surgeo

Release 1.1.2

Apr 04, 2021

Contents

1 Contributors **3**

2 Overview **5**

3 Background **7**

4 Usage **9**
- 4.1 As a Module . . . 9
- 4.2 As a Program . . . 10

5 BISG by Example **11**
- 5.1 Using Surname Data To Obtaining $P(i \mid j)$ . . . 12
- 5.2 Using Geocoding Data to Obtain $r(k \mid i)$ . . . 12
- 5.3 Multiplying Probabilities to Obtain $u(i, j, k)$ . . . 12
- 5.4 Normalizing Final Probability Vector $q(i \mid j, k)$ . . . 13

6 Classes **15**
- 6.1 Model Classes . . . 15
- 6.2 Application Classes . . . 20
- 6.3 Utility Classes . . . 21

7 Indices and tables **23**

Table of Contents

- *Surgeo*
- *Contributors*
- *Overview*
- *Background*
- *Usage*
- *BISG by Example*
- *Classes*
- *Indices and tables*

CHAPTER 1

Contributors

- Adam Weeden
- Algorex Health
- Theo Naunheim

CHAPTER 2

Overview

Surgeo is a collection of tools that allows you to determine race by using substitute (or "proxy") variables such as first name, last name and ZIP code. These proxies are useful for determining race in contexts where this data might not otherwise be available, such as in the public health and fair lending.

This open source software contains a number of tools for conducting this analysis, including:

1. Python modules for doing first name-based, geocode-based, surname-based, Bayesian Improved First Name Surname Geocoding (BIFSG), and Bayesian Improved Surname Geocoding (BISG) calculations;
2. A command line interface (CLI) for automating batch processing; and,
3. A graphical user interface (GUI) for one-off batch processing.

The base data for this module is sourced from publically available data, specifically:

1. 2010 United States Census Summary File 1 data set;
2. 2010 United States Census Frequently Occurring Surnames data set; and,
3. Demographics aspect of first names data set[1].

[1] Konstantinos Tzioumis, "Data for: Demographic aspects of first names". Harvard Dataverse (2018), V1 https://doi.org/10.7910/DVN/TYJKEZ

CHAPTER 3

Background

Note: If you are only looking for how to use Surgeo and are not interested in the mechanics, please skip to *Usage*.

There are a variety of mechanisms by which one can impute race based on generally available characteristics. Three popular approaches are surname-based (last name), forename-based (first name), and location-based.

The surname-based approach works because certain last names are found much more often in specific racial/ethnic groups. For example, over 90% of people with the surname "Hernandez" identify as Hispanic according to Census data. Similarly, over 90% of those with the last name of "Yoo" identify as Asian or Pacific Islander.

The forename-based approach works similarly to the surname-based approach, with first name being used as the proxy for racial/ethnic probability.

The location-based approach works because racial/ethnic groups often live in similar areas. For example, roughly 75% of Americans living in the U.S. Virgin Islands are Black. Similarly, Vermont is approximately 95% White.

These approaches all have their strengths and weaknesses. Surname-based approaches are excellent at picking out Hispanic and Asian/Pacific Islander surnames. Surname-based approaches are comparatively weaker when trying to identify Blacks, Whites, and Native Americans by name. Forename-based approaches compensate for weaknesses in the surname-only approach[2]. Location-based approaches are much better at discerning Black from White areas, but are not particularly good at separating Asian and Hispanic populations[3].

It is possible to combine these approaches using Bayesian inference; this provides result that is superior to using any approach alone[4]. For a combination of surname-based and location-based, for example, this can be performed as follows.

First one takes the probability of race given a particular surname (prior probability) and then multipies it by ratio of the population in a particular geographical area given that race. The result is an updated (posterior) probability of race

[2] Ioan Voicu "Using First Name Information to Improve Race and Ethnicity Classification". Statistics and Public Policy (2018) 5:1, 1-13, https://www.tandfonline.com/doi/full/10.1080/2330443X.2018.1427012

[3] Consumer Financial Protection Bureau. "Using Publicly Available Information to Proxy for Unidentified Race and Ethnicity". 2014. https://files.consumerfinance.gov/f/201409_cfpb_report_proxy-methodology.pdf

[4] Elliott, M.N., Morrison, P.A., Fremont, A. et al. "Using the Census Bureau's surname list to improve estimates of race/ethnicity and associated disparities". Health Serv Outcomes Res Method (2009) 9: 69. https://link.springer.com/article/10.1007/s10742-009-0047-1

given a particular surname and a particular geographic area. This can be further refined with the addition of forename data.

For additional detail on how this works, please see *BISG by Example*.

CHAPTER 4

Usage

Surgeo can be used as a stand-alone executable or a Python module. Details on each of these approaches are addressed below.

4.1 As a Module

Surgeo is best used as a module with pandas and the Jupyter notebook. The general workflow is to import surgeo, then create a model, and then run the analysis using the *get_probabilities()* method and the data being analyzed.

The available models are:

1. *BIFSGModel*, which takes a series of first names, a series of surnames, and a series of ZIP codes and gives the BIFSG results of those inputs (e.g. someone named HECTOR DIAZ in ZIP 79902 has a 80% probability of being hispanic).
2. *FirstNameModel*, which uses a series of first names and gives the racial percentage of that first name (e.g. 92% of people with the first name AARON are white).
3. *GeocodeModel*, which uses a series of ZIP codes and gives the racial percentage make up of that ZIP code (e.g. 81% of those people in ZIP 65201 are white).
4. *SurnameModel*, which uses a series of surnames and gives the racial percentage of that surname (e.g. 5% of people with the surname DIAZ are white); and,
5. *SurgeoModel*, which takes a series of surnames and a series of ZIP codes and gives the BISG results of those inputs (e.g. someone named DIAZ in ZIP 65201 has a 26% probability of being white).

```
import pandas as pd
import surgeo

# Instatiate your model (or all five)
fsg = surgeo.BIFSGModel()
sg = surgeo.SurgeoModel()
f = surgeo.FirstNameModel()
g = surgeo.GeocodeModel()
```

(continues on next page)

(continued from previous page)

```
s = surgeo.SurnameModel()

# Create pd.Series objects to analze (or load them)
first_names = pd.Series(['HECTOR', 'PHILLIP', 'JANICE'])
surnames = pd.Series(['DIAZ', 'JOHNSON', 'WASHINGTON'])
zctas = pd.Series(['65201', '63144', '63110'])

# Get results using the get_probabilities() function
fsg_results = fsg.get_probabilities(first_names, surnames, zctas)
sg_results = sg.get_probabilities(surnames, zctas)
f_results = f.get_probabilities(first_names)
g_results = g.get_probabilities(zctas)
s_results = s.get_probabilities(surnames)

# Show Surgeo BIFSG results
fsg_results
```

4.2 As a Program

To use the GUI, type in "surgeo_gui" into your shell or open the application after installation in your Start Menu (Windows only).

Simply supply the input and output file paths as required and then select the model type. Then select any column names in your .xlsx/.csv associated with surname/ZIP, and click "Execute". The results will be written to the output file.

To use the CLI, type in "surgeo_cli" followed by your arguments. The first argument is the input file path, the second argument is the output file path, and the third argument is the model being used. There are also optional argument to define the name of the ZCTA and surname columns if they are not the defaults. The default values are:

1. "zcta5" for geographical area;
2. "name" for surname; and,
3. "first_name" for first name.

CHAPTER 5

BISG by Example

This section is intended to provide a high-level walkthrough of how one might perform BISG by hand. For this example we are going to determine the probability of the surname "Garcia" in the ZIP code "63144". The formula we use to calculate these probabilties is:

$$q(i \mid j,k) = \frac{u(i,j,k)}{u(1,j,k) + u(2,j,k) + u(3,j,k) + u(4,j,k) + u(5,j,k) + u(6,j,k)}$$

Where:

$$u(i,j,k) = P(i \mid j) \times r(k \mid i)$$

And where:

$P(i \mid j)$ is the probability of a selected race given surname
$r(k \mid i)$ is the probability of a selected ZCTA of residence given race
k is Census Block
j is Surname
i is Race

And where:

1 is $i =$ Hispanic
2 is $i =$ White
3 is $i =$ Black
4 is $i =$ Asian or Pacific Islander
5 is $i =$ American Indian / Alaska Native
6 is $i =$ Multi Racial

The steps for performing this calculation are broken down below. Generally speaking, the steps are:

1. Obtain $P(i \mid j)$ (probability of race given surname) and $r(k \mid i)$ (proprtion of ZCTA given race);
2. Multiply each probability with each proprtion on a race-by-race basis to obtain $u(i,j,k)$ (probability intermediate) for each race; and then,

3. Divide the posterior probability of each individual race dividing the intermediate above by the sum of all intermediates (normalization constant) to give $q(i \mid j, k)$ (the probability of race given a surname and ZIP code).

5.1 Using Surname Data To Obtaining $P(i \mid j)$

According to the Census Bureau data, the surname "GARCIA" has the following racial/ethnic probabilities.

White	Black	API	Native	Multiple	Hispanic
0.0538	0.0045	0.0141	0.0047	0.0026	0.9203

Note: You may notice that "Hispanic" (i.e. whether someone comes from a Spanish-speaking culture) is treated like a race, even though there are white hispanics, black hispanics, etc.). Consequently "White" is actually "White, Non-Hispanic", and "Hispanic" is all races that come from a Spanish-speaking culture.

In other words, these are the probabilities (P) of a race (i) given the surname (j) "Garcia", which is represented mathematically as $P(i \mid j)$. It is the so-called "prior probability" that we will be updating with our location information.

5.2 Using Geocoding Data to Obtain $r(k \mid i)$

According to Census Bureau data, the ZIP code "63144" contains this proportion of the United State's race populations.

White	Black	API	Native	Multiple	Hispanic
0.000039	0.000007	0.000037	0.000005	0.000025	0.000005

Note: ZIP Code Tabulation Areas (ZCTAs) are not identical to ZIP codes. Because ZIP codes change from year-to-year, the Census Bureau uses what it calls ZCTAs. These ZCTAs are rough approximations of the ZIP code geographic area–but it remains static after being created.

Put simply, .0039% of the United State's White population lives within the 63144 ZIP code; .0007% of the US's Black populaiton lives within 63144, etc. In other words, these are the proportions (r) of a selected ZIP (k) given a particular race (i), which is represented mathematically as $r(k \mid i)$.

5.3 Multiplying Probabilities to Obtain $u(i, j, k)$

The next step is to multiply our vectors together to obtain $u(i, j, k)$, which is an intermediate used to calculate the final posterior probability. This uses is a simple element-wise multiplication operation defined by:

$u(i, j, k) = P(i \mid j) \times r(k \mid i)$

Step-by-step, we can take the numbers above from $P(i \mid j)$:

White	Black	API	Native	Multiple	Hispanic
0.0538	0.0045	0.0141	0.0047	0.0026	0.9203

... and then multiply it by our numbers for $r(k \mid i)$:

White	Black	API	Native	Multiple	Hispanic
0.000039	0.000007	0.000037	0.000005	0.000025	0.000005

... which then results in $u(i, j, k)$:

White	Black	API	Native	Multiple	Hispanic
2.07e-06	3.04e-08	5.21e-07	2.30e-08	6.48e-08	4.33e-06

As you can see from the above, the "White" probability for this surname is 0.0538 and the "White" proportion for this ZIP is .000039. If we multiply 0.0538 times .000039, we get 0.00000207. This is also done for the remaining races.

5.4 Normalizing Final Probability Vector $q(i \mid j, k)$

The final step is defined by the following equation:

$q(i \mid j, k) = \frac{u(i,j,k)}{u(1,j,k) + u(2,j,k) + u(3,j,k) + u(4,j,k) + u(5,j,k) + u(6,j,k)}$

What this means is simply that in order to obtain our final probability for a given race i, we must take the intermediate value for that race and then divide it by the sum of all races. This normalizes our probabilities and ensures they will sum to 1. For example, to run this calculation for "White" the formula would read:

$q(\text{white} \mid \text{Garcia}, 63144) = \frac{u(\text{white,Garcia,63144})}{u(\text{hispanic,Garcia,63144}) + u(\text{white,Garcia,63144}) + u(\text{black,Garcia,63144}) + u(\text{api,Garcia,63144}) + u(\text{native,Garcia,63144}) + u(\text{multi,Garcia,}}$

And pluging in the intermediate values from $u(i, j, k)$:

White	Black	API	Native	Multiple	Hispanic
2.07e-06	3.04e-08	5.21e-07	2.30e-08	6.48e-08	4.33e-06

We would have the following calculation for "White" (29.4%):

$.294 = \frac{2.07e{-}06}{4.33e{-}06 + 2.07e{-}06 + 3.04e{-}08 + 5.21e{-}07 + 2.30e{-}08 + 6.48e{-}08}$

And the following final percentages for "GARCIA" and "63144":

White	Black	API	Native	Multiple	Hispanic
.294	.004	.084	.003	.009	.615

This comes out very much like we might expect–the 63144 ZIP skews White, but "GARCIA" is overwhelmingly a Hispanic.

