Lecture 2
Categorical Variables

Dennis Sun
Stanford University
DATASCI / STATS 112

January 11, 2023

1. Review
2. One Categorical Variable
3. Two (or More) Categorical Variables
4. Reminders

Tabular Data

- **How is tabular data commonly stored on disk?**
  in a CSV (comma-separated values) file
- **How is tabular data represented in Python?**
  as a Pandas `DataFrame`.

```
[2] # Read in the Titanic data set using the Pandas `read_csv` function.
    df_titanic = pd.read_csv("https://dlsun.github.io/stats112/data/titanic.csv")

    # To look at the data, we make `df_titanic` the last line of the cell so that
    # the output is printed.
    df_titanic
```

	name	pclass	survived	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	Allen, Miss. Elisabeth Walton	1	1	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	Allison, Master. Hudson Trevor	1	1	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	Allison, Miss. Helen Loraine	1	0	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	Allison, Mr. Hudson Joshua Creighton	1	0	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	1	0	female	25.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1304	Zabour, Miss. Hileni	3	0	female	14.5000	1	0	2665	14.4542	NaN	C	NaN	328.0	NaN
1305	Zabour, Miss. Thamine	3	0	female	NaN	1	0	2665	14.4542	NaN	C	NaN	NaN	NaN
1306	Zakarian, Mr. Mapriededer	3	0	male	26.5000	0	0	2656	7.2250	NaN	C	NaN	304.0	NaN
1307	Zakarian, Mr. Ortin	3	0	male	27.0000	0	0	2670	7.2250	NaN	C	NaN	NaN	NaN
1308	Zimmerman, Mr. Leo	3	0	male	29.0000	0	0	315082	7.8750	NaN	S	NaN	NaN	NaN

1309 rows × 14 columns

Any Questions?

1 Review

2 One Categorical Variable

3 Two (or More) Categorical Variables

4 Reminders

Summarizing One Categorical Variable

To summarize a categorical variable, we can report the **counts** of each possible class.

We can get the possible classes and their counts using the method `Series.value_counts()`.

For example, we can summarize the variable `"pclass"` by:

```
df_titanic["pclass"].value_counts()
```

```
3    709
1    323
2    277
Name: pclass, dtype: int64
```

Summarizing One Categorical Variable

Instead of reporting counts, we can instead report the **proportion** of times each possible class occurs.

$$\text{proportion} = \frac{\text{count}}{\text{total}}.$$

For example, we can also summarize the variable `"pclass"` by:

```
df_titanic["pclass"].value_counts(normalize=True)
```

```
3    0.541635
1    0.246753
2    0.211612
Name: pclass, dtype: float64
```

Notice that the values in a distribution add up to 1.0!

This is called the **distribution** of the variable `"pclass"`.

What information have we lost by reporting the distribution instead of counts?

Summarizing One Categorical Variable

Proportions can be converted to percentages for readability:

$$\text{percentage} = \text{proportion} \times 100\% = \frac{\text{count}}{\text{total}} \times 100\%.$$

```
df_titanic["pclass"].value_counts(normalize=True) * 100
```

```
3    54.163484
1    24.675325
2    21.161192
Name: pclass, dtype: float64
```

If we multiply a `Series` by 100, every value gets multiplied by 100. That's because operations in Pandas are **vectorized**.

This should remind you of math! If you have a vector $\vec{v} = (v_1, v_2, ..., v_n)$, then

