SURPY 1.1.10 user manual

Translational Medicine Research Center (TMRC),

together with the Department of Surgery, Prince of Songkla University, Thailand

March 2021

Latest updated version August 2022

SURPY (เซอ-ปี้) is a Python module built by a group of surgical researchers with a primary aim to be used as a command line interface (CLI) tool for outcome data analysis, beginning from taking input file in .xlsx format, aggregate the basic statistics and basic hypothesis testing for association between a variable and a defined outcome. The following table describes packages within the module. SURPY and related packages are available from https://github.com/sasurasa/Surgical-Outcome-Analysis-on-Python/tree/SURPY. The package uses Pandas, Numpy, Scipy and Matplotlob as its base packages.

Package	Functions
soap.soapsheetin(*path*)	Acquire a data sheet in EXCEL format (.xlsx) to a Pandas (PD) dataframe. Take a file path to the EXCEL sheet as an input and return a dataframe as an output.
soap.soaplore(data, oc)	Taking dataframe derived from the 'soapsheetin(path)' and return aggregate data for each variable in the spreadsheet, including summary statistics. Also provide mean and frequency for each binary variable by the defined outcome (oc).
soap.excludif(data, condition)	Excluding row that meet a specific condition return a dataframe.
soap.soap_TU(data, oc, var)	Performing independent student-T test (T) and Ranksum (Mann-Whitney U) test for interested continuous variable (var on defined outcome (oc).

soap.soap_multi_T(data, oc)	Performing T and U test for multiple variables at a time, on a fixed binomial outcome (oc). Receiving 3 variables: data (a dataframe from sp.soapsheetin(path)), oc, and factors (a list from sp.soapvarin( ))
soap_T_for_multibinary(data, var)	Taking a list of binomial variables to be tested against a defined continuous ordinal variable, using the next command.
soap.soap_x_tab(data, var_a, var_b)	Cross tabulating 2 categorial variables (var_a and var_b) in the dataframe (data). Also show a tabular table and a percentage bar graph of var_b on var_a.
soap.soap_x_across(data, outcome)	Cross tabulating between a defined outcome and other categorial variables across the board.
soap.single_kmc(data, status, interval)	Draw a single Kaplan-Meier curve from follow-up interval (interval) and status within a dataframe (data).
soap.compare_kmc(data, factor, status, interval)	Survival comparison between groups in a variable (factor).

1. Installing the SURPY module and importing the soap package.

As the package depends on various python modules: Pandas, Numpy, Scipy and Matplotlib, those modules should be installed before installing SURPY. The module can be installed through pip protocol on CLI; as followed. To check if pip has been installed in your environment, use the command: python3 -m pip —version. To install pip, please consult https://pip.pypa.io/en/stable/installing/. To import soap package from SURPY, use the following code on Python.

```
>>> import SURPY
>>> from SURPY import soap as sp
```

2. Importing an EXCEL file (filename.xlsx) into the system.

The package uses pd.dataframe with the openpyxl to import spreadsheet in .xlsx format. To import a file, set path to the spreadsheet first and use the following code with a designated name of the dataframe, ‘data’ in this case.

Download demo data from: https://github.com/sasurasa/Surgical-Outcome-Analysis-on-Python/blob/SURPY/sodata/intuss_demo.xlsx

```
>>> path = ‘Desktop/intuss_demo.xlsx'
>>> data = sp.soapsheetin(path)
```

3. Explore and aggregate for summary data of variables in the spreadsheet. In this example, the variable ‘fail’ is committed as an outcome. The variable ‘fail’ is a binary or, in the other word, has only 2 values, usually 0 and 1.

```
>>> sp.soaplore(data, ‘fail’)
```

Expected outputs include:

- Data dimension and list of all variables

- Types and numbers of non-null count (row excluding missing data) for each variable.

- Shapiro-Wilk test (test for normal distribution) results for each variable, separating between float and integer values.

- General summary statistics for each variable (count, mean, standard deviation, minimum value, percentile 25th, percentile 50th (median), percentile 75th, maximum value.

- Number (count) of each binary variable by the designated outcome.

- Mean value for numeric variable, grouped by the defined outcome.

4. Exclude rows with specific condition

```
>>> data_1 = sp.soap_excludif(data, data.sex == 0)
```

5. Performing mean and median comparisons using independent T test and Mann-Whitney U test, respectively. Receiving 3 inputs, data as the dataframe, ‘oc’ as the defined outcome (binary) and ‘var’ as the variable to be tested against ‘oc’.

```
>>> sp.soap_TU(data, 'fail', 'age')
```

Expected outputs include a table showing mean with standard deviation, median with interquartile range, p-values from T and U tests and a boxplot comparing the 2 groups.

6. On a binary outcome, performing T-test and U-test for multiple variables all in one row. First excerpt a list of quantitative parameters to be tested against the outcome from the list of parameters in the spreadsheet.

```
>>> sp.soaptumulti(data, 'fail')
```

The program will ask to put in continuous variables to be tested against 'fail'.

```
Enter a list of factors to be analysed,separated by a space: age weight
sod plt
```

Expected output will be an aggregate table for each variable, grouped by the defined outcome, followed by p-values from T and U tests.

7. In the opposite way, one continuous parameter can be tested with multiple binary variables in one click. Beginning by taking the binary variables into a list using the following command:

```
>>> sp.soaptubivar(data, bivar, 'sod')
```

The program will ask to put in binary variables to be tested against 'sod'.

```
Enter a list of binary variables to be used for comparisons of
var,separated by a space: vomit recb abdd
```

Expected output will be p-values from T test between `'sod'` and binomial variables in the list `'bivar'` including a table comparing mean values of `'sod'`, grouped by each element in the list `'bivar',` put in the new heading `sod_vomit`, `sod_recb`, etc.

8. On soap package of the SURPY module, cross-tabulating between 2 categorial variables is performed by the following example code:

```
>>> sp.soap_x_tab(data, 'recb', 'fail')
```

Expected output include a tabular table, chi-square value, chi-square p-value and a bar plot comparing between percentage of var_b, grouped by var_a or, in this case, `%fail` comparing between the patients without and with **`rectb`** (rectal bleeding).

9. To perform tabulating between a defined outcome and across the whole spreadsheet, use the following code:

```
>>> sp.soap_x_across(data,'fail')
```

Following this code, the program will perform chi-square test between the outcome, 'fail' in this case, and all variables with number of unique value not more than 5. Expected output is a table of chi-square p-value.

10. To draw a Kaplan-Meier curve showing the survival probability as a function of time in a single line without comparison, use the following code:

Try this data set: https://github.com/sasurasa/Surgical-Outcome-Analysis-on-Python/blob/SURPY/sodata/breast_demo.xlsx

```
>>> path = 'Desktop/breast_demo.xlsx'
>>> data_1 = sp.soapsheetin(path)
>>> status = 'osstatus' #set status variable and interval variable
>>> interval = 'osint'

>>> sp.single_kmc(data_1, status, interval)
```

'data' is the dataframe, 'status' is an array of binary value showing censor status (e.g. dead, recurrence), and 'interval' is an array of duration from the beginning to the point of status (usually in the unit of day).

11. To compare survival functions between 2 or more groups with Kaplan-Meier curve and Log-rank test, use the following code:

```
>>> factor = 'n' #set the factor to be analysed with survival function using Logrank
>>> sp.compare_kmc(data, factor, status, interval)
```

The factor is the variable to be tested against survival functions. If there are 2 groups in the 'factor' the program will draw a Kaplan-Meier graph and provide Log-rank statistics with p-value. If there are more than 2 groups, the program will draw the survival graph only.

THIP.py

THIP is a package used for basic data analysis from Thai Health Information Portal (THIP) which is a large database of in-hospital patients in Thailand.
(https://thip.nbt.or.th)

```
% pip install SURPY==1.1.13
>>> import SURPY
>>> from SURPY import thip as tp
```

The package contain commands for data import (from .csv)

```
>>> path = 'yourpath'
>>> thip = tp.thipimport(path)
```

```
#Age selection
>>> thip = tp.set_age(thip, lower, upper)
#Format correction
>>> thip = tp.clean_datetime(thip)
```

```
#Select based on ICD-10
>>> q_disease = ['Q123', 'Q234']
>>> anom = tp.select_Q(source, condition, dupdel = True)
##Set dupdel = False if do not want to remove personal ID duplication
```

```
#Find sex ratio
>>> tp.sex_ratio(anom)
```

#Count the matched records by year and estimate crude incidence/ 10,000 livebirths

```
>>> tp.count_by_year(anom)
>>> tp.count_by_month(anom)
```

#Count the matched records by health region (12+1 Thai-Health region)

```
>>> tp.count_by_region(anom)
```

```
#Export selected anom to .csv

>>> tp.export(anom, filename)


#Find  mortality  and  mortality  rate  (count  records  with
'death_date')

>>> tp.mortal(anom)



#Count associate Down syndrome/Congenital heart disease.

>>> tp.count_down(anom)

>>> tp.count_heart(anom)

>>> tp.count_assoc(anom, assoc)


#Explore the procedure code and return procedure_dict.

>>> dict = tp.procdict(anom)
```