DataGraft Service for Data Publishing and Transformation: User Guide

Contents

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 transformations

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 sometimes 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 in DataGraft platform is that you may modify existing transformations, share them with other users, reuse them repeatedly on other datasets and create new transformations by extending ones that you or other users created and shared(fork transformations).

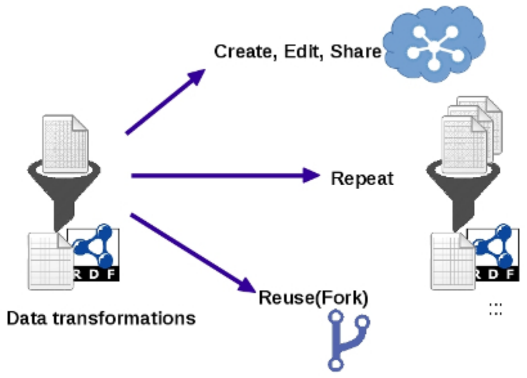
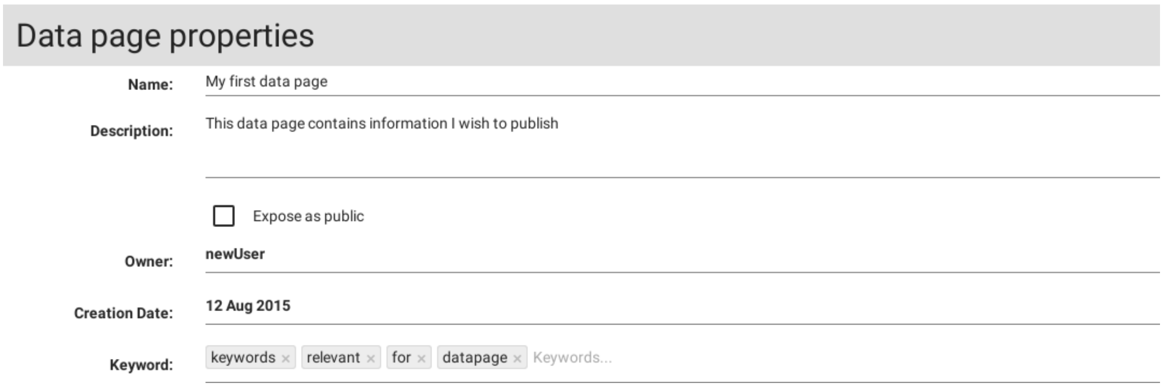


Figure 1: Transformations features

Data transformations are stored with some metadata. The transformation properties include transformation name, description, owner and public/private property. They are defined on a data transformation creation stage (see Section).

Data pages

Another type of asset that users may create and share in DataGraft is **data page**. Data pages contain cleaned and transformed data you want to publish. As well as data transformations, data pages are stored with some metadata, including data page name; short description; keywords, describing a data page; owner; creation date and public/private property (see Figure 2) . 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.



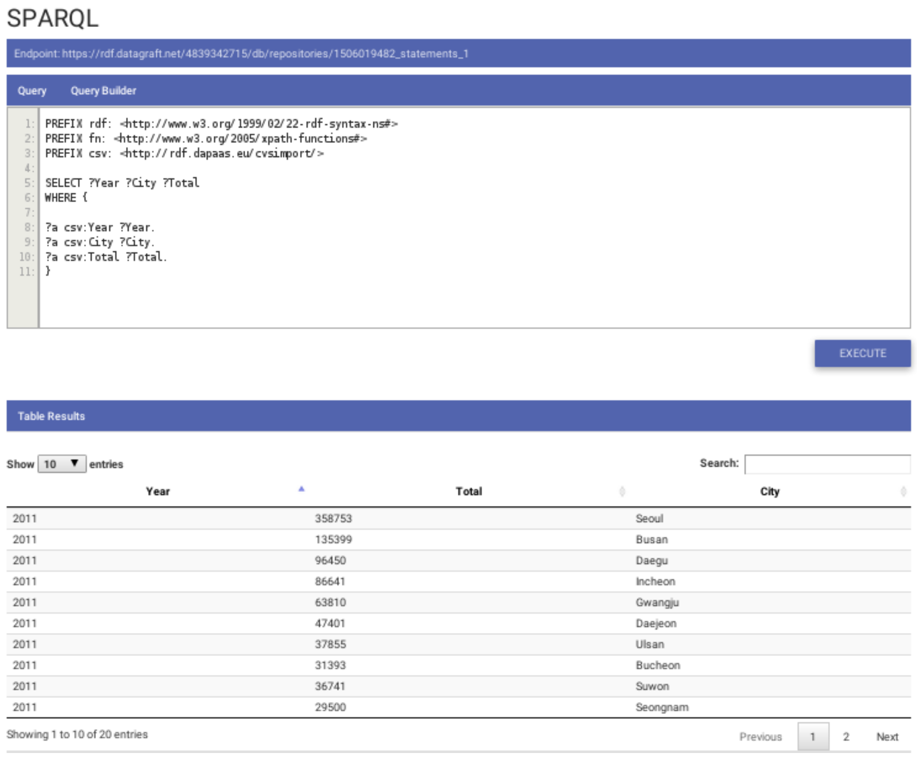
Data page properties

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 2: 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).

Data pages containing RDF also allow you to perform SPARQL querying on data they contain. The example of such query and selection results are shown below.