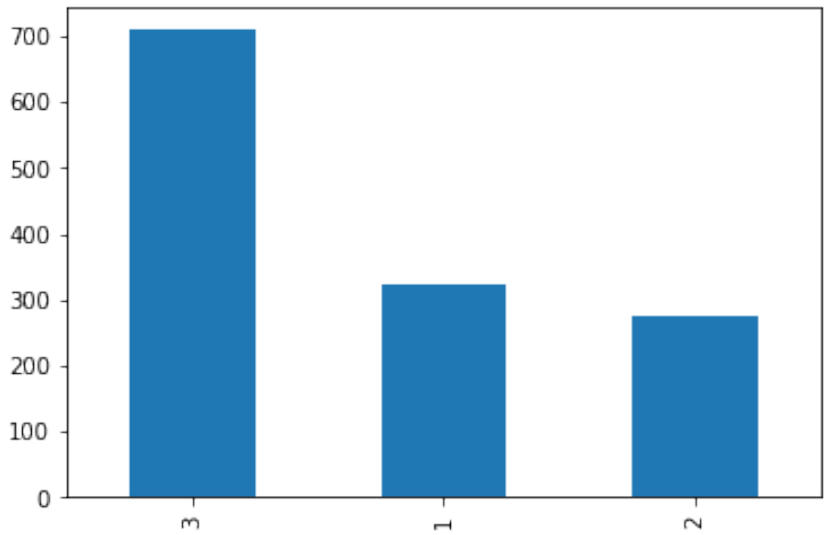
$$100\vec{v} = (100v_1, 100v_2, ..., 100v_n).$$

Visualizing One Categorical Variable

To visualize a categorical variable, we make a **bar plot**.

For example, we can visualize the variable `"pclass"` by:

```
df_titanic["pclass"].value_counts().plot.bar()
```



Hmm...why are the classes out of order?

Why is this graph not ideal?

Visualizing One Categorical Variable

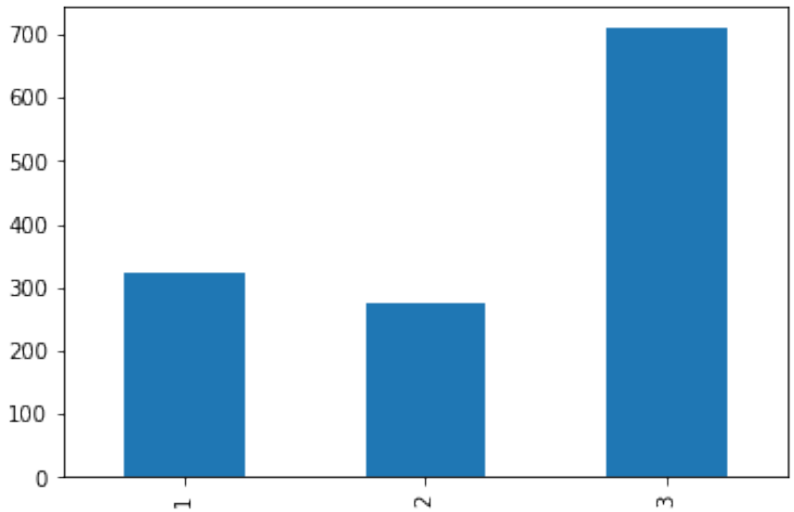
When you use `.value_counts()`, the classes are ordered by count:

```
3    709
1    323
2    277
Name: pclass, dtype: int64
```

