

# hw04

September 28, 2024

```
[1]: # Initialize Otter
import otter
grader = otter.Notebook("hw04.ipynb")
```

## 1 Homework 4: Functions, Tables, and Groups

Please complete this notebook by filling in the cells provided. Before you begin, execute the previous cell to load the provided tests.

**Helpful Resource:** - [Python Reference](#): Cheat sheet of helpful array & table methods used in this course!

### Recommended Readings:

- [Visualizing Numerical Distributions](#)
- [Functions and Tables](#)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to setup the notebook by importing some helpful libraries. Each time you start your server, you will need to execute this cell again.

For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. **Moreover, throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook!** For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Otherwise, you will fail tests that you thought you were passing previously!

**Note: This homework has hidden tests on it. That means even though the tests may say 100% passed, it doesn't mean your final grade will be 100%. We will be running more tests for correctness once everyone turns in the homework.**

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged.

You should start early so that you have time to get help if you're stuck.

## 1.1 1. Burrito-ful San Diego

```
[2]: # Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

import warnings
warnings.simplefilter('ignore', FutureWarning)
warnings.filterwarnings("ignore")
```

Mira, Sofia, and Sara are trying to use Data Science to find the best burritos in San Diego! Their friends Jessica and Sonya provided them with two comprehensive datasets on many burrito establishments in the San Diego area taken from (and cleaned from): <https://www.kaggle.com/srcole/burritos-in-san-diego/data>

The following cell reads in a table called `ratings` which contains names of burrito restaurants, their Yelp rating, Google rating, as well as their overall rating. The `Overall` rating is not an average of the Yelp and Google ratings, but rather it is the overall rating of the customers that were surveyed in the study above.

It also reads in a table called `burritos_types` which contains names of burrito restaurants, their menu items, and the cost of the respective menu item at the restaurant.

```
[3]: # Just run this cell
ratings = Table.read_table("ratings.csv")
ratings.show(5)
burritos_types = Table.read_table("burritos_types.csv").drop(0)
burritos_types.show(5)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

**Question 1.** It would be easier if we could combine the information in both tables. Assign `burritos` to the result of joining the two tables together, so that we have a table with the ratings for every corresponding menu item from every restaurant. Each menu item has the same rating as the restaurant from which it is from.

*Note:* It doesn't matter which table you put in as the argument to the table method, either order will work for the autograder tests.

*Hint:* If you need help on using the `join` method, refer to the [Python Reference Sheet](#) or [Section 8.4](#) in the textbook.

