ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-spreadsheet

Peter Reutemann

June 10, 2013

DRAFT

©2012

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-sa/3.0/

Contents

1 Introduction **7**

2 Flow **9**

3 Tools **13**
3.1 Filtering and processing 14
3.2 Plug-ins 14
3.2.1 View plug-ins 14
3.2.2 Data plug-ins 15

4 Formulas **17**

Bibliography **19**

List of Figures

3.1 Viewer for spreadsheet files. 13
3.2 Row finder setup. 14
3.3 The filtered spreadsheet. 14

Chapter 1

Introduction

Tabular data is a very common data format, not only for machine learning. The *spreadsheet* module offers some basic spreadsheet support for reading/writing and some generic actors. Other modules, like the *odf* or *excel* one, offer other native readers and writers. The data read by these readers can be processed with the same actors.

Chapter 2

Flow

A lot of the following actors and conversion schemes offer either an index of a column or a range of columns as a parameter. In addition to the usual *first/last/...* placeholders, you can use the actual column names (case-insensitive). This makes an actor less error-prone, in case the order of the columns may change.

The following sources are available:

- *NewSpreadSheet* – for creating an empty spreadsheet with pre-defined columns.
- *SpreadSheetDbReader* – turns results from SQL queries into spreadsheet objects.[1]
- *LookUp* – Outputs a value from a stored lookup table, identified by a user-supplied key.[2]
- *LookUpTable* – Outputs a stored lookup table as spreadsheet.[3]

The following transformers are available:

- *LookUp* – obtains the value associated with the string received as input from an internally stored lookup table.[4]
- *LookUpAdd* – adds a key/value pair to a lookup table.[5]
- *LookUpInit* – initializes a lookup table by using two columns from a spreadsheet, one acting as key, the other as value.[6]
- *LookUpRemove* – removes a key/value pair from a lookup table.[7]
- *SpreadSheetAnonymize* – for anonymizing columns.[8]
- *SpreadSheetColumnFilter* – filters columns using a column finder scheme.[9]
- *SpreadSheetColumnIterator* – iterates over all the columns in the spreadsheet and outputs the names.[10]

[1] adams-spreadsheet-database_access.flow
[2] adams-spreadsheet-lookup.flow
[3] adams-spreadsheet-lookup.flow
[4] adams-spreadsheet-lookup.flow
[5] adams-spreadsheet-lookup.flow
[6] adams-spreadsheet-lookup.flow
[7] adams-spreadsheet-lookup.flow
[8] adams-spreadsheet-anonymize_columns.flow
[9] adams-spreadsheet-filter_columns.flow
[10] adams-spreadsheet-iterate_cols.flow

- *SpreadSheetColumnsByName* – generates a new spreadsheet with only the columns that match a regular expression (inverting is possible as well).
- *SpreadSheetConvertCells* – applies arbitrary conversion schemes to individual cells.
- *SpreadSheetCopyColumns* – copies the content of a range of columns into new columns.
- *SpreadSheetDifference* – computes the difference between two spreadsheets.
- *SpreadSheetFileReader* – for reading spreadsheet files; depending on the reader, multiple sheets can get read at once.[11]
- *SpreadSheetGetCell* – retrieves the value of a specific cell in the spreadsheet.[12]
- *SpreadSheetGetColumnIndex* – retrieves the 1-based index columns which name matches a regular expression.[13]
- *SpreadSheetGetHeaderCell* – retrieves the value of a specific cell in the header row of a spreadsheet.
- *SpreadSheetInfo* – generates basic information on the spreadsheet object.
- *SpreadSheetInsertColumn* – inserts a column in the spreadsheet, initializes the cells with a user-defined value.
- *SpreadSheetInsertRow* – inserts a row in the spreadsheet, initializes the cells with a user-defined value.
- *SpreadSheetMerge* – merges multiple spreadsheets into a single one.[14]
- *SpreadSheetPlotGenerator* – turns a spreadsheet into plot containers to be displayed in the *SequencePlotter* sink.[15]
- *SpreadSheetRemoveColumn* – removes columns from a spreadsheet.
- *SpreadSheetRemoveRow* – removes rows from a spreadsheet.
- *SpreadSheetReorderColumns* – reorders columns in a spreadsheet, also allows duplicating/dropping of columns.
- *SpreadSheetReplaceCellValue* – replaces cell values that match a regular expression.
- *SpreadSheetRowFilter* – filters rows using a row finder scheme.[16]
- *SpreadSheetSetCell* – sets the value of a specific cell in the spreadsheet.[17]
- *SpreadSheetSetHeaderCell* – sets the value of a specific cell in the header row of a spreadsheet.
- *SpreadSheetSort* – sorts a spreadsheet using an arbitrary number of columns (ascending or descending).
- *SpreadSheetStatistic* – calculates statistics using the data stored in the spreadsheet.[18]
- *SpreadSheetSubset* – for obtaining a subset of the spreadsheet object (subset of columns and/or rows).
- *SpreadSheetStorageRowIterator* – iterates over a range of rows/columns

