Python-Callable SUDAAN User Guide

PC and Windows Installation
64-bit Operating Systems

Release 11.0.4 for
Python-Callable

Table of Contents

Getting Started

Python-Callable SUDAAN release 11.0.4 is based on SUDAAN version 11.0.4. This guide covers how to submit SUDAAN procedures within a Python environment. Details on how to write SUDAAN procedures can be found in the SUDAAN Language Manual

Requirements

The following are required to access SUDAAN within a Python environment:

1. Windows 10, 64-bit operating system[1]
2. Python version 3.11 or higher
3. Installation of Python-Callable SUDAAN 11.0.4
4. Installation of pandas 1.3.5 or higher[2]

Instructions on installation of Python-Callable SUDAAN 11.0.4 can be found in the SUDAAN Installation Guide included in your download or on the SUDAAN website (https://sudaanorder.rti.org).

Using SUDAAN Python for the First Time

After installing Python-Callable SUDAAN, it can be accessed within python by importing it from the sudaan library:

```
>>> from sudaan import proc
```

Then call one of the two SUDAAN functions:

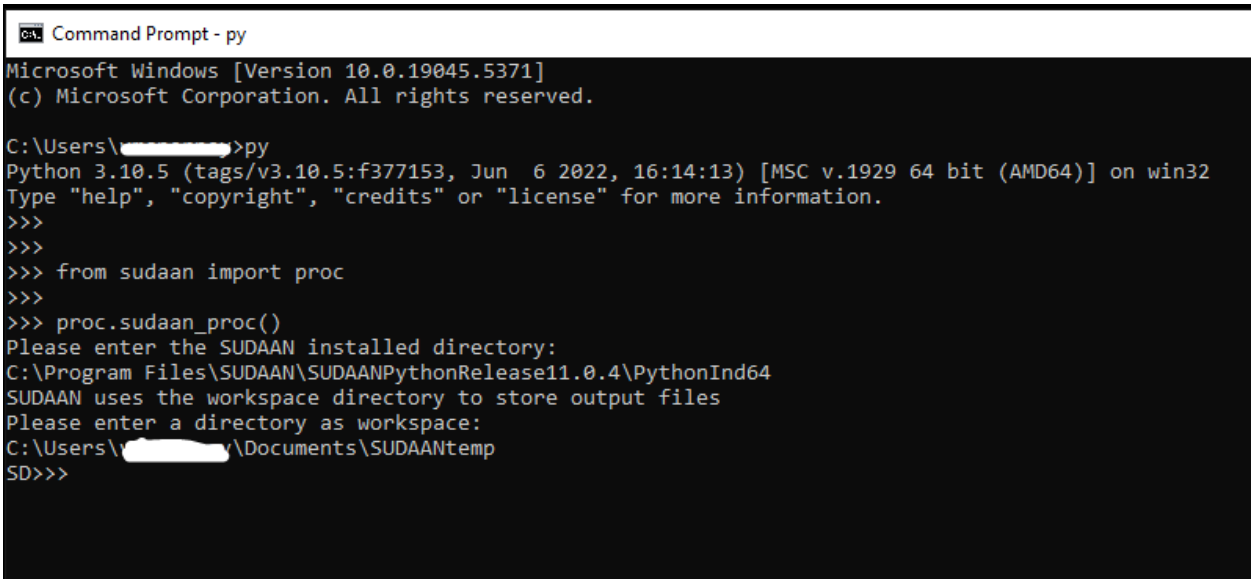
```
proc.sudaan_proc(), or
proc.sudaan_batch()
```

The first time you call one of these functions after installation, you will be asked for the following:

1. The SUDAAN installation directory. This is the location you selected in Step 3 of the installation process. The default directory is **C:\Program Files\SUDAAN\SUDAANPythonRelease11.0.4\PythonInd64**.
2. A workspace directory. This is where SUDAAN will store temporary files and output during processing. If you will be working with large files, we recommend that you set this to a location on your local machine. This should be a location that other users of Python-callable SUDAAN do not use as a workspace directory.