SPARQL
Endpoint: https://rdf.datagraft.net/4839342715/db/repositories/1506019482_statements_1
Query
Query Builder
EXECUTE
Table Results
Show 10 entries
Search:
Previous
1
2
Next

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
PREFIX csv: <http://rdf.dapaas.eu/cvsimport/>

SELECT ?Year ?City ?Total
WHERE {

?a csv:Year ?Year.
?a csv:City ?City.
?a csv:Total ?Total.
}
```

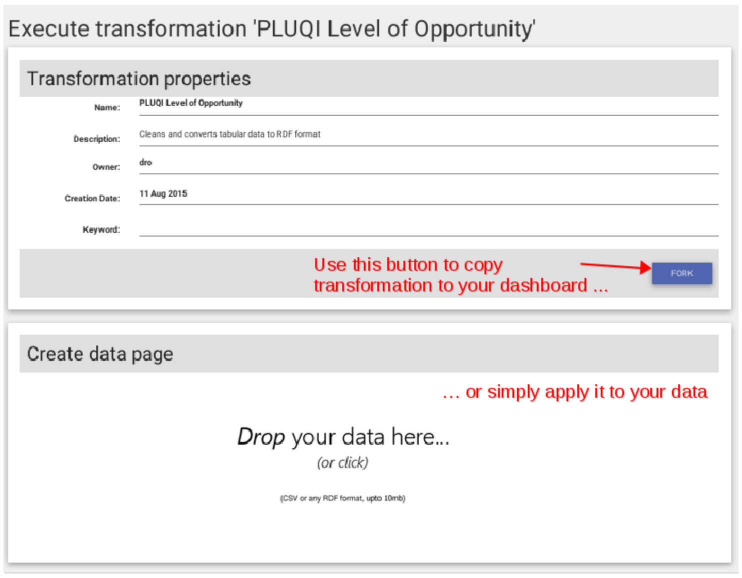
Year	Total	City
2011	358753	Seoul
2011	135399	Busan
2011	96450	Daegu
2011	86641	Incheon
2011	63810	Gwangju
2011	47401	Daejeon
2011	37855	Ulsan
2011	31393	Bucheon
2011	36741	Suwon
2011	29500	Seongnam

Showing 1 to 10 of 20 entries

Figure 3: Data page data querying

Exploring public transformations

To explore data pages and data transformations created by other users switch to the Explore tab. Here you can see a list of public assets. You may receive basic information about any public data page or data transformation by clicking its name. If in a process of exploring data transformation you find it to be suitable for your needs, you can apply it to your data directly from the explore view. To do so, you just drag and drop your datafile in the white frame labeled “Create data page”. If you want to make some changes to the transformation before you use it or just find it interesting and want to explore it in detail, use FORK button. The transformation thus will be copied to the list of your transformations.


Execute transformation 'PLUQI Level of Opportunity'

Transformation properties

Name:	PLUQI Level of Opportunity
Description:	Cleans and converts tabular data to RDF format
Owner:	dro
Creation Date:	11 Aug 2015
Keyword:	

Use this button to copy transformation to your dashboard ...

Create data page

... or simply apply it to your data

Drop your data here...
(or click)
(CSV or any RDF format, upto 10mb)

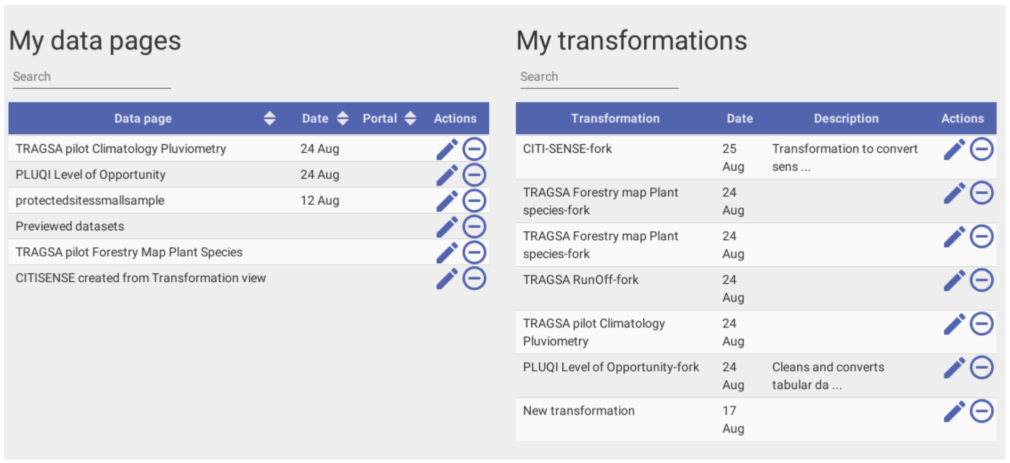
Figure 4: Ways to use public transformations

User Registration

In order to use public or 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 . After you have registered you can change your profile settings by clicking on the user name in top right corner of the website and choosing "My account" menu item.

Dashboard ??

User dashboard helps to manage data pages and data transformations created by user. The dashboard view gives you an overview of data pages and transformations you have created. From here you can search through your assets, delete them, edit their properties; fork and execute transformations and download data associated with data pages.



My data pages

Search

Data page	Date	Portal	Actions
TRAGSA pilot Climatology Pluviometry	24 Aug		
PLUQI Level of Opportunity	24 Aug		
protectedsitessmallsample	12 Aug		
Previewed datasets			
TRAGSA pilot Forestry Map Plant Species			
CITISENSE created from Transformation view			

My transformations

Search

Transformation	Date	Description	Actions
CITI-SENSE-fork	25 Aug	Transformation to convert sens ...	
TRAGSA Forestry map Plant species-fork	24 Aug		
TRAGSA Forestry map Plant species-fork	24 Aug		
TRAGSA RunOff-fork	24 Aug		
TRAGSA pilot Climatology Pluviometry	24 Aug		
PLUQI Level of Opportunity-fork	24 Aug	Cleans and converts tabular da ...	
New transformation	17 Aug		

Figure 5: The catalog of user data pages and transformations

The workflow overview

There are a lot of things that can be done with help of DataGraft portal. To give you a general overview of DataGraft functionality here is a summary of possible scenarios(see Figure 6).

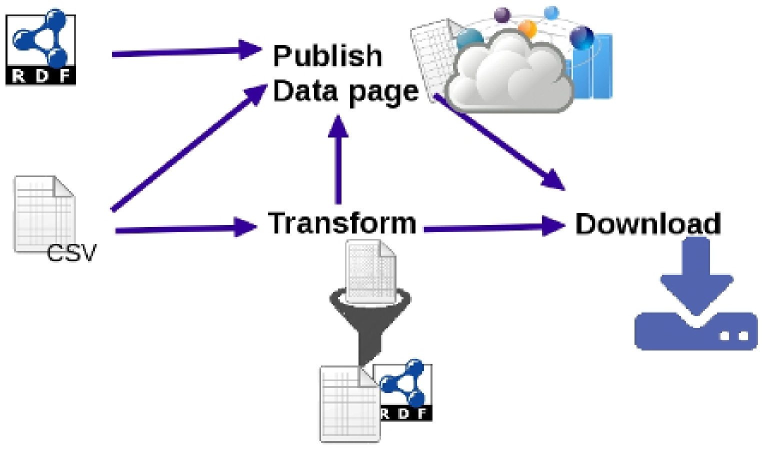


Figure 6: The workflow overview

First, if you are interested in publishing your data in form of Linked Data and already have its RDF representation, you can simply upload it to the platform and create a data page.

If data you are working with is in tabular format, there are more alternatives. First one is creating data page from raw tabular data. It is as simple process as creating data page from RDF data and can be done just in few clicks (see section for details). In case when your tabular data needs to go through some transformations, you can create data transformation and apply it to data. This may include dataset modification, data conversion from tabular to RDF form or both these procedures in one transformation. If all that you want is to transform data, on this stage you can download processed data to your computer, either in tabular or in linked data form. Alternatively you may go further and publish transformed data as a data page.

Publishing data

Publishing data with help of DataGraft platform is rather simple process. You can create a data page from several different points. You may start by switching to a Publish tab in a main menu. The first thing you do when creating a data page this way 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(see Figure 7).

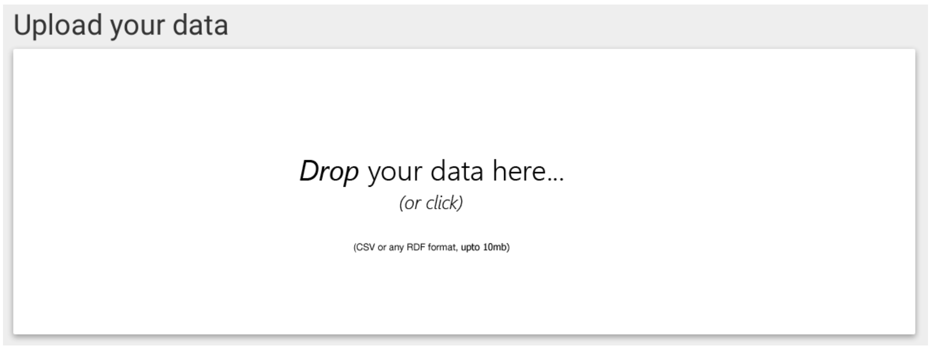


Figure 7: Uploading data

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

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

Let's go through the most simple scenario by choosing the first alternative. To do this you just click CREATE RAW button. This automatically takes you to the next page where you specify data page properties(see Figure2). After everything is in order, you simply click CREATE DATA PAGE. And that's it, you have just created your very first data page. Now you (and other users in case if you defined this data page as public) have access to the data page, are able to download associated data, add more information and features to the created asset.

However in most cases you still need to process your data before publishing it. In this case you should use the transformation service. By clicking CREATE USING A NEW TRANSFORMATION button you may start transforming your data. Details on how data transformations are created are given in Section .

Data cleaning and transformation

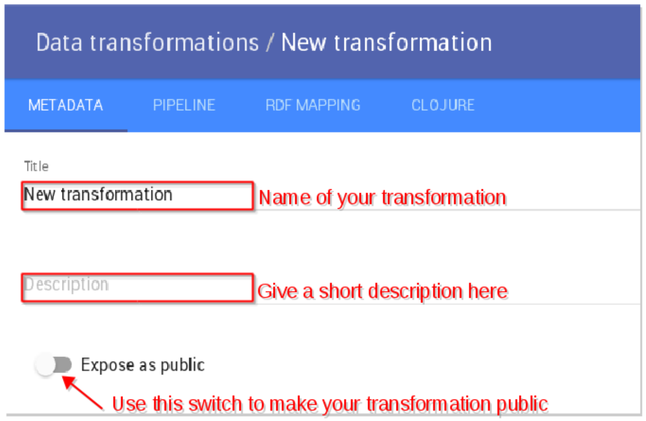
This section explains how tabular data is transformed in DataGraft platform and gives you the best strategies for data transformation.

At the basis of DataGraft data transformations there lies a Grafter DSL (Domain Specific Language), which in its turn is implemented in Clojure. Therefore, to take maximum advantage of the service, one should be acquainted with mentioned languages. However number of transformations, depending on their complexity, can be done through intuitive and user-friendly GUI without any coding.

Transformation metadata

The first tab seen in the transformation creation window is "Metadata". Here user defines transformation title and gives a short description of how this transformation processes targeted data. If you wish 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 metadata, you may save transformation by clicking "Save" button in the top right corner. Transformation may be as well saved later at any moment.