[11] adams-spreadsheet-output_cells.flow
[12] adams-spreadsheet-output_cells.flow
[13] adams-spreadsheet-get_column_index.flow
[14] adams-spreadsheet-simple_merge.flow, adams-spreadsheet-merge_using_id.flow
[15] adams-spreadsheet-spreadsheet_plot1.flow, adams-spreadsheet-spreadsheet_plot2.flow, adams-spreadsheet-statistic.flow
[16] adams-spreadsheet-filter_rows.flow
[17] adams-spreadsheet-set_cells.flow
[18] adams-spreadsheet-statistic.flow

and stores the cell values in internal storage.
- *SpreadSheetTransformCells* – applies an arbitrary global transfromer to individual cells.
- *SpreadSheetVariableRowIterator* – iterates over a range of rows/columns and stores the cell values in variables.[19]

The following sinks are available:

- *SpreadSheetDbWriter* – for storing a spreadsheet in a database.[20]
- *SpreadSheetDisplay* – for displaying spreadsheet objects in tabular form.[21]
- *SpreadSheetFileWriter* – writes spreadsheet objects to a file with the chosen writer class; depending on the writer either a single or multiple sheets can get written at once.

The following conversion schemes are available:

- *AggregateSpreadSheet* – aggregates rows in a spreadsheet (min, max, average, standard deviation, etc).
- *ContainerToSpreadSheet* – for converting any flow container (e.g., a prediction container) into a spreadsheet for better visualization.
- *ConvertSpreadSheetRows* – converts the data rows in a spreadsheet into a different format, e.g., into sparse representation.
- *DoubleMatrixToSpreadSheet* – converts a two-dimensional double array (i.e., matrix) into a spreadsheet object.
- *NotesToSpreadSheet* – generates a spreadsheet from a Notes object.
- *RenameSpreadSheetColumn* – renames a single column in a spreadsheet.
- *SpreadSheetAddFormulaColumn* – adds a column with a row-wise, user-supplied formula.
- *SpreadSheetAddFormulaRow* – adds a row with a column-wise, user-supplied formula.
- *SpreadSheetAddSumColumn* – adds a column with a row-wise sum formula.
- *SpreadSheetAddSumRow* – adds a row with a column-wise sum formula.
- *SpreadSheetAnyColumnToString* – converts any data types in a column to strings.
- *SpreadSheetDoubleColumnToLong* – converts any floating values to integer (= long) values.
- *SpreadSheetDoubleColumnToString* – converts any floating values to strings.
- *SpreadSheetInsertCellLocation* – replaces a placeholder in a string with a cell location (e.g., "A1").
- *SpreadSheetJoinColumns* – joins two or more columns in a spreadsheet into a single one.
- *SpreadSheetStringSplitColumn* – splits the string representation of a column into multiple columns using a regular expression.
- *SpreadSheetStringColumnToBoolean* – converts strings values in a column to boolean objects.

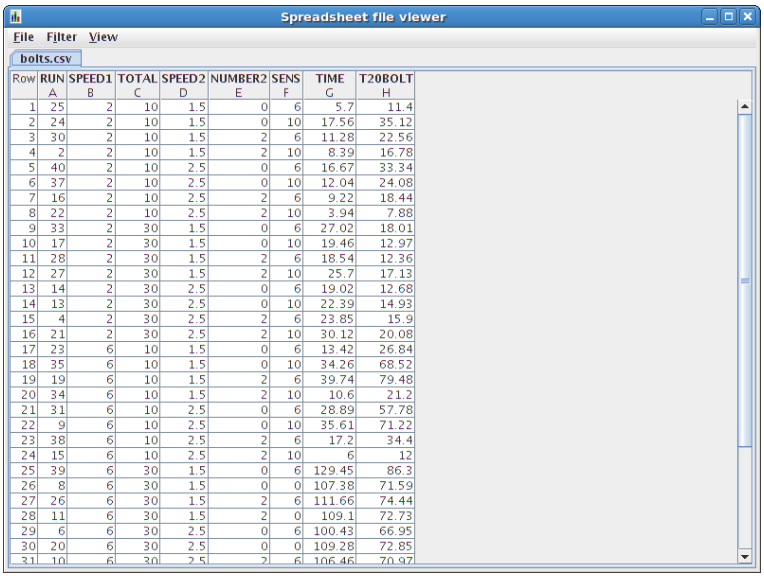
[19] adams-spreadsheet-variable_row_iterator.flow
[20] adams-spreadsheet-database_access.flow
[21] adams-spreadsheet-display.flow