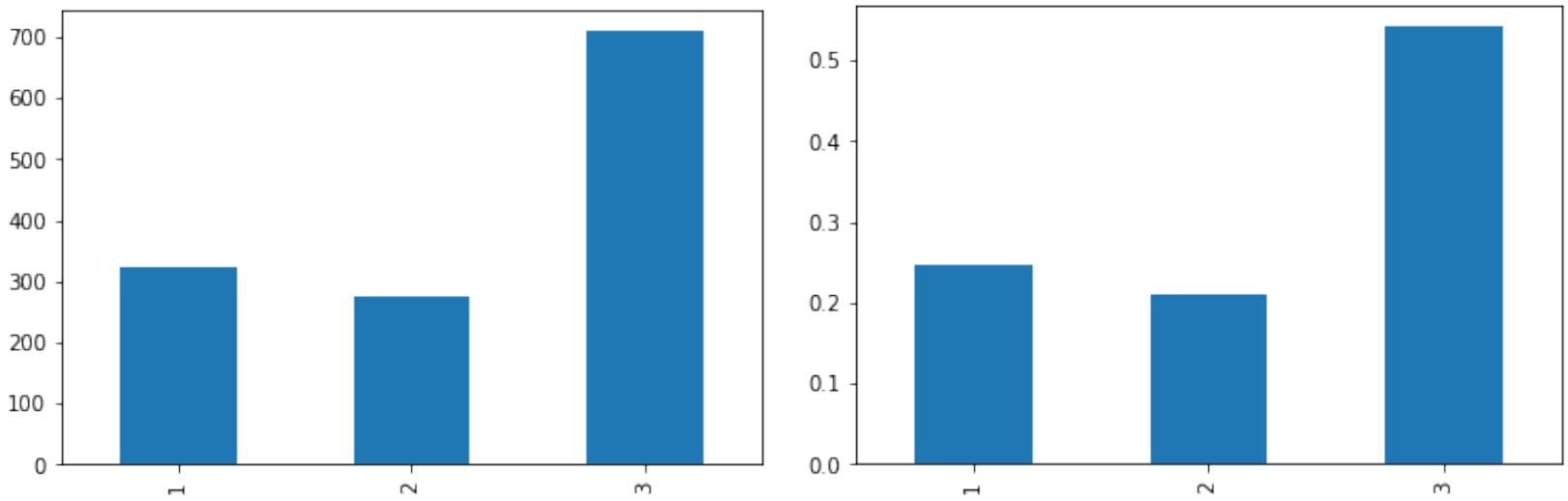
But in this case, there is a natural ordering: 1st, 2nd, 3rd.

```
df_titanic["pclass"].value_counts().sort_index().plot.bar()
```

```
counts_pclass = df_titanic["pclass"].value_counts()
counts_pclass.sort_index().plot.bar()
```



Visualizing One Categorical Variable



How would the bar plot change if we replaced counts by proportions?

The scale on the y-axis changes, but the shape is the same.

To appreciate why proportions are useful, we need to look at *more than* 1 categorical variable.

Summarizing 2+ Categorical Variables

To summarize 2+ categorical variables, we report the **counts** of every possible combination.

We can use the method `DataFrame.value_counts()`.

For example, we can summarize the variables `"pclass"` and `"survived"` by:

```
df_titanic[["pclass", "survived"]].value_counts()
```

```
pclass  survived
3       0           528
1       1           200
3       1           181
2       0           158
1       0           123
2       1           119
dtype: int64
```

Summarizing 2+ Categorical Variables

```
pclass  survived
3       0           528
1       1           200
3       1           181
2       0           158
1       0           123
2       1           119
dtype: int64
```

Let's make this information easier to digest by arranging one variable along the rows and the other along the columns.

```
counts = df_titanic[["pclass", "survived"]].value_counts()
counts.unstack("survived")
```

survived	0	1
pclass		
1	123	200
2	158	119
3	528	181

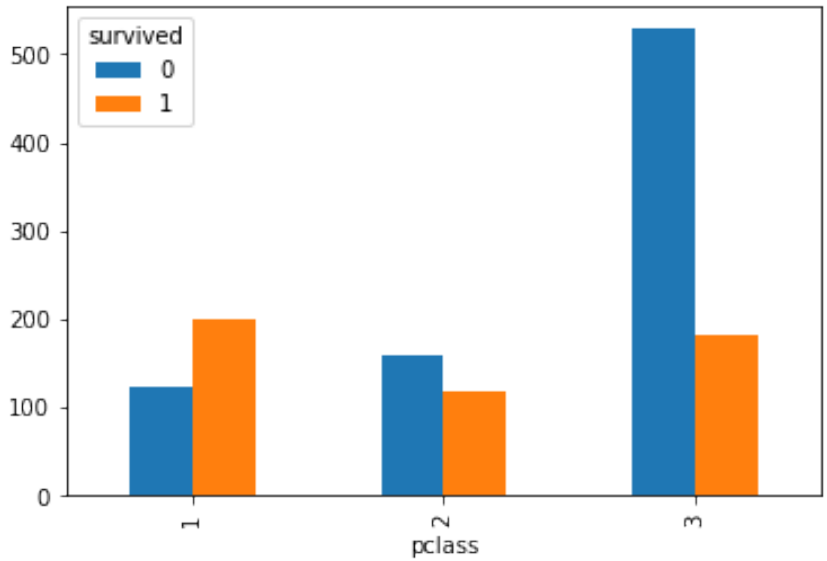
This representation is called a **two-way table** or a **crosstab** (short for "cross-tabulation").

