

Automated Data Linkage

Nov 28th, 2024

Title Automated Data Linkage – User Facing Documentation

Version 0.2.22

Description The user facing documentation for the automated data linkage package is meant to help familiarize oneself with how to navigate the GUIs, accessible functions, what inputs are expected of each function, and possible errors that you may come across with instructions on how to handle them.

Depends R (\geq 4.4.1)

Built On R 4.4.1

Encoding UTF-8

Language en-CA

Author Cole Chuchmach [aut/cre]
Barret Monchka [aut/ctb]

Maintainer The George & Fay Yee Centre for Healthcare Innovation

Date/Publication 2024-11-28 12:00:50 CST

R topics documented

Installing and Loading the Automated Data Linkage Package	3
populate_linkage_metadata.R	4
data_linkage_scripts.R	5
linkage_helper_functions.R	7
get_blocking_keys()	7
get_matching_keys()	8
get_acceptance_thresholds()	9
get_linkage_technique()	10
get_linkage_technique_description()	10
get_implementation_name()	11
get_iteration_name()	12
get_algorithm_name()	12
get_ground_truth_fields()	13
get_linkage_output_fields()	14
apply_output_cutoffs()	15
get_standardized_names()	16
load_linkage_file()	16
create_extra_parameters_list()	17
linkage_metadata_ui.R	20
Home	20
Datasets	20
Linkage Methods	22
Testable Linkage Algorithms	23
Archived Linkage Algorithms	24
Published Linkage Algorithms	25
View Linkage Iterations (Passes)	26
Modify Linkage Iterations (Passes)	27
Comparison Methods	39
Acceptance Methods	30
Ground Truth Variables	31
Linkage Algorithm Output Fields	32
Linkage Audits	33
Run Algorithm(s)	34
Regenerate Report	35

Installing and Loading the Automated Data Linkage Package

Details

The automated data linkage package (*autolink*) used to allow for an easier and automated linkage process is hosted on GitHub, from which it can be installed. To do so, follow the steps as described:

Step 1 – Installing the *Devtools* Package in R:

Install and load the **devtools** library in **R** such that you can gain access to the **install_github()** function:

```
# install.packages("devtools")
library(devtools)
```

Step 2 – GitHub Installation:

Once the **devtools** package has been installed and loaded, use the **install_github()** function to install the package from GitHub.

During installation, you will need to install or update other **R Libraries** that are used by the data linkage package.

Once all libraries have been successfully installed or updated, the data linkage package should successfully appear in your list of packages and can be enabled for use. Giving you access to all scripts and applications.

```
devtools::install_github("CHIMB/autolink")
```

Step 3 – Local Installation (*if step 2 fails*):

If installing the **autolink** package using **install_github()** is unsuccessful, consider installing the package locally, by first selecting the most recent release of the package and downloading the source zip file, then using **install_local()** in **R Studio**:

```
path_to_pkg <- file.choose() # Select the unmodified package you downloaded from GitHub.
devtools::install_local(path_to_pkg)
```

populate_linkage_metadata.R

Description

populate_linkage_metadata.R is the script that will create an **SQLite** file for storing metadata related to your datasets and linkage algorithms.

Details

The **SQLite** file generated is used in both the **Linkage Metadata GUI**, and the programmatic **run_main_linkage()** function. The generated file will initially be blank, with some default values already pre-populated for selection.

Arguments

file_name	A path to the left dataset file, where the prefix of the file matches a dataset code in the metadata file.
output_folder	A path to the right dataset file, where the prefix of the file matches a dataset code in the metadata file.

data_linkage_scripts.R

Description

data_linkage_scripts.R is the script that will perform the automated data linkage by dynamically calling linkage classes which inherit an abstract class. The output and results depend on what the user would like to be exported, with more information on what they user can decide found in the **extra_parameters** variable.

The script contains **one** main function with **2** default linkage classes to perform the linkage process.

Details

The data linkage script is paired with an **SQLite** metadata file to pull linkage information, including algorithms, passes, the fields it uses, and what output to keep. Of which requires valid datasets and algorithms to be enabled in the database.

To perform data linkage using metadata information, call the **run_main_linkage()** function and supply **five** input parameters.

- **Left Dataset:** The left dataset should be a file of type csv, excel, txt, SQLite, etc. Where the prefix of the file name should match the dataset code entry in the database.
- **Right Dataset:** The right dataset should be a file of type csv, excel, txt, SQLite, etc. Where the prefix of the file name should match the dataset code entry in the database.
- **Linkage Metadata File:** The SQLite file that contains all the users linkage information (*algorithms, passes, datasets, etc*).
- **Algorithm IDs:** A vector of integers that pertain to the unique ID of each algorithm, each algorithm ID will be used to obtain the individual passes, and output of each algorithm.
- **Extra Parameters:** A list of logical, numeric, and character values used to dictate what kind of data to export to the user, reports to generate, and values to calculate.

Additionally, the **run_main_linkage()** will return a list of relevant data if the user requested linked data to be returned within the extra parameters field, otherwise NULL is returned.

Lastly, error messages are provided to the user to help inform them on what went wrong during runtime and how they may be able to fix it.

Arguments

left_dataset_file	A path to the left dataset file, where the prefix of the file matches a dataset code in the metadata file.
right_dataset_file	A path to the right dataset file, where the prefix of the file matches a dataset code in the metadata file.
linkage_metadata_file	A path to the SQLite linkage metadata file.
algorithm_ids	A vector of integer values, each pertaining to a specific algorithm in the database.
extra_parameters	A list of key-value pairs, each of which have logical, numeric, or character value which dictates what kinds of output to export during or after data linkage takes place.

Example

```
> run_main_linkage(left_dataset, right_dataset, linkage_metadata_file, algorithm_id, extra_params_v3)
[1] "Beginning Linkage for BMD Algorithm 1"
[1] "Beginning Linkage pass (PASS1)"
[1] "PASS1 using the Reclin2Linkage class finished in 0.770 seconds"
[1] "Beginning Linkage pass (PASS2)"
[1] "PASS2 using the Reclin2Linkage class finished in 0.520 seconds"
[1] "Beginning Linkage pass (PASS3)"
[1] "PASS3 using the Reclin2Linkage class finished in 1.120 seconds"
[1] "Beginning Linkage pass (PASS4)"
[1] "PASS4 using the Reclin2Linkage class finished in 0.410 seconds"
[1] "Beginning Linkage pass (PASS5)"
[1] "PASS5 using the Reclin2Linkage class finished in 0.740 seconds"
[1] "Beginning Linkage pass (PASS6)"
[1] "PASS6 using the Reclin2Linkage class finished in 0.480 seconds"
[1] "Linkage Plot Saved As: C:/Users/chuchma1/Desktop/outputs/BMD Algorithm 1 (PASS6) - Decision Boundary Plot (1).png"
[1] "Beginning Linkage pass (PASS7)"
[1] "PASS7 using the Reclin2Linkage class finished in 0.490 seconds"
[1] "Linkage Plot Saved As: C:/Users/chuchma1/Desktop/outputs/BMD Algorithm 1 (PASS7) - Decision Boundary Plot (1).png"
Setting theme "language: en"
Setting theme "language: en"
```

linkage_helper_functions.R

Description

data_linkage_scripts.R is the script that provides the user with many functions called before, after, or during data linkage to acquire specific database information, apply data standardization, or create pre-processing options.

Details

The available specifications are:

- `get_blocking_keys()`
- `get_matching_keys()`
- `get_acceptance_thresholds()`
- `get_linkage_technique()`
- `get_linkage_technique_description()`
- `get_implementation_name()`
- `get_iteration_name()`
- `get_algorithm_name()`
- `get_ground_truth_fields()`
- `get_linkage_output_fields()`
- `apply_output_cutoffs()`
- `get_standardized_names()`
- `load_linkage_file()`
- `create_extra_parameters_list()`

get_blocking_keys()

Description

`get_blocking_keys()` will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a data frame of the blocking keys and linkage rules.

Usage

```
get_blocking_keys(linkage_metadata_conn, 2)
```

Arguments

- | | |
|---------------------|---|
| linkage_db | A database connection to the linkage metadata file being used for data linkage. |
| iteration_id | An integer ID specific to an algorithm's iteration. |

Details

The function performs a query on the provided database connection, using the iteration ID to limit the blocking keys for that specific iteration.

Once the query is returned, additional cleaning is performed to replace linkage rule IDs with key value pairs stored in a JSON format.

A data frame is returned containing **three** columns, the field name of the blocking pair in the left dataset, the field name of the blocking pair in the right dataset, and key-value pairs of what linkage rules should be applied during data linkage. If no blocking keys exist for the iteration, an empty data frame is returned.

Example

```
> get_blocking_keys(linkage_metadata_conn, 2)
  left_dataset_field right_dataset_field linkage_rules
1      birth_year      birth_year      NA
2      gender      gender      NA
3 primary_given_name primary_given_name      NA
```

get_matching_keys()

Description

get_matching_keys() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a data frame of the matching keys, linkage rules, and comparison rules.

Usage

```
get_matching_keys(linkage_metadata_conn, 2)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- iteration_id** An integer ID specific to an algorithm's iteration.

Details

The function performs a query on the provided database connection, using the iteration ID to limit the matching keys for that specific iteration.

Once the query is returned, additional cleaning is performed to replace linkage rule IDs and comparison rule IDs with key value pairs stored in a JSON format.

A data frame is returned containing **four** columns, the field name of the matching pair in the left dataset, the field name of the matching pair in the right dataset, and key-value

pairs of what linkage and comparison rules should be applied during data linkage. If no matching keys exist for the iteration, an empty data frame is returned.

Example

```
> get_matching_keys(linkage_metadata_conn, 2)
  left_dataset_field right_dataset_field linkage_rules comparison_rules
1   primary_surname   primary_surname      <NA>          NA
2    birth_month     birth_month      <NA>          NA
3    birth_day       birth_day       <NA>          NA
4   postal_code      postal_code      <NA>          NA
5 secondary_given_names secondary_given_names {"substring_length":1} NA
> get_matching_keys(linkage_metadata_conn, 1)
  left_dataset_field right_dataset_field linkage_rules comparison_rules
1      lastname      lastname      NA {"jw_score":0.9}
2     firstname     firstname      NA {"jw_score":0.9}
3       address       address      NA {"jw_score":0.9}
4         sex         sex         NA {"jw_score":0.9}
```

get_acceptance_thresholds()

Description

get_acceptance_thresholds() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a list of key value pair acceptance thresholds.

Usage

```
get_acceptance_thresholds(linkage_metadata_conn, 1)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- iteration_id** An integer ID specific to an algorithm's iteration.

Details

The function performs a query on the provided database connection, using the iteration ID to obtain all the parameter keys and values.

Once the query is returned, additional cleaning is performed to create the key-value pairs, appending each to a list which will be returned.

A list is returned, where each acceptance value is a key-value pair of the variable name and its numeric value. If no acceptance threshold was found under the provided iteration ID, an empty list (*list of length 0*) is returned.

Example

```
> get_acceptance_thresholds(linkage_metadata_conn, 1)
$match_weight
[1] 4

> get_acceptance_thresholds(linkage_metadata_conn, 2)
$posterior_threshold
[1] 0.8
```

get_linkage_technique()

Description

get_linkage_technique() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a character string of the technique.

Usage

```
get_linkage_technique(linkage_metadata_conn, 1)
```

Arguments

linkage_db A database connection to the linkage metadata file being used for data linkage.

iteration_id An integer ID specific to an algorithm's iteration.

Details

The function performs a query on the provided database connection, using the iteration ID to obtain the technique label being used within the class being called for this iteration.

A character string of the linkage technique is returned if there exists an iteration under this ID, otherwise, NA is returned.

Example

```
> get_linkage_technique(linkage_metadata_conn, 1)
[1] "P"

> get_linkage_technique(linkage_metadata_conn, 7)
[1] "D"
```

get_linkage_technique_description()

Description

get_linkage_technique_description() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a character string of the technique's description.

Usage

```
get_linkage_technique_description(linkage_metadata_conn, 1)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- iteration_id** An integer ID specific to an algorithm's iteration.

Details

The function performs a query on the provided database connection, using the iteration ID to obtain the description of the technique label being used within the class being called for this iteration.

A character string of the linkage technique description is returned if there exists an iteration under this ID, otherwise, NA is returned.

Example

```
> get_linkage_technique_description(linkage_metadata_conn, 1)
[1] "Probabilistic linkage pass using the Reclin2 package."
> get_linkage_technique_description(linkage_metadata_conn, 7)
[1] "Deterministic linkage pass using the Reclin2 package."
```

get_implementation_name()

Description

get_implementation_name() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a character string of linkage implementation which will be called when running that iteration of an algorithm for data linkage.

Usage

```
get_implementation_name(linkage_metadata_conn, 1)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- iteration_id** An integer ID specific to an algorithm's iteration.

Details

The function performs a query on the provided database connection, using the iteration ID to obtain the implementation name to identify which class to call and pass parameters to for data linkage.

A character string of the implementation name is returned if there exists an iteration under this ID, otherwise, NA is returned.

Example

```
> get_implementation_name(linkage_metadata_conn, 1)
[1] "RecLin2Linkage"
```

get_iteration_name()

Description

get_iteration_name() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific pass of an algorithm. Returning a character string of the identifiable iteration name defined by the user.

Usage

```
get_iteration_name(linkage_metadata_conn, 1)
```

Arguments

- | | |
|---------------------|---|
| linkage_db | A database connection to the linkage metadata file being used for data linkage. |
| iteration_id | An integer ID specific to an algorithm's iteration. |

Details

The function performs a query on the provided database connection, using the iteration ID to obtain the identifying iteration name for summary or timing purposes.

A character string of the iteration name is returned if there exists an iteration under this ID, otherwise, NA is returned.

Example

```
> get_iteration_name(linkage_metadata_conn, 1)
[1] "PASS1"
> get_iteration_name(linkage_metadata_conn, 3)
[1] "PASS2"
```

get_algorithm_name()

Description

get_algorithm_name() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific algorithm. Returning a character string of the identifiable algorithm name defined by the user.

Usage

```
get_algorithm_name(linkage_metadata_conn, 1)
```

Arguments

linkage_db A database connection to the linkage metadata file being used for data linkage.

algorithm_id An integer ID specific to an algorithm.

Details

The function performs a query on the provided database connection, using the algorithm ID to obtain the identifying algorithm name for summary or timing purposes.

A character string of the algorithm name is returned if there exists an algorithm under this ID, otherwise, NA is returned.

Example

```
> get_algorithm_name(linkage_metadata_conn, 1)
[1] "Test Algorithm 1"
> get_algorithm_name(linkage_metadata_conn, 3)
[1] "Test Algorithm 2"
```

get_ground_truth_fields()

Description

get_ground_truth_fields() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific algorithm. Returning a data frame of the left and right dataset fields, along with any linkage rules they may be using.

Usage

```
get_ground_truth_fields(linkage_metadata_conn, 1)
```

Arguments

linkage_db A database connection to the linkage metadata file being used for data linkage.

algorithm_id An integer ID specific to an algorithm.

Details

The function performs a query on the provided database connection, using the algorithm ID to obtain the ground truth fields of the algorithm.

A data frame of the left and right dataset fields is returned along with any linkage rules if there exists an algorithm under this ID, otherwise, an empty data frame is returned.

Example

```
> get_ground_truth_fields(linkage_metadata_conn, 1)
  left_dataset_field right_dataset_field linkage_rules
1                id                id           NA
> get_ground_truth_fields(linkage_metadata_conn, 2)
  left_dataset_field right_dataset_field linkage_rules
1                phin                phin           NA
```

get_linkage_output_fields()

Description

get_linkage_output_fields() will take in a connection to the linkage database you plan on using, along with an integer which correlates to a specific algorithm. Returning a data frame of the output labels and their associated field names (*from the left dataset*).

Usage

```
get_linkage_output_fields(linkage_metadata_conn, 1)
```

Arguments

linkage_db A database connection to the linkage metadata file being used for data linkage.

algorithm_id An integer ID specific to an algorithm.

Details

The function performs a query on the provided database connection, using the algorithm ID to obtain the output fields of the algorithm.

A data frame of the label and field name pairs is returned if there exists an algorithm under this ID, otherwise, an empty data frame is returned.

Example

```
> get_linkage_output_fields(linkage_metadata_conn, 1)
dataset_label field_name
1      Sex      sex
2  Postal Code  postcode
> get_linkage_output_fields(linkage_metadata_conn, 2)
dataset_label field_name
1  Birth Year  birth_year
2      Gender  gender
3  Postal Code  postal_code
```

apply_output_cutoffs()

Description

`apply_output_cutoffs()` will take in a connection to the linkage database you plan on using, an integer which correlates to a specific algorithm, and a data frame of values to apply cutoffs to. Returning an updated version of the data frame passed with cutoffs apply to certain values.

Usage

```
apply_output_cutoffs(linkage_metadata_conn, 1, input_data)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- algorithm_id** An integer ID specific to an algorithm.
- output_df** A data frame of the output fields and associated values obtained during data linkage.

Details

The function performs a query on the provided database connection, using the algorithm ID to identify how to modify the input data frame and apply cutoffs to group many values together for easier comparisons.

An updated version of the input data frame is returned if there exists an algorithm under this ID, otherwise, no modifications are made.

Example

birth_year	gender	postal_code
2024	F	R
2024	F	H
2024	M	H
2024	M	H
1960	M	R
1960	M	H
2024	M	H
1960	F	R
1960	M	R
1960	F	H
2024	F	R
1960	F	H
1960	F	H
2024	M	R
1960	M	H
1960	F	R

birth_year_cat	gender_untouched_cat	postal_code_initial_cat
Birth Year	Gender	Postal Code
2015-2024	F	R
2015-2024	F	H
2015-2024	M	H
2015-2024	M	H
<1975	M	R
<1975	M	H
2015-2024	M	H
<1975	F	R
<1975	M	R
<1975	F	H
2015-2024	F	R
<1975	F	H
<1975	F	H
2015-2024	M	R
<1975	M	H
<1975	F	R

get_standardized_names()**Description**

get_standardized_names() will take in a connection to the linkage database you plan on using, an integer which correlates to a specific iteration of an algorithm, and a vector of names to standardize. Returning a vector of unique names standardized to a common spelling.

Usage

```
get_standardized_names(linkage_metadata_conn, 1, data_field)
```

Arguments

- linkage_db** A database connection to the linkage metadata file being used for data linkage.
- iteration_id** An integer ID specific to an algorithm's iteration.
- data_field** A vector of unique names to standardize.

Details

The function performs a query on the provided database connection, using the iteration ID to identify how to modify the input vector of names, using a user defined look up table, or the packages default lookup values if none is found.

An updated version of the persons name is returned if there exists a value in the lookup table that matches the input name, otherwise, NA is returned in place of the name.

Example

```
> get_standardized_names(linkage_metadata_conn, 1,
+ c("John", "Johnny", "Jon", "Sara", "Sarah", "Kris", "Chris", "?!"))
[1] "JOHN" "JOHN" "JOHN" "SARAH" "SARAH" "CHRIS" "CHRIS" NA
```

load_linkage_file()

Description

load_linkage_file() will take in a file path of types **csv**, **txt**, **sas7bdat**, **xlsx**, **sqlite** (*datastan output*), and will return a data frame of the read in data.

Usage

```
load_linkage_file(file_path)
```

Arguments

dataset_file A path to the dataset file that should be loaded in and returned as a data frame.

Details

The function checks for the file extension of the passed file path, and uses that extension to determine which package will read in the data and return it as a data frame.

Example

```
> load_linkage_file(left_dataset)
  id lastname  firstname address sex postcode
<int> <char> <char> <char> <char> <char>
1:   1   Smith    Anna 12 Mainstr  F  1234 AB
2:   2   Smith    George 12 Mainstr  M  1234 AB
3:   3 Johnson    Anna 61 Mainstr  F  1234 AB
4:   4 Johnson  Charles 61 Mainstr  M  1234 AB
5:   5 Johnson   Charly 61 Mainstr  M  1234 AB
6:   6 Schwartz    Ben  1 Eaststr  M  6789 XY
```

create_extra_parameters_list()

Description

create_extra_parameters_list() will allow users to modify any of the default parameter values to allow for more specific linkage options such as exporting or saving certain linkage results.

Usage

```
load_linkage_file(...)
```

Arguments

linkage_output_folder	An output directory where results will be stored during the linkage process.
include_unlinked_records	Includes the unlinked records in the final linkage results data frame. (Options: TRUE, FALSE)
output_linkage_iterations	Exports the linked iteration pairs for each pass of an algorithm, which includes the pair values and their scores. (Options: TRUE, FALSE)
output_unlinked_iteration_pairs	Exports the unlinked iteration pairs for each pass of an algorithm, including their scores. (Options: TRUE, FALSE)
linkage_report_type	Determines what type of report should be generated after an algorithm has been run. (Options: 1 = No Report/CSV Linkage Results, 2 = Intermediate Report for All Algorithms Being Ran, 3 = Final Report for Each Algorithm Being Ran)
calculate_performance_measures	Exports the performance measures of each algorithm being ran, which includes PPV, NPV, Sensitivity, Specificity, F1-Score, and Linkage Rates. (Options: TRUE, FALSE)
data_linker	The name of the person performing the linkage.
generate_algorithm_summary	Exports the algorithm summary of each algorithm being ran, which includes each passes blocking and matching criteria, acceptance threshold, technique, and linkage rate. (Options: TRUE, FALSE)
generate_threshold_plots	Exports PNG images of plots taken to help establish valid acceptance thresholds, or to show an acceptance threshold cutoff once selected. (Options: TRUE, FALSE)
save_all_linkage_results	Saves the linkage results, algorithm summary, performance measures, and missing data indicators in an .Rdata file in the packages AppData which can be used to regenerate reports for later use. (Options: TRUE, FALSE)

collect_missing_data_indicators	Will browse the input dataset for missing values in the output fields included in the linkage results and keep them for report purposes. (Options: TRUE, FALSE)
save_audit_performance	Will perform a query after an algorithm finishes completion which involves writing its performance measures to the database so that it may be viewed or exported for later use. (Options: TRUE, FALSE)

Details

The function will accept any number of input values and will override the default NULL or FALSE values with user input, returning a list of the parameters which can be passed during linkage.

Example

```
> create_extra_parameters_list(include_unlinked_records = T, output_linkage_iterations = T,
+                             output_unlinked_iteration_pairs = T, data_linker = "Awesome Data Analyst")
$include_unlinked_records
[1] TRUE

$output_linkage_iterations
[1] TRUE

$output_unlinked_iteration_pairs
[1] TRUE

$data_linker
[1] "Awesome Data Analyst"
```

linkage_metadata_ui.R

Description

linkage_metadata_ui.R is the shiny application that will allow users to make various changes to and run their linkage algorithms.

Details

The available specifications/pages are:

- Home
- Datasets
- Linkage Methods
- Testable Linkage Algorithms
- Archived Linkage Algorithms
- Published Linkage Algorithms
- View Linkage Iterations
- Modify Linkage Iterations
- Comparison Methods
- Acceptance Methods
- Ground Truth Variables
- Linkage Algorithm Output Fields
- Linkage Audits
- Run Algorithm(s)
- Regenerate Report

Home

Description

The home page provides users with a descriptive summary of the common work-flow from adding a new dataset to algorithm creation, while also providing them with the other accessible pages within the GUI and how to get from the start page to them.

Datasets

Description

The datasets page allows users to add or modify datasets to be used for linkage algorithms. A minimum of two datasets will need to exist before constructing an algorithm takes place.

Details

When adding a new dataset, there are **five inputs** that the user must make note off.

- The **first** is the **dataset code** which is used when reading files to determine which dataset entry (*and correspondingly, which algorithms*) to access. The code should be a *prefix* of the input files name that appears before any punctuation.
- The **second** is the **dataset name** which serves as a longer descriptive/more identifiable name appearing in reports, algorithm summaries, and the GUI often.
- The **third** is the **dataset version**, which will be a numerical value to help differentiate datasets that may share the same name or code.
- The **fourth** is an optional toggle which will read in the dataset as **fixed-width format**, allow the user to modify the field widths if they must.
- The **fifth** is the **input file** which will store the local file location (*as uploaded*) and provide you with the fields and field types read from the file before you submit.

When updating an existing dataset, the same rules apply as above.

Example – Adding a New Database

The following examples will make use of the **Reclin2 sample dataset**. Suppose we have two datasets want to link, the first step to complete is adding both datasets. To add the new dataset, we'd enter the following information and press the “**Add Dataset**” button. Resulting in a new entry and success notification.

Select An Existing Dataset to Update:

NOTE: For datasets that use the same dataset code/prefix, only one be enabled at a time.

Dataset Code	Dataset Name	Local Dataset File Location	Version	Is Fixed-Width	Enabled
No data available in table					

Previous Next

Or, Add the Dataset Information & View Uploaded Fields Here:

Add Dataset Information

Dataset Code/File Prefix:

Dataset Name:

Dataset Version:

Is The Dataset of Fixed-Width Format?

Uploaded File (Path):

View Uploaded Dataset Fields

Field Name	Field Type	Field Width
id	integer	
lastname	character	
firstname	character	
address	character	
sex	character	
postcode	character	

Previous 1 Next

Field Width:

Update Field Width

+ Add Dataset

Example – Updating an Existing Dataset

To update an existing dataset in the event of wanting to change the dataset code, name, or file location, simply select a dataset from the table and update any of the stored information, pressing “**Update Dataset**” to save any changes back to the database.

Example – Toggling (*Enabled/Disabling*) A Dataset

If you want to use a more up to date version of a dataset without wanting to overwrite the previous file, you may disable the selected dataset by pressing the green “**Toggle Selected Dataset**” button. Doing so will hide that dataset selection when adding or modifying algorithms until it is re-enabled.

When a dataset is disabled, a dataset may use the exact same dataset code since there is no conflict, however attempting to re-enable a dataset while an enabled dataset is using that code will result in an error and will require you to first either rename the enabled dataset or disable it.

Linkage Methods

Description

The linkage methods page allows the user to submit the name and method their own custom linkage class (*written and scripted outside of the UI*). The linkage method will be usable when choosing the implementation of each individual pass of an algorithm.

Details

There are **four** inputs that the user must fill before adding a new usable linkage method.

- The **first** is the **implementation/class name**, which should exactly match the name of the linkage class sourced into the users R environment.
- The **second** is the **linkage method**, which should dictate the different linkage techniques available in the implementation.
- The **third** is the **implementation description**, and serves as a definition of the implementation and method being used which appears in the algorithm summary footnotes of a linkage report (*if generated*).
- The **fourth** is the **versioning implementation**, which helps differentiate implementations and techniques that use the same identifiers.

Example – Adding a New Linkage Method

Consider that we have written our own custom linkage iteration class and want to add it as a selectable method in the UI. We must make sure that the name matches exactly, so say our class in the R environment is called **CustomLinkageClass** and has only one

method which is **DET**. We’d enter the following information and press “**Add Linkage Method**” to submit it.

View the Currently Usable Linkage Methods:

NOTE: Only one combination of implementation name and technique label can exist at a time.

Implementation Name	Linkage Method	Implementation Description	Version
Reclin2Linkage	D	Deterministic linkage pass using the Reclin2 package.	v1
Reclin2Linkage	P	Probabilistic linkage pass using the Reclin2 package.	v1
Reclin2Linkage	M	Machine learning linkage pass using the Reclin2 package.	v1

Previous 1 Next

Or, Add a New Linkage Method Here:

Implementation/Class Name: ?

Linkage Method: ?

Implementation Description: ?

Implementation Version: ?

+ Add Linkage Method

Testable Linkage Algorithms

Description

The testable linkage algorithms page allows the user to select the **left** and **right** dataset that they want to add, modify, view the algorithms of. From this page, you can run the algorithms, regenerate reports, while also modifying the passes, ground truth, and output fields of each algorithm.

Details

The user may add a **new empty algorithm** by filling in the input at the bottom of the page and submitting the algorithm.

Once an algorithm exists, the user may **select** the algorithm and perform any number of modifications on it, such as **archiving** or **publishing** the algorithm, setting it as **default**, enabled or disabling it for **testing purposes**, modifying the **passes**, **ground truth**, or **output**, or export and view **performance measures**.

Example – Adding a New Linkage Algorithm

At the bottom of the page, the user should provide a descriptive (*and unique*) algorithm name and submit it using the green “**Add Linkage Algorithm**” button.

Or, create an empty linkage algorithm here:

Note 1: Algorithm name/descriptor must be unique to the algorithm.

Note 2: One algorithm may be enabled at a time, creating a new algorithm will require you to enable it manually

Algorithm Name/Descriptor: ?

+ Add Linkage Algorithm

Example – Archiving an Algorithm

Archiving an algorithm will remove the selected algorithm from the testable algorithms page and store it for later use in the archived algorithms page. Any number of algorithms may be archived by first selecting an algorithm and pressing and confirming the red “**Archive Selected Algorithm**” button.

Example – Publishing an Algorithm

Publishing an algorithm will remove the selected algorithm from the testable algorithms page and move it to the published algorithms page. Any number of algorithms may be published so long as they don’t share the same algorithm name. This can be done by first selecting an algorithm and pressing and confirming the green “**Publish Selected Algorithm**” button.

Example – Setting an Algorithm as the Default Algorithm

Setting an algorithm as the default algorithm will have affects only in the linkage report generated for that algorithm. When an algorithm is set as **default**, all other algorithms enabled for sensitivity testing will appear in the appendix of that linkage report showing the other algorithms that were considered. An algorithm can be set as default by selecting it and pressing the green “**Set as Default Algorithm**” button. Note that only one algorithm may be set as default at a time.

Example – Toggling (*Enabling/Disabling*) an Algorithm for Testing

Toggling an algorithm for testing will determine whether it appears in the appendix of the default algorithms report. An algorithm can be toggled for testing by selecting it and pressing the green “**Toggle for Testing**” button. Multiple algorithms may be enabled for testing a time.

Archived Linkage Algorithms

Description

The archived linkage algorithms page allows users to view and restore the algorithms they decided were unfit for publication or current testing purposes. Selecting an algorithm will allow the user to briefly look over the algorithm details, but will be unable to modify anything until the user restores the algorithm, bringing it back to the testing page.

Details

The user may select an archived algorithm to view the details of, and to restore. Note that an algorithm can’t be restored if an algorithm being currently tested is using the same name.

Example – Restoring an Algorithm

Once an algorithm is selected, at the bottom of the page, the user can view the individual passes of the algorithm and restore it for testing by clicking the green “**Restore Linkage Algorithm**” button.

Published Linkage Algorithms

Description

The published linkage algorithms page allows users to view and unpublish the algorithms they decided were fit for publication or current testing purposes. Selecting an algorithm will allow the user to briefly look over the algorithm details but will be unable to modify anything until the user unpublishes the algorithm, bringing it back to the testing page.

Details

The user may select a published algorithm to view the details of, and to unpublish. Note that an algorithm can’t be unpublished if an algorithm being currently tested is using the same name.


Example – Unpublishing an Algorithm

Once an algorithm is selected, at the bottom of the page, the user can view the individual passes of the algorithm and unpublish it for testing by clicking the green “**Unpublish Linkage Algorithm**” button.

Example – Running and Saving Published Algorithms


Once the users’ algorithms are published, they may select an output folder where each ran algorithm will have data output to. Each algorithm will be ran like normal but will save all linkage results, algorithm summaries, performances measures, into individual **.Rdata** files labeled in the output folder. This can be done by selecting the output folder and pressing the green “**Run & Save All Published Algorithms**”.

Add Linkage Output Fields

No Folder Has Been Chosen


Select a folder where all output will be saved.

Data will be saved as a singular **.Rdata** file containing a variety of linkage information, including the linked and unlinked records pairs, algorithm summary, performance measures, and missing data indicators.

 Run & Save All Published Algorithms

View Linkage Iterations (Passes)

Description

The view linkage iterations page allows users to view all of the currently added passes for the currently selected algorithm at the top of page, which includes the each pass along with its order, method/technique, blocking and matching criteria, acceptance rules, modification date, modification author, and whether or not it is enabled.

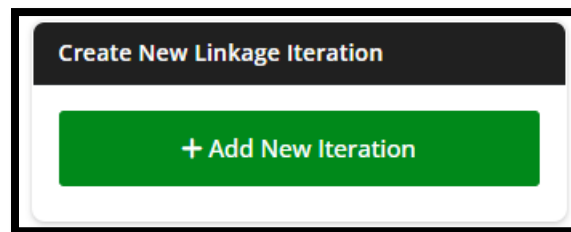
Details

The user may view the details of the algorithm and all its passes at the top of the page, where selecting a pass will allow the user to either modify the selected pass or toggle it (*enable/disable*) from being used when the algorithm is ran.

The user may also add a new pass from scratch, or start from a baseline, using the existing pass of another algorithm.

Example – Creating a New Pass from Scratch

To create a new pass from scratch, the user can press the green “**Add New Iteration**” button which will bring the user to the *Modify Linkage Iterations* page without any inputs filled in.



Example – Creating a New Pass from a Baseline

To create a new pass from a baseline, the user can move to the bottom of the page and view the previously used passes of other algorithms, the user can then select any singular pass and press the green “**Review And Add Iteration**” button which will bring the user to the *Modify Linkage Iterations* page with inputs filled in for the user which can be dropped or updated.

Example – Modifying an Existing Pass

To modify an existing pass, the user can select one of the existing passes from the top of page, and once selected, can press the yellow “**Modify Selected Iteration**” button which will bring the user to *Modify Linkage Iterations* page, pre-populating the inputs with that passes information.

Example – Toggling (*Enabling/Disabling*) an Existing Pass

To modify an existing pass, the user can select one of the existing passes from the top of page, and once selected, can press the green “**Toggle Selected Iteration**” button which will either disable or enable the selected pass. When disabled, the pass will be ignored when the algorithm it belongs to is ran. Note that two iterations with the same name can’t be enabled at the same time.

Modify Linkage Iterations (Passes)

Description

The modify linkage iterations page allows users to create the individual passes that make up an algorithm to their liking, choosing the order they run in, implementations they’ll use, blocking and matching criteria, and acceptance thresholds.

Details

Each iteration being added or modified involves progressing through **six steps**:

- **Step 1:** Providing a name, order, and implementation for the iteration being added.
- **Step 2:** Optionally, if the user is using the linkage rule that involves standardizing names, the user can upload or select a dataset that will act as a lookup table. This dataset must include the columns **unique** and **common** to be considered valid.
- **Step 3:** Select the blocking criteria for this pass, with optional linkage rules for each pair of fields.
- **Step 4:** Select the matching criteria for this pass, with optional linkage and comparison rules for each pair of fields.
- **Step 5:** Optionally, if the user does not know what acceptance threshold to use yet, the user can save their current information, and preview the passes they currently have, opting to get the results from the pass without an acceptance threshold to make an educated decision on a threshold.
- **Step 6:** Select an acceptance threshold to use for the pass being ran.

Example – Creating a Pass for an Algorithm

Consider that for our sample algorithm, we want a pass where we block on the **postal code**, while matching on the **last name**, **first name**, **address** and **sex**, all with a **jaro-winkler threshold** of **0.9**. Our acceptance threshold will be a **match weight** value of **0**.

The **first** step will be filling in the basic information, this can be done by filling in the **three input boxes** under **Step 1**:

Step 1: Enter General Information About The Linkage Iteration**NOTE:** Iterations cannot contain the same name.

Create New Linkage Iteration

Iteration/Pass Name: <input type="text" value="Pass One"/>	Iteration Order/Priority: <input type="text" value="1"/>	Linkage Implementation: <input type="text" value="Reclin2Linkage (P)"/>
---	---	--

The **second** step is optional and since we are not standardizing any names here, we can skip it.

The **third** step involves selecting our blocking criteria. Since we are blocking on, we'll select the **postcode** field for both the left and right data set, then clicking the green “**Add Blocking Variables**” button:

Step 3: Select the Blocking Variables**NOTE:** Blocking variables cannot contain duplicate pairs.

Blocking Variables

Blocking Left Field	Blocking Right Field	Linkage Rules
postcode	postcode	

Previous Next

Left Dataset Field:

Right Dataset Field:

Linkage Rules:

The **fourth** step involves selecting our matching variables. Since we want to match on the first name, last name, address, and sex with a jaro-winkler score of 0.9, we'll add each pair and select the jaro-winkler comparison rule by typing 0.9 for each, pressing the green “**Add Matching Variables**” for all four pairs:

Step 4: Select the Matching Variables**NOTE:** Matching variables cannot contain duplicate pairs.

Matching Variables

Matching Left Field	Matching Right Field	Linkage Rules	Comparison Rules
firstname	firstname		Reclin-Jarowinkler (0.9)
lastname	lastname		Reclin-Jarowinkler (0.9)
address	address		Reclin-Jarowinkler (0.9)
sex	sex		Reclin-Jarowinkler (0.9)

Previous Next

Left Dataset Field:

Right Dataset Field:

Linkage Rules:

Comparison Rules:

The **fifth** step involves saving our current pass information, and running what we have in our algorithm to find a good acceptance threshold. However, since we already want to use a **match weight** of **0** then we can skip this step.

The **sixth/final** step involves selecting an acceptance threshold. Since we want to use a **match weight** of **0**, we press the **Green Pencil Button** and then select **Match Weight** as our acceptance method, typing **0** and confirming using the green “**Add Acceptance Rule**” button:

The last thing we must do now is save our iteration/pass by pressing the green “**Save and Modify Iteration**”.

Comparison Methods

Description

The comparison methods page is accessible from the *Modify Linkage Iterations* page by pressing the blue “**Create Comparison Method**” button within the add comparison rule (*green pencil*) button. Within this page, you can add or modify comparison methods, including the method name, description, and parameters.

Details

When adding a new comparison method, there are **two** inputs for the general information, one for the **name** of the comparison method, and one for a **description** of what the method does. There are also **two** input fields for the parameters section which will allow the user to enter any number of parameter key and description pairs in the database.

Additionally, the user may select an existing comparison method and update the general information, or the parameter information if they wish to make any changes.

Example – Adding a New Comparison Method

Suppose that within our custom linkage class, we also have a custom comparison function that needs a parameter. What we can do is add a **name** and **description** for the method, along with the **parameter key** and **description** of our parameter. Of which, the key must match the name in the code *exactly*. We can submit our values by pressing the green “**Add Comparison Method & Parameters**”:

Or, Add a New Comparison Method and Parameters Here:

Comparison Method General Information ?

Comparison Method Name:

 ?

Comparison Method Description:

 ?

Comparison Method Parameter Information ?

Parameters to be Added:
NOTE: No parameters can share the same name.

Comparison Parameter Key	Comparison Parameter Description
parameter_key	Description of my parameter.

Previous 1 Next

Comparison Parameter Key: ?

Comparison Parameter Description: ?

[+ Add Comparison Parameter](#)

[+ Add Comparison Method & Parameters](#)

Example – Updating an Existing Comparison Method

To update an existing comparison method, the user must first select the comparison method that they wish to modify. Once selected, they may change the method name, description, and select the parameters that should be updated with a new key and description. Once all fields are set to the users liking, they may press the yellow “**Update Comparison Method & Parameters**” button to confirm the changes.

Acceptance Methods

Description

The acceptance methods page is accessible from the *Modify Linkage Iterations* page by pressing the blue “**Create Acceptance Method**” button within the add acceptance rule (*green pencil*) button. Within this page, you can add or modify acceptance methods, including the method name, description, and parameters.

Details

When adding a new acceptance method, there are **two** inputs for the general information, one for the **name** of the acceptance method, and one for a **description** of what the method does. There are also **two** input fields for the parameters section which will allow the user to enter any number of parameter key and description pairs in the database.

Additionally, the user may select an existing acceptance method and update the general information, or the parameter information if they wish to make any changes.

Example – Adding a New Acceptance Method

Suppose that within our custom linkage class, we also have a custom acceptance function that needs a parameter. What we can do is add a **name** and **description** for the method,

along with the **parameter key** and **description** of our parameter. Of which, the key must match the name in the code *exactly*. We can submit our values by pressing the green “Add Acceptance Method & Parameters”:

Or, Add a New Acceptance Method and Parameters Here:

Acceptance Method General Information ?

Acceptance Method Name:

 ?

Acceptance Method Description:

 ?

Acceptance Method Parameter Information ?

Parameters to be Added:

NOTE: No parameters can share the same name.

Acceptance Parameter Key	Acceptance Parameter Description
No data available in table	

Previous Next

Acceptance Parameter Key:

 ?

Acceptance Parameter Description:

 ?

+ Add Acceptance Parameter

+ Add Acceptance Method & Parameters

Example – Updating an Existing Acceptance Method

