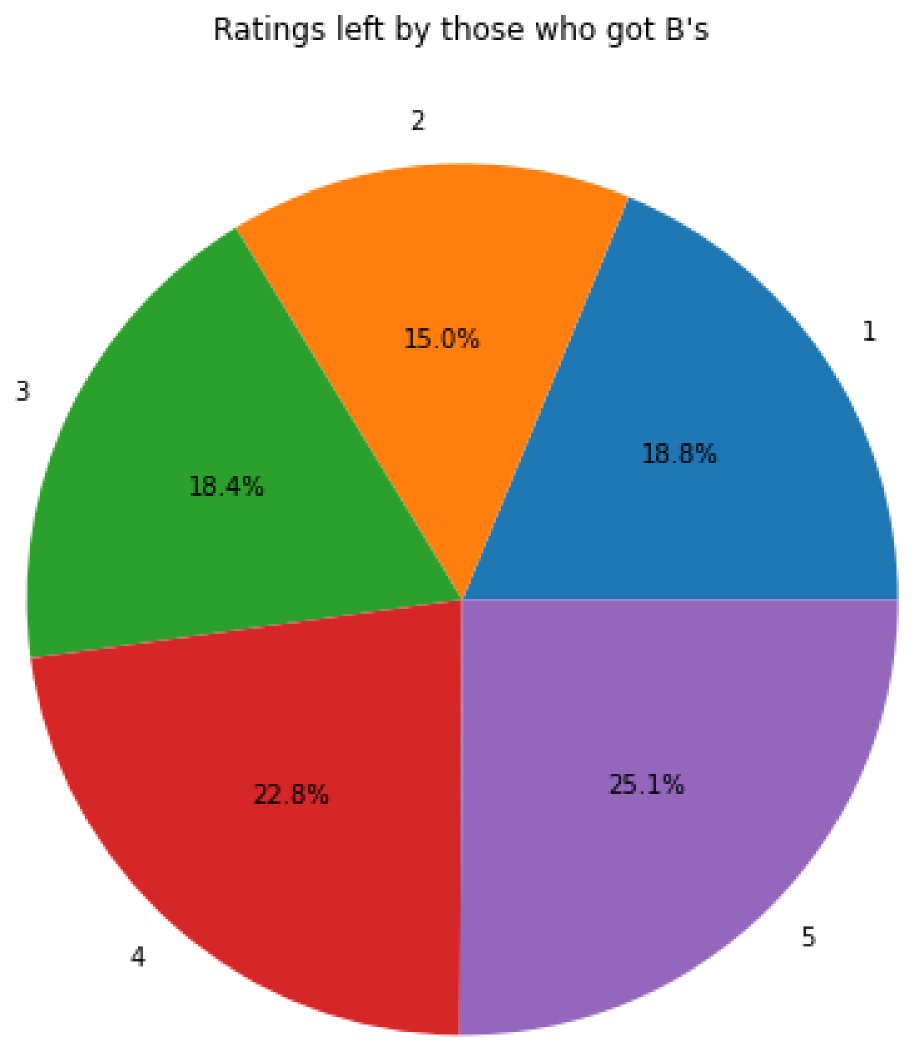
It is worth noting that all of the trends that we anticipated seem to be born out in this heat map. If we look at each of the different grade categories, we can see that people who got A's are overwhelmingly more likely to leave high reviews. The people who got B's, on the other hand, seem to have a more mixed relationship with a pretty even spread among the different ratings. People who got C's or lower were skewed in the opposite direction as those who got A's, with more people leaving low ratings. This is also true of those who passed or withdrew from the class.

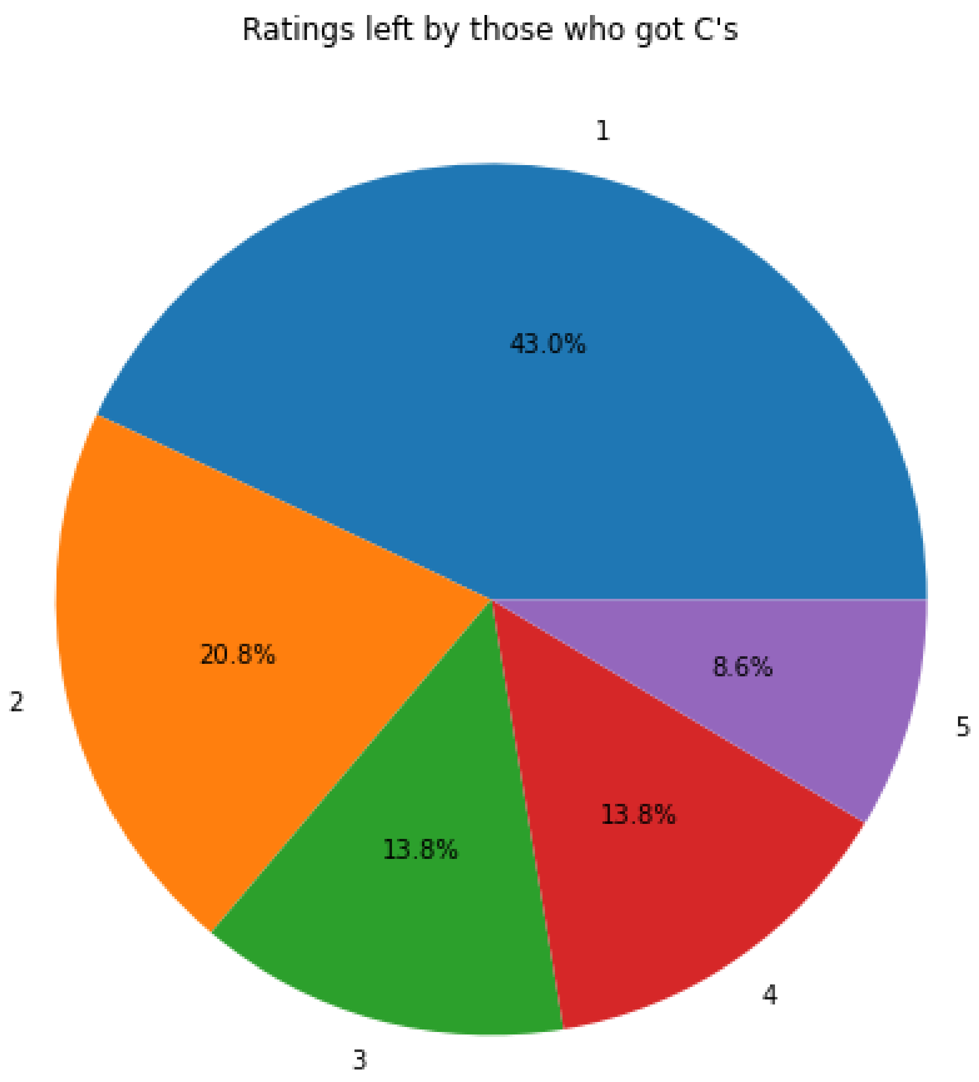
It is also interesting to observe that there doesn't seem to be a clear relationship between not reporting one's grade and the rating. In the group of people who didn't report their grades, we see that about the same number of people of people left 1-star and 5-star ratings, with the rest pretty evenly spread among 2 to 4-star ratings.