Data transformations / New transformation

METADATA PIPELINE RDF MAPPING CLOJURE

Title

New transformation

Name of your transformation

Description

Give a short description here

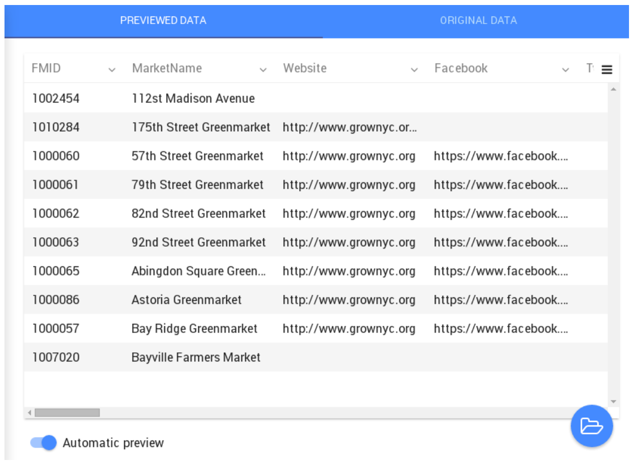
Expose as public

Use this switch to make your transformation public

Figure 8: Transformation properties

Transformation Preview

After your transformation was saved, in the bottom right corner you may see this icon: . By clicking on it you may immediately apply transformation being created to target data. This may be whether distribution already uploaded to a platform, or alternatively you may upload a new file. For the first option use button depicted as , and for the second – this one . In this way you are able to see the instant effect of created transformation on your data in the preview area.



PREVIEWED DATA ORIGINAL DATA

FMID	MarketName	Website	Facebook	T
1002454	112st Madison Avenue			
1010284	175th Street Greenmarket	http://www.grownyc.or...		
1000060	57th Street Greenmarket	http://www.grownyc.org	https://www.facebook....	
1000061	79th Street Greenmarket	http://www.grownyc.org	https://www.facebook....	
1000062	82nd Street Greenmarket	http://www.grownyc.org	https://www.facebook....	
1000063	92nd Street Greenmarket	http://www.grownyc.org	https://www.facebook....	
1000065	Abingdon Square Green...	http://www.grownyc.org	https://www.facebook....	
1000086	Astoria Greenmarket	http://www.grownyc.org	https://www.facebook....	
1000057	Bay Ridge Greenmarket	http://www.grownyc.org	https://www.facebook....	
1007020	Bayville Farmers Market			

Automatic preview

Figure 9: Transformation preview

Preview area is located in the right part of transformation window. You can see two tabs there – one with original data and another with changes made through transformation pipeline. Each time you modify a pipeline, the transformation is applied to the previewed dataset immediately, so you can see the effect of each performed step. You may adjust preview settings to check and evaluate transformation steps you are creating. Thus, it is possible to hide columns and to sort visible data. The changes made through these settings are not part of transformation and affect just previewed data. However at any time you may export tabular data either as it looks in preview or in a format it has at current stage of your transformation.

Constructing Transformation Pipeline

Data cleaning and transformation in DataGraft platform is performed with help of a "pipeline" concept. To begin with, each single transformation step is defined as a pipe – a function that performs simple data conversion on its input. 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. Obviously, the input/output data, that travels through this pipeline is dataset being transformed. This way of composing operations gives a great flexibility and allows to perform rather complex data conversions.

The first implicit step of every transformation pipeline is getting the very first pipeline input. Therefore, each transformation starts from reading a dataset from uploaded file. However you do not need to include this step into your pipeline manually, since this action is performed automatically for each transformation.

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

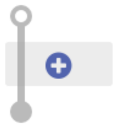


Figure 10: Add pipe function

Now you can see the list of functions you may use to modify uploaded dataset. Available functions are logically grouped according to the type of effect they have on data. Consequently, 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.

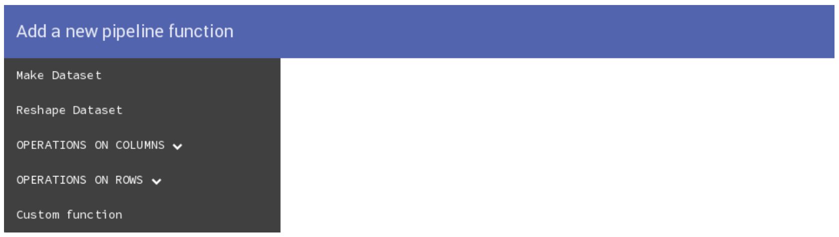


Figure 11: List of available functions

For each operation you can see a short documentation with simple illustrated example by clicking "show/hide documentation" button. For every pipeline function you create you may leave a short description note in the "Comment" field. This information helps you and other users of your transformation to understand operations that are performed here. If you ignore this field, the note will be created automatically based on function parameters you have specified.

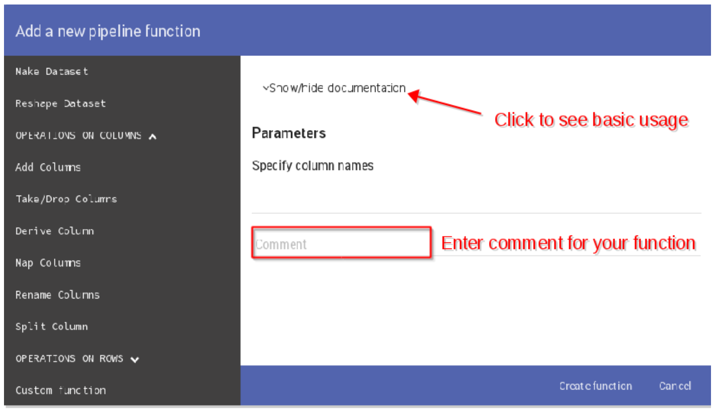


Figure 12: Common operations for all pipe functions

Once you have added a new function to the pipeline, it will instantly appear in the pipeline view. You are free to change function parameters any time you need it by simply clicking on correspondent function icon. To get a short information about actions performed by this function you may just hover mouse pointer over its name. In many cases function order significantly affects the transformation result. It is very simple to change this order by just dragging function icons along the pipeline. To remove a function click ⊖ button next to the function you would like to remove.

The following sections provide you with detailed guidelines for each function usage.

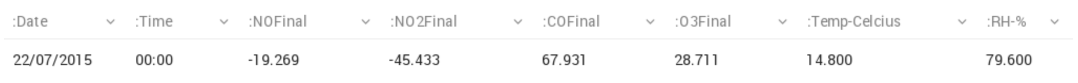
Make Dataset

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 see in a new dataset or fetch first n columns.

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. The best way to explain this function is through an example.

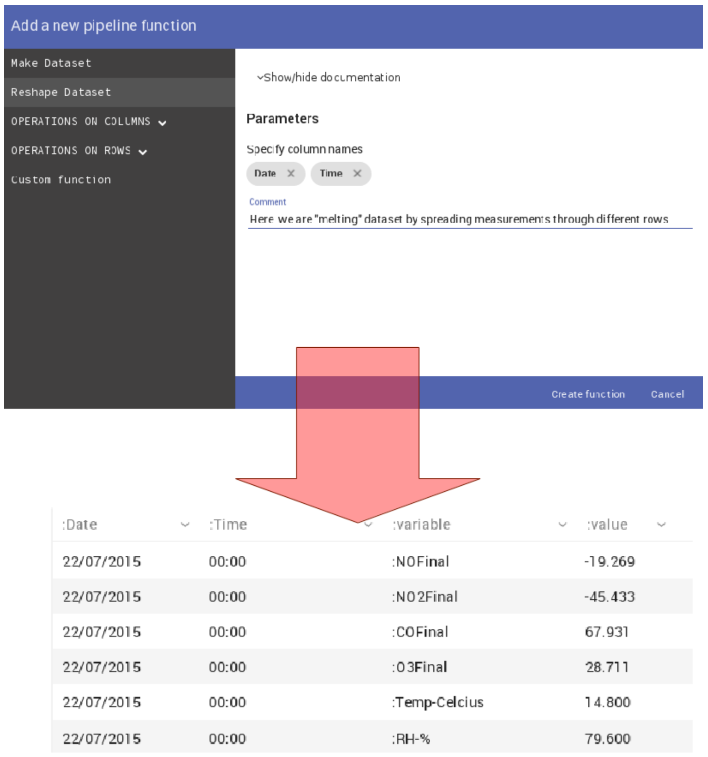
Example:
Let’s consider the case, when dataset you are interested in is represented by some sensor measurements data(Figure 13).



:Date	:Time	:NOFinal	:NO2Final	:COFinal	:O3Final	:Temp-Celcius	:RH-%
22/07/2015	00:00	-19.269	-45.433	67.931	28.711	14.800	79.600

Figure 13: Original measurement data

Now you may want to put each measurement into a different row. To do so, you reshape your dataset on fields Date and Time. In this way combination of date and time will be considered as unique id key, while other data will be bound to this key as map of variables and values(see Figure 14).



Here we are "melting" dataset by spreading measurements through different rows

:Date	:Time	:variable	:value
22/07/2015	00:00	:NOFinal	-19.269
22/07/2015	00:00	:NO2Final	-45.433
22/07/2015	00:00	:COFinal	67.931
22/07/2015	00:00	:O3Final	28.711
22/07/2015	00:00	:Temp-Celcius	14.800
22/07/2015	00:00	:RH-%	79.600

Figure 14: Reshaped measurement data

Add Columns

The "Add columns" results in adding a new column(s) to a given dataset. To add a new column you should specify column name 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, row index 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. You are able to add as many columns as you need within one operation. To add one more column simply click

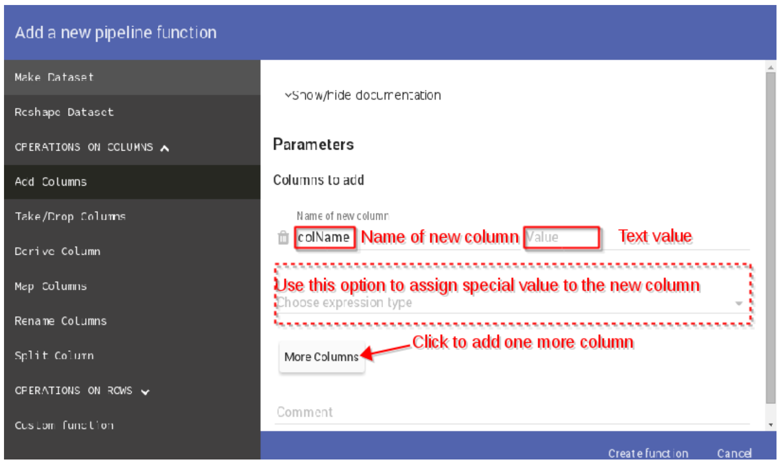


