hw05

April 17, 2022

Name: Allan Gongora

Section: 0131

1 Homework 5: Pivot Tables and Iteration

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

[1]:
```
pip install gofer-grader
```

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-
packages (1.1.0)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
packages (from jinja2->gofer-grader) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
# Don't change this cell; just run it.

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)

# These lines load the tests.
```

```
from gofer.ok import check
```

Recommended Reading: * Cross-Classifying by More Than One Variable * Bike Sharing * Randomness

1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include:
   A) Sentence reponses to questions that ask for an explanation
   B) Numeric responses to multiple choice questions
   C) Programming code
2) 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!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

1.1 1. Causes of Death by Year

This exercise is designed to give you practice using the Table method `pivot`. Here is a link to the Python reference page in case you need a quick refresher.

We'll be looking at a dataset from the California Department of Public Health that records the cause of death, as recorded on a death certificate, for everyone who died in California from 1999 to 2013. The data is in the file `causes_of_death.csv.zip`. Each row records the number of deaths by a specific cause in one year in one ZIP code.

To make the file smaller, we've compressed it. Run the next cell to unzip and load it.

[3]:
```
!unzip -o causes_of_death.csv.zip
causes = Table.read_table('causes_of_death.csv')
causes
```

```
Archive:  causes_of_death.csv.zip
  inflating: causes_of_death.csv
```

[3]:
```
Year | ZIP Code | Cause of Death | Count | Location
1999 | 90002    | SUI            | 1     | (33.94969, -118.246213)
1999 | 90005    | HOM            | 1     | (34.058508, -118.301197)
1999 | 90006    | ALZ            | 1     | (34.049323, -118.291687)
1999 | 90007    | ALZ            | 1     | (34.029442, -118.287095)
```

```
1999 | 90009    | DIA            | 1     | (33.9452, -118.3832)
1999 | 90009    | LIV            | 1     | (33.9452, -118.3832)
1999 | 90009    | OTH            | 1     | (33.9452, -118.3832)
1999 | 90010    | STK            | 1     | (34.060633, -118.302664)
1999 | 90010    | CLD            | 1     | (34.060633, -118.302664)
1999 | 90010    | DIA            | 1     | (34.060633, -118.302664)
… (320142 rows omitted)
```

The causes of death in the data are abbreviated. We've provided a table called `abbreviations.csv` to translate the abbreviations.

[4]:
```
abbreviations = Table.read_table('abbreviations.csv')
abbreviations.show()
```

```
<IPython.core.display.HTML object>
```

The dataset is missing data on certain causes of death for certain years. It looks like those causes of death are relatively rare, so for some purposes it makes sense to drop them from consideration. Of course, we'll have to keep in mind that we're no longer looking at a comprehensive report on all deaths in California.

Question 1.1 Let's clean up our data. First, filter out the HOM, HYP, and NEP rows from the table for the reasons described above. Next, join together the abbreviations table and our causes of death table so that we have a more detailed description of each disease in each row. Lastly, drop the column which contains the acronym of the disease, and rename the column with the full description 'Cause of Death'. Assign the variable `cleaned_causes` to the resulting table.

Hint: You should expect this to take more than one line. Use many lines and many intermediate tables to complete this question.

Hint 2: The total number of rows should be 251,548.

[5]:
```
cleaned_causes = abbreviations.where("Cause of Death", are.not_equal_to("HOM")).
 ↪where("Cause of Death", are.not_equal_to("HYP"))
cleaned_causes = cleaned_causes.where("Cause of Death", are.not_equal_to("NEP"))
cleaned_causes = cleaned_causes.join("Cause of Death", causes)
cleaned_causes = cleaned_causes.drop("Cause of Death").relabel("Cause of Death␣
 ↪(Full Description)", "Cause of Death")
cleaned_causes
```

[5]:
```
Cause of Death      | Year | ZIP Code | Count | Location
Alzheimer's Disease | 1999 | 90006    | 1     | (34.049323, -118.291687)
Alzheimer's Disease | 1999 | 90007    | 1     | (34.029442, -118.287095)
Alzheimer's Disease | 1999 | 90012    | 1     | (34.061396, -118.238479)
Alzheimer's Disease | 1999 | 90015    | 1     | (34.043439, -118.271613)
Alzheimer's Disease | 1999 | 90017    | 1     | (34.055864, -118.266582)
Alzheimer's Disease | 1999 | 90020    | 1     | (34.066535, -118.302211)
Alzheimer's Disease | 1999 | 90031    | 1     | (34.078349, -118.211279)
Alzheimer's Disease | 1999 | 90033    | 1     | (34.048676, -118.208442)
Alzheimer's Disease | 1999 | 90042    | 1     | (34.114527, -118.192902)
```

