Lecture 2
Relationships between Categorical Variables

1 Review

2 Two (or More) Categorical Variables

3 Proportions and Probabilities

4 Joint and Conditional Distributions

1 Review

2 Two (or More) Categorical Variables

3 Proportions and Probabilities

4 Joint and Conditional Distributions

Types of Data

Type	On Disk	In Python
tabular		

Types of Data

Type	On Disk	In Python
tabular	CSV	

Types of Data

Type	On Disk	In Python
tabular	CSV	DataFrame

Types of Data

Type	On Disk	In Python
tabular	CSV	DataFrame
hierarchical		

Types of Data

Type	On Disk	In Python
tabular	CSV	DataFrame
hierarchical	JSON	

Types of Data

Type	On Disk	In Python
tabular	CSV	DataFrame
hierarchical	JSON	dict

```
{
  "name": "Alice",
  "age": 28,
  "married": false,
  "children": null,
  "scores": [90, 85, 88],
  "address": {
    "city": "London",
    "zip": 12345
  }
}
```

Types of Data

Type	On Disk	In Python
tabular	CSV	`DataFrame`
hierarchical	JSON	`dict`
textual		

Types of Data

Type	On Disk	In Python
tabular	CSV	`DataFrame`
hierarchical	JSON	`dict`
textual	plaintext	

Types of Data

Type	On Disk	In Python
tabular	CSV	`DataFrame`
hierarchical	JSON	`dict`
textual	plaintext	`string`

Types of Data

Type	On Disk	In Python
tabular	CSV	DataFrame
~~hierarchical~~	~~JSON~~	~~dict~~
textual	plaintext	string
geospatial	???	???

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-0.1276, 51.5072]
  },
  "properties": {
    "name": "London",
    "country": "United Kingdom",
    "population": "8,982,000"
  }
}
```

- **Vector data:** `.shp`, `.geojson`, `.kml`, `.gpx`, `.gpkg`
- **Raster data:** `.tif`, `.img`, `.asc`
- **Databases / Mixed:** `.gdb`, `.gpkg`

Example for geojson file.

Types of Variables

```python
import pandas as pd
df = pd.read_csv("https://datasci112.stanford.edu/data/titanic.csv")
df
```

`pd.read_json("file.json")` or `json.load()`

pandas / json

Types of Variables

```
import pandas as pd
df = pd.read_csv("https://datasci112.stanford.edu/data/titanic.csv")
df
```

	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

Types of Variables

```python
import pandas as pd
df = pd.read_csv("https://datasci112.stanford.edu/data/titanic.csv")
df
```

variables

observational units

	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

Types of Variables

```python
import pandas as pd
df = pd.read_csv("https://datasci112.stanford.edu/data/titanic.csv")
df
```

variables

observational units

	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

quantitative variables

categorical variables

One Categorical Variable

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

```
df["pclass"].value_counts().sort_index()
```

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

One Categorical Variable

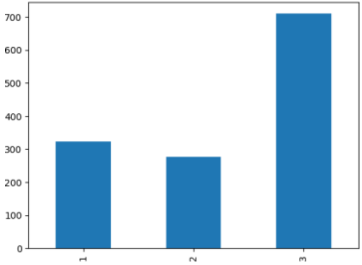
To *summarize* a categorical variable, we report the **counts** of each possible category.

```
df["pclass"].value_counts().sort_index()
```

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

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

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



1 Review

2 Two (or More) Categorical Variables

3 Proportions and Probabilities

4 Joint and Conditional Distributions

Selecting Columns

Note that we selected a single column by passing the column name as a key to the DataFrame.

```
df["pclass"]
```

```
0       1
1       1
2       1
       ..
1306    3
1307    3
1308    3
Name: pclass, Length: 1309, dtype: int64
```

Selecting Columns

Note that we selected a single column by passing the column name as a key to the `DataFrame`.

```
df["pclass"]
```

```
0        1
1        1
2        1
        ..
1306     3
1307     3
1308     3
Name: pclass, Length: 1309, dtype: int64
```

The result is a one-dimensional `pandas` object called a `Series`.

Selecting Columns

Note that we selected a single column by passing the column name as a key to the `DataFrame`.