CHAPTER 6

Classes

6.1 Model Classes

6.1.1 SurgeoModel

class surgeo.models.surgeo_model.**SurgeoModel**(*geo_level='ZCTA'*)

Subclass for running a Bayesian Improved Surname Geocode model.

This class:

1. Loads the appropriate surname and geocode lookup dataframes upon instantiation;
2. Exposes a public get_probabilities() function to compute race probabilities based on proxy data (namely surnames and ZIP codes); and,
3. Contains a number of helper functions for cleaning ZCTA/names, multiplying probabilities, checking input values, and obtaining ZCTA/name data components.

Notes

The surname probability dataframe for this model is identical to that used for the SurnameModel (*prob_race_given_surname_2010.csv*); the geocode probability dataframe for this model is not the same as that used for the GeocodeModel. This model uses the *prob_zcta_given_race_2010.csv* file, which has the percentage of a particular race that falls within that ZCTA (e.g. .002% of all White US citizens live within this ZIP code). The GeocodeModel uses the *prob_race_given_zcta_2010.csv* file, which has the race percentages for a given ZCTA (e.g. 90% of ZCTA 63144 is White).

The manner in which the geography data file was created can be found in the "fetch_geography" Jupyter notebook.

This is based of the following general formula from Elliott et al[5].

[5] Elliott, M.N., Morrison, P.A., Fremont, A. et al. Using the Census Bureau's surname list to improve estimates of race/ethnicity and associated disparities. Health Serv Outcomes Res Method (2009) 9: 69. https://link.springer.com/article/10.1007/s10742-009-0047-1

$q(i \mid j,k) = \frac{u(i,j,k)}{u(1,j,k)+u(2,j,k)+u(3,j,k)+u(4,j,k)+u(5,j,k)+u(6,j,k)}$

Where:

$u(i,j,k) = P(i \mid j) \times r(k \mid i)$

And where:

$P(i \mid j)$ is the probability of a selected race given surname
$r(k \mid i)$ is the probability of a selected census block of residence given race
k is Census Block
j is Surname
i is Race

And where:

1 is $i =$ Hispanic
2 is $i =$ White
3 is $i =$ Black
4 is $i =$ Asian or Pacific Islander
5 is $i =$ American Indian / Alaska Native
6 is $i =$ Multi Racial

References

get_probabilities(*names*, *geo_df*)

Obtain a set of BISG probabilities for name/ZCTA series

This method first takes the data and checks to see if the data is formatted appropriately. It triggers the _get_surname_probs() and _get_geocode_probs() helper function to merge the probabilities for the inputs with their looked-up values. It then runs the _combined_probs() helper function to actually conduct the data calculation and obtain the BISG probabilities. It finally runs the _adjust_frame() method to concatenate the inputs and outputs in a single convenient frame.

Parameters

- **names** (*pd.Series*) – A series of names to use for the BISG algorithm
- **geo_df** (*Union[pd.Series, pd.DataFrame]*) – A series of target ZIP/ZCTA codes or State County Tract for the BISG algorithm

Returns Dataframe of BISG probability results

Return type pd.DataFrame

6.1.2 BIFSGModel

class surgeo.models.bifsg_model.**BIFSGModel**

Subclass for running a Bayesian Improved First Name Surname Geocode model.

This class:

1. Loads the appropriate first name, surname, and geocode lookup dataframes upon instantiation;
2. Exposes a public get_probabilities() function to compute race probabilities based on proxy data (namely first names, surnames and ZIP codes); and,

3. Contains a number of helper functions for cleaning ZCTA/names, multiplying probabilities, checking input values, and obtaining ZCTA/name data components.

Notes

The surname probability dataframe for this model is identical to that used for the SurnameModel (*prob_race_given_surname_2010.csv*); the first name probability dataframe for this modelis not the same as that used for the FirstNameModel. his model uses the *prob_first_name_given_race_harvard.csv* file, which has the percentage of a particular race that uses that first name (e.g. 3% of all White US citizens have the first name AARON). The FirstNameModel uses the *prob_race_given_first_name_harvard.csv* file, which has the race percentages for a given first name (e.g. 92% of people with the first name AARON are White); the geocode probability dataframe for this model is not the same as that used for the GeocodeModel. This model uses the *prob_zcta_given_race_2010.csv* file, which has the percentage of a particular race that falls within that ZCTA (e.g. .002% of all White US citizens live within this ZIP code). The GeocodeModel uses the *prob_race_given_zcta_2010.csv* file, which has the race percentages for a given ZCTA (e.g. 90% of ZCTA 63144 is White).

