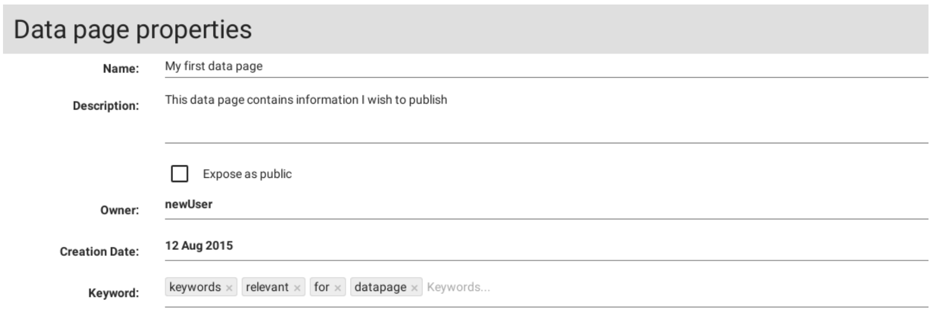
DataGraft Service for Data Publishing and Transformation: User Guide

This tutorial describes core functionality of the DataGraft platform and gives detailed step-by-step explanation of data publishing and transformation process with help of DataGraft portal.

Platform overview

The most demonstrable way to get an overview of what can be done with help of DataGraft platform is to explore data pages and data transformations that other users of this platform chose to share. The two terms mentioned above, **data pages** and **data transformations**, are two main concepts you work with while using DataGraft platform, therefore it may be useful to understand what each of them means in a context of the service.

Data pages contain cleaned and transformed data you want to publish. Each data page is stored with some metadata, including data page name, short description, keywords, describing a data page, owner, creation date and public/private property (see Figure **??**). The latter is defined by data page owner and specifies whether this data page can be explored by other users of a platform or not.


Data page properties

Name:	My first data page
Description:	This data page contains information I wish to publish
	☐ Expose as public
Owner:	newUser
Creation Date:	12 Aug 2015
Keyword:	keywords × relevant × for × datapage × Keywords...

Figure 1: Data page properties

Users that have access to the data page (i.e. just owners in case of private pages and everyone else for public pages) can locally download information associated with a data page in a raw tabular format by pressing EXPORT RAW button; or as RDF by pressing EXPORT RDF. The list of supported RDF formats includes RDF/XML(.rdf), n-triple(.nt), turtle(.ttl), n3(.n3), nquads(.nq), RDF/JSON(.rj). View portal, SPARQL

Another type of asset that users may create and share in DataGraft is **data transformation**. Before publishing data, in most cases you will need to transform the original dataset – clean messy data, remove unnecessary information, probably add some new data fields and convert tabular data to RDF. This sequence of operations you perform on your data to convert it to desirable form is called data transformation. The greatest thing about data transformations is that you may reuse them repeatably on another datasets, share them with other users, modify existing transformations and create new transformations by extending those you or other users created and shared.

User Registration

In order to create your own data pages and transformations through a platform, you should first sign up for DataGraft account. After registration you are automatically redirected to data page creation service, from where you may start a process of creating your first data page. This process is described in detail in section **??**.

User Dashboard

User dashboard helps to manage data pages and data transformations created by user. The dashboard view gives you an overview of your data pages and transformations with search capabilities.

Publishing data

Publishing data with help of DataGraft platform is rather simple process. You may start by switching to a Publish tab in a main menu. The first thing you do is uploading data. To do so you just drop your dataset file in a raw CSV or RDF format in a white frame under "Upload your data" label.