- *SpreadSheetStringColumnToDate* – converts strings values in a column to date objects.
- *SpreadSheetStringColumnToDateTime* – converts strings values in a column to date/time objects.
- *SpreadSheetStringColumnToDouble* – converts strings values in a column to floating values.
- *SpreadSheetStringColumnToLong* – converts strings values in a column to integer (= long) values.
- *SpreadSheetStringColumnToTime* – converts strings values in a column to time objects.
- *SpreadSheetToDoubleMatrix* – turns all the numeric columns of a spreadsheet into a two-dimensional double matrix.
- *SpreadSheetUniqueColumnNames* – ensures that the column names uniquely identify a column.
- *StringToSpreadSheet* – parses a string in CSV format and turns it into a spreadsheet object.
- *TransposeSpreadSheet* – swaps columns with rows.

Chapter 3

Tools

The *Spreadsheet file viewer* is a simple tool for viewing all spreadsheet file formats that ADAMS supports. Figure 3.1 shows a dataset that was loaded from a CSV (comma-separated values) file.



Row	RUN	SPEED1	TOTAL	SPEED2	NUMBER2	SENS	TIME	T20BOLT
	A	B	C	D	E	F	G	H
1	25	2	10	1.5	0	6	5.7	11.4
2	24	2	10	1.5	0	10	17.56	35.12
3	30	2	10	1.5	2	6	11.28	22.56
4	2	2	10	1.5	2	10	8.39	16.78
5	40	2	10	2.5	0	6	16.67	33.34
6	37	2	10	2.5	0	10	12.04	24.08
7	16	2	10	2.5	2	6	9.22	18.44
8	22	2	10	2.5	2	10	3.94	7.88
9	33	2	30	1.5	0	6	27.02	18.01
10	17	2	30	1.5	0	10	19.46	12.97
11	28	2	30	1.5	2	6	18.54	12.36
12	27	2	30	1.5	2	10	25.7	17.13
13	14	2	30	2.5	0	6	19.02	12.68
14	13	2	30	2.5	0	10	22.39	14.93
15	4	2	30	2.5	2	6	23.85	15.9
16	21	2	30	2.5	2	10	30.12	20.08
17	23	6	10	1.5	0	6	13.42	26.84
18	35	6	10	1.5	0	10	34.26	68.52
19	19	6	10	1.5	2	6	39.74	79.48
20	34	6	10	1.5	2	10	10.6	21.2
21	31	6	10	2.5	0	6	28.89	57.78
22	9	6	10	2.5	0	10	35.61	71.22
23	38	6	10	2.5	2	6	17.2	34.4
24	15	6	10	2.5	2	10	6	12
25	39	6	30	1.5	0	6	129.45	86.3
26	8	6	30	1.5	0	0	107.38	71.59
27	26	6	30	1.5	2	6	111.66	74.44
28	11	6	30	1.5	2	0	109.1	72.73
29	6	6	30	2.5	0	6	100.43	66.95
30	20	6	30	2.5	0	0	109.28	72.85
31	10	6	30	2.5	2	6	106.46	70.97

Figure 3.1: Viewer for spreadsheet files.

If there are more spreadsheet file formats registered, you can save the currently displayed spreadsheet in another format. Printing, of course, is available throught the *Send to* menu. By default, the viewer displays each cell with as many digits after the decimal point as necessary. But you can also unify this and specify how many digits should be used for all floating point cells.

3.1 Filtering and processing

The viewer supports some basic filtering and processing:

- *columns* – creates a subset of the spreadsheet by selecting a subset of columns, e.g., all columns which name starts with a certain string.
- *rows* – creates a subset of the spreadsheet by selecting a subset of rows, e.g., rows with a certain value in a column.
- *convert* – applies a conversion scheme specific to spreadsheets, e.g., transposing a spreadsheet.
- *transform* – applies a flow transformer specific to spreadsheets, e.g., inserting a column or creating a subset.

Figures 3.2 and 3.3 show the setup for a row finder filter and the resulting new spreadsheet.

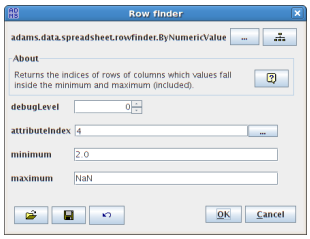


Figure 3.2: Row finder setup.