```
[4]: burritos = ratings.join('Name', burritos_types, 'Name')
burritos.show(5)
```

<IPython.core.display.HTML object>

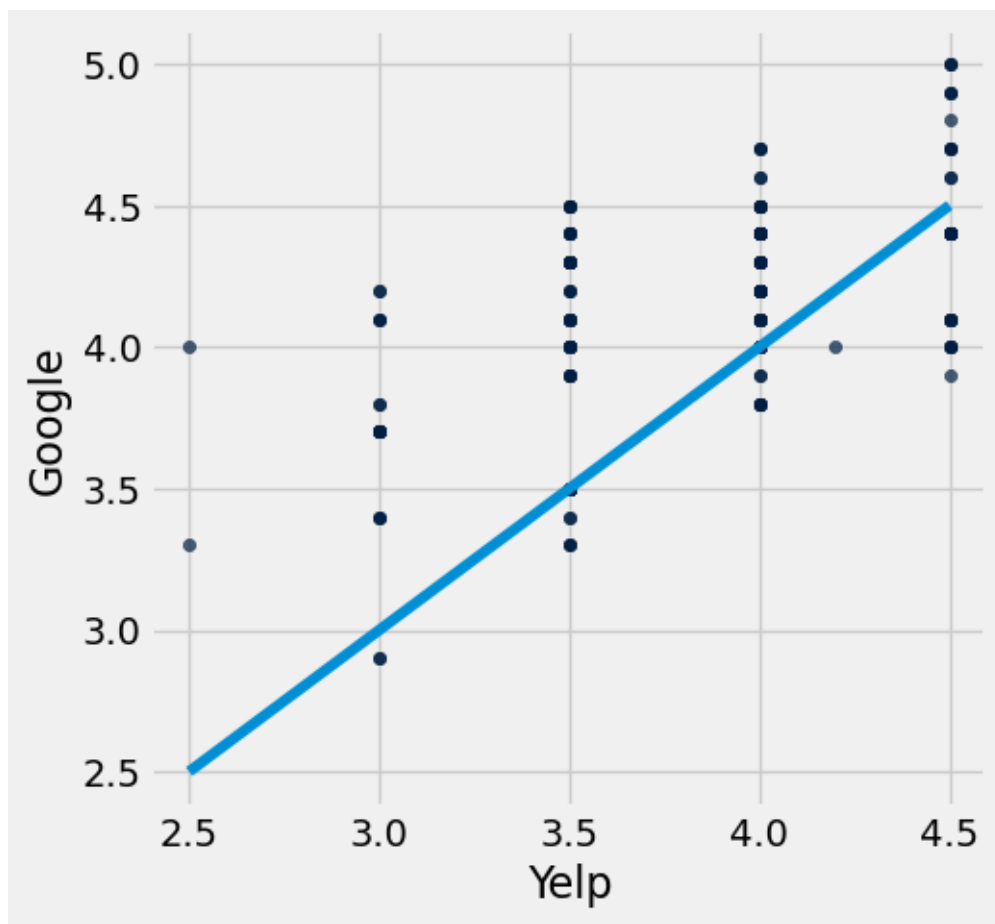
```
[5]: grader.check("q1_1")
```

[5]: q1\_1 results: All test cases passed!

**Question 2.** Let's look at how the Yelp scores compare to the Google scores in the `burritos` table. First, assign `yelp_and_google` to a table only containing the columns `Yelp` and `Google`. Then, make a scatter plot with Yelp scores on the x-axis and the Google scores on the y-axis.

```
[6]: yelp_and_google = burritos.select('Yelp', 'Google')
yelp_and_google.scatter('Yelp', 'Google')

# Don't change/edit/remove the following line.
# To help you make conclusions, we have plotted a straight line on the graph.
↪ (y=x).
plt.plot(np.arange(2.5,5,.5), np.arange(2.5,5,.5));
```