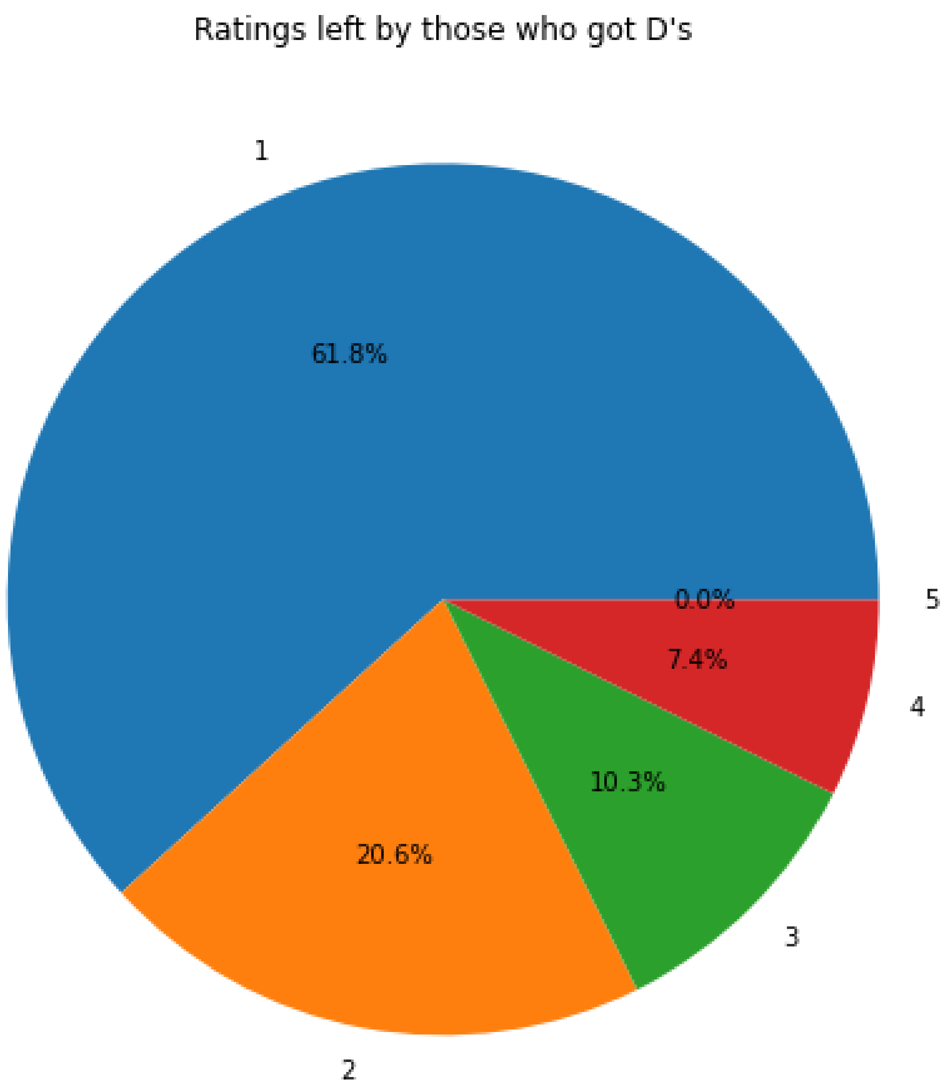
You might find the pie charts below to be helpful in seeing these proportions among the different grade categories more clearly.

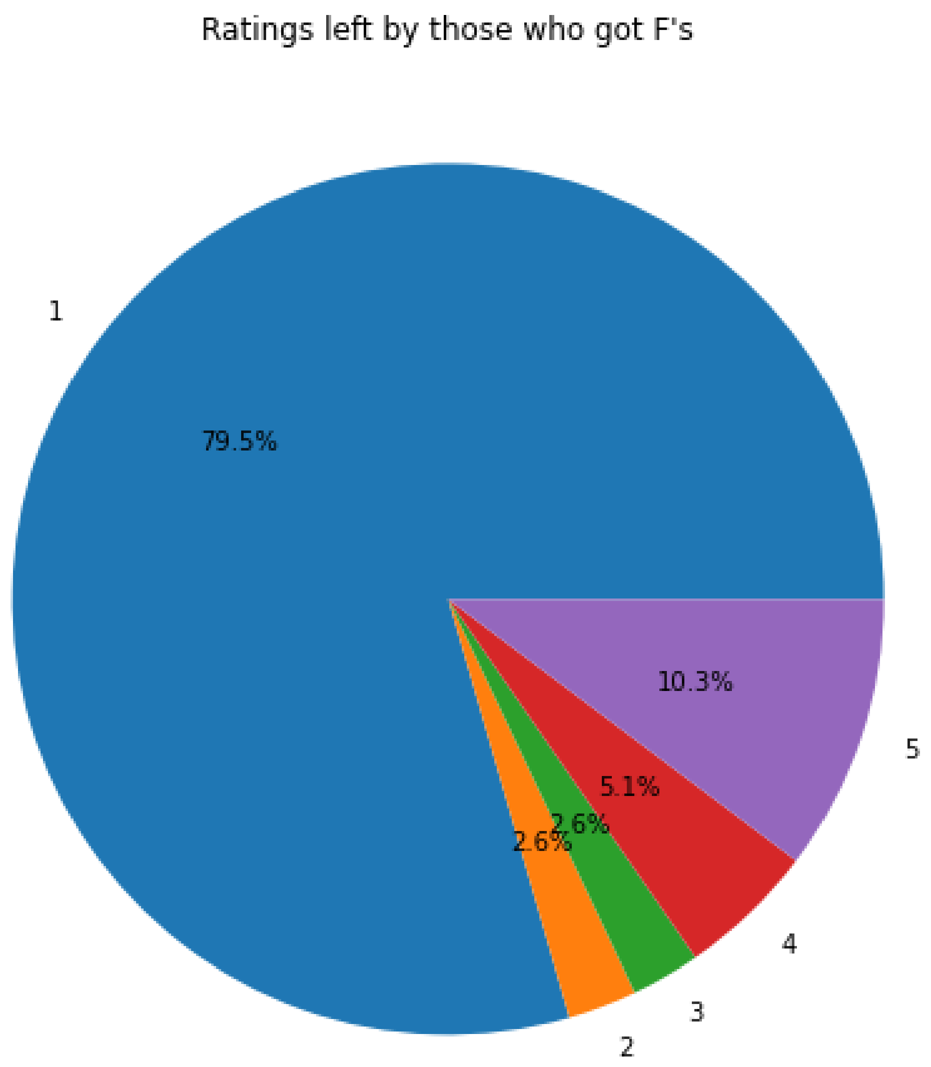
[213]:
```
def get_title(grade):
    switcher={
        "A": "those who got A's",
        "B": "those who got B's",
        "C": "those who got C's",
        "D": "those who got D's",
        "F": "those who got F's",
        "P": "those who passed",
        "W": "those who withdrew",
        "NR": "those who didn't report their grade"
    }
    return switcher.get(grade)

for grade in grades.keys():
    title = f"Ratings left by {get_title(grade)}"
    ratings = grades[grade]
    rate_vals = []
    for rating in ratings.values():
        rate_vals.append(rating)
    labels = ["1", "2", "3", "4", "5"]
    plt.pie(rate_vals,labels=labels, autopct='%1.1f%%')
    plt.title(title)
    plt.show()
```

