3.1 Relationships between Categorical Variables

May 9, 2019

1 Chapter 3. Relationships between Variables

So far, we have seen different ways to summarize and visualize *individual* variables in a data set. But we have not really discussed how to summarize and visualize relationships between *multiple* variables. This chapter is all about how to understand relationships between the columns in a `DataFrame`. The methods will be different, depending on whether the variables are categorical or quantitative.

2 3.1 Relationships between Categorical Variables

In this section, we look at ways to summarize the relationship between two *categorical* variables. To do this, we will again use the Titanic data set.

```
In [1]: %matplotlib inline
        import pandas as pd

        titanic_df = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/mas
```

Suppose we want to understand the relationship between where a passenger embarked and what class they were in. We can completely summarize this relationship by counting the number of passengers in each class that embarked at each location. We can create a pivot table that summarizes this information.

```
In [2]: embarked_pclass_counts = titanic_df.pivot_table(
            index="embarked", columns="pclass",
            values="name",  # We can pretty much count any column, as long as there are no NaN
            aggfunc="count" # The count function will count the number of non-null values.
        )
        embarked_pclass_counts
```

```
Out[2]: pclass      1    2    3
        embarked
        C         141   28  101
        Q           3    7  113
        S         177  242  495
```

A pivot table that stores counts is also called a **contigency table** or a **cross-tabulation**. This type of pivot table is common enough that there is a specific function in `pandas` to calculate one, allowing you to bypass `.pivot_table`:

```
In [3]: counts = pd.crosstab(titanic_df.embarked, titanic_df.pclass)
        counts
```

```
Out[3]: pclass      1    2    3
        embarked
        C         141   28  101
        Q           3    7  113
        S         177  242  495
```

2.1 Joint Distributions

It is common to normalize the counts in a table so that they add up to 1. These proportions represent the **joint distribution** of the two variables.

To calculate the joint distribution, we need to divide the table of counts above by the total count. To find the total count, we call `.sum()` twice; the first call gives us the sum of each column, and the second call adds those numbers together.

```
In [4]: print(counts.sum().sum())
        joint = counts / counts.sum().sum()
        joint
```

```
1307
```

```
Out[4]: pclass           1         2         3
        embarked
        C         0.107881  0.021423  0.077276
        Q         0.002295  0.005356  0.086458
        S         0.135425  0.185157  0.378730
```

Note that this is yet another example of broadcasting. When we divided the `DataFrame` `counts` by the number 1307, the division was applied elementwise, producing another `DataFrame`.

Each cell in this `DataFrame` tells us a joint proportion. For example, the cell in the bottom right tells us the proportion of all passengers that embarked at Southampton and were in 3rd class. We notate this joint proportion as follows:

$$P(\text{embarked at Southampton and in 3rd class}) = .379.$$

The joint distribution above could also have been obtained by specifying `normalize=True` when the contingency table was first created:

```
In [5]: pd.crosstab(titanic_df.embarked, titanic_df.pclass,
                    normalize=True)
```

```
Out[5]: pclass           1         2         3
        embarked
        C         0.107881  0.021423  0.077276
        Q         0.002295  0.005356  0.086458
        S         0.135425  0.185157  0.378730
```

The above joint distribution is not, strictly speaking, a contingency table. A contingency table is a table of all counts, while the above table is a table of proportions.

2.2 Marginal Distributions

The **marginal distribution** of a variable is simply the distribution of that variable, ignoring the other variables. To calculate the marginal distribution from a joint distribution of two variables, we sum the rows or the columns of the joint distribution.

For example, to calculate the marginal distribution of `embarked`, we have to sum the joint distribution over the columns—in other words, *roll-up* or *marginalize over* the `pclass` variable:

```
In [6]: joint.sum(axis=1)
```

```
Out[6]: embarked
        C    0.206580
        Q    0.094109
        S    0.699311
        dtype: float64
```

We can check this answer by calculating the distribution of `embarked` directly from the original data, ignoring `pclass` entirely.

```
In [7]: embarked_counts = titanic_df.groupby("embarked")["name"].count()
        embarked_counts / embarked_counts.sum()
```

```
Out[7]: embarked
        C    0.206580
        Q    0.094109
        S    0.699311
        Name: name, dtype: float64
```

The numbers match!