```
[7]: grader.check("q1_2")
```

[7]: q1\_2 results: All test cases passed!

**Question 3.** Looking at the scatter plot you just made in Question 1.2, do you notice any pattern(s) (i.e. is one of the two types of scores consistently higher than the other one)? If so, describe them **briefly** in the cell below.

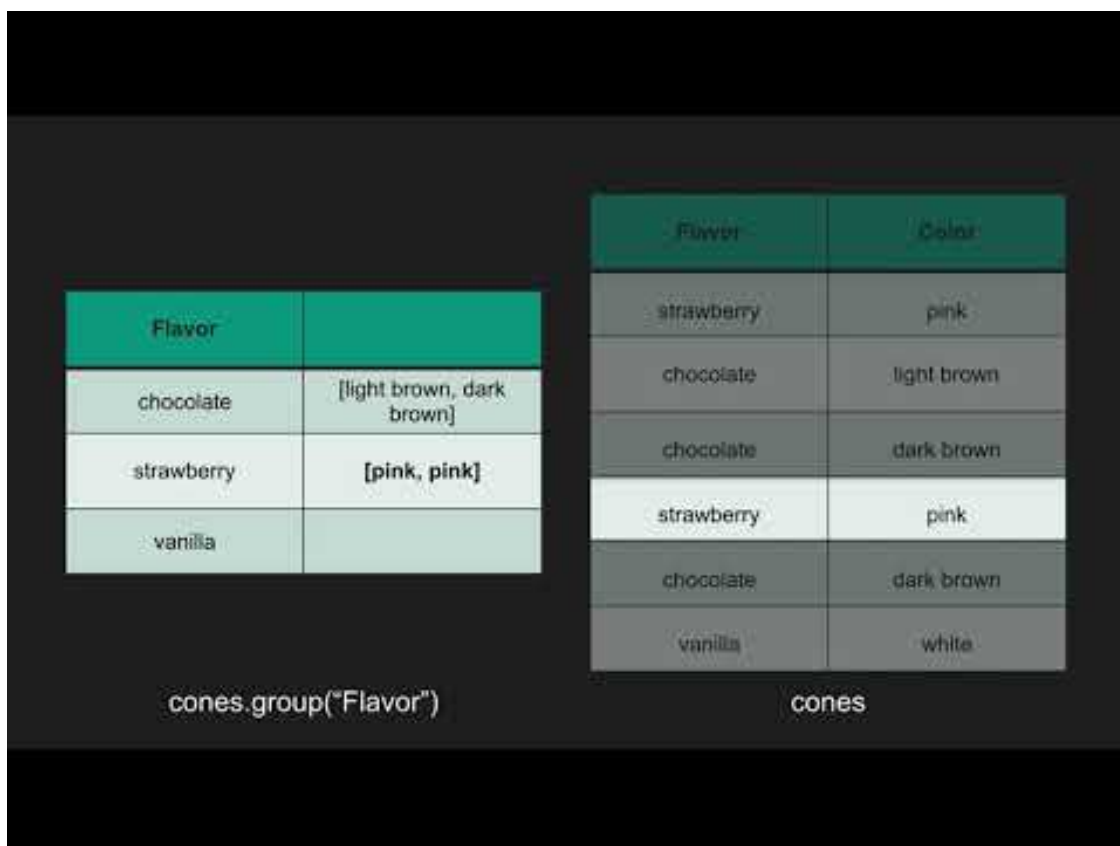
By looking at the scatter plot a pattern I noticed is it looks like the Google scores are usually higher than the Yelp scores. I can see that most of the points are above the line which means for the same Yelp score the Google score tends to be higher. So Google ratings seem to be more positive overall compared to Yelp ratings.

Here's a refresher on how `.group` works! You can read how `.group` works in the [textbook](#), or you can view the video below. The video resource was made by a past staff member, Divyesh Chotai!

You can also use the [Table Functions Visualizer](#) to get some more hands-on experience with the `.group` function.

```
[5]: from IPython.display import YouTubeVideo
      YouTubeVideo("HLoYTCUPOfc")
```

[5]:



Flavor	
chocolate	[light brown, dark brown]
strawberry	[pink, pink]
vanilla	

Flavor	Color
strawberry	pink
chocolate	light brown
chocolate	dark brown
strawberry	pink
chocolate	dark brown
vanilla	white

cones.group("Flavor")

cones

**Question 4.** There are so many types of California burritos in the `burritos` table! Sara wants to know which type is the highest rated across all restaurants. For the sake of these questions, we are treating each menu item's rating the same as its respective restaurant's, as we do not have the rating of every single item at these restaurants. You do not need to worry about this fact, but we thought to mention it!

Create a table with two columns: the first column include the names of the burritos and the second column should contain the average overall rating of that burrito across restaurants. **In your calculations, you should only compare burritos that contain the word “California”.** For example, there are “California” burritos, “California Breakfast” burritos, “California Surf And Turf” burritos, etc.

*Hint:* If multiple restaurants serve the “California - Chicken” burrito, what table method can we use to aggregate those together and find the average overall rating?

*Note:* For reference, the staff solution only used one line. However, feel free to break up the solution into multiple lines and steps; just make sure you assign the final output table to `california_burritos`!