```
df["pclass"]
```

```
0       1
1       1
2       1
       ..
1306    3
1307    3
1308    3
Name: pclass, Length: 1309, dtype: int64
```

The result is a one-dimensional `pandas` object called a `Series`.

We can select multiple columns by passing a `list` of column names.

Selecting Columns

Note that we selected a single column by passing the column name as a key to the `DataFrame`.

```
df["pclass"]
```

```
0       1
1       1
2       1
       ..
1306    3
1307    3
1308    3
Name: pclass, Length: 1309, dtype: int64
```

The result is a one-dimensional pandas object called a `Series`.

We can select multiple columns by passing a `list` of column names.

```
df[["pclass", "survived"]]
```

Selecting Columns

Note that we selected a single column by passing the column name as a key to the DataFrame.

```
df["pclass"]
```

```
0       1
1       1
2       1
       ..
1306    3
1307    3
1308    3
Name: pclass, Length: 1309, dtype: int64
```

The result is a one-dimensional pandas object called a Series.

We can select multiple columns by passing a list of column names.

```
df[["pclass", "survived"]]
```

	pclass	survived
0	1	1
1	1	1
2	1	0
3	1	0
4	1	0
...	...	...
1304	3	0
1305	3	0
1306	3	0
1307	3	0
1308	3	0

1309 rows × 2 columns

The result is two-dimensional, another smaller DataFrame.

Selecting Columns

Note that we selected a single column by passing the column name as a key to the DataFrame.

```
df["pclass"]
```

```
0       1
1       1
2       1
        ..
1306    3
1307    3
1308    3
Name: pclass, Length: 1309, dtype: int64
```

The result is a one-dimensional pandas object called a Series.

We can select multiple columns by passing a list of column names.

```
df[["pclass", "survived"]]
```

	pclass	survived
0	1	1
1	1	1
2	1	0
3	1	0
4	1	0
...	...	...
1304	3	0
1305	3	0
1306	3	0
1307	3	0
1308	3	0

1309 rows × 2 columns

The result is two-dimensional, another smaller DataFrame.

How do we make sense of multiple variables at once?

Summarizing Multiple Categorical Variables

To summarize multiple categorical variables, we report the **counts** of every possible combination of categories.

Summarizing Multiple Categorical Variables

To summarize multiple categorical variables, we report the **counts** of every possible combination of categories.

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

Summarizing Multiple Categorical Variables

To summarize multiple categorical variables, we report the **counts** of every possible combination of categories.

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

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

Summarizing Multiple Categorical Variables

To summarize multiple categorical variables, we report the **counts** of every possible combination of categories.

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

```
df[["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 Multiple Categorical Variables

To summarize multiple categorical variables, we report the **counts** of every possible combination of categories.

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

```
df[["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
```

Note that the result is a `Series`, with a multi-level index, one for each variable!

Summarizing Multiple 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 read by arranging one variable along the rows and the other along the columns.

Summarizing Multiple 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 read by arranging one variable along the rows and the other along the columns.

```
(df[["pclass", "survived"]].value_counts().
 unstack())
```

Summarizing Multiple 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 read by arranging one variable along the rows and the other along the columns.

```
(df[["pclass", "survived"]].value_counts().
 unstack())
```

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

Summarizing Multiple 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 read by arranging one variable along the rows and the other along the columns.

```
(df[["pclass", "survived"]].value_counts().
 unstack())
```

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 Multiple Categorical Variables

Visualizing Multiple Categorical Variables

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

Visualizing Multiple Categorical Variables

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

From a crosstab, we can make a bar plot to visualize the data.

Visualizing Multiple Categorical Variables

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

From a crosstab, we can make a bar plot to visualize the data.

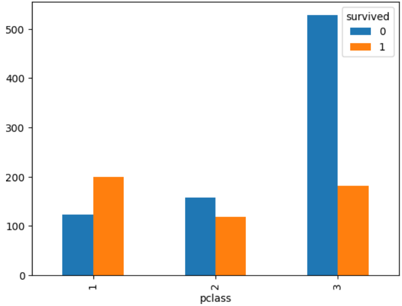
```
(df[["pclass", "survived"]].value_counts().
 unstack().
 plot.bar())
```

Visualizing Multiple Categorical Variables

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

From a crosstab, we can make a bar plot to visualize the data.

```
(df[["pclass", "survived"]].value_counts().
 unstack().
 plot.bar())
```

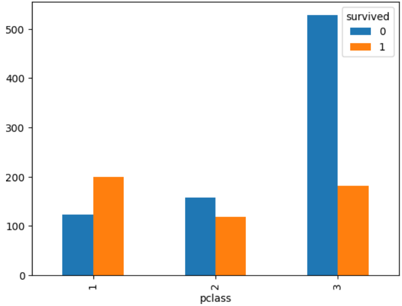


Visualizing Multiple Categorical Variables

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

From a crosstab, we can make a bar plot to visualize the data.

```
(df[["pclass", "survived"]].value_counts().
 unstack().
 plot.bar())
```



This is called a **grouped bar plot**.