Visualizing 2+ Categorical Variables

survived	0	1
pclass		
1	123	200
2	158	119
3	528	181

From a crosstab, it is easy to make a visualization.

```
crosstab = counts.unstack("survived")
crosstab.plot.bar()
```



Remember when I said that it's sometimes easier to compare proportions than counts? Let's calculate proportions!

survived	0	1
pclass		
1	123	200
2	158	119
3	528	181

```
pclass
1    323
2    277
3    709
dtype: int64
```

```
crosstab
```

```
crosstab.sum(axis="columns")
```

Now we need to divide the crosstab by the sum.

```
totals_pclass = crosstab.sum(axis="columns")
crosstab.divide(totals_pclass, axis="rows")
```

survived	0	1
pclass		
1	0.380805	0.619195
2	0.570397	0.429603
3	0.744711	0.255289

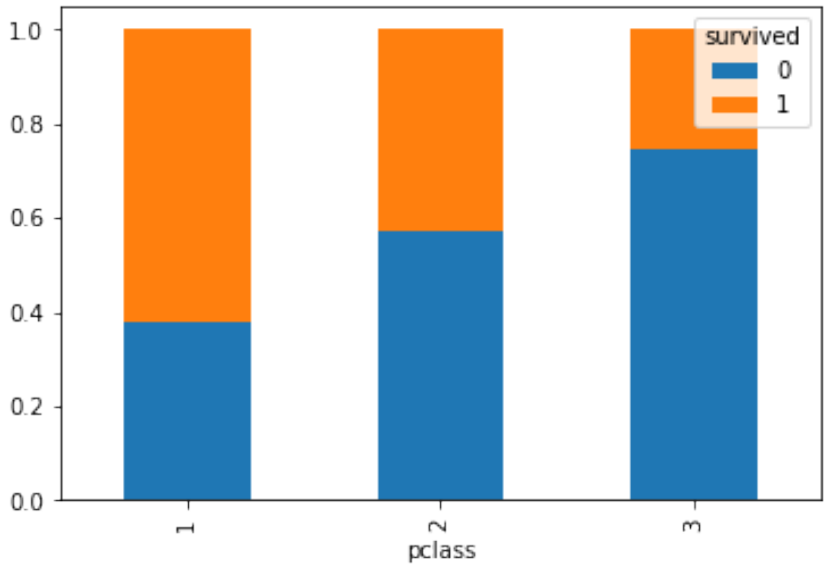
This is the **conditional distribution** of `survived`, given `pclass`:

$$p(\text{survived}|\text{pclass}).$$

Visualizing Conditional Distributions

To visualize a conditional distribution, we use a **stacked bar plot**:

```
survived_given_pclass = crosstab.divide(totals_pclass, axis="rows")
survived_given_pclass.plot.bar(stacked=True)
```



In-Class Exercise

We calculated $p(\text{survived}|\text{pclass})$ above.

survived	0	1
pclass		
1	0.380805	0.619195
2	0.570397	0.429603
3	0.744711	0.255289

What would $p(\text{pclass}|\text{survived})$ look like?

Click here to calculate it in Colab!

The conditional distributions are not the same! What's different?

$$p(\text{survived}|\text{pclass}) = \frac{\#\text{ survived \& pclass}}{\#\text{ pclass}}$$

$$p(\text{pclass}|\text{survived}) = \frac{\#\text{ survived \& pclass}}{\#\text{ survived}}$$

What changes is the Baseline / denominator!

1 Review

2 One Categorical Variable

3 Two (or More) Categorical Variables

4 Reminders

- Work through the Colab “Case Study - Comparing COVID Rates”.
- Come to section tomorrow prepared to present the exercises in this Colab.
- I have office hours now until noon. (Sorry, I have a meeting at noon. I will be around in the afternoon. Amber has office hours 1:30-2:30.)
- Post on the Ed Discussion board if you have any questions!