


Grafterizer Tool for Data Cleaning and Transformation: User Guide

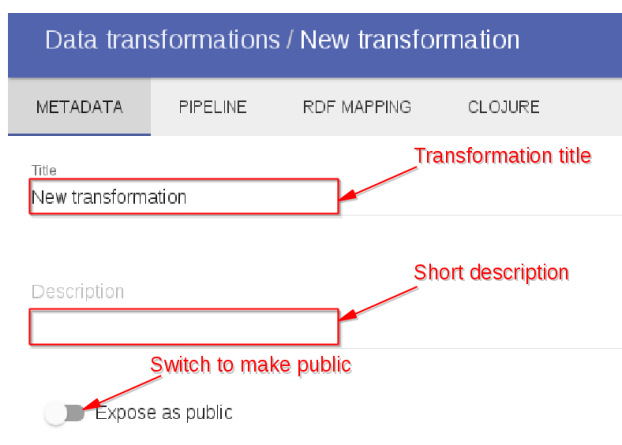
July 24, 2015

This tutorial describes core functionality of the Grafterizer data cleaning and transformation tool ...

1 Transformation metadata

In this section user defines transformation title and gives a short description of how this transformation processes given data. If user wishes to share transformation, it is possible to expose it as public. In this case other platform users will be able to explore and use given transformation.

After describing transformation metadata user may save transformation by clicking “Save” button  in the top right corner. Transformation may be as well saved later at any moment.



The screenshot shows a web interface for creating a new transformation. At the top is a blue header bar with the text "Data transformations / New transformation". Below this is a tabbed interface with four tabs: "METADATA", "PIPELINE", "RDF MAPPING", and "CLOSURE". The "METADATA" tab is currently selected. The form contains three main sections: 1. "Title" section with a text input field containing the text "New transformation". A red arrow points from the label "Transformation title" to this input field. 2. "Description" section with a text input field. A red arrow points from the label "Short description" to this input field. 3. A section with a toggle switch and the text "Expose as public". A red arrow points from the label "Switch to make public" to the toggle switch.

Figure 1: Transformation metadata

2 Transformation pipeline

The Grafterizer tool performs tabular data cleaning and transformation with help of “pipeline” concept. To begin with, each single transformation step is defined as a pipe – a function that performs simple data conversion. The greatest fact about these functions is that they may be combined together in a such way, that output of one pipe acts as an input for another. This way of composition gives a great flexibility and allows to perform rather complex data conversions.

Each transformation starts from reading a dataset, however user do not need to include this step into the transformation pipeline, since this action is performed automatically for each transformation.

To add a first transformation step click the  button next to the pipeline



Figure 2: Add pipe function

Now you can see the list of functions you may use to transform your data. Basically each function supports either functionality of Grafter function or combination of several functions to perform a single logical operation(like ”Split column”).

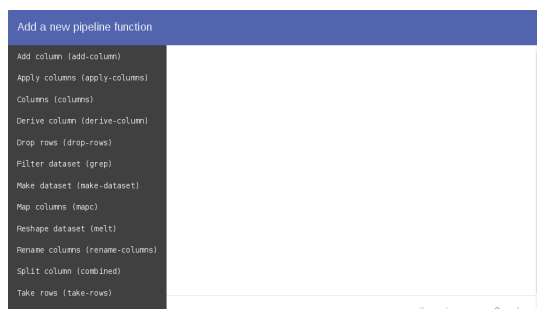


Figure 3: List of available functions

For each option you can see the function description with simple illustrated example by clicking ”show/hide documentation” button. You may also leave a

note about function you are creating in the "Comment" field. This information helps you and other users of your transformation to understand operations that are performed by this particular function. If you ignore this field, the note will be created automatically based on function parameters you have specified.

Add a new pipeline function

~Show/hide documentation Click to see basic usage

Parameters

Name of new column

Value

Expression

Comment Enter comment for your function

Create function Cancel

Figure 4: Common operations for all pipe functions

The sections 2.1-2.13 provide you with detailed guidelines for each function usage.

2.1 Add Column

The "Add column" function results in adding a new column to a dataset. To add a new column you should specify column name(this one will be converted to a Clojure keyword automatically) and a value for a new column. This value will be copied into every row within dataset. You allowed as well to use a Clojure expression to obtain value for a new column(this may be useful for instance to obtain current date, time or dataset name). Note that the "Expression" field is prioritized, in other words if you define both value and expression, only expression will be used to get a value for a new column.

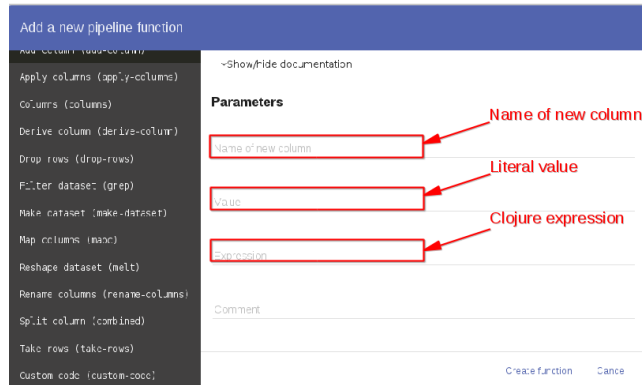


Figure 5: Add column function

2.2 Apply Columns

2.3 Columns

The "Columns" function narrows given dataset to just the specified columns. You may define these columns explicitly by writing their names in "Specify column names" field or by assigning number of columns to fetch. In second case, dataset will be narrowed to n columns with their names assigned as alphabetical sequence ("A", "B", "C" etc.). If more than 26 columns are fetched column names will count "AA", "AB", "AC" ... "BA", "BB", "BC" etc.