After data is succesfully uploaded(this is indicated by green mark in the top right corner of a file icon) you have several options:

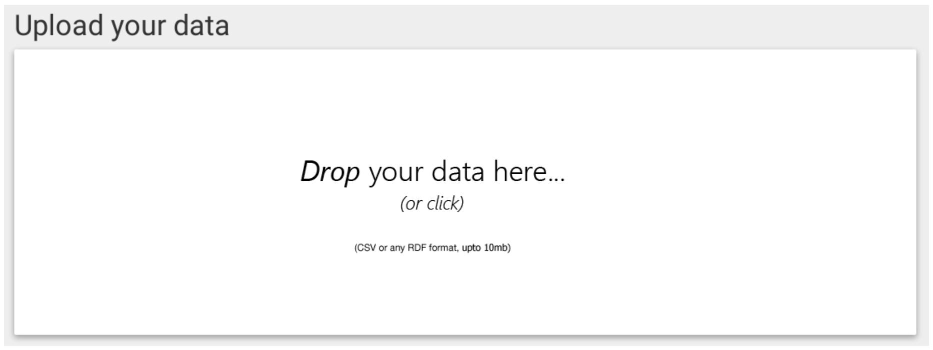


Figure 2: Uploading data

1. Create data page from a row data without applying any transformation on it.
2. Transform your data before creating a datapage. This in turn can be done in two ways
   (a) By creating a new transformation to use on your data
   (b) By applying an existing transformation to your data

Data cleaning and transformation

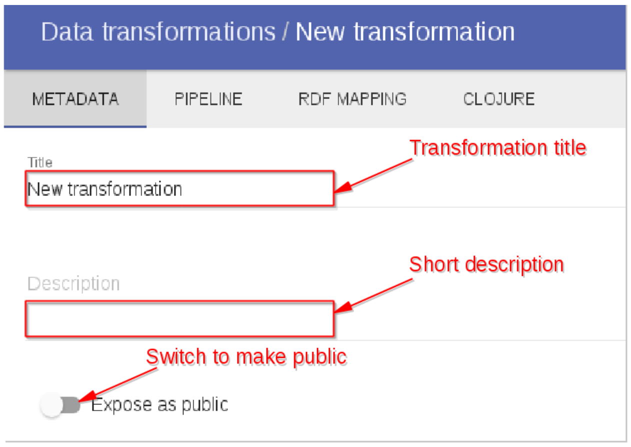
5.1 Transformation metadata

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.

5.2 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.



Title: New transformation

Description:

Figure 3: Transformation metadata

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

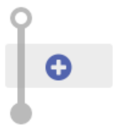


Figure 4: Add pipe function

Now you can see the list of functions you may use to transform your data. Available functions are logically grouped according to the type of effect they have on a dataset. OPERATIONS ON COLUMNS add, remove or modify dataset columns, while OPERATIONS ON ROWS extract certain rows from a dataset based on row numbers or some condition that user defines. Operations "Make dataset" and "Reshape dataset" affect the entire dataset.

For each operation you can see a short documentation with simple illustrated example by clicking "show/hide documentation" button. For any pipeline function you create you may leave a short description note in the "Comment" field.

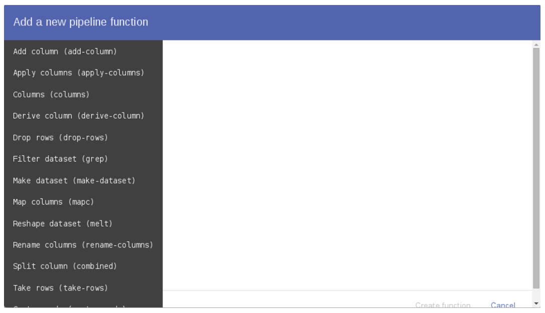


Figure 5: List of available functions

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.

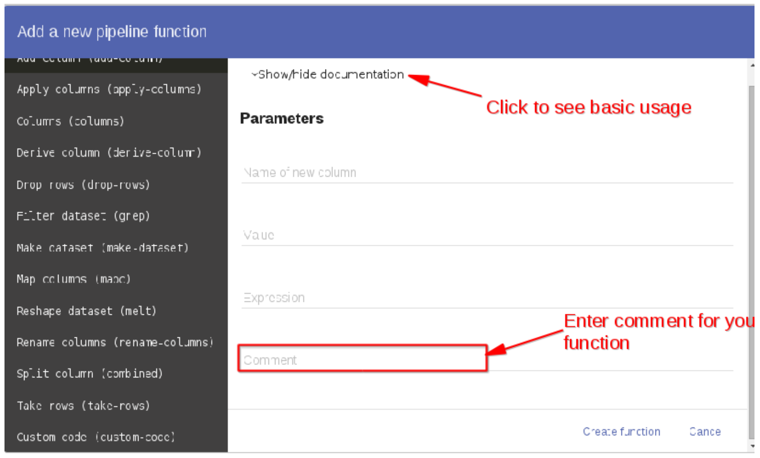


Figure 6: Common operations for all pipe functions

The sections **??**-**??** provide you with detailed guidelines for each function usage.