Likewise, we calculate the marginal distribution of `pclass` by summing the joint distribution over the rows—in other words, by *rolling-up* or *marginalizing over* the `embarked` variable:

```
In [8]: joint.sum(axis=0)
```

```
Out[8]: pclass
        1    0.245601
        2    0.211936
        3    0.542464
        dtype: float64
```

So given the joint distribution of two categorical variables, there are two marginal distributions: one for each of the variables. These marginal distributions are obtained by summing the joint distribution table over the rows and over the columns.

The *marginal distribution* is so-named because these row and column totals would typically be included alongside the joint distribution, in the *margins* of the table. A contingency table with the marginal distributions included can be obtained by specifying `margins=True` in `pd.crosstab`:

```
In [9]: pd.crosstab(titanic_df.embarked, titanic_df.pclass,
                    normalize=True, margins=True)
```

```
Out[9]: pclass           1         2         3       All
        embarked
        C         0.107881  0.021423  0.077276  0.206580
        Q         0.002295  0.005356  0.086458  0.094109
        S         0.135425  0.185157  0.378730  0.699311
        All       0.245601  0.211936  0.542464  1.000000
```

2.3 Conditional Distributions

The **conditional distribution** tells us about the distribution of one variable, *conditional on* the value of another. For example, we might want to know the proportion of 3rd class passengers that embarked at each location. In other words, what is the distribution of where a passenger embarked, *conditional on* being in 3rd class?

If we go back to the contingency table:

```
In [10]: embarked_pclass_counts
```

```
Out[10]: pclass      1    2    3
         embarked
         C         141   28  101
         Q           3    7  113
         S         177  242  495
```

there were $101 + 113 + 495 = 709$ passengers in 3rd class, of whom

- $101/709 = .142$ embarked at Cherbourg
- $113/709 = .159$ embarked at Queenstown and
- $495/709 = .698$ embarked at Southampton.

We can calculate these proportions in code by dividing the `pclass=3` column by its sum:

```
In [11]: embarked_pclass_counts[3] / embarked_pclass_counts[3].sum()
```

```
Out[11]: embarked
         C    0.142454
         Q    0.159379
         S    0.698166
         Name: 3, dtype: float64
```

Notice that these three proportions add up to 1, making this a proper distribution.

This conditional distribution helps us answer questions such as, "What proportion of 3rd class passengers embarked at Southampton?" We notate this conditional proportion as follows:

$$P\left(\text{embarked at Southampton} \,\middle|\, \text{in 3rd class}\right) = 0.698.$$

The pipe | is read "given". So we are interested in the proportion of passengers who embarked at Southampton, *given* that they were in 3rd class.

We could have also calculated this conditional distribution from the joint distribution (i.e., proportions instead of counts):

```
In [12]: joint[3] / joint[3].sum()

Out[12]: embarked
         C    0.142454
         Q    0.159379
         S    0.698166
         Name: 3, dtype: float64
```

We have just calculated *one* of the conditional distributions of `embarked`: the distribution conditional on being in 3rd class. There are two more conditional distributions of `embarked`:

- the distribution conditional on being in 1st class
- the distribution conditional on being in 2nd class

It is common to report *all* of the conditional distributions of one variable given another variable.

Of course, it is straightforward to calculate these conditional distributions manually:

```
In [13]: embarked_pclass_counts[1] / embarked_pclass_counts[1].sum()

Out[13]: embarked
         C    0.439252
         Q    0.009346
         S    0.551402
         Name: 1, dtype: float64

In [14]: embarked_pclass_counts[2] / embarked_pclass_counts[2].sum()

Out[14]: embarked
         C    0.101083
         Q    0.025271
         S    0.873646
         Name: 2, dtype: float64
```

But there is a nifty trick for calculating all three conditional distributions at once. By summing the counts over `embarked`, we obtain the total number of people in each `pclass`:

```
In [15]: pclass_counts = embarked_pclass_counts.sum(axis=0)
         pclass_counts
```