```
[8]: california_burritos = burritos.where('Menu_Item', are.containing('California')).
      ↪group('Menu_Item', np.average).select('Menu_Item', 'Overall average')

california_burritos
```

```
[8]: Menu_Item          | Overall average
     California        | 3.5242
     California (Only Cheese) | 4.1
     California + Guac + Sour Cream | 3.4
     California - Chicken | 3.45839
     California - Pork Adobada | 3.26429
     California - Steak | 3.26429
     California Breakfast | 2.75833
     California Chicken | 3.54815
     California Chipotle | 4.36667
     California Everything | 4.1
     ... (9 rows omitted)
```

```
[9]: grader.check("q1_4")
```

[9]: q1\_4 results: All test cases passed!

**Question 5.** Given this new table `california_burritos`, Sara can figure out the name of the California burrito with the highest overall average rating! Assign `best_california_burrito` to a line of code that outputs the string that represents the name of the California burrito with the highest overall average rating. If multiple burritos satisfy this criteria, you can output any one of them.

```
[10]: best_california_burrito = california_burritos.sort('Overall average',
      ↪descending=True).column('Menu_Item').item(0)
```

```
best_california_burrito
```

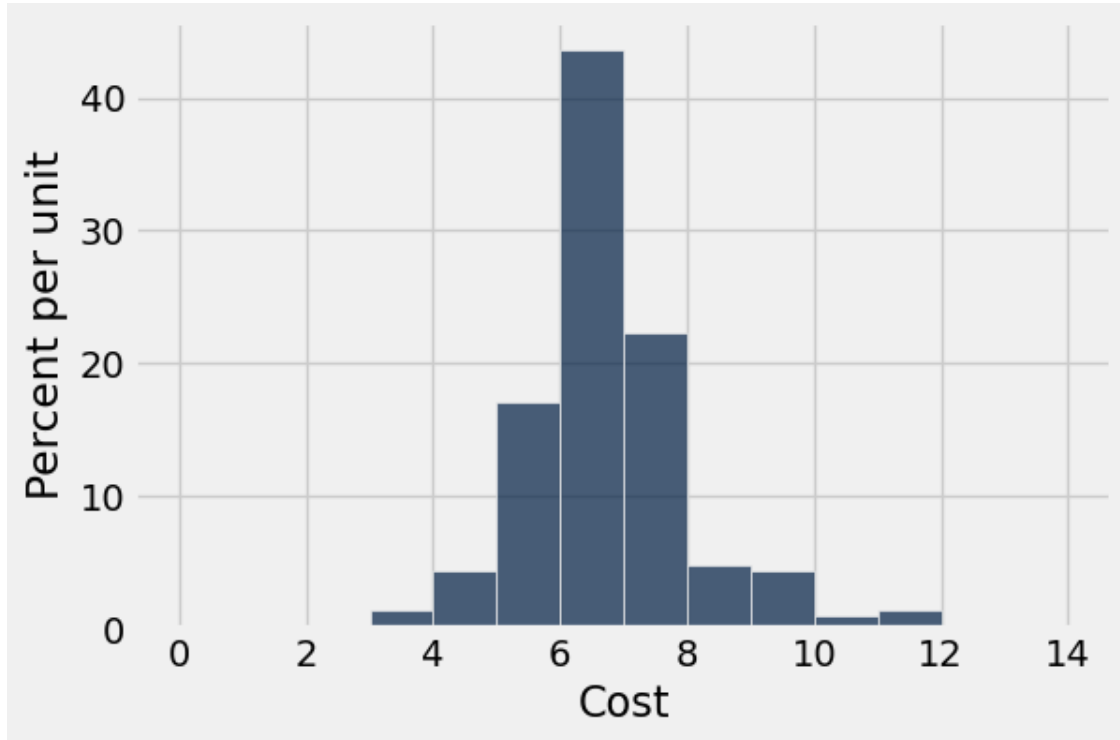
```
[10]: 'California Chipotle'
```

```
[13]: grader.check("q1_5")
```

```
[13]: q1_5 results: All test cases passed!
```

**Question 6.** Mira thinks that burritos in San Diego are cheaper (and taste better) than the burritos in Berkeley. Plot a histogram that visualizes that distribution of the costs of the burritos from San Diego in the `burritos` table. Also use the provided `bins` variable when making your histogram, so that the histogram is more visually informative.