Figure 15: Add Columns function

Take/Drop Columns

The "Take/Drop Columns" function narrows given dataset. You may explicitly define columns you wish to see in a dataset by listing their names, get first n columns or remove unnecessary columns from a dataset. In a second case, dataset will be narrowed to specified number of 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.

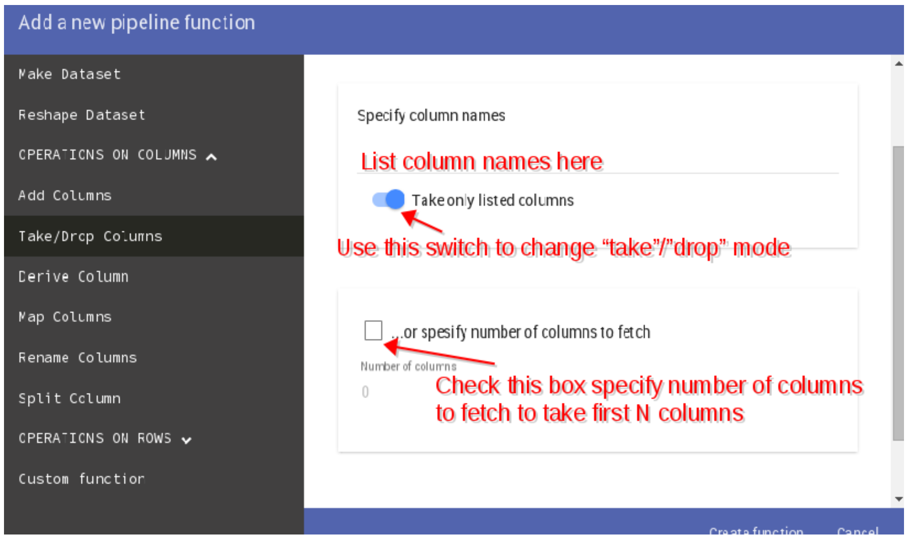


Figure 16: Take/Drop Columns function

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. For your convenience these functions are grouped together based on type of operation they perform.

If you think that functionality ordered by standard provided functions is not enough to build your transformation, you may define custom utility functions by yourself using Clojure code(see) or use utility functions written by other users.

Some of functions may expect input parameters in addition to columns, they will operate on. If this is the case, you may specify such a parameter by pressing ⊞ icon and entering parameter in a field that appears.

Tips and tricks:

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.

Map Columns

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

Tips and tricks:
As you probably noticed, the function that appears by default each time you add new mapping pair is string-literal. This function belongs to the group "CONVERT DATATYPE" and plays important role in tabular-to-RDF conversion.
Whenever you need to treat a column value as a constant literal node, you should first define, which datatype it represents. String-literal will mark column as one having datatype "string". Another functions used for defining data type are integer-literal and double-literal.

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.

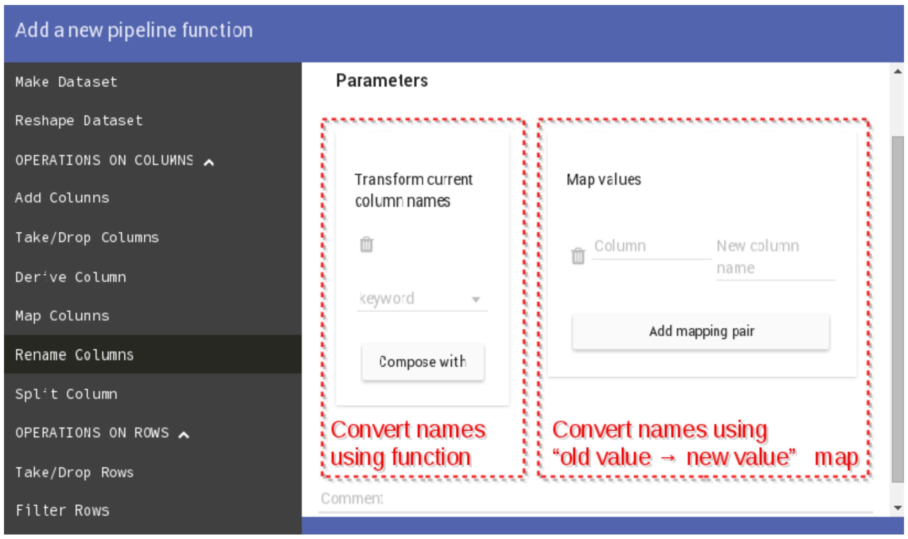


Figure 17: Rename Columns function

Tips and tricks:

You may see the function named "keyword" assigned by default as a function used to transform column names. This is done because column names in Grafter DSL should be treated as keywords – special symbolic identifiers used in Clojure language. Having this function as a first one in a list of functions used for column names conversion guarantees column names will be recognized as valid keywords.

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.

Take/Drop Rows

This function allows you to take or drop first n rows in a given dataset. Use switch Take rows/Drop rows in the top of the function parameter list to choose between these two options

Filter Rows

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).

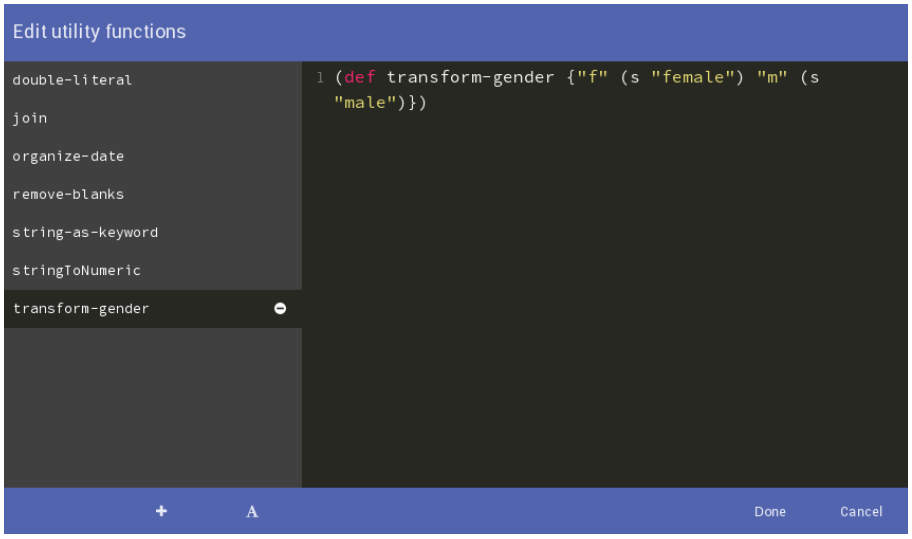
Defining Auxiliary Functions

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

Creating and Editing Custom Utility Functions

Creating utility functions for transformations is a great way to encapsulate the logic of data modifications, thus making them independent of data they are applied to. For instance, if to convert data in several dataset columns you use the same formula, you can define a function, that performs all the necessary calculations based on its input parameters and call it any time and on any data you need.

To create your custom function use </> Edit utility functions button. In window that opens you can see list of available functions at the left side and code editor at the right side.