The manner in which the first name data file was created can be found in the "fetch_first_names" Jupyter notebook.

The manner in which the geography data file was created can be found in the "fetch_geography" Jupyter notebook.

This is based of the following general formula from Voicu[6].

$$q(r \mid s, f, g) = \frac{u(r,s,f,g)}{u(1,s,f,g) + u(2,s,f,g) + u(3,s,f,g) + u(4,s,f,g) + u(5,s,f,g) + u(6,s,f,g)}$$

Where:

$u(r, s, f, g) = p(r \mid s) \times p(g \mid r) \times p(f \mid r)$

And where:

$p(r \mid s)$ is the probability of a selected race given surname
$p(g \mid r)$ is the probability of a selected census block of residence given race
$p(f \mid r)$ is the probability of a selected first name given race
g is Census Block
f is First Name
s is Surname
r is Race

And where:

1 is $r =$ Hispanic
2 is $r =$ White
3 is $r =$ Black
4 is $r =$ Asian or Pacific Islander
5 is $r =$ American Indian / Alaska Native
6 is $r =$ Multi Racial

[6] Ioan Voicu "Using First Name Information to Improve Race and Ethnicity Classification". Statistics and Public Policy (2018) 5:1, 1-13, https://www.tandfonline.com/doi/full/10.1080/2330443X.2018.1427012

References

get_probabilities(*first_names*, *surnames*, *zctas*)

Obtain a set of BIFSG probabilities for first_name/surname/ZCTA series

This method first takes the data and checks to see if the data is formatted appropriately. It triggers the _get_surname_probs(), _get_first_name_probs(), and _get_geocode_probs() helper functions to merge the probabilities for the inputs with their looked-up values. It then runs the _combined_probs() helper function to actually conduct the data calculation and obtain the BIFSG probabilities. It finally runs the _adjust_frame() method to concatenate the inputs and outputs in a single convenient frame.

Parameters

- **first_names** (*pd.Series*) – A series of first names to use for the BIFSG algorithm
- **surnames** (*pd.Series*) – A series of surnames to use for the BIFSG algorithm
- **zctas** (*pd.Series*) – A series of ZIP/ZCTA codes for the BIFSG algorithm

Returns Dataframe of BIFSG probability results

Return type pd.DataFrame

6.1.3 FirstNameModel

class surgeo.models.first_name_model.**FirstNameModel**

Provides a way to look up race percentages by first name.

This class uses a get_probabilities() method to provide a simple mechanism for obtaining race data. It is created using a simple join of a race data table and the first names that are input.

Notes

The manner in which the first name data file was created can be found in the "fetch_first_names" Jupyter notebook.

The first name probability dataframe for this model is generated from the *prob_race_given_first_name_harvard.csv* file.

get_probabilities(*names*)

Obtain race probabilities for a set of first names.

Parameters **names** (*pd.Series*) – names to which to attach race probability data

Returns Dataframe of race probability results

Return type pd.DataFrame

6.1.4 GeocodeModel

class surgeo.models.geocode_model.**GeocodeModel**(*geo_level='ZCTA'*)

Provides a way to look up race percentages by ZIP/ZCTA code

This class uses a get_probabilities() method to provide a simple mechanism for obtaining race data. It is created using a simple join of a race data table and the ZIPs/ZCTAs that are input.

Notes

ZIP Code Tabulation Areas (ZCTAs) are approximations for US Postal ZIP codes. While ZIP codes change, ZCTAs are static for a given census cycle. They are not identical.

The manner in which the geography data file was created can be found in the "fetch_geography" Jupyter notebook.

This does not use the same Geocode data as the Surgeo class. This model uses the *prob_race_given_zcta_2010.csv* file, which has the race percentages for a given ZCTA (e.g. 90% of ZCTA 63144 is White). The SurgeoModel uses the *prob_zcta_given_race_2010.csv* file, which has the percentage of a particular race that falls within that ZCTA (e.g. .002% of all White US citizens live within this ZIP code).

get_probabilities(*zctas*)

Obtain race probabilities for a set of ZIP codes or ZCTAs.

Parameters **zctas** (*pd.Series*) – ZIPs/ZCTAs to which to attach race probability data

Returns Dataframe of race probability results

Return type pd.DataFrame

get_probabilities_tract(*geo_df*)

Obtain race probabilities for a set of State, County, Tract.