```
[11]: bins = np.arange(0, 15, 1)
      # Please also use the provided bins
      burritos.hist('Cost', bins=bins)
```



**Question 7.** What percentage of burritos in San Diego are less than \$6? Assign `burritos_less_than_6` to your answer, which should be between 0 and 100. You should only use the histogram above to answer the question. Do not use code on the table to find the answer, just eyeball the heights and use Python to evaluate your arithmetic!

*Note:* Your answer does not have to be exact, but it should be within a couple percentages of the staff answer.

```
[12]: burritos_less_than_6 = 25
```

```
[13]: grader.check("q1_7")
```

[13]: q1\_7 results: All test cases passed!

## 1.2 2. Faculty Salaries

This exercise is designed to give you practice with using the Table methods `.pivot` and `.group`. Here is a link to the [Python Reference Sheet](#) in case you need a quick refresher. The [Table Function Visualizer](#) may also be a helpful tool.

Run the cell below to view a demo on how you can use pivot on a table. (Thank you to past staff Divyesh Chotai!)

```
[14]: from IPython.display import YouTubeVideo
      YouTubeVideo("4WzXo8eKLAg")
```

[14]:



In the next cell, we load a dataset created by the [Daily Cal](#) which contains Berkeley faculty, their departments, their positions, and their gross salaries in 2015.

```
[15]: raw_profs = Table.read_table("faculty.csv").where("year", are.equal_to(2015)).
      ↪drop("year", "title")
      profs = raw_profs.relabeled("title_category", "position")
      profs
```

```
[15]: name          | department          | position          |
      gross_salary
CYNTHIA ABAN      | South & Southeast Asian Studies | lecturer          | 64450
PIETER ABBEEL    | Computer Science    | associate professor | 184998
SALLY ABEL       | Law                 | lecturer          | 3466
ELIZABETH ABEL   | English             | professor         | 138775
DOR ABRAHAMSON   | Education           | associate professor | 100300
KATHRYN ABRAMS   | Law                 | professor         | 319693
BARBARA ABRAMS   | Public Health       | professor         | 191162
SARAH ACCOMAZZO  | Social Welfare      | lecturer          | 14779
CHARISMA ACEY    | City and Regional Planning | assistant professor | 101567
DAVID ACKERLY    | Biology             | professor         | 182288
... (2049 rows omitted)
```

We want to use this table to generate arrays with the names of each professor in each department.

**Question 1.** Set `prof_names` to a table with two columns. The first column should be called `department` and have the name of every department once, and the second column should be called `faculty` with each row in that second column containing an *array* of the names of all faculty members in that department.

*Hint:* Think about how `group` works: it collects values into an array and then applies a function to that array. We have defined two functions below for you, and you will need to use one of them in your call to `group`.

```
[33]: # Pick one of the two functions defined below in your call to group.
      def identity(array):
          '''Returns the array that is passed through'''
          return array

      #def first(array):
      #    '''Returns the first item'''
      #    return array.item(0)

      # Make a call to group using one of the functions above when you define
      ↪prof_names
      prof_names = profs.select('department', 'name').group('department',
      ↪collect=identity).relabeled('name identity', 'faculty')
      prof_names
```