Marginal Counts

How do we recover the counts for each individual variable from a crosstab?

```
crosstab = df[["pclass", "survived"]].value_counts().unstack()
crosstab
```

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

Marginal Counts

How do we recover the counts for each individual variable from a crosstab?

```
crosstab = df[["pclass", "survived"]].value_counts().unstack()
crosstab
```

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

We could sum over the columns (across each row) to obtain the counts for **pclass**...

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

Marginal Counts

How do we recover the counts for each individual variable from a crosstab?

```
crosstab = df[["pclass", "survived"]].value_counts().unstack()
crosstab
```

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

We could sum over the columns (across each row) to obtain the counts for **pclass**...

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

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

Marginal Counts

How do we recover the counts for each individual variable from a crosstab?

```
crosstab = df[["pclass", "survived"]].value_counts().unstack()
crosstab
```

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

...or sum over the rows (down each column) to obtain the counts for **survived**.

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

Marginal Counts

How do we recover the counts for each individual variable from a crosstab?

```
crosstab = df[["pclass", "survived"]].value_counts().unstack()
crosstab
```

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

...or sum over the rows (down each column) to obtain the counts for **survived**.

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

```
survived
0     809
1     500
dtype:  int64
```

1 Review

2 Two (or More) Categorical Variables

3 Proportions and Probabilities

4 Joint and Conditional Distributions

Proportions

Proportions

Instead of counts, it can be useful to report **proportions**, where we normalize by the total.

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

Proportions

Instead of counts, it can be useful to report **proportions**, where we normalize by the total.

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

For example, the proportions of the three passenger classes are:

```
df["pclass"].value_counts() / len(df)
```

Proportions

Instead of counts, it can be useful to report **proportions**, where we normalize by the total.

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

For example, the proportions of the three passenger classes are:

```
df["pclass"].value_counts() / len(df)
```

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

Proportions

Instead of counts, it can be useful to report **proportions**, where we normalize by the total.

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

For example, the proportions of the three passenger classes are:

```
df["pclass"].value_counts() / len(df)
```

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

Together, the proportions of a categorical variable are called the **distribution** of the variable **pclass**.

Proportions

Instead of counts, it can be useful to report **proportions**, where we normalize by the total.

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

For example, the proportions of the three passenger classes are:

```
df["pclass"].value_counts() / len(df)
```

```
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!

Together, the proportions of a categorical variable are called the **distribution** of the variable **pclass**.

Probabilities

What does it mean to say, “The proportion of passengers in 3rd class is 0.541635?”

Probabilities

What does it mean to say, “The proportion of passengers in 3rd class is 0.541635?”

One interpretation is as a **probability**.

Probabilities

- What does it mean to say, "The proportion of passengers in 3rd class is 0.541635?"
- One interpretation is as a **probability**.
- "If we were to pick a passenger on the Titanic at random, the probability that they are in 3rd class is 0.541635."

Probabilities

- What does it mean to say, "The proportion of passengers in 3rd class is 0.541635?"
- One interpretation is as a **probability**.
- "If we were to pick a passenger on the Titanic at random, the probability that they are in 3rd class is 0.541635."
- We notate this as
    - $P(\text{3rd class}) = 0.541635$

Probabilities

- What does it mean to say, “The proportion of passengers in 3rd class is 0.541635?”
- One interpretation is as a **probability**.
- “If we were to pick a passenger on the Titanic at random, the probability that they are in 3rd class is 0.541635.”

- We notate this as
  - $P(\text{3rd class}) = 0.541635$
- or, if we want to be explicit about the variable and the category,
  - $P(\textbf{pclass} = 3) = 0.541635$

Vectorization

Let's take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Vectorization

Let’s take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Notice that we divided a `Series` by a number! Is that even legal?

Vectorization

Let's take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Notice that we divided a `Series` by a number! Is that even legal?

In pandas, operations are **vectorized**. A `Series` behaves like a vector.

Vectorization

Let's take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Notice that we divided a `Series` by a number! Is that even legal?

In pandas, operations are **vectorized**. A `Series` behaves like a vector.

Vectors in pandas work like vectors in math!

Vectorization

Let's take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Notice that we divided a `Series` by a number! Is that even legal?

In pandas, operations are **vectorized**. A `Series` behaves like a vector.

Vectors in pandas work like vectors in math!

Math Review

To multiply a vector

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

by a scalar (a.k.a. number) a,

$$a\vec{v} =$$

Vectorization

Let's take a closer look at the code for calculating the proportions.