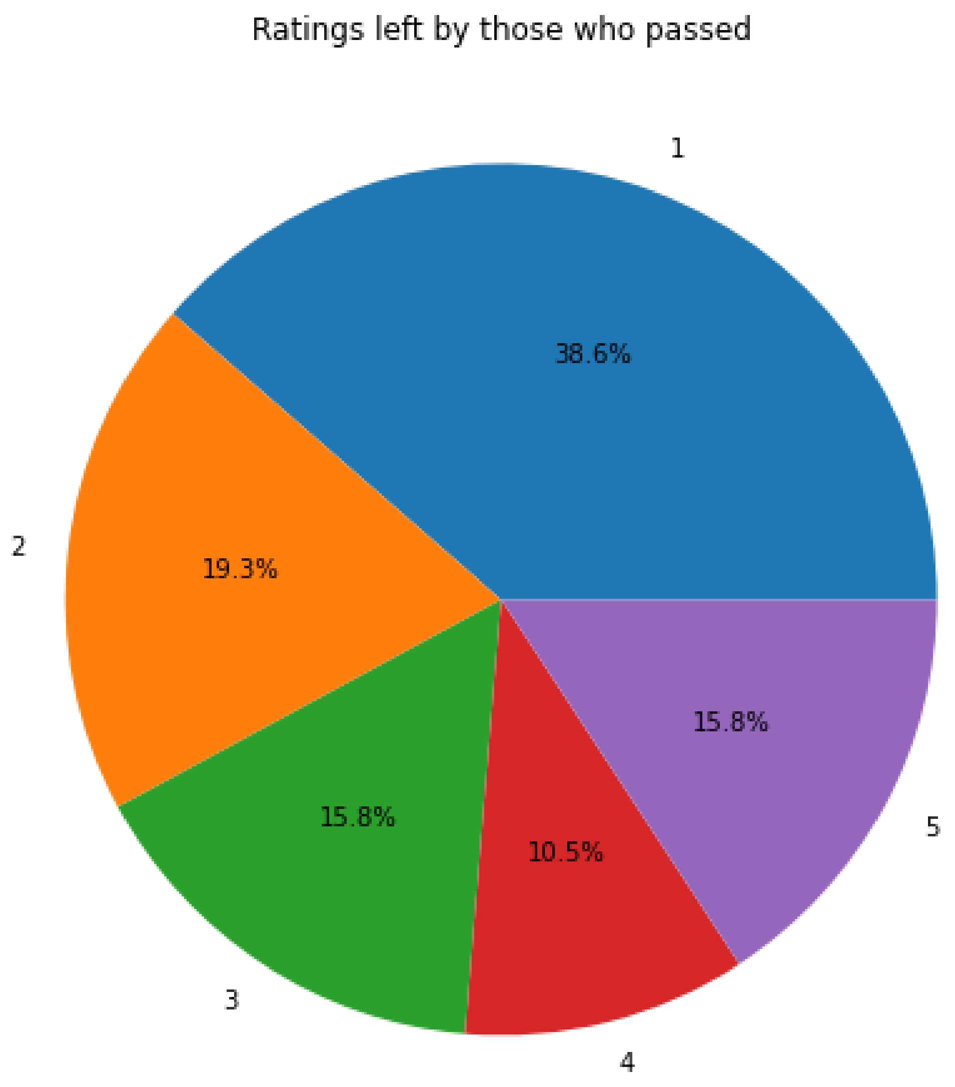

Ratings left by those who got A's
1
6.0%
2
5.4%
3
8.9%
4
20.2%
5
59.5%

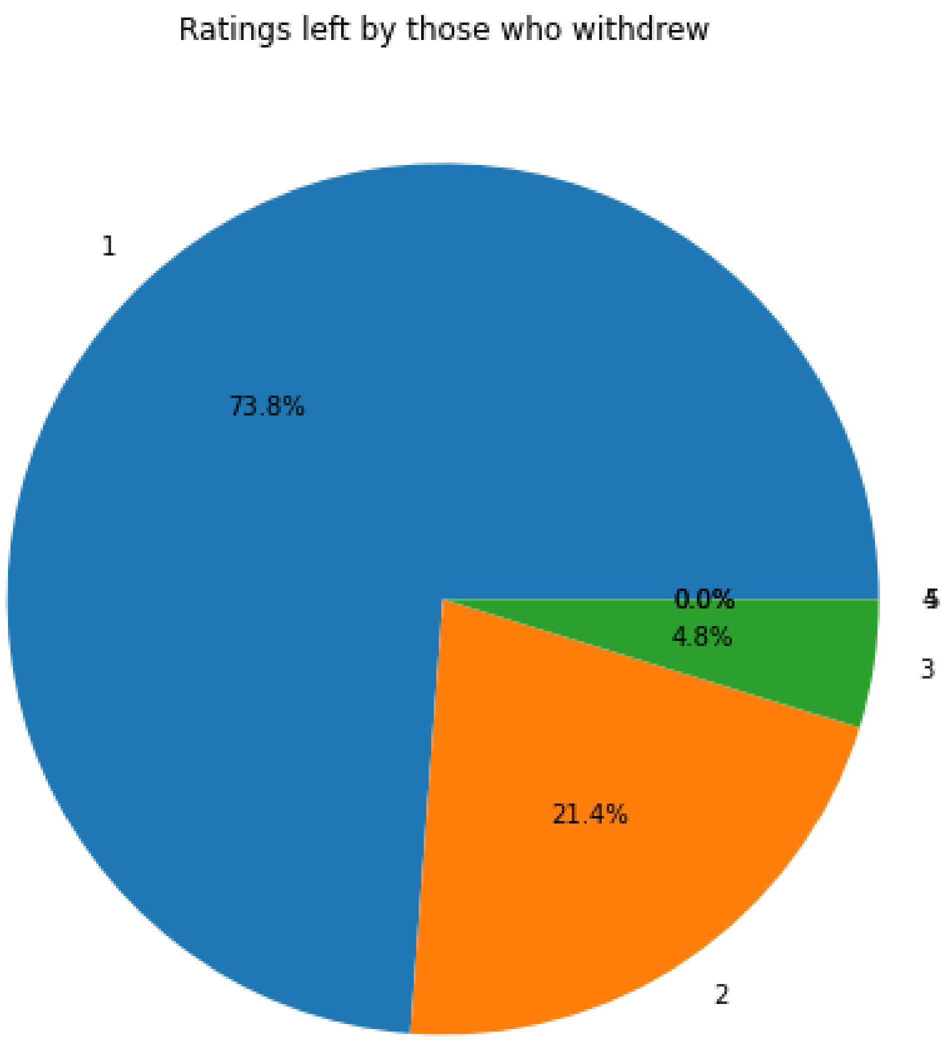

Ratings left by those who got B's
1
18.8%
2
15.0%
3
18.4%
4
22.8%
5
25.1%

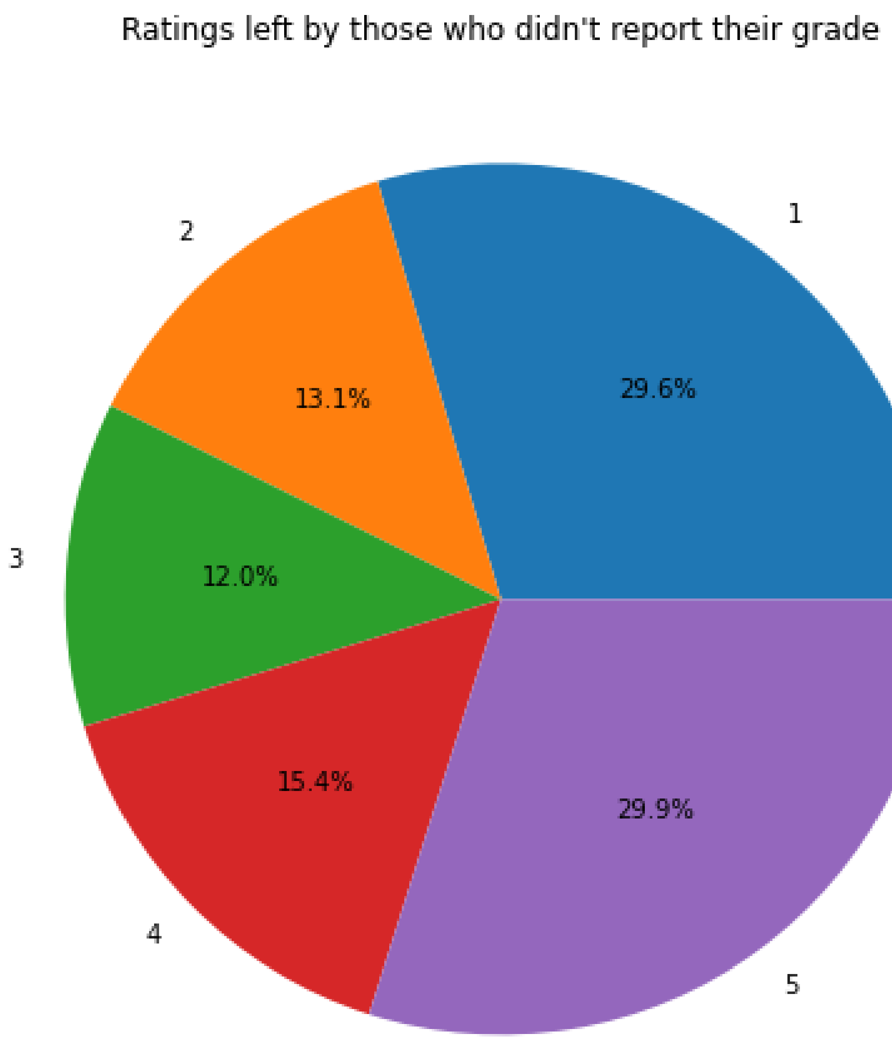

Ratings left by those who got C's
1
43.0%
2
20.8%
3
13.8%
4
13.8%
5
8.6%


Ratings left by those who got D's
1
61.8%
2
20.6%
3
10.3%
4
7.4%
5
0.0%


Ratings left by those who got F's
1
79.5%
2
2.6%
3
2.6%
4
5.1%
5
10.3%


Ratings left by those who passed
1
38.6%
2
19.3%
3
15.8%
4
10.5%
5
15.8%


Ratings left by those who withdrew
1
73.8%
2
21.4%
3
4.8%
4
5
0.0%



## 4 A new measure?

Given our analysis so far, it seems fair to conclude that people who leave reviews on PlanetTerp are more likely to be people who received high grades in the courses they are reviewing. This presents a problem for those who are relying on PlanetTerp reviews and rating data to determine the quality of a professor, as these ratings might be biased towards professors who simply teach easy classes or grade easily. In this last part of the analysis, let's try and see if we can devise a way to measure the quality of a professor that doesn't suffer from the issues we face with the current information. I think that it's fair to say that if a student receives a high grade in a class, but rates the professor very low, then that indicates that the professor might not be very good. Similarly, if a student receives a low grade, but still rates the professor very highly, that would be a strong indication that the professor is very good. Obviously, it's not perfect. For example, some professors might still be very bad, but have cool personalities that cause students to still leave high ratings and vice versa. However, analyzing the review data in this way might yield some interesting results.

I'm going to only create this score for professors that have more than 20 ratings, as it likely wouldn't be very meaningful otherwise.

