Tablicious for Octave

version 0.1.0-SNAPSHOT, March 2019

Andrew Janke

Short Contents

Table of Contents

1 Introduction

This is the manual for the Tablicious package version 0.1.0-SNAPSHOT for GNU Octave.

Tablicious provides Matlab-compatible tabular data support for GNU Octave. This includes a `table` class with support for filtering and join operations, Missing Data support, and `string` and `categorical` data types.

This document is a work in progress. You are invited to help improve it and submit patches.

Tablicious was written by Andrew Janke `<floss@apjanke.net>`. Support can be found on the Tablicious project GitHub page (`https://github.com/apjanke/octave-tablicious`).

2 Getting Started

The easiest way to obtain Tablicious is by using Octave's `pkg` package manager. To install the development prerelease of Tablicious, run this in Octave:

```
    pkg install https://github.com/apjanke/octave-tablicious/releases/download/v0.1.0-SNAP
```

(Check the releases page at `https://github.com/apjanke/octave-tablicious/releases` to find out what the actual latest release number is.)

For development, you can obtain the source code for Tablicious from the project repo on GitHub at `https://github.com/apjanke/octave-tablicious`. Make a local clone of the repo. Then add the `inst` directory in the repo to your Octave path.

3 Table Representation

Tablicious provides the `table` class for representing tabular data.

3.1 `table` Class

A `table` is an array object that represents a tabular data structure. It holds multiple named “variables”, each of which is a column vector, or a 2-D matrix whose rows are read as records.

4 Missing Functionality

Tablicious is based on Matlab's table API and supports most of its major functionality. But not all of it is implemented yet. The missing parts are currently:

- `timetable`
- Moving window methods in `fillmissing`
- `summary()` for `table` and `categorical`
- Assignment to table variables using `.`-indexing
- File I/O like `readtable()` and `writetable()`

It is the author's hope that all these will be implemented some day.

5 Function Reference

5.1 Functions by Category

5.1.1 Tables

Section 5.2.19 [table], page 7
    Tabular data array containing multiple columnar variables.

Section 5.2.1 [array2table], page 5
Section 5.2.3 [cell2table], page 5
Section 5.2.18 [struct2table], page 6
Section 5.2.20 [tableOuterFillValue], page 8

5.1.2 Data Types

Section 5.2.17 [string], page 6
Section 5.2.16 [startsWith], page 6
Section 5.2.8 [endsWith], page 6
Section 5.2.5 [contains], page 6
Section 5.2.2 [categorical], page 5
Section 5.2.6 [discretize], page 6

5.1.3 Missing Data

Section 5.2.9 [fillmissing], page 6
Section 5.2.10 [ismissing], page 6
Section 5.2.14 [rmmissing], page 6
Section 5.2.15 [standardizemissing], page 6
Section 5.2.12 [missing], page 6
Section 5.2.11 [isnannish], page 6

5.1.4 Miscellaneous

Section 5.2.4 [colvecfun], page 6
Section 5.2.7 [dispstrs], page 6
Section 5.2.13 [pp], page 6

5.2 Functions Alphabetically

5.2.1 array2table

Not documented

5.2.2 categorical

Not documented

5.2.3 cell2table

Not documented

5.2.4 colvecfun

Not documented

5.2.5 contains

Not documented

5.2.6 discretize

Not documented

5.2.7 dispstrs

Not documented

5.2.8 endsWith

Not documented

5.2.9 fillmissing

Not documented

5.2.10 ismissing

Not documented

5.2.11 isnannish

Not documented

5.2.12 missing

Not documented

5.2.13 pp

Not documented

5.2.14 rmmissing

Not documented

5.2.15 standardizemissing

Not documented

5.2.16 startsWith

Not documented

5.2.17 string

Not documented

5.2.18 struct2table

Not documented

5.2.19 table

`table` [Class]

Tabular data array containing multiple columnar variables.

A `table` is a tabular data structure that collects multiple parallel named variables. Each variable is treated like a column. (Possibly a multi-columned column, if that makes sense.) The types of variables may be heterogeneous.

A table object is like an SQL table or resultset, or a relation, or a DataFrame in R or Pandas.

A table is an array in itself: its size is *nrows*-by-*nvariables*, and you can index along the rows and variables by indexing into the table along dimensions 1 and 2.

`cellstr VariableNames` [Instance Variable of `table`]

The names of the variables in the table, as a cellstr row vector.

`cell VariableValues` [Instance Variable of `table`]

A cell vector containing the values for each of the variables. `VariableValues(i)` corresponds to `VariableNames(i)`.

`cellstr RowNames` [Instance Variable of `table`]

An optional list of row names that identify each row in the table. This is a cellstr column vector, if present.

5.2.19.1 table.table

`obj` `= table ()` [Constructor]

Constructs a new empty (0 rows by 0 variables) table.

`obj` `= table (`*`var1`*`,` *`var2`*`, ...,` *`varN`*`)` [Constructor]

Constructs a new table from the given variables. The variables passed as inputs to this constructor become the variables of the table. Their names are automatically detected from the input variable names that you used.

`obj` `= table ('Size',` *`sz`*`, 'VariableTypes',` *`varTypes`*`)` [Constructor]

Constructs a new table of the given size, and with the given variable types. The variables will contain the default value for elements of that type.

`obj` `= table (..., 'VariableNames',` *`varNames`*`)` [Constructor]
`obj` `= table (..., 'RowNames',` *`rowNames`*`)` [Constructor]

Specifies the variable names or row names to use in the constructed table. Overrides the implicit names garnered from the input variable names.

5.2.19.2 table.summary

`summary (`*`obj`*`)` [Method]
`s` `= summary (`*`obj`*`)` [Method]

Summary of table's data.

Displays or returns a summary of data in the input table. This will contain some statistical information on each of its variables.

This method is not implemented yet.

5.2.19.3 table.prettyprint

`prettyprint (`*`obj`*`)` [Method]

Display table's values in tabular format. This prints the contents of the table in human-readable, tabular form.

Variables which contain objects are displayed using the strings returned by their `dispstrs` method, if they define one.

5.2.19.4 table.table2cell

`c` `= table2cell (`*`obj`*`)` [Method]

Converts table to a cell array. Each variable in *obj* becomes one or more columns in the output, depending on how many columns that variable has.

Returns a cell array with the same number of rows as *obj*, and with as many or more columns as *obj* has variables.

5.2.20 tableOuterFillValue

Not documented

6 Copying

6.1 Package Copyright

Tablicious for Octave is covered by the GNU GPLv3.

All the code in the package is GNU GPLv3.

The Fisher Iris dataset is Public Domain.

6.2 Manual Copyright

This manual is for Tablicious, version 0.1.0-SNAPSHOT.