

Introduction to Data Science

Data Wrangling Continued

Joanna Bieri DATA101

Combining Masks

PREDICT

DF

	hotel	is_canceled	country
0	Resort Hotel	1	USA
1	City Hotel	1	GBR
2	Resort Hotel	0	PRT
3	City Hotel	0	USA
4	Resort Hotel	1	FRA

```
mask = (DF['hotel'] == 'Resort Hotel') \
        & (DF['is_canceled'] == 1)
DF[mask]
```

Which rows come out?

Combining Masks

ANSWER

```
mask = (DF['hotel'] == 'Resort Hotel') \
        & (DF['is_canceled'] == 1)
DF[mask]
```

	hotel	is_canceled	country
0	Resort Hotel	1	USA
4	Resort Hotel	1	FRA

Rows 0 and 4: the only ones where both questions are True. Row 2 is a resort but was not canceled, so it is out. Parentheses around each question are required.

Combining Masks

PREDICT

DF

	hotel	is_canceled	country
0	Resort Hotel	1	USA
1	City Hotel	1	GBR
2	Resort Hotel	0	PRT
3	City Hotel	0	USA
4	Resort Hotel	1	FRA

```
mask = (DF['country'] == 'USA') \  
        | (DF['country'] == 'GBR')  
DF[mask]
```

Which rows come out this time?

Combining Masks

ANSWER

```
mask = (DF['country'] == 'USA') \
        | (DF['country'] == 'GBR')
DF[mask]
```

	hotel	is_canceled	country
0	Resort Hotel	1	USA
1	City Hotel	1	GBR
3	City Hotel	0	USA

Rows 0, 1, and 3: either question True is enough. Notice you cannot write `country == 'USA' or 'GBR'`. Each side has to be a full question.

Combining Masks

ASK

Last class we used `and` and `or`.

Today it is `&` and `|`.

Why the change? What happens if you use `and` here?

Combining Masks

ANSWER

- and asks one True/False question about the whole column. Pandas does not know how to answer that for 5 rows at once, so you get `ValueError: The truth value of a Series is ambiguous.`
- `&` and `|` work row by row: True and True on row 0, True and False on row 2, and so on.
- Same idea as last class, one row at a time, all rows at once.

Unique Values

PREDICT

DF

	market_segment	customer_type
0	Online TA	Transient
1	Online TA	Group
2	Direct	Transient
3	Online TA	Transient
4	Direct	Transient

```
DF[['market_segment']].drop_duplicates()
```

```
my_columns = ['market_segment', 'customer_type']  
DF[my_columns].drop_duplicates()
```

How many rows come out of each? Which row labels survive?

Unique Values

ANSWER

one column

	market_segment
0	Online TA
2	Direct

two columns

	market_segment	customer_type
0	Online TA	Transient
1	Online TA	Group
2	Direct	Transient

2 rows, then 3. With two columns a row is a duplicate only if the whole pair repeats. The first copy of each stays, with its original label.

Unique Values

PREDICT

DF

	market_segment	customer_type
0	Online TA	Transient
1	Online TA	Group
2	Direct	Transient
3	Online TA	Transient
4	Direct	Transient

```
my_columns = ['market_segment', 'customer_type']  
DF[my_columns].value_counts()
```

What comes out? How is it different from `drop_duplicates()`?

Unique Values

ANSWER

```
my_columns = ['market_segment', 'customer_type']  
DF[my_columns].value_counts()
```

market_segment	customer_type	count
Direct	Transient	2
Online TA	Transient	2
Online TA	Group	1

Same three pairs, but now with how many times each showed up, biggest first. `drop_duplicates` tells you what is there, `value_counts` tells you how much of it.

Unique Values

REVERSE

DF

	market_segment	customer_type
0	Online TA	Transient
1	Online TA	Group
2	Direct	Transient
3	Online TA	Transient
4	Direct	Transient

I want those counts as a regular data frame, three columns, so I can add a proportion column next. What do I add to the command?

