ipprl_tools Documentation v1.0

July 2019

Contents

1 Compatibility

The *ipprl_tools* package was written using Python 3.6, but should be compatible with any version of Python 3 (Python 3.x).

2 Required Dependencies

The following packages are required dependencies for the *ipprl_tools* package. If you installed *ipprl_tools* through PIP, these dependencies should be installed automatically.

- **Pandas** $\geq$ v0.23
  - `https://pandas.pydata.org`
- **NumPy** $\geq$ v1.16
  - `https://www.numpy.org`
- **SciPy** $\geq$ v1.2

  - `https://www.scipy.org`

3 Optional Dependencies

The following packages are optional dependencies for the *ipprl_tools* package. These dependencies will not be installed automatically when installing *ipprl_tools* with PIP, so they must be installed manually if needed.

- **Fuzzy** ≥ v1.2.2
  - This package is required for the Soundex corruption method. For more information about the package, visit `https://pypi.org/project/Fuzzy/`.
- **Jupyter** ≥ v1.0.0
  - This package is required to view and run the tutorial Jupyter notebook. For more information about Jupyter, visit `https://jupyter.org/`

4 Installation

4.1 PIP Method (Recommended)

To install the package via PIP run the command:

```
pip install git+git://github.com/cu-recordlinkage/ipprl_tools
```

through a command-line interface.

This command will install the *ipprl_tools* package into your default Python environment. This command will also install the required dependencies (Pandas, NumPy, SciPy, etc) if they are not already installed.

4.2 GitHub Method

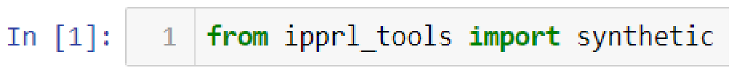
: The source code can also be cloned directly from GitHub using the following command from a command-line interface.

```
git clone https://github.com/cu-recordlinkage/ipprl_tools
```

5 Usage

5.1 Importing the Package

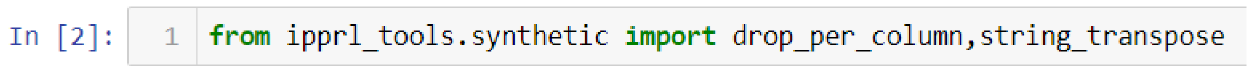
To use *ipprl_tools*, first import the `synthetic` submodule.



```
In [1]: from ipprl_tools import synthetic
```

Figure 1: Importing *ipprl_tools*

This command will import all of the functions defined in the **synthetic** submodule. If you only need a subset of the functions, you can specify the exact functions to import as well:



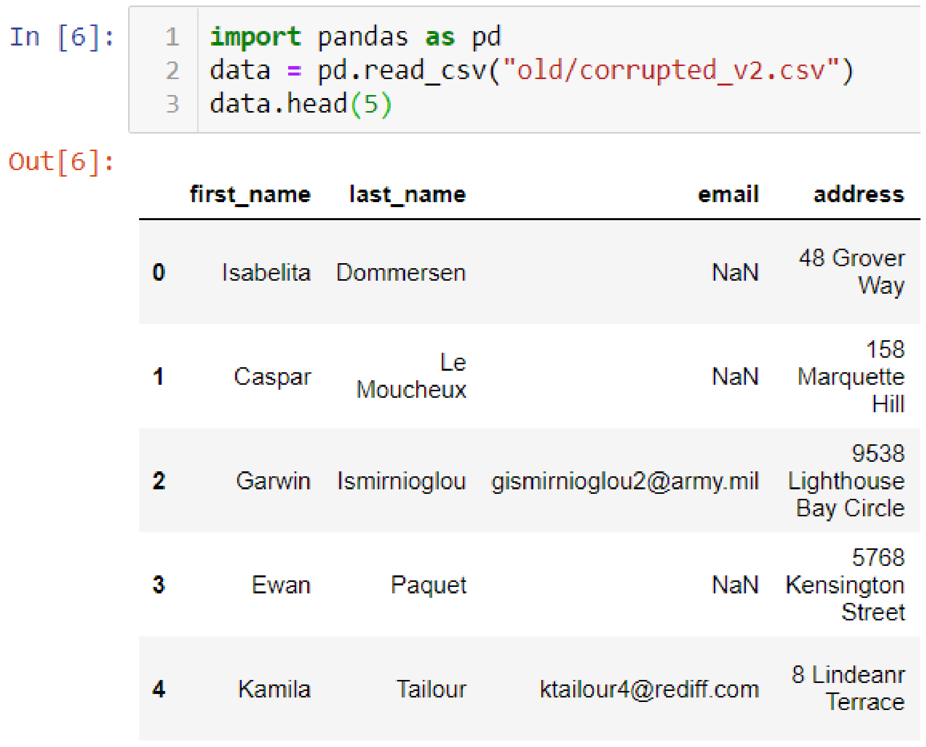
```
In [2]: from ipprl_tools.synthetic import drop_per_column,string_transpose
```

Figure 2: Importing specific functions

5.2 Data Prerequisites

The synthetic data functions are designed to operate on Pandas DataFrame objects. Pandas DataFrame objects are data structures, similar to R dataframes, which contain data organized in named columns.

Before using the synthetic data methods, first read the raw data in as a Pandas DataFrame. The example below shows reading a CSV file in using Pandas.
For additional ways to import data using Pandas, refer to the Pandas Documentation here: Pandas Documentation: IO



```
In [6]: import pandas as pd
        data = pd.read_csv("old/corrupted_v2.csv")
        data.head(5)
Out[6]:
```

	first_name	last_name	email	address
0	Isabelita	Dommersen	NaN	48 Grover Way
1	Caspar	Le Moucheux	NaN	158 Marquette Hill
2	Garwin	Ismirnioglou	gismirnioglou2@army.mil	9538 Lighthouse Bay Circle
3	Ewan	Paquet	NaN	5768 Kensington Street
4	Kamila	Tailour	ktailour4@rediff.com	8 Lindeanr Terrace

Figure 3: Reading in a CSV file with Panda

6 Function Documentation

6.1 String Manipulation Methods

String manipulation methods are designed to mimic common typographic errors that can occur when data is entered into a record system.

6.1.1 *string_delete(data, indicators, delete_num, delete_freq, columns=None)*

This function randomly deletes characters from strings in one or more columns of the DataFrame. The operation is performed in-place, and information about which values were changed is available in the *indicators* variable.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *delete_num* - Parameter choosing an upper bound on the number of deletions per value (i.e. Delete up to 3 characters). This value can either be a scalar *integer* or a list of *integer* to specify a different number of characters for each column to be modified.
- *delete_freq* - The percentage of elements in a column which should be modified (i.e. 0.2 == modify 20% of the values in a column). This value can either be a scalar *float* or a list of *float* to specify a different frequency for each column.
- *columns* - An optional list of columns to operate on. These columns should correspond to column names in the DataFrame being operated on. If this parameter is not passed, defaults to operating on all of the columns in the DataFrame.

6.1.2 *string_transpose(data, indicators, trans_num, trans_freq, columns=None)*

This function randomly transposes character positions within a string. This operation is intended to simulate a mis-typed entry into a record system.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *trans_num* - Parameter specifying an upper bound on the number of transpositions per string value. This value can either be a scalar *integer* or a list of *integer* to specify a different number of characters for each column in *columns*.
- *trans_freq* - The percentage of elements in a column which should be modified (i.e. 0.2 == modify 20% of the values in the column). This value can also be scalar *float* or a list of *float*.
- *columns* - An optional list of columns to operate on. These columns should correspond to column names in the DataFrame being operated on. If this parameter is not passed, defaults to operating on all of the columns in the DataFrame.

6.1.3 *string_insert_alpha(data, indicators, insrt_num, insrt_freq, columns=None)*

This function inserts lowercase alphabetical characters (a-z) into a strings randomly. This is to simulate if a field was mistakenly entered with some extraneous characters.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *insrt_num* - Parameter specifying the upper bound on the number of characters to insert into a string. This value can either be a scalar *integer* or a list of *integer* to specify a different upper bound for each column in *columns*
- *insrt_freq* - The percentage of elements in a column which should be modified. (i.e. 0.2 == modify 20% of the rows in the column. This can also be a scalar *float* or a list of *float* to specify a different percentage for each column in *columns*.