[1] Python-Callable SUDAAN was tested under a Windows 10 operating system. We expect that it will also work under Windows 11, but we have not tested that capability, yet.
[2] Installation of pandas is only required when reading from or writing to pandas dataframes.

Figure 1. Setting SUDAAN Python Directories



You will only need to set these directories once, the first time you call a SUDAAN function. If you need to change the workspace directory at a future point, see Changing the Workspace Directory.

Submitting SUDAAN Code via Python

There are two ways to submit SUDAAN code within the Python environment, interactively, or through a batch submission. These methods are invoked by calling the functions proc.sudaan_proc() or proc.sudaan_batch(), respectively.

When submitting SUDAAN code via one of these functions, the syntax of the SUDAAN code is the same as that of Standalone, Command Prompt, and SAS-Callable SUDAAN.

Run Python-Callable SUDAAN in Interactive Mode

Within a Python terminal, you can run SUDAAN interactively by calling the function proc.sudaan_proc(). This function takes the following arguments:

Positional Parameters (optional):

'screen = off' – By default, SUDAAN will print output and run information to the terminal. Include this parameter, including the surrounding quotation marks, to stop SUDAAN from sending output to the terminal. Output will only be sent to Output.txt in the workspace directory, or to the file specified in the 'output =' parameter.

'output = <*path*>' – By default, SUDAAN sends printed output to a file named Output.txt in the workspace directory. To send this to a different file, include this parameter, replacing <path> with the path to your desired output location. For example, 'output=W:/Imputation/Programs/Results.txt'.

Notes

1. If included, the positional parameters 'screen =' and 'output =' can be in any order, but they must come before any keyword parameters.
2. Make sure that you include the name of the positional parameter within the quote (e.g. 'screen = off', **not** screen = 'off').
3. Only forward slashes (/) are permitted in the path for the output file. Backslash (\) and double backslash (\\) are not permitted and will cause an error.

Keyword Parameters (optional):

<*input_data_name*> = <*pandas_df_name*> - To pass a pandas dataframe for use by SUDAAN, use this parameter. The name of the parameter is the name of the input dataset listed in your SUDAAN code. For example, if your SUDAAN code is

```
PROC RECORDS DATA = educ FILETYPE = PANDAS contents countrec;
PRINT idnum sex age yrs_in_school / maxrec = 25;
```

and your data are saved in a pandas dataframe named df, then your function call would be

```
proc.sudaan_proc( educ = df)
```

You may include as many keyword parameters as necessary to list all the input pandas

dataframes you would like to pass to SUDAAN. For example, if your SUDAAN code references data sets educ1, educ2, and educ3, and these are saved in pandas dataframes named df1, df2, and df3, respectively, your function call would be

```
proc.sudaan_proc(educ1 = df1, educ2 = df2, educ3 = df3)
```

Note

Keyword parameters listing input data names are only required if you are passing pandas dataframes to SUDAAN. If your input data are in any other format (e.g. CSV, SUDXPORT), omit them.

Returned Results

A log of submitted SUDAAN statements and output from any PRINT statements in your SUDAAN code will be printed to the terminal and to Output.txt in the workspace directory. If you included the 'screen = off' parameter in your call to the proc.sudaan_proc() function, the log of submitted SUDAAN statements and output will not be printed to the terminal, but will be printed to the Output.txt file in the workspace directory. If you included the 'output =' parameter in your call to the proc.sudaan_proc() function, output will be printed to the output file specified instead of to Output.txt.

To access output data created by SUDAAN OUTPUT statements with FILETYPE = PANDAS outside of SUDAAN, assign the proc.sudaan_proc() function to a variable, e.g. result = proc.sudaan_proc(). When you exit the SUDAAN interface, the variable *result* will contain a list of pandas dataframes. See Example 1. Interactive Submission with pandas Dataframes as Input and Output.

Output data created by SUDAAN OUTPUT statements with any other filetype (e.g. CSV, SUDXPORT) will be created according to the path defined in the FILENAME = parameter of the SUDAAN OUTPUT statement.