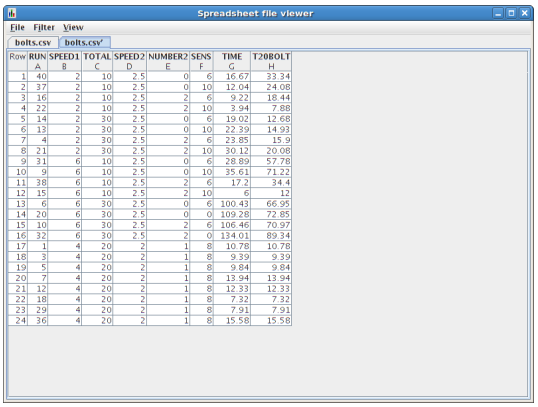


Figure 3.3: The filtered spreadsheet.

3.2 Plug-ins

The viewer can be extended with two sorts of plug-ins:

- ones that generate a view based on the current sheet ("view")
- ones that process the current sheet ("data")

3.2.1 View plug-ins

A view plug-in is derived from the following super-class:

```
adams.gui.tools.spreadsheetviewer.AbstractViewPlugin
```

There are three methods that need implementing:

- *getMenuText()* – returns the text used for the menu item and the title of the dialog displaying the generated view.
- *getMenuIcon()* – returns the name of the icon (no path) that should be displayed in the menu (use null to display no icon).

- *doGenerate(SpreadSheet)* – this method generates the actual view in form of a *adams.gui.core.BasePanel*.

An example is the *Statistics* plug-in, which shows simple statistics for a spreadsheet, number of rows and colums and what types of columns are present:

```
adams.gui.tools.spreadsheetviewer.Statistics
```

3.2.2 Data plug-ins

A view plug-in is derived from the following super-class:

```
adams.gui.tools.spreadsheetviewer.AbstractDataPlugin
```

There are four methods that need implementing:

- *getMenuText()* – returns the text used for the menu item and the title of the dialog displaying the generated view.
- *getMenuIcon()* – returns the name of the icon (no path) that should be displayed in the menu (use null to display no icon).
- *doProcess(SpreadSheet)* – this method processes the current spreadsheet and returns a new spreadsheet object.
- *isInPlace()* – returns whether the generated spreadsheet object should simply replace the current one ("in-place") or added as new tab.

Chapter 4

Formulas

ADAMS supports formulas with a range of basic functions. The following lists describe briefly what functionality is available. A full list is available through the *Help -¿ Formulas* menu in the *Spreadsheet file viewer*.

Operands:

- $NUM + NUM$ – addition
- $NUM - NUM$ – subtraction
- $NUM * NUM$ – multiplication
- NUM/NUM – division
- $NUM \wedge NUM$ – exponential
- $NUM\%NUM$ – modulo

Boolean operations:

- $<$ – less than
- $<=$ – less than or equal
- $>$ – greater than
- $>=$ – greated than or equal
- $=$ – equals
- $!=$ – does not equal
- ! – negation
- & – and
- | – or

Numeric functions:

- *abs* – absolute value of a cell/number.
- *average* – average computed from a range of cells.
- *ceil* – smallest value that is greater than or equal to the cell/number and is equal to a mathematical integer.
- *cos* – the trigonometric cosine of a number/cell.
- *exp* – Euler's number e raised to the power of a number/cell.
- *floor* – largest value that is less than or equal to the cell/number and is equal to a mathematical integer.

- *if[else]* – if-then-else construct.
- *log* – natural logarithm (base e) of a number/cell.
- *max* – largest value from range of cells.
- *min* – smallest value from range of cells.
- *pow[er]* – the first number/cell raised to the power of the second number/cell.
- *rint* – returns number that is closest in value to the number/cell and is equal to a mathematical integer.
- *sin* – trigonometric sine of a number/cell.
- *sqrt* – the correctly rounded positive square root of a number/cell.
- *stdev* – the sample standard deviation from a range of cells.
- *stdevp* – the population standard deviation from a range of cells.
- *sum* – the sum over a range of cells.
- *tan* – trigonometric tangent of a number/cell.

String functions:

- *length* – the length of a string.
- *lowercase* – converts string to a lowercase one.
- *matches* – matches a string against a regular expression.
- *substr* – creates a substring from a string, given start/end position.
- *uppercase* – converts string to an uppercase one.

Bibliography

[1] *ADAMS* – Advanced Data mining and Machine learning System
`https://adams.cms.waikato.ac.nz/`