[214]:
```
def get_number_ratings(prof_df):
    ratings_count = []
    for idx, row in prof_df.iterrows():
        print(idx)
        reviews = requests.get(f"https://api.planetterp.com/v1/professor?
↪name={row['name']}&reviews=true").json()
        if reviews is None or "reviews" not in reviews:
            ratings_count.append(float("NaN"))
            continue
        reviews = reviews["reviews"]
        ratings_count.append(len(reviews))
    return ratings_count

num_ratings = get_number_ratings(professor_df)
professor_df["num_ratings"] = num_ratings
professor_df
```

```
1
2
4
6
8
10
15
19
27
42
45
47
53
58
61
64
68
74
76
79
92
97
104
105
```

```
113
115
120
124
129
132
136
138
139
140
146
173
176
178
181
183
184
187
194
198
200
205
206
214
215
216
223
224
227
229
236
240
241
242
245
247
249
251
252
276
279
282
291
301
305
307
319
329
```

```
331
334
341
344
349
350
351
357
362
365
368
369
371
372
376
377
378
381
399
413
417
426
428
432
433
438
453
465
467
472
478
482
485
489
491
492
495
503
504
506
511
512
518
525
529
536
537
550
```

```
833
842
848
849
850
872
873
890
892
894
902
914
915
916
917
922
931
932
934
942
960
961
967
968
988
989
998
2
3
5
11
24
25
26
31
33
35
36
37
38
40
42
45
46
49
56
63
70
```

```
71
76
77
79
83
88
90
102
105
110
111
112
114
116
125
128
129
140
142
151
155
156
159
168
176
182
183
186
201
203
210
213
214
218
221
224
232
234
248
252
256
259
263
265
285
289
295
296
```

301
307
308
309
310
322
325
332
336
340
343
374
398
405
407
408
416
424
425
428
442
444
448
457
462
476
490
501
519
531
534
539
542
543
548
555
566
569
578
579
581
587
590
597
600
607
609
618

```
859
863
865
875
878
880
894
899
902
904
907
908
914
918
919
924
930
934
936
938
946
951
955
961
964
967
979
985
986
993
6
10
11
12
17
21
31
37
47
51
52
58
59
63
67
74
78
98
```

```
111
121
126
127
128
132
137
141
148
152
171
176
185
186
195
196
197
200
206
209
218
220
226
229
236
239
247
248
257
264
286
287
288
289
295
297
301
306
319
326
329
342
345
356
359
363
366
369
```

768
778
780
781
783
788
800
801
803
804
810
820
827
836
838
845
847
849
859
860
863
864
868
870
874
883
886
887
889
891
893
896
903
909
921
926
933
935
939
940
943
944
947
948
957
958
962
966

```
967
974
976
977
978
980
986
992
997
13
15
19
21
26
28
32
33
42
51
55
59
62
65
68
70
73
79
84
88
91
92
103
105
123
132
137
150
152
154
162
167
169
171
179
183
184
186
190
```

192
193
195
196
197
198
200
203
209
214
221
237
244
245
254
256
277
280
281
285
290
303
306
311
317
320
323
324
327
329
347
349
355
358
360
365
377
379
390
397
399
406
407
408
409
412
425
430

438
441
447
449
469
477
479
483
484
486
487
489
490
493
494
495
503
516
521
531
538
540
543
544
545
546
548
555
558
559
563
571
573
575
578
579
580
584
592
595
597
604
609
614
617
624
630
631

```
833
852
853
866
870
873
880
882
888
896
897
901
906
909
910
911
914
921
923
928
933
943
947
949
951
952
954
959
961
962
964
979
981
985
990
9
10
14
20
21
25
27
29
30
33
35
40
41
```

```
42
66
67
78
79
80
85
92
97
98
103
113
130
131
134
138
141
147
173
178
181
184
189
198
205
206
214
216
219
221
225
232
235
237
239
240
241
245
246
251
254
256
267
277
278
282
284
286
```

293
296
298
301
302
305
310
326
334
338
339
345
347
348
351
352
360
364
367
368
377
379
385
394
400
401
408
409
415
417
418
429
434
437
439
443
450
452
453
464
465
467
468
471
481
484
487
492

```
768
784
786
788
793
796
799
802
803
804
805
813
814
817
820
822
829
836
838
843
850
851
854
857
860
862
863
867
874
875
876
877
882
884
885
888
890
892
900
907
914
916
934
938
945
946
950
956
```

```
960
961
966
967
971
976
982
987
996
999
1
10
12
17
22
24
25
32
35
38
46
49
50
52
62
67
74
77
78
80
84
89
91
101
103
107
117
120
121
130
131
132
137
140
142
143
144
159
```

```
168
169
178
183
186
200
201
202
204
208
211
220
224
231
233
234
235
236
243
245
250
251
257
258
259
263
264
267
269
275
302
310
314
320
323
327
329
330
331
335
338
345
354
359
360
363
365
371
```

375
377
380
381
383
384
385
386
391
394
411
414
416
418
423
424
426
428
434
437
443
447
449
450
452
455
459
469
477
487
491
496
497
505
506
513
524
529
531
546
547
560
569
571
581
592
616
620

```
855
869
874
876
881
888
898
909
913
934
935
938
941
944
945
946
956
958
970
972
980
994
995
999
2
5
20
23
28
30
31
46
51
52
53
54
62
66
68
69
71
77
79
81
83
84
86
91
```

103
105
106
109
110
115
129
132
137
140
155
161
173
176
178
185
190
200
205
207
209
213
219
236
238
239
241
242
243
248
249
253
262
266
270
271
274
276
288
290
298
301
307
308
309
311
313
314

315
324
330
331
332
336
337
345
355
359
364
370
375
376
378
386
393
403
417
419
424
425
430
450
462
476
479
485
490
495
497
504
521
525
533
537
541
551
553
557
564
565
583
588
593
604
607
610

611
629
631
634
635
636
638
645
649
657
661
663
668
677
678
683
699
700
701
704
707
709
715
721
723
729
740
764
775
788
789
790
795
804
815
818
820
822
828
830
832
834
835
837
838
842
856
859

```
860
871
875
878
884
890
899
904
910
912
913
914
916
917
921
928
940
944
949
950
958
962
964
965
980
990
992
996
999
8
10
14
17
21
23
26
35
46
50
59
65
71
75
79
80
82
91
92
```

94
101
104
109
116
122
123
126
128
133
136
137
142
154
155
157
158
164
175
179
193
194
205
208
212
225
227
232
235
236
237
241
242
257
258
269
272
277
278
279
281
283
284
286
296
316
317
319

