1.2 Summarizing Variables

May 8, 2019

1 1.2 Summarizing Variables

In the previous section, we emphasized the difference between quantitative and categorical variables. The distinction is not merely pedantic; pandas will actually behave differently depending on whether it thinks a variable is quantitative or categorical.

It is not easy for a human to make sense of *all* the values of a variable. In this section, we focus on ways to reduce the values to just a handful of summary statistics. Our working example will again be the Titanic data set, which contains both quantitative and categorical variables.

```
In [1]: import pandas as pd
        pd.options.display.max_rows = 8

        df = pd.read_csv("/data301/data/titanic.csv")
        df
```

Out[1]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	0	Zabour, Miss. Thamine	female	NaN	1	0	2665	14.4542	NaN	C	NaN	NaN
1306	3	0	Zakarian, Mr. Mapriededer	male	26.5000	0	0	2656	7.2250	NaN	C	NaN	304.0
1307	3	0	Zakarian, Mr. Ortin	male	27.0000	0	0	2670	7.2250	NaN	C	NaN	NaN
1308	3	0	Zimmerman, Mr. Leo	male	29.0000	0	0	315082	7.8750	NaN	S	NaN	NaN

```
                              home.dest
0                          St Louis, MO
1     Montreal, PQ / Chesterville, ON
2     Montreal, PQ / Chesterville, ON
3     Montreal, PQ / Chesterville, ON
...                                ...
1305                               NaN
1306                               NaN
1307                               NaN
1308                               NaN

[1309 rows x 14 columns]
```

To get a quick summary of a variable, we can use the `.describe()` function. Let's see what happens when we call `.describe()` on a quantitative variable, like `age`.

```
In [2]: df.age.describe()    #same as summary in R

Out[2]: count    1046.000000
        mean       29.881135
        std        14.413500
        min         0.166700
        25%        21.000000
        50%        28.000000
        75%        39.000000
        max        80.000000
        Name: age, dtype: float64
```

It returns the count (the number of observations with non-missing values), the mean, the standard deviation (`std`), and various percentiles (`min`, `25%`, `50%`, `75%`, `max`).

Now, what if we call `.describe()` on a categorical variable, like `embarked`? This is a variable that takes on the values `C`, `Q`, or `S`, depending on whether the passenger embarked at Cherbourg, Queenstown, or Southampton.

```
In [3]: df.embarked.describe()

Out[3]: count     1307
        unique       3
        top          S
        freq       914
        Name: embarked, dtype: object
```

The description of this variable is very different. We still get the count (of non-missing values). But instead of the mean and standard deviation (how would you calculate the mean of `Q` and `S`, anyway?), we get the number of unique values (`unique`), the value that appeared most often (`top`), and how often it appeared (`freq`). These are more natural summaries for a categorical variable, which only take on a limited set of values, where the values are often not even numeric.

The `.describe()` function only provides a handful of the many summary statistics that are available in `pandas`. We extract additional summary statistics below.

1.1 Summary Statistics for Quantitative Variables

What statistics should we use to summarize a quantitative variable? The most salient features of a quantitative variable are its **center** and **spread**.

1.1.1 Measures of Center

Some statistics measure the **center** of a variable. Two commonly used measures of the center are:

- the **mean** (a.k.a. average): the sum of the values divided by the count
- the **median**: the middle value when you sort the values (i.e., a value such that 50% of the values lie below and 50% of the values lie above)

A measure of center gives us information about the "typical" value of a variable. For example, you might not know whether a typical fare on the Titanic was č1, č10, or č100. But if we calculate the mean:

```
In [4]: df.fare.mean()
```

```
Out[4]: 33.295479281345571
```

we see that a typical fare is around č30.

Let's see what the median says about the "typical" fare:

```
In [5]: df.fare.median()
```

```
Out[5]: 14.4542
```

The median is quite different from the mean! It says that about 50% of the passengers paid less than č15 and about 50% paid more, so another reasonable value for the "typical" fare is č15.

The mean was twice the median! What explains this discrepancy? The reason is that the mean is very sensitive to extreme values. To see this, let's look at the highest fare that any passenger paid.

```
In [6]: df.fare.max()
```

```
Out[6]: 512.32920000000001
```

The highest fare paid was over č500! Even if most passengers paid less than č15, extreme values like this one will drag the mean upward. On the other hand, since the median is always the middle value, it is not affected by the extreme values, as long as the ordering of the values is not changed.

To drive this point home, let's see what would happen to the mean and median if that maximum fare were actually č10,000.

```
In [7]: fare_10k = df.fare.replace(df.fare.max(), 10000)
        fare_10k.mean(), fare_10k.median()
```

```
Out[7]: (62.309763073394485, 14.4542)
```

Notice how the mean is now over č60, but the median is unchanged.

Just to satisfy our curiosity, let's learn more about this passenger who paid the maximum fare. To do this, we have to find the row that achieved this maximum value. Fortunately, there is a convenient `pandas` function, `.idxmax()`, that returns the *row index* of the maximum fare. (A mathematician might call this the "arg max".)

```
In [8]: df.fare.idxmax()

Out[8]: 49
```

Now we can select the row corresponding to this index using `.loc`, as we learned in the previous section.

```
In [9]: df.loc[df.fare.idxmax()]

Out[9]: pclass                                            1
        survived                                          1
        name                       Cardeza, Mr. Thomas Drake Martinez
        sex                                            male
                                      ...
        embarked                                          C
        boat                                              3
        body                                            NaN
        home.dest    Austria-Hungary / Germantown, Philadelphia, PA
        Name: 49, Length: 14, dtype: object
```

The median is a number below which 50% of the values fall. What if we want to know some other percentile? We can use the `.quantile()` function, which takes a percentile rank (between 0 and 1) as input and returns the corresponding percentile.

For example, the 75th percentile is:

```
In [10]: df.fare.quantile(.75)

Out[10]: 31.275
```

which is pretty close to the mean. So only about 25% of the passengers paid more than the mean! The mean is not a great measure of center when there are extreme values, as in this data set.

To summarize, we have encountered several `pandas` functions that can be used to summarize a quantitative variable:

- `.mean()` calculates the mean or average.
- `.median()` calculates the median.
- `.quantile(q)` returns a value such that a fraction `q` of the values fall below that value (in other words, the (100q)th percentile).
- `.max()` calculates the maximum value.
- `.idxmax()` returns the index of the row with the maximum value. If there are multiple rows that achieve this value, then it will only return the index of the first occurrence.

The corresponding functions for the *minimum* value exist as well: