R-Callable SUDAAN User Guide

PC and Windows Installation
64-bit Operating Systems

Release 11.0.4 for
R-Callable

Table of Contents

Getting Started

R-Callable SUDAAN release 11.0.4 is based on SUDAAN version 11.0.4. This guide covers how to submit SUDAAN procedures within an R 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 an R environment:

1. Windows 10, 64-bit operating system[1]
2. R version 3.5 or higher
3. Installation of R-Callable SUDAAN 11.0.4

Instructions on installation of R-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).

Submitting SUDAAN Code via R

After installing the SUDAANforR package, SUDAAN is accessed via the callSUDAAN() function. SUDAAN commands can be saved as a character variable within the R session, or in a separate batch file with the extension .sud.

When submitting SUDAAN code via callSUDAAN(), the syntax of the SUDAAN code is the same as that of Standalone, Command Prompt, and SAS-Callable SUDAAN.

The callSUDAAN() Function

The callSUDAAN() function takes the following arguments:

Required Parameters:

call = A string holding the commands to submit to SUDAAN. One of call or batchfile must be included.

Batchfile = A string holding a path to a .sud file holding commands to submit to SUDAAN. One of call or batchfile must be included.

Optional Parameters:

Rdata = A data frame or named list of data frames to pass to SUDAAN. If passing a single data frame, the name of the data frame must match the name of the input dataset listed in your SUDAAN code. For example, if your SUDAAN code is

[1] R-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.

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

then your data must be saved in an R dataframe named educ.

To pass multiple data frames to SUDAAN you must first save them in a named list. The names of the list objects must match the names of the input datasets listed in your SUDAAN code. For example, if your SUDAAN code references data sets educ1, educ2, and educ3, and these are saved in R data frames named df1, df2, and df3, respectively, your list would be defined as

```
indata1 <- list(educ1 = df1, educ2 = df2, educ3 = df3)
callSUDAAN(batchfile = "path/to/my/file.sud", Rdata = indata1)
```

outfile = A string holding the name and location of the SUDAAN output log. When using the *call* parameter, the default location is ~/SUDAANTemp/Job_<date_time>/Output.log. When using the *batchfile* parameter, the default location is the same as your batch file name and location with the extension *.log*.

workdir = A string holding the location of the SUDAAN working directory. The default is ~/SUDAANTemp/Job_<date_time>

delworkdir = A logical value defining whether to delete the working directory after the job completes. When using the call parameter the default is FALSE, when using the batchfile parameter, the default is TRUE.

Note

The Rdata parameter is only required if you are passing R dataframes to SUDAAN. If your input data are in any other format (e.g. CSV, SUDXPORT), omit it.

Returned Results

The callSUDAAN() function returns a SUDAANResult object. This is a named list with the following components:

call – The code sent to SUDAAN
data – A list of tibbles returned from SUDAAN (if any)
wrk_dir – Path to the SUDAAN working directory
log_file – Path to the log produced by SUDAAN

To access output data created by SUDAAN OUTPUT statements with FILETYPE = R outside of SUDAAN, assign the callSUDAAN() function to a variable, e.g. result = callSUDAAN(). When the SUDAAN run is complete, the output data can be accessed in result$data. See Example 1. Using the *call* Parameter.

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.

Using Data Created by SUDAAN in Subsequent SUDAAN Procedures

If your SUDAAN procedure will output R dataframes or CSV files that will be used as input for subsequent SUDAAN procedures, you must submit the first SUDAAN statements in one call to callSUDAAN(). Then submit the subsequent SUDAAN procedures via a second call to callSUDAAN(). See Example 1. Using the *call* Parameter. This is not necessary if your output files are SUDXPORT, SASXPORT, or ASCII files.

Example 1. Using the *call* Parameter

This example uses the R dataframe included with the SUDAANforR package. In the example, we simulate some missing values, and then use SUDAAN's PROC IMPUTE to perform weighted sequential hot-deck imputation. The submitted code is below.

```
library(SUDAANforR)
library(dplyr)

#Simulate missing values
set.seed(123)
impute_ex$randUni <- runif(nrow(impute_ex), min = 0, max = 1)
sum(is.na(impute_ex$var1))
impute_ex$var1_x <- ifelse(impute_ex$randUni <= .9, impute_ex$var1, NA)
sum(is.na(impute_ex$var1_x))

#Sort the dataframe by IMPBY variables
impute_ex1 <- impute_ex %>% arrange(y, var2)
impute_ex1 <- impute_ex1 %>%
        mutate(
                num_var2 = case_when(
                                        var1 == "A" ~ 1,
                                        TRUE ~ 2)
                )
impute_ex1 <- impute_ex1 %>% select(-var2)
names(impute_ex1)

call1 <- '
proc impute data=impute_ex1 filetype=R method=wshd;
weight weight;
impid ID;
impby y num_var2;
impvar var1_x;
output impid imputeval donorid /filename = impute_ex1_out filetype=R REPLACE;
'

result1 <- callSUDAAN(call = call1, Rdata = impute_ex1)
```

Figure 2.1 Printed SUDAANResult

```
result1
#> SUDAAN Code Submitted:
#>
#> proc impute data=impute_ex1 filetype=R method=wshd;
#> weight weight;
#> impid ID;
#> impby y num_var2;
#> impvar var1_x;
#> output impid imputeval donorid /filename = impute_ex1_out filetype=R REPLACE;
#>
#>
#> Returned Data:
#> impute_ex1_out: 500 rows, 4 columns: PROCNUM ID IMPUTEVAL1 DONORID
#>
#>
#> SUDAAN run status:
#>
#> Working Directory: C:\Users\vmcnerney\Documents\SUDAANTemp\Job_20260706111003348
#>
#> SUDAAN Log: C:\Users\vmcnerney\Documents\SUDAANTemp\Job_20260706111003348\Output.log
```

Notes

1. 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 working directory, or the output file specified using the *outfile* parameter.
2. All R dataframes passed to SUDAAN will be converted to CSV files and saved in the workspace directory.