321
322
330
335
340
342
345
348
352
369
371
373
377
381
387
391
392
395
396
401
403
404
410
411
413
420
424
428
434
446
458
466
475
478
483
494
500
502
504
505
509
510
511
512
515
520
522
526

528
532
536
537
541
543
545
546
550
551
552
557
579
585
588
590
596
597
599
600
602
609
611
615
624
625
628
631
640
646
647
652
664
678
680
683
687
691
694
699
702
707
712
714
716
718
722
746

```
752
758
759
768
776
778
779
784
799
804
809
819
821
822
827
832
836
842
846
847
850
853
861
864
870
878
885
893
896
907
915
917
930
938
941
953
955
977
978
979
6
16
18
19
21
23
28
36
```

```
38
40
43
50
72
80
83
85
91
97
98
99
100
101
102
103
104
107
115
118
123
125
132
133
136
144
151
153
157
158
162
165
166
168
169
171
174
175
177
182
183
186
189
192
193
200
203
216
```

218
222
225
226
231
232
234
236
248
256
261
262
264
267
281
282
283
284
285
286
287
297
300
301
304
308
310
313
315
316
323
325
327
331
332
344
346
347
349
352
354
355
360
370
371
373
374
375

376
386
388
390
394
395
400
401
402
411
412
413
421
426
430
431
435
439
445
456
459
467
476
479
481
482
488
489
492
494
495
496
501
507
511
519
521
526
530
531
532
536
538
540
543
546
553
558

560
579
582
585
586
590
592
595
596
598
599
622
632
634
646
649
650
654
655
662
666
669
673
675
686
689
691
694
696
697
698
700
702
705
707
708
709
711
712
714
715
718
722
725
729
730
735
737

742
748
754
763
764
767
768
774
776
777
781
782
783
784
787
790
792
799
804
808
812
824
825
829
835
840
841
853
856
860
863
873
874
878
885
886
892
907
909
910
914
919
925
926
929
936
937
939

```
943
945
953
957
964
967
972
976
979
983
994
998
1
9
10
13
14
18
22
30
31
34
39
42
43
50
51
57
59
65
67
70
71
81
83
85
92
94
101
107
113
128
133
139
142
143
144
146
```

150
151
152
155
159
160
161
162
163
169
172
178
180
183
189
191
192
197
198
199
205
208
216
222
228
230
234
238
242
248
253
254
255
257
260
261
265
268
279
282
284
285
288
298
302
303
304
323

325
328
331
332
335
343
344
346
347
353
355
361
374
381
392
402
404
407
410
415
420
421
438
446
450
452
453
457
460
463
470
473
480
488
494
499
501
502
512
513
522
527
529
532
533
534
538
544

551
557
563
566
578
579
584
587
595
605
606
610
612
619
620
631
635
639
644
647
649
656
657
660
668
693
696
706
717
724
728
729
733
734
737
751
752
772
780
785
786
788
789
798
809
810
813
815

```
816
818
822
823
831
834
836
837
839
841
847
848
851
859
864
868
870
871
879
890
891
892
907
911
916
917
924
930
939
941
948
949
950
955
960
961
962
965
966
975
981
983
988
990
991
994
5
7
```

```
8
13
21
27
31
42
61
65
67
72
74
79
82
84
96
98
110
116
118
119
120
121
122
127
135
142
146
149
155
157
160
170
178
179
180
182
184
187
190
192
193
194
195
196
199
201
202
203
```

205
206
218
223
226
239
249
256
261
265
267
272
274
279
281
290
292
301
302
303
304
305
308
312
321
326
332
333
334
338
346
347
349
350
358
360
366
379
380
391
393
396
398
400
405
410
411
415

```
424
426
441
445
449
451
457
462
464
468
471
475
482
483
509
511
513
514
519
521
534
537
538
540
544
564
566
568
571
577
579
586
587
589
592
596
603
605
607
616
617
630
632
639
640
641
642
643
```

```
824
825
827
834
835
837
851
853
875
878
887
888
893
922
927
928
933
934
935
951
954
958
966
967
969
977
987
992
998
999
4
20
29
32
34
```

```
<ipython-input-214-ac9bd26733b1>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  professor_df["num_ratings"] = num_ratings
```

[214]:
```
              name         slug       type  \
1        A Anthony      anthony  professor
2     A Kruglanski   kruglanski  professor
4         A Sharma     sharma_a  professor
```

```
6     A.U. Shankar   shankar_a.u.  professor
8      Aaron Allen    allen_aaron  professor
..             …              …          …
4    Zhengguo Xiao  xiao_zhengguo  professor
20    Zhongchi Liu   liu_zhongchi  professor
29      Zita Nunes     nunes_zita  professor
32  Zohreh Davoudi        davoudi  professor
34    Zsuzsa Daczo          daczo  professor

                                               courses  average_rating  \
1                            [AMST202, AMST203, AMST101]            1.00
2   [PSYC743, PSYC748M, PSYC489H, PSYC489T, PSYC78…            2.00
4                                            [ASTR300]            1.80
6       [CMSC412, CMSC414, CMSC712, CMSC216, CMSC798]            2.10
8                                   [HHUM205, HHUM206]            5.00
..                                                 …              …
4   [ANSC460, ANSC214, GEMS296, ANSC212, GEMS297, …            4.00
20              [BSCI378H, BSCI410, CBMG699Y, MOCB899]            4.50
29  [ENGL234, ENGL749B, ENGL448B, ENGL300, ENGL301…            3.75
32                           [PHYS604, PHYS411, PHYS624]            2.50
34                  [SOCY105, SOCY227, SOCY325, WMST325]            4.50

   average_gpa  num_ratings
1         2.48          1.0
2         3.49          1.0
4         2.82          5.0
6         2.41         20.0
8         3.61          1.0
..         …          …
4         2.95          1.0
20        3.37          6.0
29        2.75          4.0
32        3.38          2.0
34        3.62          4.0

[2363 rows x 7 columns]
```

[215]:
```
more_than_20 = professor_df.loc[professor_df["num_ratings"] >= 20]
more_than_20
```