```
df["pclass"].value_counts() / len(df)
```

Notice that we divided a `Series` by a number! Is that even legal?

In pandas, operations are **vectorized**. A `Series` behaves like a vector.

Vectors in pandas work like vectors in math!

Math Review

To multiply a vector

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

by a scalar (a.k.a. number) a,

$$a\vec{v} = (av_1, av_2, \ldots, av_n),$$

we multiply each component of the vector by a.

1 Review

2 Two (or More) Categorical Variables

3 Proportions and Probabilities

4 Joint and Conditional Distributions

Joint Distributions

We can also calculate the distribution of multiple variables, called a **joint distribution**.

Joint Distributions

We can also calculate the distribution of multiple variables, called a **joint distribution**.

```
df[["pclass", "survived"]].value_counts().unstack() / len(df)
```

Joint Distributions

We can also calculate the distribution of multiple variables, called a **joint distribution**.

```
df[["pclass", "survived"]].value_counts().unstack() / len(df)
```

survived	0	1
pclass		
1	0.093965	0.152788
2	0.120703	0.090909
3	0.403361	0.138273

Joint Distributions

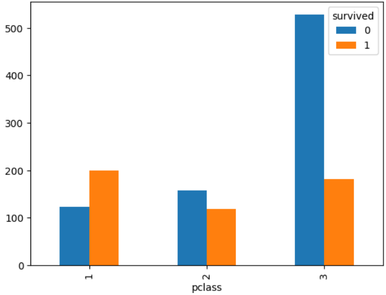
We can also calculate the distribution of multiple variables, called a **joint distribution**.

```
df[["pclass", "survived"]].value_counts().unstack() / len(df)
```

survived	0	1
pclass		
1	0.093965	0.152788
2	0.120703	0.090909
3	0.403361	0.138273

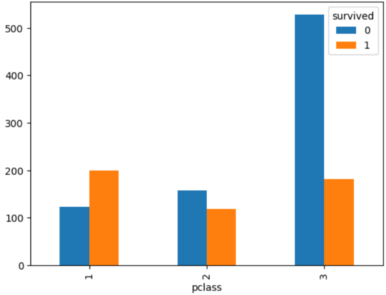
Notice that the values in the joint distribution also sum to 1.0!

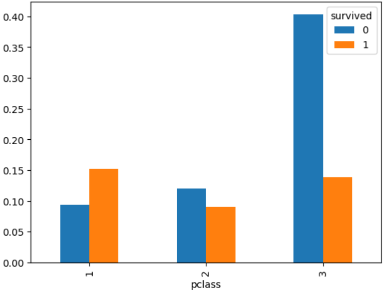
Visualizing Joint Distributions



How does the bar plot change if we plot the joint distribution instead of the counts?

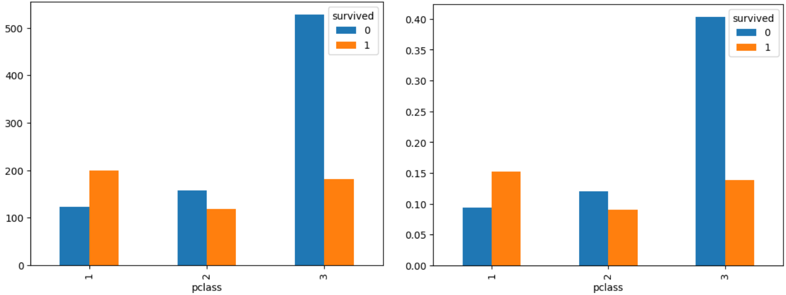
Visualizing Joint Distributions





How does the bar plot change if we plot the joint distribution instead of the counts?

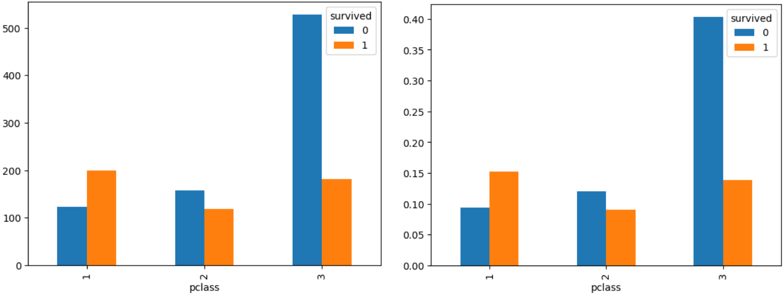
Visualizing Joint Distributions



How does the bar plot change if we plot the joint distribution instead of the counts?

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

Visualizing Joint Distributions



How does the bar plot change if we plot the joint distribution instead of the counts?

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

To appreciate the power of proportions, we need to look at conditional distributions.

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

crosstab

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

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