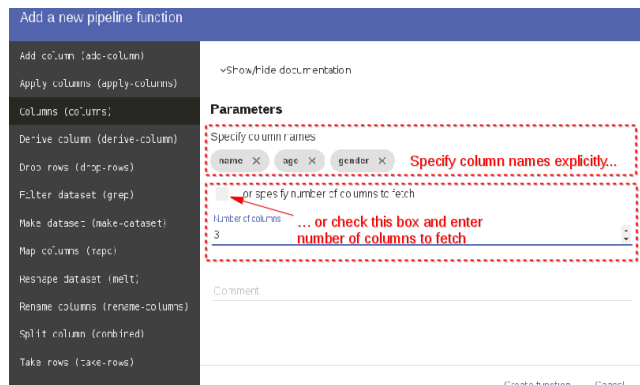




Figure 6: Columns function

2.4 Derive Column

This function creates a new column in a dataset by applying some transformation to existing column(or columns). To use this function you should specify name for a new column, define one or several columns you are going to use to obtain new value and specify a function you apply to them. This function can be chosen from a drop-down list, which contains some standard functions and custom utility functions. You may define custom utility functions by yourself using Clojure code(see 2.15) or use utility functions written by other users.

One powerful feature of derive-column and similar pipeline functions is possibility to combine functions you apply to columns in the same way you combine functions in pipelines. Essentially, these are pipelines inside your dataset manipulation pipeline. To add one more "internal" function into your pipeline function, click  button. Note, that function order is significant in this case. Functions are composed as they appear from top to down. To remove function from composition pipeline click trash icon  next to the function you wish to delete.

2.5 Drop Rows

This function just removes first n rows from a given dataset.

2.6 Filter Dataset

This method filters rows in the dataset for matches. This is a rather flexible filtering tool that allows you to filter your data in several different ways.

First field in in "Filter Dataset" parameter list specifies whether filter will be applied to specific column(s) or to all columns in dataset. For latter option just leave this field empty and matching will be performed for each column.

To perform actual data filtering you have several options.

First, it is possible to select only rows containing specified text. For this you enter the text in field marked as "Text to match".

You may as well filter dataset with help of regular expressions. By pressing  button next to the "Regular expression" field you will get short quick start tutorial for pattern usage.

Finally, you may filter dataset by applying utility functions to columns. Note, that the result of function(or combination of functions) will be treated a true/false expression.

The priority of listed option is defined as they appear - from top to down: if "Text to match" field is specified, other fields are ignored, if "Text to match" is ignored, but "Regular expression" is defined – this one will be used to filter your dataset ignoring functions below(if specified any).

2.7 Make Dataset

As its name suggests "Make dataset" operation creates new dataset from its input. If you leave all parameter fields blank new dataset will be created from all the input columns with column names given as simple alphabetic sequence. By checking "move first row to header" option you get all the column names from the first row. First row will be removed from your dataset. You may as well specify column names you wish to take to dataset being created or fetch first n columns.

2.8 Map Columns

This method allows you to apply function transformation to column and put the transformed value back to the same column. Transformation is done cell by cell. In parameter list you must specify a column you wish to change and a function that will be used to perform a transformation. You may add as many column-function pairs as you need.

2.9 Reshape Dataset

Reshape dataset "melts" given dataset in a such way, that each row of new dataset represents a unique combination of variables and values for given column array.

2.10 Rename Columns

This operation allows you to rename columns in dataset. To get new names for columns you may either apply function(or function pipeline) to current column names or assign mapping from old to new column names directly.

2.11 Split Column

2.12 Take Rows

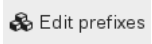
This operation allows you to narrow dataset just to first n rows.

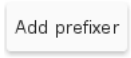
2.13 Adding Custom Function to Your Pipeline

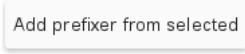
Some complex transformations cannot be done with help of operations described above. In this case you may need to define your own pipeline functions. This can be done with help of "Custom code" option using Clojure language

2.14 Creating and editing custom utility functions

2.15 Creating and editing prefixers

You may create and edit prefixers in the "Edit prefixers" window. To see this press  button in the pipeline view. Here you can see the list of all prefixers you created for current transformation. You may add a new prefixer by

specifying its name and URI and pressing  button. Created prefixer will instantly appear in the list of prefixers above. It is possible as well to create prefixer by adding some string to the existing one. In this case select a prefixer you wish to choose as base one, enter new prefixer name and string value and

press . Now, if the base prefixer is changed, changes will be as well applied to the child prefixers. To remove prefixers select all prefixers

you wish to remove and press .

Example 1. The following example illustrates ...