[215]:
```
                    name          slug       type  \
6           A.U. Shankar  shankar_a.u.  professor
124  Adrianos Papamarcou    papamarcou  professor
331       Alice Mignerey     mignerey  professor
362          Alka Gandhi        gandhi  professor
369      Allan Yashinski     yashinski  professor
..                     …             …          …
```

```
605        Wesley Lawson    lawson_wesley  professor
649      William Higgins  higgins_william  professor
676   William McClenahan       mcclenahan  professor
727         Wiseley Wong     wong_wiseley  professor
729       Wojciech Czaja            czaja  professor

                                                  courses  average_rating  \
6        [CMSC412, CMSC414, CMSC712, CMSC216, CMSC798]          2.1000
124  [ENEE222, ENEE324, ENEE620, ENEE322, ENEE322H,…        4.2400
331  [CHEM271, CHEM705, CHEM889A, CHEM403, CHEM131,…        2.4444
362               [ECON312, ECON422, ECON321, ECON325]          3.8788
369  [MATH240, MATH401, MATH406, MATH461, MATH141H,…        4.6400
..                                                  …             …
605  [ENEE131, ENEE132, ENEE200, ENEE205, ENEE381, …        3.7027
649  [BIOL608F, BSCI440, BSCI105, BSCI207, CLFS510,…        3.4082
676  [BMGT380H, BMGT381, BMGT380, BUSI764, BMGT496,…        4.1250
727  [MATH141, MATH241H, MATH401, MATH241, MATH475,…        4.8621
729  [MATH648I, MATH141H, MATH411, MATH648Q, MATH14…        3.8929

    average_gpa  num_ratings
6          2.41         20.0
124        2.87         25.0
331        2.59         36.0
362        2.14         33.0
369        2.72         25.0
..          …            …
605        3.10         37.0
649        2.66         49.0
676        3.10         24.0
727        2.49         29.0
729        2.03         28.0

[126 rows x 7 columns]
```

[216]:
```
def get_grade_point(grade):
    switcher={
        "A": 5,
        "B": 4,
        "C": 3,
        "D": 2,
        "F": 1,
        "P": 1,
        "W": 1
    }
    return switcher.get(grade)

g = ["A", "B", "C", "D", "F", "P", "W"]
```

```
scores = []
for idx, row in more_than_20.iterrows():
    reviews = requests.get(f"https://api.planetterp.com/v1/professor?
↪name={row['name']}&reviews=true").json()
    reviews = reviews["reviews"]
    cur_score = 0
    for rev in reviews:
        if rev["expected_grade"] == "" or rev["expected_grade"] is None:
            continue
        grade = rev["expected_grade"][0].upper()
        if grade not in g:
            continue
        # print(grade)
        rating = rev["rating"]
        cur_score += rating - get_grade_point(grade) # High rating, low grade␣
↪-> high score. Low rating, high grade -> low score
    scores.append(cur_score)
print(scores)
```

```
[-35, -3, -27, -13, 6, -14, -24, 1, -16, -15, -34, -16, -6, -57, -8, -23, -30,
-40, -49, -18, -65, -32, -51, 2, -3, -8, -1, -41, -10, -33, 3, -1, -62, -6, -10,
-46, -41, -24, 0, -23, -22, -13, -8, -8, -29, -75, -26, -36, -9, -21, -7, 5,
-31, -13, -36, -27, -7, -12, -16, -47, -8, -10, 21, -5, -21, -25, -4, -5, -42,
-103, 0, -22, -29, -7, -15, -53, -11, -38, -24, -22, -28, -30, -34, 3, -21, -15,
-8, 2, -99, -14, -7, 0, -13, -10, -28, -42, -25, -7, -39, -45, -5, -19, -17, -2,
-13, -42, -27, -26, -5, -55, 7, 3, -1, -36, -20, -4, -13, -12, -43, -60, -24,
-27, -40, -6, 0, -8]
```

[217]:
```
more_than_20["score"] = scores
```

```
<ipython-input-217-76e9b4c5cf59>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  more_than_20["score"] = scores
```

[218]:
```
sorted_by_score = more_than_20.sort_values(by=["score"],ascending=False)
sorted_by_score.head(40)
```