Example 2. Using the *batchfile* Parameter

This example is similar to Example 1. Using the *call* Parameter, but uses the *batchfile* parameter to send multiple procedure calls to SUDAAN.

R code:

```
library(SUDAANforR)
library(dplyr)

#Simulate missing values
set.seed(123)
impute_ex$randUni <- runif(nrow(impute_ex), min = 0, max = 1)
sum(is.na(impute_ex$var1))
impute_ex$var1_x <- ifelse(impute_ex$randUni <= .9, impute_ex$var1, NA)
sum(is.na(impute_ex$var1_x))

#Sort the dataframe by IMPBY variables
impute_ex1 <- impute_ex %>% arrange(y, var2)
impute_ex1 <- impute_ex1 %>%
        mutate(
                num_var2 = case_when(
                                        var1 == "A" ~ 1,
                                        TRUE ~ 2)
                )
impute_ex1 <- impute_ex1 %>% select(-var2)

#Create another data frame
impute_ex2 <- impute_ex

#simulate missing values
set.seed(456)
impute_ex2$randUni <- runif(nrow(impute_ex2), min = 0, max = 1)
sum(is.na(impute_ex2$var1))
impute_ex2$var1_x <- ifelse(impute_ex2$randUni <= .9, impute_ex2$var1, NA)
sum(is.na(impute_ex2$var1_x))

#prep for SUDAAN
impute_ex2_sorted <- impute_ex2 %>% arrange(y, var2)
impute_ex2_sorted <- impute_ex2_sorted %>%
        mutate(
                num_var2 = case_when(
                                        var1 == "A" ~ 1,
                                        TRUE ~ 2)
                )
```

```
impute_ex2_sorted <- impute_ex2_sorted %>% select(-var2)

indata <- list(impute_ex1 = impute_ex1, impute_ex2 = impute_ex2_sorted)

result2 <- callSUDAAN(batchfile = "test_impute.sud", Rdata = indata, outfile =
"C:/temp/test_impute.log")

#access output dataframe
result2$data$impute_ex2_out
```

Figure 2.2 First few observations of impute_ex2_out

```
result2$data$impute_ex2_out
#> # A tibble: 500 × 4
#>    PROCNUM    ID IMPUTEVAL1 DONORID
#>      <dbl> <dbl>      <dbl>   <dbl>
#>  1     16     9       2.83      NA
#>  2     16    15       2.98      NA
#>  3     16    19       3.08      NA
#>  4     16    26       3.28      NA
#>  5     16    30       2.99      NA
#>  6     16    56       2.90      NA
#>  7     16    74       3.19      NA
#>  8     16    85       2.50      NA
#>  9     16    87       2.49      NA
#> 10     16    99       2.77      NA
#> # ℹ 490 more rows
```

Contents of test_impute.sud:

```
/*test proc impute*/
proc impute data=impute_ex1 filetype=R method=wshd;
weight weight;
impid ID;
impby y num_var2;
impvar var1_x;
output impid imputeval donorid /filename = impute_ex1_out filetype=R REPLACE;

proc impute data=impute_ex2 filetype=R method=wshd;
weight weight;
impid ID;
impby y num_var2;
impvar var1_x;
output impid imputeval donorid /filename = impute_ex2_out filetype=R REPLACE;
```

Notes

1. The input data are saved in the named list *indata*. The names of the list items must match

the names of the input data in the SUDAAN code (impute_ex1 and impute_ex2), but not necessarily the names of the R data frames (impute_ex1 and impute_ex2_sorted).

2. By default, SUDAAN will write the output log to the location of the batch file. If you do not have write access to this location, you must supply a different location for *outfile*.

File Input and Output

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

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

Filetype	Description
CSV	R-Callable SUDAAN reads and writes to comma separated text files (.csv). Use FILETYPE = CSV.
R	R-Callable SUDAAN reads and writes to R data frames. Use FILETYPE = R.
SUDAAN	R-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	R-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	R-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	R-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 R Files

SUDAAN should not be used to sort CSV files or R data frames. 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 R data frames in SUDAAN is memory intensive and may cause memory overflow errors. Instead, you should sort your data in R before submitting code to SUDAAN. SUDAAN can still be used to sort SUDXPORT and SASXPORT files.

RTF Output from PROC RECORDS

In this version of R-Callable SUDAAN, PROC RECORDS cannot output to an RTF file when the input file is in CSV or R format. If you need to output from PROC RECORDS to an RTF file, first use PROC RECORDS to convert your CSV or R 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 R 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.

Thie means you must separate your SUDAAN statements into multiple call statements or .SUD files and include multiple calls to the function callSUDAAN() in your R code. The first call should include the SUDAAN statements creating the R data frames or CSV files, and the next calls would include the subsequent SUDAAN procedures.