Parameters **geo_df** (*pd.DataFrame*) – DF of ['state','county','tract'] codes to retrun probabilities for

Returns Dataframe of race probability results

Return type pd.DataFrame

6.1.5 SurnameModel

class surgeo.models.surname_model.**SurnameModel**

Provides a way to look up race percentages by surname.

This class uses a get_probabilities() method to provide a simple mechanism for obtaining race data. It is created using a simple join of a race data table and the surnames that are input.

Notes

The manner in which the surname data file was created can be found in the "fetch_surnames" Jupyter notebook.

The surname probability dataframe for this model is generated from the *prob_race_given_surname_2010.csv* file.

get_probabilities(*names*)

Obtain race probabilities for a set of surnames.

Parameters **names** (*pd.Series*) – names to which to attach race probability data

Returns Dataframe of race probability results

Return type pd.DataFrame

6.1.6 BaseModel

class surgeo.models.base_model.**BaseModel**

Base class for the first name, surname, geocode, bifsg, and surname-geocode models.

Class creation is greatly simplified by placing most of the funcionality wihtin a single base class and leaving only small areas of responsibility for the subclass. This base class does the following operations:

1. Creating functions to provide lookup dataframes; and,
2. Housing normalization routines for dirty ZIP code and name data.

Note: Names are normalized in a manner consistent with Word et. al (2007)[7]. This includes removing all whitespace/punctuation/digits, making the strings upper case, and then removing elements such as "JR", "SR", "IV" from the tail of the string. An example would be "Dav 3idson" being translated to "DAVIDSON".

ZCTAs, which serve as a proxy for ZIP codes, are normalized by simply translating them to stirngs and then .zfill()ing them. An example would be "531" to "00531".

References

6.2 Application Classes

6.2.1 SurgeoCLI

class surgeo.app.surgeo_cli.**SurgeoCLI**

A CLI application class to function as an executable

This script adds surgeo to path and runs a simple command line script. The class pulls in a parser, parses the command line arguments as needed, loads the data, processes the data, and sends the output to a file. It uses the "main()" function and then uses other methods as helpers.

Example

main()

This is the public interface function for this CLI.

In summary, this function triggers various routines that:

1. Read arguments;
2. Take a user defined path argument and load an Excel or CSV into a dataframe;
3. Route that dataframe to a speciic processing function based on the "type" function argument (e.g. first_name, surname, geocoding, bifsg, or surgeo);
4. Optional specifies the column names to analyze (if not using the default "zcta5", "name", or "first_name" headers);
5. Runs the appropriate algorithm and returns a new dataframe;
6. Writes the resulting data to a new CSV based to output path specified by user.

[7] Word, David L., Charles D. Coleman, Robert Nunziata and Robert Kominski. 2007. Demographic Aspects of Surnames from Census 2000. http://www2.census.gov/topics/genealogy/2000surnames/surnames.pdf.

Raises `surgeo.utility.SurgeoException` – Raised if 1) file endings are not correct, 2) inappropriate columns are specified, 3) an incorrect model type is supplied, or 4) an inappropriate outputs are not specified.

6.2.2 SurgeoGUI

class `surgeo.app.surgeo_gui.`**`SurgeoGUI`**

A GUI application class to function as an executable

This script creates a single window tkinter application. This application allows a user to specify inputs/outputs, specify which model that they want to run, and then define the column headers for the ZCTA, first name, and surname fields as applicable. It then runs the model and stores the results in a file.

It also has various helper functions to integrate the surgeo logic within the program.

It currently supports .xlsx, .xls, and .csv inputs; it currently supports .xlsx and .csv outputs.

main()

This is the entry point for the GUI program.

It calls a function to create a root window, then adds widgets to that root window, and then finally triggers the tkinter main loop.

6.2.3 SurgeoCommonEntry

class `surgeo.app.common_entry.`**`SurgeoCommonEntry`**

An entry point for both the GUI and CLI Surgeo applications

This class simply gets the number of args sent to the entry point. If there is a single argument, the GUI is run. If addtional arguments are supplied, the CLI is run. The CLI will then parse the arguments as nothing it is not necessary to pass the arguments from the common entry to the CLI.

main()

The entry point's main function

This gets the number of arguments supplied. If no arguments are supplied in addition to the 'surgeo' command, the GUI is run. Otherwise, the CLI is run.

6.3 Utility Classes

6.3.1 SurgeoException

exception `surgeo.utility.surgeo_exception.`**`SurgeoException`**

This is an application specific Exception class.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search