Unique Values

ANSWER

```
my_columns = ['market_segment', 'customer_type']  
my_counts = DF[my_columns].value_counts()  
my_counts = my_counts.reset_index()
```

	market_segment	customer_type	count
0	Direct	Transient	2
1	Online TA	Transient	2
2	Online TA	Group	1

`reset_index()` turns the row labels back into ordinary columns and gives you a fresh 0, 1, 2. Now `my_counts['count']` is a column you can do math on.

Adding a Column

REVERSE

DF

	hotel	adults	children	babies
0	Resort Hotel	2	1	0
1	City Hotel	1	0	0
2	Resort Hotel	2	0	1
3	City Hotel	2	2	0
4	Resort Hotel	1	0	0

I want a new column called `little_ones` that is children plus babies.
What is the command?

Adding a Column

ANSWER

```
DF['little_ones'] = DF['children'] + DF['babies']
```

	children	babies	little_ones
0	1	0	1
1	0	0	0
2	0	1	1
3	2	0	2
4	0	0	0

A name that is not already a column, on the left of a single equals sign. Pandas adds the two columns row by row and saves the answer under the new name.

Adding a Column

ASK

Last class `DF.sort_values(...)` did not change `DF`.

This command did. What is the difference?

Adding a Column

ANSWER

- The single equals sign. `DF['little_ones'] = ...` is an assignment, it writes into the frame.
- `sort_values`, `drop_duplicates`, `masks`, and `value_counts` all hand you back a new object and leave DF alone.
- Rule of thumb: nothing changes unless there is an equals sign. That is also how you keep a result: give it a name.

Grouping

PREDICT

DF

	hotel	adults	children	babies
0	Resort Hotel	2	1	0
1	City Hotel	1	0	0
2	Resort Hotel	2	0	1
3	City Hotel	2	2	0
4	Resort Hotel	1	0	0

```
my_columns = ['adults', 'children', 'babies']  
DF.groupby(by=['hotel'])[my_columns].sum()
```

How many rows come out? What is in them?

Grouping

ANSWER

```
my_columns = ['adults', 'children', 'babies']  
DF.groupby(by=['hotel'])[my_columns].sum()
```

hotel	adults	children	babies
City Hotel	3	2	0
Resort Hotel	5	1	1

One row per group, two groups. Each cell is the column total inside that group. The hotel name became the row label.

Grouping

PREDICT

DF

	hotel	is_canceled	country
0	Resort Hotel	1	USA
1	City Hotel	1	GBR
2	Resort Hotel	0	PRT
3	City Hotel	0	USA
4	Resort Hotel	1	FRA

```
DF.groupby(by=['hotel']).sum()
```

No columns picked this time. What happens to the country column?

Grouping

ANSWER

```
DF.groupby(by=['hotel']).sum()
```

hotel	is_canceled	country
City Hotel	1	GBRUSA
Resort Hotel	2	USAPRTFRA

Summing words glues them together. No error, just nonsense. Pick the numerical columns first, that is why the earlier command had the list in the middle.

Grouping

REVERSE

DF

	hotel	adults	children	babies
0	Resort Hotel	2	1	0
1	City Hotel	1	0	0
2	Resort Hotel	2	0	1
3	City Hotel	2	2	0
4	Resort Hotel	1	0	0

I want the average number of adults per booking, one number for each hotel. What is the command?

Grouping

ANSWER

```
DF.groupby(by=['hotel'])[['adults']].mean()
```

hotel	adults
City Hotel	1.500000
Resort Hotel	1.666667

Same shape as the sum, just swap the last piece. `.min()`, `.max()`, `.median()`, `.describe()` all work in the same spot.

The Exercises

ASK

Exercise 1: what proportion of bookings were made less than a day ahead

by guests not from the USA?

What two numbers do you need, and how do you get each one?

The Exercises

ANSWER

- The number of bookings that match: build the mask with `&`, apply it, then `len()` of what comes back.
- The number of bookings in the whole data set:
`len(DF_raw_hotels)`.
- Divide. A proportion is always a part over a whole, so ask yourself what the whole is before you start typing.

Questions?

Questions?