DSFA

Spring 2020

Lecture 9

Table Methods, Maps, Booleans

Announcements

- Prelim 1
  - Next Thursday 2/27, 7:30-9PM, Goldwin Smith G64
  - Practice questions posted on Piazza, review sheet coming
  - You will get a list of functions; you may bring a double-sided sheet of notes you make yourself.

Announcements

Project 1

- Part 1 due Friday 2/21 5:59PM, Part 2 due Monday 3/2 at 5:59PM.
- You may work together with a partner from your section

Office hours

- Because of the break, prelim, and project, we will be shifting office hours next week; be sure to look at the schedule on the course website.

Combining Table Methods

Important Table Methods

```
t.select(column, …) or t.drop(column, …)
t.take([row, …]) or t.exclude([row, …])

t.sort(column, descending=False, distinct=False)
t.where(column, are.condition(...))

t.apply(function, column, …)
t.group(column) or t.group(column, function)
t.group([column, …]) or t.group([column, …], function)
t.pivot(cols, rows) or t.pivot(cols, rows, vals, function)

t.join(column, other_table, other_table_column)
```

Apply

The `apply` method creates an array by calling a function on every element in input column(s)

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(function_name, 'column_label')
```

(Demo)

Group

The `group` method aggregates all rows with the same value for a column into a single row in the result

- First argument: Which column to group by
- Second argument: (Optional) How to combine values
    - `len` — number of grouped values (default)
    - `sum` — total of all grouped values
    - `list` — list of all grouped values

(Demo)

Grouping By Two Columns

The `group` method can also aggregate all rows that share the combination of values in multiple columns

- First argument: A list of which columns to group by
- Second argument: (Optional) How to combine values

(Demo)

Pivot

- Cross-classifies according to two categorical variables
- Produces a grid of counts or aggregated values
- Two required arguments:
  - First: variable that forms column labels of grid
  - Second: variable that forms row labels of grid
- Two optional arguments (include both or neither)
  - `values`=‘column_label_to_aggregate’
  - `collect`=function_with_which_to_aggregate

(Demo)

Joining Two Tables

```
drinks.join('Cafe', discounts, 'Location')
```

Keep all rows in the table that have a match ...

… for the value in this column ...

… somewhere in this other table's ...

… column that contains matching values.

drinks

Drink	Cafe	Price
Milk tea	Panda Tea	4
Espresso	Gimme	2
Latte	Gimme	3
Espresso	Cafe Gola	2

discounts

Coupon	Location
25%	Panda Tea
50%	Gimme
5%	Gimme

Cafe	Drink	Price	Coupon
Gimme	Espresso	2	50%
Gimme	Espresso	2	5%
Gimme	Latte	3	50%
Gimme	Latte	3	5%
Panda Tea	Milk Tea	4	25%

The joined column is sorted automatically

(Demo)

Discussion Question

Generate a table with one row per cafe that has the name and discounted price of its cheapest discounted drink

`drinks`

Drink	Cafe	Price
Milk tea	Panda Tea	4
Espresso	Gimme	2
Coffee	Gimme	3
Espresso	Cafe Gola	2

`discounts`

Coupon	Location
25%	Panda Tea
50%	Gimme
5%	Gimme

`cheapest`

Cafe	Drink	Discounted Price
Panda Tea	Milk Tea	3
Gimme	Espresso	1

Bikes

Maps

Maps

A table containing columns of latitude and longitude values can be used to generate a map of markers

```
________.map_table(table, ...)
```

Either `Marker` or `Circle`

Column 0: latitudes
Column 1: longitudes
Column 2: labels
Column 3: colors
Column 4: sizes

Applies to all features:
`color='blue'`
`size=200`

Comparison

Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2                y = 3
```

Assignment statements

```
x > 1                x > y                y >= 3

x == y               x != 2               2 < x < 5
```

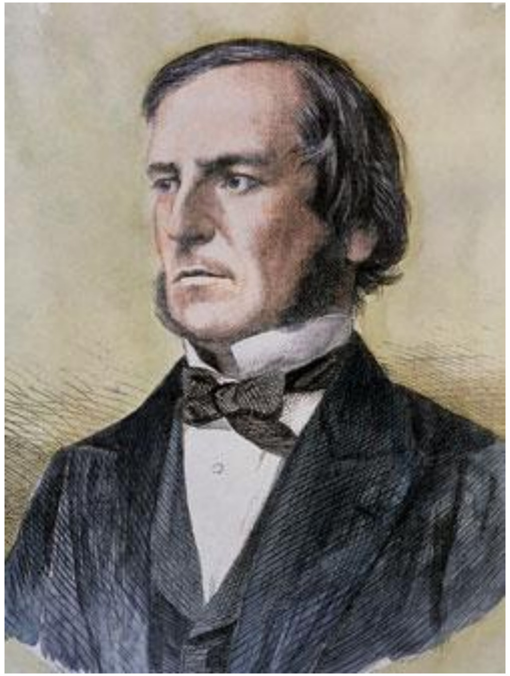
Comparison expressions

(Demo)

George Boole

The Laws of Thought (1854)

No general method for the solution of questions in the theory of probabilities can be established which does not explicitly recognise, not only the special numerical bases of the science, but also those **universal laws of thought which are the basis of all reasoning**, and which, whatever they may be as to their essence, are at least mathematical as to their form.



Combining Comparisons

Boolean operators can be applied to `bool` values

```
a = True        b = False
```

Evaluate to True

```
not b           a or b            a and not b
```

```
a and b         not (a or b)      b and b
```

Evaluate to False

(Demo)

Aggregating Comparisons

Summing an array or list of bool values will count the True values only.

```
1    + 0     + 1            == 2
True + False + True         == 2
sum([1    , 0     , 1    ]) == 2
sum([True, False, True])    == 2
```

(Demo)

Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2                   y = 3
```

Assignment statements

```
x > 1                   x > y                   y >= 3

x == y                  x != 2                  2 < x < 5
```

Comparison expressions

`t.where(array_of_bool_values)` returns a table with only the rows of `t` for which the corresponding `bool` is `True`.

(Demo)