To update an existing acceptance method, the user must first select the acceptance method that they wish to modify. Once selected, they may change the method name, description, and select the parameters that should be updated with a new key and description. Once all fields are set to the users liking, they may press the yellow “**Update Acceptance Method & Parameters**” button to confirm the changes.

Ground Truth Variables

Description

The ground truth variables page allows users to select the field that appears in both datasets that act to determine whether our linked pairs are true or false matches.

Details

When adding ground truth fields, there are **three inputs** that the user must make note off.

- The **first** is the **left dataset field**.
- The **second** is the **right dataset field**.
- The **third** are any **comparison rules** that should be applied to the ground truth.

The user may also drop a pair of ground truth variables by selecting a row from the table and pressing the red “**Drop Ground Truth Pair**” button.

Example – Adding a New Ground Truth Pair

Within our example algorithm, we have access to a field titled “**id**” which serves as ground truth for the matches. To add this, we’ll select “**id**” for both the left and right ground truth fields, skipping any linkage rules since we don’t need any. Once the two fields are selected, we can apply this ground truth by pressing the green “**Add Ground Truth Pair**” button.

Select A Pair of Ground Truth Variables to Drop:
NOTE: No duplicate ground truth pairs are allowed.



Right Dataset Field Left Dataset Field Linkage Rules

No data available in table

Previous Next

Add Ground Truth Variables

Left Dataset Field: Right Dataset Field: Linkage Rule:

id id  

+ Add Ground Truth Pair

Example – Dropping a Ground Truth Pair

If we decide that we don’t need a ground truth pair, or make a mistake adding our ground truth, then we may select the row we had just added, pressing the red “**Drop Ground Truth Pair**” button to remove it.

Linkage Algorithm Output Fields

Description

The linkage algorithms output fields page allows users to select the field that will appear in the output results after linkage takes place.

Details

When adding output fields, the user can manually enter the output fields, or select the fields used by a previous algorithm.

When manually entering output fields, the user must first select the **field type**, which will change some of the required inputs depending on which type the user selects (*i.e.*, *Derived Age*, *Standardized Values*). The user should fill in all of the required inputs which includes the **field label**, **standardizing field(s)** and **standardizing file** (*if required*).

The user may also drop an output fields by selecting a row from the table and pressing the red **“Drop Output Field”** button.

Example – Adding New Output Fields

Within our example algorithm, suppose that we want to keep Sex and Postal Code in the linkage results. Since we don’t need anything to happen to these two fields, we can select the field types of **Generic/Pass-Through** and **Postal Code Initials** respectively.

Example – Dropping Output Fields

If we decide that we don’t need an output field to appear, or make a mistake adding our output fields, then we may select the row we had just added, pressing the red **“Drop Output Field”** button to remove it.

Linkage Audits

Description

The linkage audits page allows the user to view saved performance measures of their algorithms, and to export them for later use.

Details

The audits page has a date filter that the user can use to limit the saved results by days, weeks, months, years, etc. Clicking on a row/saved record will allow the user to view the details of each record value and will also allow for exporting the selects row(s) as a CSV file by selecting an output folder below.

Run Algorithm(s)

Description

The run algorithms page allows the users to run one or more algorithms at a time, choosing a variety of run-time options for what results are exported, saved, and reports are generated.

Details

Step 1: Select one or more of the algorithms that you are currently testing by clicking on the rows of the table under step 1 that you wish to use.

Step 2: Select the output directory where results such as reports, algorithm summaries, performance measures, histograms, etc, will be saved.

Step 3: Select the type of linkage quality report. You may choose **No Report** which will forgo a PDF report in favor of a CSV output of all the linkage results (*including passes and a linkage indicator*). You may also choose an **Intermediate Report**, which will generate a single report for ALL algorithms being ran, allowing for easier comparisons between many algorithms at once. Or, you may choose **Final Report** which will generate a report for EACH algorithm being ran.

Step 4: Toggle the various output options that you would like to occur during linkage, the options and results are as follows:

- **Unlinked Pairs in Final Output:** Includes all the records that failed to link in the linkage results dataset, if left untoggled, the results dataset will contain ONLY the linked records.
- **Output Linked Iteration Pairs:** If toggled, every pass of an algorithm will export a CSV of the linked pairs, and scores obtained during linkage. If untoggled, no CSV is exported.
- **Output Unlinked Iteration Pairs:** If toggled, every pass of an algorithm will export a CSV of the unlinked pairs before the final *linkage* step takes place.
- **Algorithm Summary:** If toggled, each algorithm will have a CSV output containing each of the pass information, including the matching and blocking criteria, technique, acceptance threshold, and linkage rate.
- **Performance Measures:** If toggled, each algorithm will have a CSV of the performance measures calculated during the algorithm, this includes Linkage Rate, PPV, NPV, Specificity, Sensitivity, and F1 Score.
- **Histograms:** If toggled, probabilistic passes will export PNG images of score distribution.
- **Missing Data Indicators:** If toggled, an additional table will appear in the final report of how many records were missing from the output fields.

- **Save Algorithm Performance:** If toggled, the performance measures calculated during the linkage process will be saved to the database for later auditing purposes.

Step 5: Retain the linked data results by toggling the option. This will store the linkage results, algorithm summary, performance measures, missing data indicators, in an **.Rdata** file and store it in the packages AppData folder.

Step 6: Run the selected linkage algorithm(s) by pressing the green “**Run Linkage**” button. You can view which algorithm is being ran, and which report is being generated by paying attention to the progress bar that appears at the top of the screen after pressing.

Regenerate Report

Description

The regenerate report page allows the users to use the saved **.Rdata** data located in the packages AppData to regenerate quality reports without requiring the need for re-running the algorithm and all its steps.

Details

Step 1: Select one of the saved **.Rdata** files, using a date and time stamp.

Step 2: Select the output directory where results such as reports, algorithm summaries, performance measures, histograms, etc., will be saved.

Step 3: Press the green “**Regenerate Report**” button and watch the progress bar at the top of the screen to know when the report has finished generating.