

Introduction to Data Science

Importing Recoding and Visualizing Data.

Joanna Bieri DATA101

Renaming Columns

REVERSE

DF

	Religious tradition	Sample Size
0	Buddhist	233
1	Catholic	6137

want

	religion	n
0	Buddhist	233
1	Catholic	6137

What is the command?

Renaming Columns

ANSWER

```
DF.rename(columns={'Religious tradition':'religion',  
                  'Sample Size':'n'}, inplace=True)
```

Old name, colon, new name. Short names with no spaces mean fewer typos later.

Rearranging Data

ASK

What does it mean to make a table *longer*?

What does it mean to make it *wider*?

Rearranging Data

ANSWER

- Longer (`pd.melt()`): column names become values in a new column. More rows, fewer columns.
- Wider (`.pivot()`): values in a column become new column names. Fewer rows, more columns.
- Same information either way. Pick the shape your analysis or plot needs.

Pivot

PREDICT

df

	name	exam	grade
0	Ana	one	92
1	Ben	one	70
2	Ana	two	88
3	Ben	two	81

```
df.pivot(index='name', columns='exam', values='grade')
```

What comes out? How many rows and columns?

Pivot

ANSWER

```
df.pivot(index='name', columns='exam', values='grade')
```

name	one	two
Ana	92	88
Ben	70	81

index gives the rows, columns gives the new column names, values fills the inside.

Melt

PREDICT

df

	name	math	sci
0	Ana	85	70
1	Ben	90	82

```
pd.melt(df, id_vars=['name'], var_name='subject',  
        value_name='score')
```

What comes out? How many rows?

Melt

ANSWER

```
pd.melt(df, id_vars=['name'], var_name='subject',  
        value_name='score')
```

	name	subject	score
0	Ana	math	85
1	Ben	math	90
2	Ana	sci	70
3	Ben	sci	82

2 people times 2 subjects = 4 rows. The old column names `math` and `sci` are now values.

Melt

REVERSE

DF

	religion	low	high	n
0	Buddhist	0.36	0.13	233
1	Catholic	0.36	0.19	6137

want

	religion	n	income	proportion
0	Buddhist	233	low	0.36
1	Catholic	6137	low	0.36
2	Buddhist	233	high	0.13
3	Catholic	6137	high	0.19

What is the command?

Melt

ANSWER

```
pd.melt(DF, id_vars=['religion','n'], var_name='income',  
        value_name='proportion')
```

Every column you want to keep goes in `id_vars`. Everything else gets melted.

Category Order

PREDICT

DF

	income
0	50-99k
1	under 30k
2	50-99k
3	30-49k

```
list(DF['income'].drop_duplicates())
```

What comes out?

Category Order

ANSWER

```
list(DF['income'].drop_duplicates())
```

```
['50-99k', 'under 30k', '30-49k']
```

Each value once, in the order it first shows up. Plotly uses that order for colors and the legend unless you say otherwise:

```
category_orders={'income':  
                  ['under 30k', '30-49k', '50-99k']}
```

Frequency or Proportion?

ASK

233 Buddhists and 7462 Evangelical Protestants
answered the survey.

To compare how each group's income is split, do you
plot frequency or proportion?

Frequency or Proportion?

ANSWER

- Proportion. Every bar has the same length (1), so you compare the splits directly.
- Frequency mostly shows which group had more people in the survey.
- Frequency is still useful when the size of the group is the point.

Filling in NaNs

PREDICT

DF

	brand
0	Brand 1
1	NaN
2	NaN
3	Brand 2
4	NaN

DF.ffill()

What comes out?

Filling in NaNs

ANSWER

`DF.ffill()`

	brand
0	Brand 1
1	Brand 1
2	Brand 1
3	Brand 2
4	Brand 2

Forward fill: each NaN gets the last real value above it.

Masking Out Rows

PREDICT

DF

	id	n
0	Brand 1	n
1	1234	8
2	8721	2
3	Brand 2	n
4	2156	3

```
mask = DF['id'].apply(lambda x: 'Brand' not in str(x))  
DF[mask]
```

Which rows are left?

Masking Out Rows

ANSWER

DF[mask]

	id	n
1	1234	8
2	8721	2
4	2156	3

The mask is False for the two Brand rows, so they drop out. The index keeps its old numbers.

Questions?

Questions?