```
Alzheimer's Disease | 1999 | 90044    | 1     | (33.955089, -118.290119)
… (251538 rows omitted)
```

[6]:
```
answer_cleaned_causes = cleaned_causes.copy()
check('tests/q1_1.py')
```

[6]:
```
<gofer.ok.OKTestsResult at 0x7f83b66a5ad0>
```

We're going to examine the changes in causes of death over time. To make a plot of those numbers, we need to have a table with one row per year, and the information about all the causes of death for each year.

Question 1.2 Create a table with one row for each year and a column for each kind of death, where each cell contains the number of deaths by that cause in that year. Call the table `cleaned_causes_by_year`.

Hint: You need to use the pivot method with values = 'Column Label', collect = function.

[7]:
```
cleaned_causes_by_year = cleaned_causes.pivot(columns="Cause of Death",␣
↪rows="Year", values="Count", collect=sum)
cleaned_causes_by_year.show(15)
```

```
/opt/conda/lib/python3.7/site-packages/datascience/tables.py:920:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  values = np.array(tuple(values))
```

```
<IPython.core.display.HTML object>
```

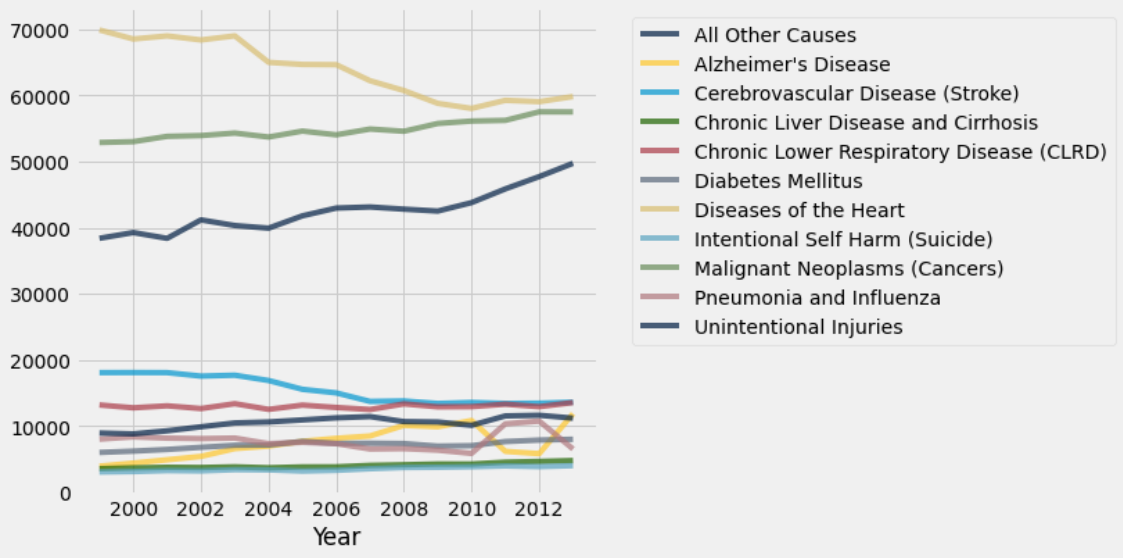
[8]:
```
answer_cleaned_causes_by_year = cleaned_causes_by_year.copy()
check('tests/q1_2.py')
```

[8]:
```
<gofer.ok.OKTestsResult at 0x7f83f835cad0>
```

Question 1.3 Make a plot of all the causes of death by year, using your cleaned-up version of the dataset. There should be a single plot with one line per cause of death.

Hint: Use the Table method `plot`. If you pass only a single argument, a line will be made for each of the other columns.

[9]:
```
answer_cleaned_causes_by_year.plot("Year")
```



After seeing the plot above, we would now like to examine the distributions of diseases over the years using percentages. Below, we have assigned `distributions` to a table with all of the same columns, but the raw counts in the cells are replaced by the percentage of the total number of deaths from a particular disease that happened in that specific year.

Try to understand the code below.

[10]:
```
def percents(array_x):
    return np.round( (array_x/sum(array_x))*100, 2)