```
(def transform-gender {"f" (s "female") "m" (s
"male")})
```

Figure 18: Create custom utility function

Each function can be removed by pressing ⊖ icon next to the function name. New functions are added by plus icon in the toolbox below function list.

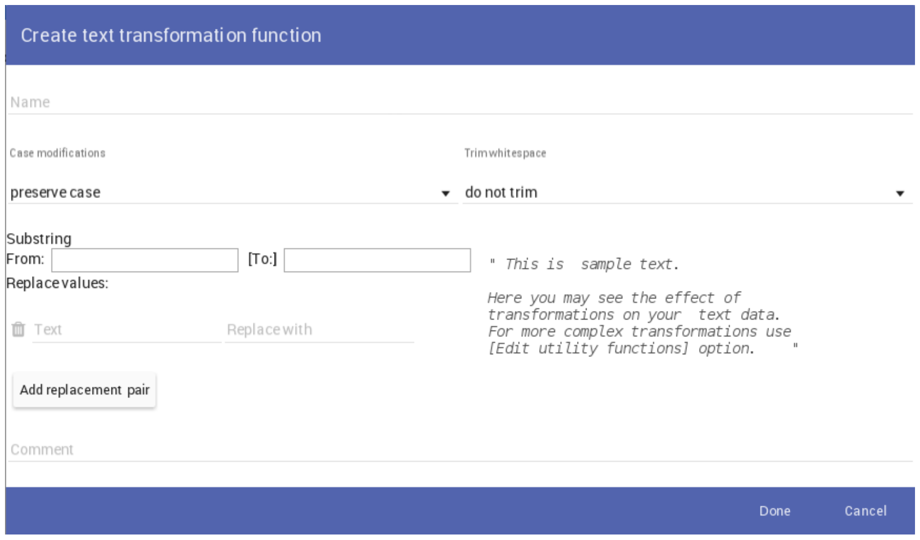
Immediately after you have created a utility function you may start using it as a parameter in pipeline functions.

Tips and tricks:

If you create a function having a dataset as an input parameter, it is possible to make it a part of the transformation pipeline directly. You may add this function to a pipeline by choosing ”Custom function” option from a pipeline function lists.

Creating String Transformation Functions through Graphical Interface

However to specify text transformations you need to perform on your data, you may define new function in a more user-friendly way. To do so press A button in the same window, that is used for creating custom utility functions. Now you can see the dialog, where you specify transformations you want to perform on text values(see Figure 19). All modifications you define are instantly shown with help of sample text at the right bottom corner.


Create text transformation function

Name

Case modifications: preserve case

Trim whitespace: do not trim

Substring
From: [To:]

Replace values:

Text | Replace with