```
crosstab
```

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

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

```
crosstab
```

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

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

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

```
crosstab
```

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

Next, we divide the crosstab by the total in each class.

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

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

```
crosstab
```

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

Next, we divide the crosstab by the total in each class.

```
crosstab.divide(pclass_marginal, axis="rows")
```

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

Conditional Distributions

To compare survival across the classes, we should normalize by the total in each class.

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

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

```
crosstab
```

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

Next, we divide the crosstab by the total in each class.

```
crosstab.divide(pclass_marginal, axis="rows")
```

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

These are the **conditional distributions** of `survived` given `pclass`.

Visualizing Conditional Distributions

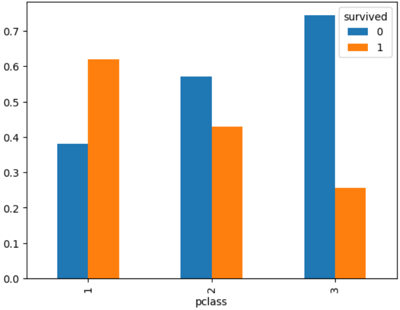
To visualize the conditional distributions, we could make a grouped bar plot...

```
(crosstab.divide(pclass_marginal, axis="rows").
 plot.bar())
```

Visualizing Conditional Distributions

To visualize the conditional distributions, we could make a grouped bar plot...

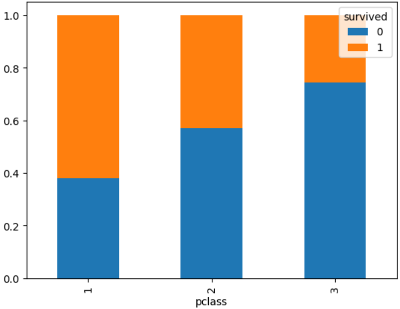
```
(crosstab.divide(pclass_marginal, axis="rows").
 plot.bar())
```



Visualizing Conditional Distributions

To visualize a conditional distribution, we could make a grouped bar plot...

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



...but it is better to make a **stacked bar plot**.

Conditional Probabilities

What does it mean to say, “The conditional proportion of survival given 3rd class is 0.255289”?

Conditional Probabilities

What does it mean to say, “The conditional proportion of survival given 3rd class is 0.255289”?

One interpretation is as a **conditional probability**.

Conditional Probabilities

- What does it mean to say, "The conditional proportion of survival given 3rd class is 0.255289"?

- One interpretation is as a **conditional probability**.

- "If we were to pick a 3rd class passenger on the Titanic at random, the probability that they survived is 0.255289."

Conditional Probabilities

- What does it mean to say, "The conditional proportion of survival given 3rd class is 0.255289"?

- One interpretation is as a **conditional probability**.

- "If we were to pick a 3rd class passenger on the Titanic at random, the probability that they survived is 0.255289."

- We notate this as
  - $P(\text{survived}|\text{3rd class}) = 0.255289$

Conditional Probabilities

- What does it mean to say, "The conditional proportion of survival given 3rd class is 0.255289"?

- One interpretation is as a **conditional probability**.

- "If we were to pick a 3rd class passenger on the Titanic at random, the probability that they survived is 0.255289."

- We notate this as
  - $P(\text{survived}|\text{3rd class}) = 0.255289$