Submitting SUDAAN Code Interactively

After submitting the call to proc.sudaan_proc(), your SUDAAN session will begin, and you may type or paste in your SUDAAN statements into the terminal. In order to submit your statements for SUDAAN to process, type *run* and press ENTER. SUDAAN will then process your statements, and you will be able to type or paste additional SUDAAN statements to the terminal. When you have finished with SUDAAN, type *exit* and press ENTER.

Using Data Created by SUDAAN in Subsequent SUDAAN Procedures

If your SUDAAN procedure will output pandas dataframes or CSV files that will be used as input for subsequent SUDAAN procedures, you must submit the first SUDAAN statements by typing *run* and pressing ENTER. Then you may submit more SUDAAN procedures without calling the proc.sudaan_proc() function again. See Example 1. Interactive Submission with pandas Dataframes as Input and Output. This is not necessary if your output files are SUDXPORT, SASXPORT, or ASCII files.

Example 1. Interactive Submission with pandas Dataframes as Input and Output

This example reads in a SAS data set, simulates some missing values, and then uses SUDAAN's PROC IMPUTE to perform weighted sequential hot-deck imputation. The submitted code is below. Note that text following '>>>' is Python code, and text blocs between 'SD>>>' and 'quit' are SUDAAN statements.

```
>>>import pandas as pd
>>>from sudaan import proc
>>>import numpy as np

>>>np.random.seed(42)

>>>artif1 = pd.read_sas("artif1.sas7bdat")

>>># Add an ID variable
>>>artif1["ID"]=range(1,len(artif1)+1)

>>># Check that weights are positive and non-missing
>>>artif1[["weight"]].describe()
>>>artif1["weight2"]=artif1["weight"].where(artif1["weight"]>0,0.01)
>>>artif1[["weight2"]].describe()

>>># Simulate missing values
>>>artif1["randUni"]=np.random.rand(250)
>>>artif1["var1_x"]=artif1["var1"].where(artif1["randUni"]<=.9)
>>>artif1[["var1_x"]].value_counts(dropna=False)
>>>artif1[["var1","var1_x"]].value_counts(dropna=False)

>>># Sort the dataframe by IMPBY variables
>>>artif1=artif1.sort_values(by=["y","var2"])

>>># Call SUDAAN Interactive
>>>impute = proc.sudaan_proc(artif1=artif1)
SD>>>
/*These are SUDAAN statements*/
proc impute data=artif1 filetype=pandas method=wshd;
weight weight2;
impid ID;
impby y var2;
impvar var1_x;
output impid imputeval donorid / filename = artif2 filetype=pandas REPLACE;
run

/*Output will be printed to the terminal. Figure 2.1 Submitting SUDAAN Code in an
Interactive Session shows a portion of this output.*/
```

```
SD>>>
proc crosstab data=artif2 filetype=pandas;
nest _one_;
weight _one_;
class imputeval1;
tables imputeval1;
run

/*Output will be printed to the terminal. Figure 2.2 Submitting SUDAAN Code in an
Interactive Session: Output from PROC Crosstab shows a portion of this output.*/

exit

# Now we have exited our SUDAAN session

# Merge imputed values with original data
>>>artif2=impute['artif2'].sort_values(by="ID")
>>>artif3=artif2.set_index('ID').join(artif1.set_index('ID'))
>>>artif3[["var1_x","IMPUTEVAL1"]].value_counts(dropna=False)
```

Figure 2.1 Submitting SUDAAN Code in an Interactive Session: Output from PROC impute

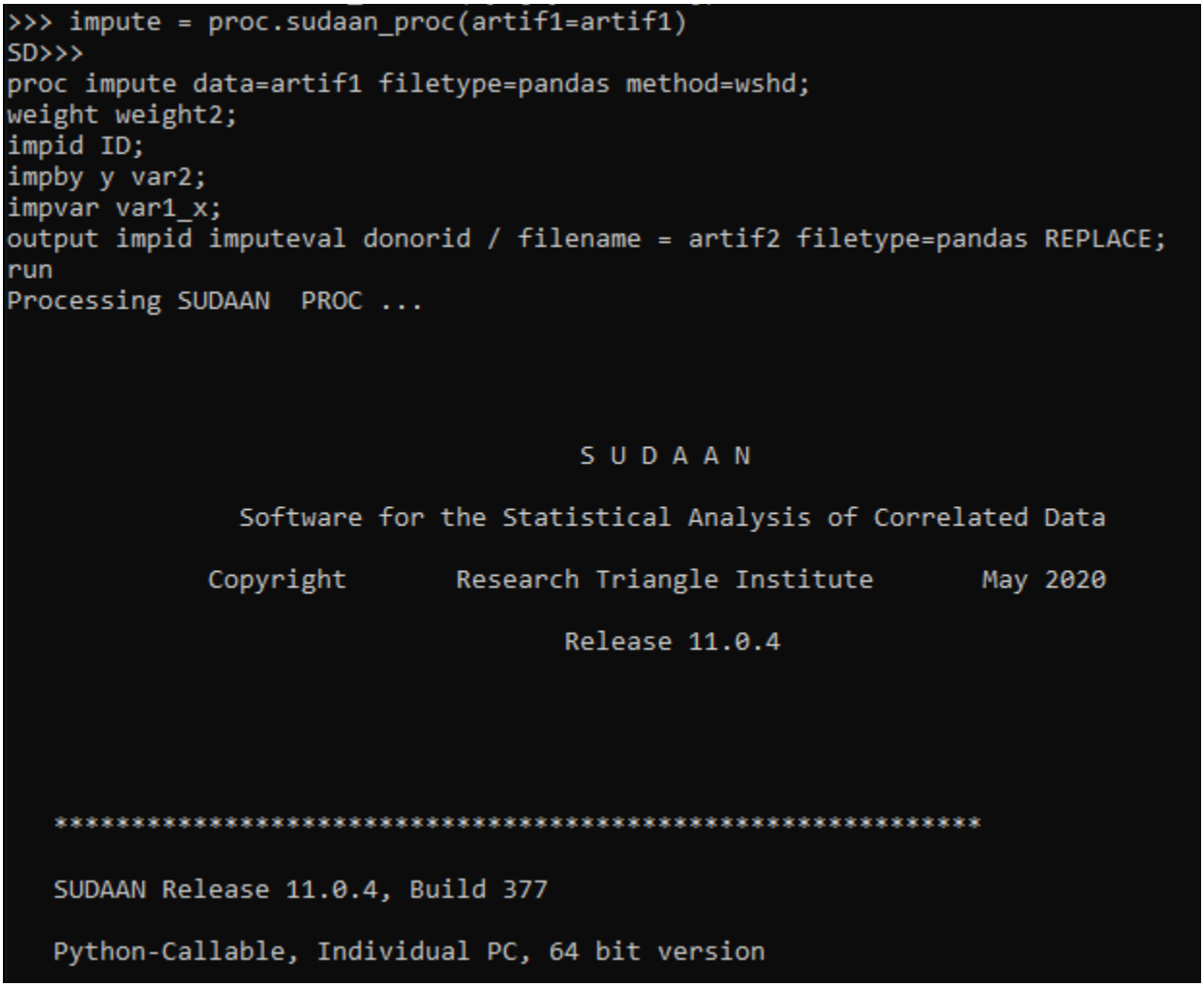


Figure 2.2 Submitting SUDAAN Code in an Interactive Session: Output from PROC Crosstab

```
Frequencies and Values for CLASS Variables

by: IMPUTEVAL1.


---------------------------------

IMPUTEVAL1      Frequency    Value

---------------------------------

Ordered

  Position:

  1                   82        1

Ordered

  Position:

  2                   85        2

Ordered

  Position:

  3                   83        3

---------------------------------
```