[218]:
```
                      name            slug       type  \
386   Justin Wyss-Gallifent    wyss-gallifent  professor
724       Stefan Doboszczak        doboszczak  professor
369         Allan Yashinski         yashinski  professor
467           Jeffery Davis     davis_jeffery  professor
352      Michael Galczynski        galczynski  professor
```

```
997      Elizabeth Griffith            griffith  professor
823         Steven Chadwick            chadwick  professor
434            Michael Ross        ross_michael  professor
513           David Straney             straney  professor
803              Anne Simon          simon_anne  professor
503            Gary Pertmer             pertmer  professor
822            Naveen Sarna               sarna  professor
207     Lawrence Washington  washington_lawrence  professor
727            Wiseley Wong        wong_wiseley  professor
209      Erich Studer-Ellis        studer-ellis  professor
939          Susan Mazzullo            mazzullo  professor
706      Dionisios Margetis            margetis  professor
774      Robert Bonenberger         bonenberger  professor
124     Adrianos Papamarcou          papamarcou  professor
527             David Yager               yager  professor
84             Terence Long        long_terence  professor
756         Kendall Williams     williams_kendall  professor
496          Kasso Okoudjou            okoudjou  professor
829           Kevin Calabro             calabro  professor
711            Richard King        king_richard  professor
50              Ryan Curtis              curtis  professor
416          Bryan Eichhorn            eichhorn  professor
355              Fawzi Emad          emad_fawzi  professor
676      William McClenahan           mcclenahan  professor
495               Louisa Wu           wu_louisa  professor
804          Nathan Manning      manning_nathan  professor
411            Progyan Basu                basu  professor
982              John Pease               pease  professor
465             Jeff Miller         miller_jeff  professor
403  Michael Montague-Smith      montague-smith  professor
258             Judd Nelson         nelson_judd  professor
25             Ibrahim Ades        ades_ibrahim  professor
985             Hugh Turner         turner_hugh  professor
729          Wojciech Czaja               czaja  professor
684   Chandrasekhar Thamire             thamire  professor

                                                courses  average_rating  \
386  [MATH206, MATH241, MATH410, MATH695, HONR219O,…          4.9149
724  [AMSC460, CMSC460, MATH140, MATH410, MATH120, …          4.9500
369  [MATH240, MATH401, MATH406, MATH461, MATH141H,…          4.6400
467  [CHEM241, CHEM611, CHEM889D, CHEM648F, CHEM640…          4.7407
352  [ENES100, ENES100A, ENES113, ENES102, ENES102H…          4.9677
997  [CHEM135, CHEM482, CHEM177, CHEM271, CHEM131, …          4.4167
823  [MATH140, MATH140H, STAT410, MATH141, MATH401,…          4.2000
434  [HIST289K, HIST455, HIST454, HIST619R, HIST131…          4.4286
513            [BSCI222, CPSP219L, BSCI348R, BSCI415]          4.8621
803  [BSCI105C, BSCI105S, CBMG688B, CBMG688X, BSCI1…          3.2245
```

```
503  [ENES102, ENES102H, ENME489V, ENME472, ENME201…          4.6000
822                              [ECON201, ECON317]           4.4444
207  [MATH456, MATH620, MATH406, MATH808N, MATH220,…          4.8571
727  [MATH141, MATH241H, MATH401, MATH241, MATH475,…          4.8621
209                             [BMGT230, BMGT230B]           4.2973
939  [STAT100, STAT400, STAT430, STAT401, STAT440, …          4.8696
706  [MATH648M, MATH241, MATH424, MATH463, MATH858M…          4.5581
774                              [ENES102, ENES220]           4.3810
124  [ENEE222, ENEE324, ENEE620, ENEE322, ENEE322H,…          4.2400
527  [NACS728Q, PSYC301, HONR209O, PSYC407, PSYC606…          3.8333
84   [MATH213, MATH140, MATH113, MATH115, MATH246, …          4.4000
756  [MATH241, MATH241H, MATH310, MATH274, MATH402,…          2.8929
496  [MATH141, MATH140, MATH630, MATH631, MATH858G,…          3.8400
829  [ENES100, ENES221, ENES100A, ENES232, ENES499,…          4.2917
711  [MUSC205, MUSC230, HONR209Z, MUSC639H, MUSC320…          4.4286
50   [PSYC100, PSYC420, PSYC334, PSYC798B, HONR219E…          4.1613
416  [CHEM131, CHEM608B, CHEM602, CHEM401, CHEM271,…          3.9231
355     [CMSC122, CMSC131, CMSC132, CMSC131H, CMSC250]        4.4086
676  [BMGT380H, BMGT381, BMGT380, BUSI764, BMGT496,…          4.1250
495  [BSCI378H, BSCI416, BSCI410, MOCB899, HLSC322,…          4.3333
804  [MATH220, MATH445, MATH140, MATH401, MATH140H,…          3.4231
411  [BMGT313, BMGT220, BMGT398C, BUAC743, BUSI610,…          3.9697
982  [SOCY441, SOCY200, SOCY200H, SOCY699T, SOCY653…          4.1500
465  [BMGT110, BMGT289I, BMGT372, BMGT110S, BMGT466…          4.1064
403  [CHEM241, CHEM395, CHEM231, CPSP218L, CHEM399X…          4.1081
258             [BSCI105, ENTM788C, BSCI105M, TOXI609]        4.2424
25    [BSCI338P, BSCI433, BSCI330, BSCI338G, BSCI339Q]        4.1892
985  [BMGT198F, BMGT372, BMGT110F, BULM758G, BMGT37…          4.4400
729  [MATH648I, MATH141H, MATH411, MATH648Q, MATH14…          3.8929
684  [ENES221, ENME371, ENME632, ENPM620, ENES232, …          4.2727

     average_gpa  num_ratings  score
386         3.29        141.0     21
724         2.72         20.0      7
369         2.72         25.0      6
467         2.81         27.0      5
352         3.56         31.0      3
997         2.57         24.0      3
823         2.57         20.0      3
434         3.14         21.0      2
513         3.01         29.0      2
803         3.17         49.0      1
503         3.08         20.0      0
822         2.73         27.0      0
207         3.12         28.0      0
727         2.49         29.0      0
209         2.58         37.0     -1
```

```
939         2.98        23.0     -1
706         2.70        43.0     -1
774         2.62        21.0     -2
124         2.87        25.0     -3
527         2.48        30.0     -3
84          2.73        20.0     -4
756         2.47        28.0     -4
496         2.36        25.0     -5
829         2.80        24.0     -5
711         3.08        21.0     -5
50          2.74        31.0     -5
416         2.29        26.0     -6
355         2.41        93.0     -6
676         3.10        24.0     -6
495         3.69        21.0     -7
804         2.01        26.0     -7
411         2.44        66.0     -7
982         3.00        20.0     -7
465         2.90        47.0     -7
403         2.33        37.0     -8
258         2.83        33.0     -8
25          2.80        37.0     -8
985         3.35        25.0     -8
729         2.03        28.0     -8
684         2.93        22.0     -8
```

Ok so it looks like my new measure might be a little bit too harsh, as the mean is -21. On the other hand, it has also reaffirmed to me that at the top of all things teaching at UMD is always Justin Wyss-Gallifent. His score is triple the next-highest score. Looking at the data in this table, it's really interesting to see where different teachers land. As a student who has taken classes with a number of these professors, I notice certain ones that appear lower than I would have thought, and others that appear higher than I would have thought.

## 5 Conclusion

Students lead stressful lives, and trying to determine which professors will give the best learning experience is a very tricky task. Consulting different sources of data, such as ratings on PlanetTerp, is one approach to tackling this task. However, as we've seen through this analysis, there are problems with the kind of data that you see on PlanetTerp ratings (and likely with other similar review sites). It does seem to be the case that these sites are dominated by students who receive high marks in the courses they are reviewing, which could lead to biased data that paints an inaccurate picture of which professors will be best. However, it does still seem to be possible to harness the ratings data in a way that combats these biases.

For future research, I believe there is a lot of potential for using grade and rating data to help students. One possibility would be some way for students to see how well they are likely to do in future courses based on their past performance. A big issue that I've encountered as a student is knowing how difficult a courseload will likely be. For example, I might be planning out my schedule for next semester and go on the UMD subreddit to ask if my schedule is manageable. I may get wildly different answers. Moreover, because everyone has different levels of ability, what might be an unmanageable workload for me might be a cakewalk for someone on Reddit responding to my proposed schedule. One way you could combat this would be to have some sort of resource where each student would have their grades uploaded to an account. Then you would have some sort of machine-learning model that predicts future performance based on past performance. So as a CMSC student, if I got an A in CMSC131, a B in CMSC132, and a B- in CMSC216, the model would be able to take in that data and predict how well I'm likely to do in CMSC330 and CMSC351 based on the grades of upperclassmen who also got those same grades in CMSC 131, 132, and 216. This could help give students a better idea of how difficult certain courses and workloads will be. Currently, I don't believe this is possible since grade data would have to be associated with individual student accounts, and that sort of thing would only be available if students voluntarily give it up. But if these kinds of barriers could be overcome, I believe this kind of tool would be very helpful in helping students plan out their academic roadmap.

[ ]:

```

```