5.2.1 Add Columns

The "Add columns" results in adding a new column(s) 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 may as well populate new column with some custom value. This may be current date, dataset filename or custom Clojure code. 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.

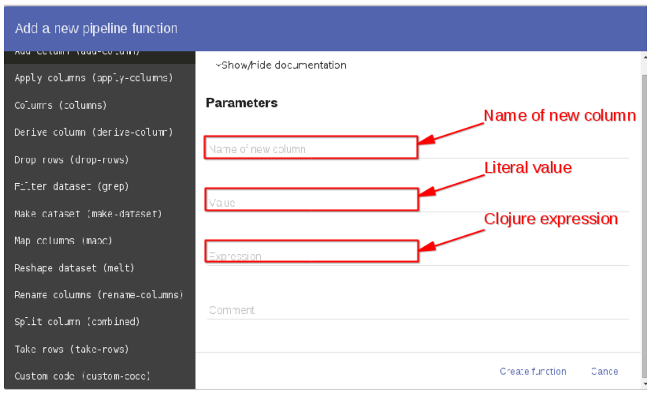


Figure 7: Add column function

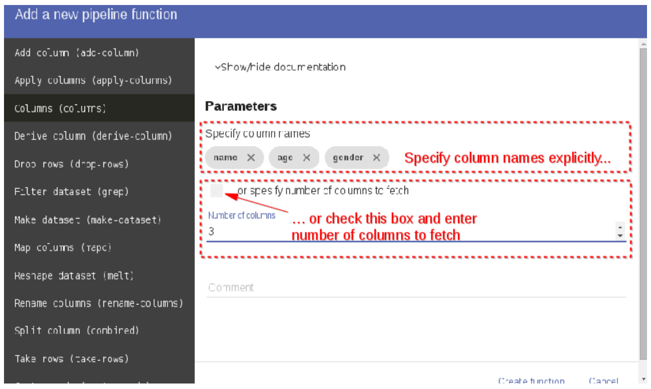
5.2.2 Apply Columns

5.3 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 alphabletical sequence("A","B","C" etc.). If more than 26 columns are fetched column names will count "AA", "AB", "AC" ... "BA", "BB", "BC" etc.

5.3.1 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



Parameters

Specify column names: name, age, gender — Specify column names explicitly...

or specify number of columns to fetch — ... or check this box and enter number of columns to fetch

Number of columns: 3

Comment

Figure 8: Columns function

custom utility functions. You may define custom utility functions by yourself using Clojure code(see **??**) 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 Compose with 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.

5.3.2 Drop Rows

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

5.4 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". By default this text is treated as case-sensitive. To perform a case insensitive filtering check the box "ignore case" next to the text field.

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).

5.4.1 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.

5.4.2 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.

5.4.3 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.

5.4.4 Remove Columns

This operation allows you to narrow dataset to all but specified columns.

5.4.5 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.

5.4.6 Split Column

This operation allows to split column on some separator. Column is splitted once, leaving substring obtained as original text before first instance of separator in the source column and copying text after separator into new column. Separator itself is extracted from text. Name of new column is optional.

5.4.7 Take Rows

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

5.4.8 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

5.4.9 Creating and editing custom utility functions

5.4.10 Creating and editing prefixers

You may create and edit prefixers in the "Edit prefixers" window. To see this press Edit prefixes 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 Add prefixer buttom. 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 Add prefixer from selected.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 Remove selected.

Example 1. The following example illustrates ...