Notes

1. If you save your Python code and your SUDAAN code in the same file, you will not be able to submit that entire file for Python to run. The Python compiler will not recognize the SUDAAN code as valid and will produce an error. To run code in batch, go to the section Run Python-Callable SUDAAN in Batch Mode.
2. If you are developing your code in an integrated development environment (IDE) like PyCharm or VSCode, you may not be able to use the "run interactively" functionality of the IDE to access SUDAAN in interactive mode. Instead, open a terminal and paste your Python and SUDAAN code there.
3. Syntax errors in submitted SUDAAN statements will produce an error "SUDAAN Error (4)," which will be printed to the terminal. To see details about the error, check Output.txt in the workspace directory, or the output file specified using the 'output =' parameter.
4. All pandas dataframes passed to SUDAAN will be converted to CSV files and saved in the workspace directory.

Run Python-Callable SUDAAN in Batch Mode

Within a Python program, you can batch run SUDAAN statements by calling the function proc.sudaan_batch(). This function takes the same arguments as proc.sudaan_proc(), with the addition of the following required positional parameter:

Positional Parameter:

'<*path/mycode.sud*>' – (Required) Include the path to the file containing your SUDAAN statements. This file must have an extension of **.sud**.

Using Data Created by SUDAAN in Subsequent SUDAAN Procedures

If your SUDAAN procedure will output pandas dataframes or CSV files that will be used as input for subsequent SUDAAN procedures, you must include multiple calls to the function proc.sudaan_batch(). The first call should include the path to the file containing SUDAAN statements creating the pandas dataframes or CSV files, and the next calls would include the path to the file containing the subsequent SUDAAN procedures. See Example 2. Batch Submission with pandas Dataframes as Input and Output. This is not necessary if your output files are SUDXPORT, SASXPORT, or ASCII files.

Example 2. Batch Submission with pandas Dataframes as Input and Output

This example is the same as Example 1. Interactive Submission with pandas Dataframes as Input and Output, but uses the batch submission function proc.sudaan_batch().

Python code:

```
import pandas as pd
from sudaan import proc
import numpy as np

np.random.seed(42)

artif1 = pd.read_sas("artif1.sas7bdat")

# Add an ID variable
artif1["ID"]=range(1,len(artif1)+1)

# Check that weights are positive and non-missing
artif1[["weight"]].describe()
artif1["weight2"]=artif1["weight"].where(artif1["weight"]>0,0.01)
artif1[["weight2"]].describe()

# Simulate missing values
artif1["randUni"]=np.random.rand(250)
artif1["var1_x"]=artif1["var1"].where(artif1["randUni"]<=.9)
artif1[["var1_x"]].value_counts(dropna=False)
artif1[["var1","var1_x"]].value_counts(dropna=False)

# Sort the dataframe by IMPBY variables
artif1=artif1.sort_values(by=["y","var2"])

# Complete the imputation using SUDAAN batch
impute = proc.sudaan_batch('Impute.sud',  'output = ImputeLog.txt', 'screen = off',
artif1=artif1)

# Get artif2 from the result of our batch submission
artif2=impute['artif2'].sort_values(by="ID")

# Run PROC CROSSTAB – There are no output files
proc.sudaan_batch('Crosstab.sud', 'output = CrosstabLog.txt', 'screen = off', artif2=artif2)

# Merge imputed values with original data
artif2=impute['artif2'].sort_values(by="ID")
artif3=artif2.set_index('ID').join(artif1.set_index('ID'))
artif3[["var1_x","IMPUTEVAL1"]].value_counts(dropna=False)
```

Contents of Impute.sud:

```
proc impute data=artif1 filetype=pandas method=wshd;
weight weight2;
impid ID;
impby y var2;
impvar var1_x;
output impid imputeval donorid / filename = artif2 filetype=pandas REPLACE;
```

Contents of Crosstab.sud:

```
proc crosstab data=artif2 filetype=pandas;
nest _one_;
weight _one_;
class imputeval1;
table imputeval1;
```

File Input and Output

Python-Callable SUDAAN supports all input and output file types supported by Standalone SUDAAN, with the exception of SPSS. Python-Callable SUDAAN does not write to SPSS files. However, Python-Callable SUDAAN does read and write to CSV files and pandas data frames. See Table 1, below, for details.

