Tables (`datascience.tables`)

Summary of methods for Table. Click a method to see its documentation.

One note about reading the method signatures for this page: each method is listed with its arguments. However, optional arguments are specified in brackets. That is, a method that's documented like

`Table.foo` (first_arg, second_arg[, some_other_arg, fourth_arg])

means that the `Table.foo` method must be called with first_arg and second_arg and optionally some_other_arg and fourth_arg. That means the following are valid ways to call `Table.foo`:

```
some_table.foo(1, 2)
some_table.foo(1, 2, 'hello')
some_table.foo(1, 2, 'hello', 'world')
some_table.foo(1, 2, some_other_arg='hello')
```

But these are not valid:

```
some_table.foo(1) # Missing arg
some_table.foo(1, 2[, 'hi']) # SyntaxError
some_table.foo(1, 2[, 'hello', 'world']) # SyntaxError
```

If that syntax is confusing, you can click the method name itself to get to the details page for that method. That page will have a more straightforward syntax.

At the time of this writing, most methods only have one or two sentences of documentation, so what you see here is all that you'll get for the time being. We are actively working on documentation, prioritizing the most complicated methods (mostly visualizations).

Creation

`Table.__init__`([labels, _deprecated, formatter])	Create an empty table with column labels.
`Table.empty`([labels])	Creates an empty table.
`Table.from_records`(records)	Create a table from a sequence of records (dicts with fixed keys).
`Table.from_columns_dict`(columns)	Create a table from a mapping of column labels to column values.
`Table.read_table`(filepath_or_buffer, *args, …)	Read a table from a file or web address.
`Table.from_df`(df)	Convert a Pandas DataFrame into a Table.
`Table.from_array`(arr)	Convert a structured NumPy array into a Table.

Extension (does not modify original table)

`Table.with_column`(label, values, *rest)	Return a new table with an additional or replaced column.
`Table.with_columns`(*labels_and_values)	Return a table with additional or replaced columns.
`Table.with_row`(row)	Return a table with an additional row.
`Table.with_rows`(rows)	Return a table with additional rows.
`Table.relabeled`(label, new_label)	Return a new table with `label` specifying column label(s) replaced by corresponding `new_label`.

Accessing values

`Table.num_columns`	Number of columns.
`Table.columns`	
`Table.column`(index_or_label)	Return the values of a column as an array.
`Table.num_rows`	Number of rows.
`Table.rows`	Return a view of all rows.
`Table.row`(index)	Return a row.
`Table.labels`	Return a tuple of column labels.

`Table.column_index`(label)	Return the index of a column by looking up its label.
`Table.apply`(fn, *column_or_columns)	Apply `fn` to each element or elements of `column_or_columns`.

Mutation (modifies table in place)

`Table.set_format`(column_or_columns, formatter)	Set the format of a column.
`Table.move_to_start`(column_label)	Move a column to the first in order.
`Table.move_to_end`(column_label)	Move a column to the last in order.
`Table.append`(row_or_table)	Append a row or all rows of a table.
`Table.append_column`(label, values)	Appends a column to the table or replaces a column.
`Table.relabel`(column_label, new_label)	Changes the label(s) of column(s) specified by `column_label` to labels in `new_label`.

Transformation (creates a new table)

`Table.copy`(*[, shallow])	Return a copy of a table.
`Table.select`(*column_or_columns)	Return a table with only the columns in `column_or_columns`.
`Table.drop`(*column_or_columns)	Return a Table with only columns other than selected label or labels.
`Table.take`()	Return a new Table with selected rows taken by index.
`Table.exclude`()	Return a new Table without a sequence of rows excluded by number.
`Table.where`(column_or_label[, …])	Return a new `Table` containing rows where `value_or_predicate` returns True for values in `column_or_label`.
`Table.sort`(column_or_label[, descending, …])	Return a Table of rows sorted according to the values in a column.
`Table.group`(column_or_label[, collect])	Group rows by unique values in a column; count or aggregate others.
`Table.groups`(labels[, collect])	Group rows by multiple columns, count or aggregate others.
`Table.pivot`(columns, rows[, values, …])	Generate a table with a column for each unique value in `columns`, with rows for each unique value in `rows`.
`Table.stack`(key[, labels])	Takes k original columns and returns two columns, with col.
`Table.join`(column_label, other[, other_label])	Creates a new table with the columns of self and other, containing rows for all values of a column that appear in both tables.
`Table.stats`([ops])	Compute statistics for each column and place them in a table.
`Table.percentile`(p)	Return a new table with one row containing the pth percentile for each column.
`Table.sample`([k, with_replacement, weights])	Return a new table where k rows are randomly sampled from the original table.
`Table.split`(k)	Return a tuple of two tables where the first table contains `k` rows randomly sampled and the second contains the remaining rows.
`Table.bin`(*columns, **vargs)	Group values by bin and compute counts per bin by column.

Exporting / Displaying

`Table.show`([max_rows])	Display the table.
`Table.as_text`([max_rows, sep])	Format table as text.
`Table.as_html`([max_rows])	Format table as HTML.
`Table.index_by`(column_or_label)	Return a dict keyed by values in a column that contains lists of rows corresponding to each value.
`Table.to_array`()	Convert the table to a structured NumPy array.
`Table.to_df`()	Convert the table to a Pandas DataFrame.
`Table.to_csv`(filename)	Creates a CSV file with the provided filename.

Visualizations

`Table.plot`([column_for_xticks, select, …])	Plot line charts for the table.
`Table.bar`([column_for_categories, select, …])	Plot bar charts for the table.
`Table.barh`([column_for_categories, select, …])	Plot horizontal bar charts for the table.
`Table.pivot_hist`(pivot_column_label, …[, …])	Draw histograms of each category in a column.
`Table.hist`(*columns[, overlay, bins, …])	Plots one histogram for each column in columns.
`Table.scatter`(column_for_x[, select, …])	Creates scatterplots, optionally adding a line of best fit.
`Table.boxplot`(**vargs)	Plots a boxplot for the table.