```
" This is  sample text.

Here you may see the effect of
transformations on your  text data.
For more complex transformations use
[Edit utility functions] option.    "
```

Add replacement pair

Comment

Done Cancel

Figure 19: Create text transformation function

Creating and Editing Prefixers

One special type of utility functions you may define is prefixer – function that adds some prefix to its argument in a such way, that the result represents an URI. This is something you may need in constructing RDF mapping part. If all you need is just constructing URI nodes by adding the same type of prefix to all values in a column, you should define prefixers in “Edit RDF mapping prefixes” dialog (see Building RDF Mapping section). However, if you need to add some logic in assigning prefixes depending on column values, you should define prefixer functions here in pipeline view. You may create and edit prefixers in the "Edit prefixers" window. To see this window 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 .

Building RDF Mapping

One great method to publish your data on the Web is to use Linked Data for it. You can read about all the opportunities you get with linked data here: `<http://www.w3.org/standards/semanticweb/data>`. If you've chosen to convert your data in a format of linked data, the first thing you need to do is to build an RDF map, that will determine way of getting linked data nodes from tabular data.

When tabular data is converted to a desirable format, you can start creating RDF mappings. In the process of creating RDF skeleton you can immediately see a clear visualization of nodes and corresponding relations.

To start creating RDF mapping of your dataset use switch "Map tabular data to RDF". The first step here would be to specify the base graph URI. To create and edit prefixes you are going to need for the RDF generation use "Edit RDF prefixes button". By pressing it you open a dialog, where you can as well see RDF vocabularies available by default.

Now you can start to create, edit and remove RDF nodes and their properties. The three groups of nodes that can be created here are URI nodes, literal nodes and blank nodes. To construct a URI node both as one obtained from dataset column and as some constant URI, you need to define some prefix. Prefixes are separated from values they complement with semicolon. Note, that dataset columns may be converted to the URI form during the tabular transformation step. In this case you don't have to specify prefixers to make them URI nodes in RDF schema. If you, however try to create a URI node from data column that is not recognized as a valid URI, and do not assign any prefix value to this column, it would be automatically converted to column literal node. Literal nodes are represented just by their value in the resulted RDF file. An example of RDF mapping skeleton is shown below.

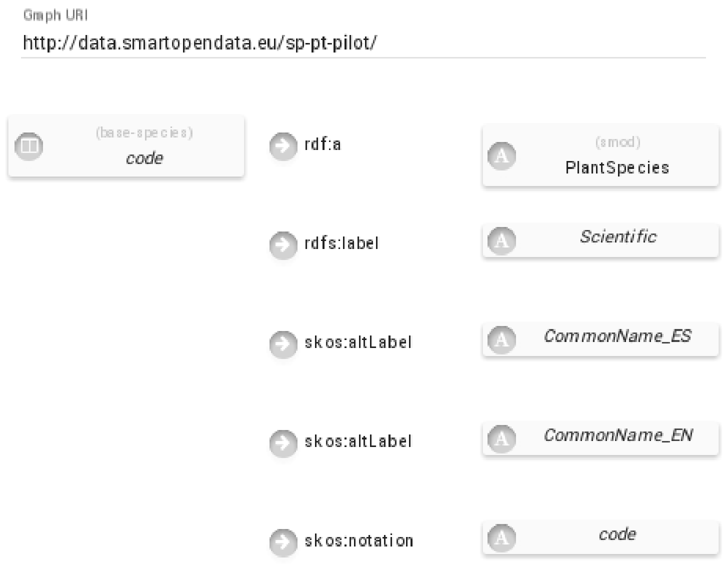


Figure 20: Example of RDF mapping

Executing transformation

After you've completed creating transformation either with or without RDF mapping part you may apply transformation to your data right in the transformation view. To do so press Edit prefixes. After you did this you can see a dialog, where you can choose between publishing transformed data as a data page and downloading results locally on your computer.

Alternatively transformations can be executed from the "Dashboard" and "Explore" views as it have been discussed in sections **??** and .

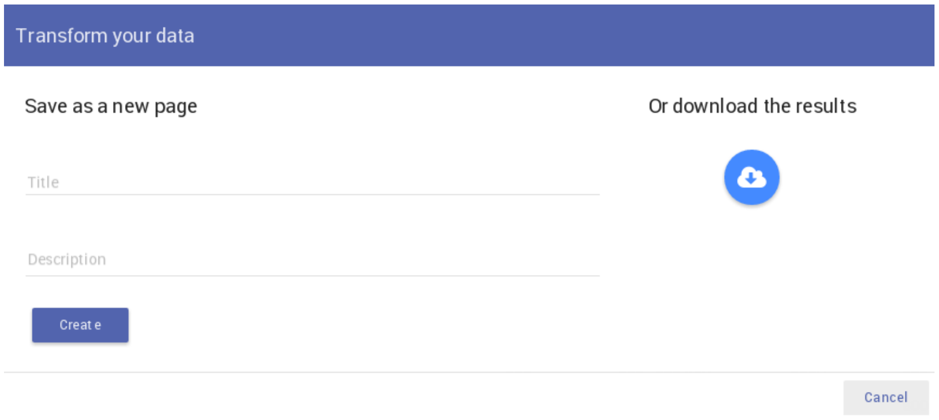


Figure 21: Apply transformation from the transformation view