labels = cleaned_causes_by_year.labels
distributions = Table().with_columns(labels[0], cleaned_causes_by_year.
 ↪column(0),
                                     labels[1], percents(cleaned_causes_by_year.
 ↪column(1)),
                                     labels[2], percents(cleaned_causes_by_year.
 ↪column(2)),
                                     labels[3], percents(cleaned_causes_by_year.
 ↪column(3)),
                                     labels[4], percents(cleaned_causes_by_year.
 ↪column(4)),
                                     labels[5], percents(cleaned_causes_by_year.
 ↪column(5)),
                                     labels[6], percents(cleaned_causes_by_year.
 ↪column(6)),
                                     labels[7], percents(cleaned_causes_by_year.
 ↪column(7)),
                                     labels[8], percents(cleaned_causes_by_year.
 ↪column(8)),
```

```
                                        labels[9], percents(cleaned_causes_by_year.
 ↪column(9)),
                                        labels[10],␣
 ↪percents(cleaned_causes_by_year.column(10)),
                                        labels[11],␣
 ↪percents(cleaned_causes_by_year.column(11)))
distributions.show()
```

```
<IPython.core.display.HTML object>
```

Question 1.4 What is the sum (roughly) of each of the columns (except the Year column) in the table above? Why does this make sense?

100. Makes sense because percentages

Question 1.5 We suspect that the larger percentage of stroke-related deaths over the years 1999-2013 happened in the earlier years, while the larger percentage of deaths related to Chronic Liver Disease over this time period occured in the most recent years. Draw a bar chart to display both of the distributions for these diseases over the time period.

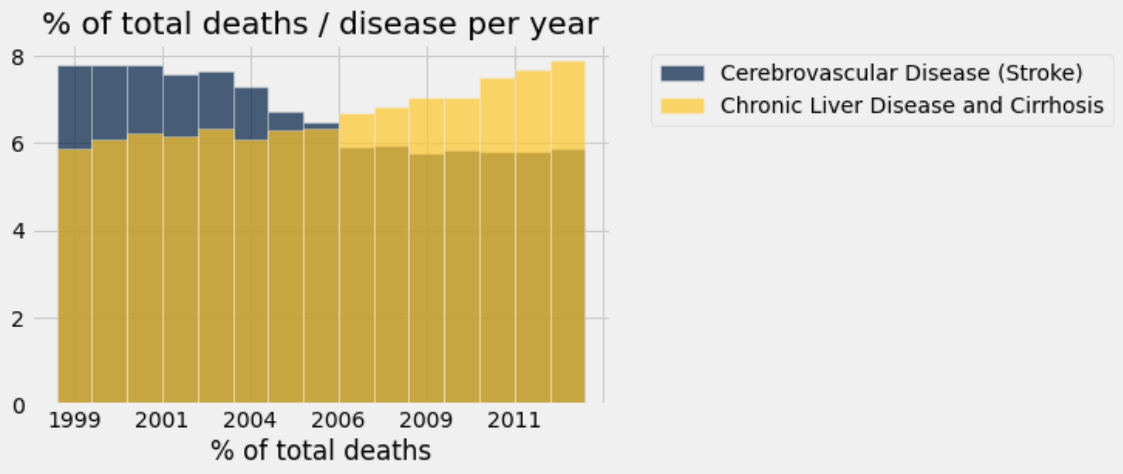
Hint: The relevant column labels are "Cerebrovascular Disease (Stroke)" and "Chronic Liver Disease and Cirrhosis."

Hint 2: You could use either tbl.barh() or tbl.bar() methods. In order to make the bar chart display the information correctly, your table should contain 3 columns.

[31]:
```
distributions.select("Cerebrovascular Disease (Stroke)", "Chronic Liver Disease␣
 ↪and Cirrhosis", "Year").bar("Year")

# Don't change the code below this comment.
plt.title("% of total deaths / disease per year")
plt.xlabel("% of total deaths")
```

[31]:
```
Text(0.5, 0, '% of total deaths')
```



1.2 2. Unrolling Loops (Optional)

The rest of this homework is optional. Do it for your own practice, but it will not be incorporated into the final grading!

"Unrolling" a `for` loop means to manually write out all the code that it executes. The result is code that does the same thing as the loop, but without the structure of the loop. For example, for the following loop:

```
for num in np.arange(3):
    print("The number is", num)
```

The unrolled version would look like this:

```
print("The number is", 0)
print("The number is", 1)
print("The number is", 2)
```

Unrolling a `for` loop is a great way to understand what the loop is doing during each step. In this exercise, you'll practice unrolling `for` loops.

In each question below, write code that does the same thing as the given code, but with any `for` loops unrolled. It's a good idea to run both your answer and the original code to verify that they do the same thing. (Of course, if the code does something random, you'll get a different random outcome than the original code!)

First, run the cell below to load data that will be used in a few questions. It's a table with 52 rows, one for each type of card in a deck of playing cards. A playing card has a "suit" (" ", " ", " ", or " ") and a "rank" (2 through 10, J, Q, K, or A). There are 4 suits and 13 ranks, so there are $4 \times 13 = 52$ different cards.

[12]:
```
deck = Table.read_table("deck.csv")
deck
```

[12]:
```
Rank | Suit
2    |
2    |
2    |
2    |
3    |
3    |
3    |
3    |
4    |
4    |
… (42 rows omitted)
```

Optional Question 2.1 Unroll the code below.

[13]:
```
# This table will hold the cards in a randomly-drawn hand of
# 5 cards.  We simulate cards being drawn as follows: We draw
# a card at random from the deck, make a copy of it, put the
# copy in our hand, and put the card back in the deck.  That
# means we might draw the same card multiple times, which is
# different from a normal draw in most card games.
hand = Table().with_columns("Rank", make_array(), "Suit", make_array())
for suit in np.arange(5):
    card = deck.row(np.random.randint(deck.num_rows))
    hand = hand.with_row(card)
hand
```

[13]:

Rank	Suit
8	
3	
10	
2	
4	

[14]:
```
hand = Table().with_columns("Rank", make_array(), "Suit", make_array())
...
```

[15]:
```
check('tests/q2_1.py')
```

[15]:
```
<gofer.ok.OKTestsResult at 0x7f83a658de90>
```

Optional Question 2.2 Unroll the code below.

[16]:
```
for joke_iteration in np.arange(4):
    print("Knock, knock.")
    print("Who's there?")
    print("Banana.")
    print("Banana who?")
print("Knock, knock.")
print("Who's there?")
print("Orange.")
print("Orange who?")
print("Orange you glad I didn't say banana?")
```

```
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
```

```
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Orange.
Orange who?
Orange you glad I didn't say banana?
```

[17]:
```
...
```

[17]:
```
Ellipsis
```

Optional Question 2.3 Unroll the code below.

Hint: `np.random.randint` returns a random integer between 0 (inclusive) and the value that's passed in (exclusive).

[18]:
```
# This table will hold the cards in a randomly-drawn hand of
# 4 cards.  The cards are drawn as follows: For each of the
# 4 suits, we draw a random card of that suit and put it into
# our hand.  The cards within a suit are drawn uniformly at
# random, meaning each card of the suit has an equal chance of
# being drawn.
hand_of_4 = Table().with_columns("Rank", make_array(), "Suit", make_array())
for suit in make_array(" ", " ", " ", " "):
    cards_of_suit = deck.where("Suit", are.equal_to(suit))
    card = cards_of_suit.row(np.random.randint(cards_of_suit.num_rows))
    hand_of_4 = hand_of_4.with_row(card)
hand_of_4
```

[18]:
```
Rank | Suit
J    |
Q    |
A    |
4    |
```

[19]:
```
...
```

[19]:
```
Ellipsis
```

[20]:
```
check('tests/q2_3.py')
```

[20]:
```
<gofer.ok.OKTestsResult at 0x7f83a6625250>
```

1.3 3. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the auto-grader does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this homework. When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

[ ]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('hw05.ipynb', sorted(glob.glob('tests/q*.py'))))
```

```
Archive:  causes_of_death.csv.zip
  inflating: causes_of_death.csv
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Banana.
Banana who?
Knock, knock.
Who's there?
Orange.
Orange who?
Orange you glad I didn't say banana?
```

Name: Allan Gongora

Section: 0131