Table 1. Input and Output File Types Supported by Python-Callable SUDAAN

Filetype	Description
CSV	Python-Callable SUDAAN reads and writes to comma separated text files (.csv). Use FILETYPE = CSV.
pandas	Python-Callable SUDAAN reads and writes to pandas data frames. Use FILETYPE = PANDAS.
SUDAAN	Python-Callable SUDAAN reads and writes to SUDAAN files. SUDAAN files are binary, so there is no loss of precision when floating point values are saved and read back later. Use FILETYPE = SUDAAN.
SUDXPORT	Python-Callable SUDAAN reads and writes to files in a format that can be used by any version of SUDAAN on any platform. Though the method is slightly slower than reading/writing SUDAAN file types, it permits transport of SUDAAN data from one platform to another. Use FILETYPE = SUDXPORT.
ASCII	Python-Callable SUDAAN reads and writes to text data files. Documentation files accompany these text files so that SUDAAN (or other programs) can read and interpret the data later. Use FILETYPE = ASCII.
SASXPORT	Python-Callable SUDAAN reads and writes to SAS's XPORT format, which can be used by SAS. Use FILETYPE = SASXPORT.

Recommendations and Limitations

Sorting CSV and pandas Files

SUDAAN should not be used to sort CSV files or pandas dataframes. This includes using PROC RECORDS with the SORTBY statement, as well as using the NOSORT option available in most SUDAAN procedures. Sorting CSV files and pandas dataframes in SUDAAN is memory intensive and may cause memory overflow errors. Instead, you should sort your data in Python before submitting code to SUDAAN. SUDAAN can still be used to sort SUDXPORT and SASXPORT files.

Multi-Line Comments in Interactive Mode

Comments in SUDAAN are formatted as /* comment */. This format can be used for one-line comments and multi-line comments. For example:

```
/*This is a one-line comment in SUDAAN*/
/*
This is a multi-
line comment in SUDAAN
*/
```

When running SUDAAN in batch mode, both comment styles are acceptable. However, in interactive mode, only one-line comments are accepted. For example:

```
/*This comment is acceptable in both batch and interactive modes*/
/*
  This comment is only acceptable in batch mode.
  It will cause an error in interactive mode.
*/
```

RTF Output from PROC RECORDS

In this version of Python-Callable SUDAAN, PROC RECORDS cannot output to an RTF file when the input file is in CSV or pandas format. If you need to output from PROC RECORDS to an RTF file, first use PROC RECORDS to convert your CSV or pandas data to SUDXPORT, then use that SUDXPORT file as the input to PROC RECORDS to produce the RTF file.

Using SUDAAN Output Data as Input

If your SUDAAN procedure will output pandas dataframes or CSV files that will be used as input for subsequent SUDAAN procedures, the first SUDAAN procedure must be processed before the subsequent SUDAAN procedures are submitted.

If you are running your SUDAAN code interactively, this means you must submit the first SUDAAN statements by typing *run* and pressing ENTER. Then you may submit more SUDAAN procedures without calling the proc.sudaan_proc() function again. See Example 1. Interactive Submission with pandas Dataframes as Input and Output.

If you are running your SUDAAN code in batch mode, you must separate your SUDAAN statements into multiple .SUD files and include multiple calls to the function proc.sudaan_batch() in your Python code. The first call should include the path to the .SUD file containing SUDAAN statements creating the pandas dataframes or CSV files, and the next calls would include the path to the .SUD file containing the subsequent SUDAAN procedures. See Example 2. Batch Submission with pandas Dataframes as Input and Output.

Changing the Workspace Directory

To change the workspace directory, you will need to navigate to the sudaan folder within your Python installation. The default location is **C:\Users\<username>\AppData\Local\Programs\Python\Python###\Lib\site-packages\sudaan**, where ### is your python version number (e.g. 311). Within that folder is a file named config.ini. Saved in that file is the variable **sudaan_workdir**. Change the value of that variable to the new working directory location.

Figure 2. SUDAAN Configuration File