```
[33]: department          | faculty
      African American Studies | ['AYA DE LEON' 'CHIYUMA
      ELLIOTT' 'NIKKI JONES' 'DAVID KY ...
```



Agricultural and Resource Economics and Policy	['MAXIMILIAN AUFFHAMMER'
'CHARLES GIBBONS' 'JEFFREY PERL ...	
Anthropology	['SABRINA AGARWAL' 'STANLEY
BRANDES' 'CHARLES BRIGGS'	
' ...	
Architecture	['MARK ANDERSON' 'JACOB
ATHERTON' 'WILLIAM ATWOOD' 'R.GA ...	
Art History	['DILIANA ANGELOVA' 'PATRICIA
BERGER' 'JULIA BRYAN-WILSO ...	
Art Practice	['ALLAN DESOUZA' 'AIDA GAMEZ'
'RANDY HUSSONG' 'JENNIFER ...	
Astronomy	['GIBOR BASRI' 'STEVEN
BECKWITH' 'LEO BLITZ' 'EUGENE CHI ...	
Bioengineering	['ADAM ARKIN' 'IRINA CONBOY'
'STEVEN CONOLLY' 'JOHN DUEB ...	
Biology	['DAVID ACKERLY' 'HILLEL
ADESNIK' 'KELLY AGNEW' 'DORIS B ...	
Buddhist Studies	['JANN RONIS']
... (61 rows omitted)	

```
[32]: grader.check("q2_1")
```

```
[32]: q2_1 results: All test cases passed!
```

Understanding the code you just wrote in 2.1 is important for moving forward with the class! If you made a lucky guess, take some time to look at the code, step by step. Lab time and office hours is always a great resource! **Question 2.** At the moment, the `name` column of the `profs` table is sorted by last name. Would the arrays you generated in the `faculty` column of the previous part be the same if we had sorted by first name instead before generating them?

Two arrays are the **same** if they contain the same number of elements and the elements located at corresponding indexes in the two arrays are identical. An example of arrays that are NOT the same: `array([1,2]) != array([2,1])`. Explain your answer.

*Hint:* If you are not sure, you can always add another coding cell below this one and try sorting the table first. What do you see?

If we sort the `name` column by the first name instead of the last name then the array in the `faculty` column of the `prof_names` table won't stay the same. That's because the `group` function collects names based on how they show up in the original table. Sorting by first name will mix up the order of names in each department making the arrays different from when they were sorted by last name.

**Question 3.** Set `department_ranges` to a table containing departments as the rows, and the position as the columns. The values in the rows should correspond to a salary range, where range is defined as the **difference between the highest salary and the lowest salary in the department for that position**.

*Hint:* First you'll need to define a new function `salary_range` which takes in an array of salaries

and returns the range of salaries in that array.

*Hint 2:* What table function allows you to specify the rows and columns of a new table? You probably watched a video on it earlier in the homework!

```
[34]: # Define compensation_range first
def salary_range(salaries):
    return max(salaries) - min(salaries)

department_ranges = profs.pivot('position', 'department',
    ↪values='gross_salary', collect=salary_range)
department_ranges
```

```
[34]: department                | assistant professor | associate
professor | lecturer | professor
African American Studies          | 128828              | 48814
| 83309      | 0
Agricultural and Resource Economics and Policy | 0                  | 0
| 0          | 29650
Anthropology                      | 11193              | 103271
| 11131      | 166527
Architecture                     | 47675              | 103204
| 72977      | 167892
Art History                      | 57288              | 27439
| 31861      | 100894
Art Practice                     | 0                  | 26632
| 93923      | 0
Astronomy                       | 0                  | 0
| 0          | 138346
Bioengineering                   | 5513               | 2794
| 69008      | 134739
Biology                         | 209667             | 113341
| 137087     | 288554
Buddhist Studies                | 0                  | 0
| 0          | 0
... (61 rows omitted)
```

```
[35]: grader.check("q2_3")
```

```
[35]: q2_3 results: All test cases passed!
```

**Question 4.** Give an explanation as to why some of the row values are 0 in the `department_ranges` table from the previous question.\*

Some of the rows in the `department_ranges` table have 0 values because there are no faculty members in those departments who hold that position specifically. Because the range is collected using the max and min salaries without any information for that position the result will be just 0.

### **1.3 Congratulations! You're done with Homework 4!**

Be sure to run the tests and verify that they all pass, then choose Download as PDF from the File menu and submit the .pdf file on canvas.