6.1.4 *string_insert_numeric(data, indicators, insrt_num, insrt_freq, columns=None)*

This function is identical in operation to *string_insert_alpha()*, but it randomly inserts numerical characters (0-9) instead of alphabetical characters. This function is suitable for performing corruption on values where alphabetical errors would not be realistic, like in a Phone Number or SSN field.

Paramters:
See Section 6.1.3 for details on parameters.

6.2 Replacement Methods

Unlike the string manipulation methods in Section 6.1, which alter the existing contents of a value by performing string-based manipulations, replacement methods completely change the values contained in a column by replacing each value with another.

6.2.1 *soundex_string_corrupt(data, indicators, corrupt_name_pct, columns=None)*

The Soundex string corruption method works by scanning over the values in each column, and building a Soundex code lookup table that maps Soundex codes to all of the values within the column that contain that Soundex value.

The function then iterates through each of the values to be replaced, and replaces the original value with another value from the column containing the same Soundex code.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *corrupt_name_pct* - The percentage of values within the column to replace with their Soundex equivalents. This percentage can either be a scalar *float*, or a list of *float* to specify a different percentage for each column in *columns*.
- *columns* - An optional list of columns to operate on. These columns should correspond to column names in the DataFrame being operated on. If this parameter is not passed, defaults to operating on all of the columns in the DataFrame.

6.2.2 *drop_per_column(data, indicators, columns=None, drop_num = None, drop_pct=None)*

The drop-per-column operation randomly chooses some values in the selected columns to delete completely. This operation is intended to simulate missing or incomplete information within a record system.

A deleted or null value is represented by the empty string ”” in the DataFrame.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *columns* - An optional list of columns to operate on. These columns should correspond to column names in the DataFrame being operated on. If this parameter is not passed, defaults to operating on all of the columns in the DataFrame.
- *drop_num* - An optional parameter specifying the number of values to drop as an absolute number of rows. This value may either be a *integer* number of rows, or a list of *integer* corrsesponding to the columns in *columns.* Either this parameter or *drop_pct* should be specified when calling the function, but not both.
- *drop_pct* - An optional parameter specifying the number of values to drop as a percentage of the total number of rows. This value may either be a *float* or a list of *float.* Either this value or *drop_num* should be specified, but not both.

6.2.3 *edit_values(data, swap_set, indicators, pct_edits, columns=None)*

This function swaps randomly chosen rows from *data* with randomly chosen rows (from the same column name) from *swap_set.*

This function is intended to replicate natural information change over time.

Parameters:

- *data* - The Pandas DataFrame to be modified. The DataFrame will be modified in-place.
- *swap_set* - This parameter should be a Pandas DataFrame with column names that match the column names in *data.* This DataFrame represents the ”set” from which values will be randomly chosen to replace values in *data.*
- *indicators* - A sparse matrix containing information about modified entries in the DataFrame.
- *pct_edits* - This parameter represents the percentage of values within a column with should be edited. This can either be a scalar *float* or a list of *float* to specify a different percentage for each element of *columns.*
- *columns* - An optional list of columns to operate on. These columns should correspond to column names in the DataFrame being operated on. If this parameter is not passed, defaults to operating on all of the columns in the DataFrame.

7 Utilities

In addition to the synthetic data corruption methods, *ipprl_tools* also provides some utility functions to make generating synthetic datasets easier.

7.1 Data Utilities

7.1.1 *split_dataset(raw_data, overlap_pct)*

This function is a simple utility that takes a DataFrame as input, then returns two DataFrames containing two separate Datasets, where *overlap_pct* of the rows from the dataset occur in both DataFrames, and the rests of the rows are unique to their repsective DataFrame.
This function also returns a *ground_truth* variable, which is a list of tuples containing the IDs of the matching rows in the *left* and *right* datasets.
Parameters:

- *raw_data* - The DataFrame containing the raw data to build the dataset from.
- *overlap_pct* - The percentage of rows in *raw_data* which should be used as ground-truth positive linkages. (i.e. 0.5 == 50% of the rows will be considered as positive linkages in the ground-truth data.

Returns:

- *left* - One half of the generated synthetic dataset.
- *right* - The other half of the generated synthetic dataset.
- *ground_truth* - A list of tuples containing the IDs of the row pairs that overlap between *left* and *right*