Paper 183-2019

Python and R made easy for the SAS® Programmer

Janet J. Li, Pfizer Inc.; Varaprasad Ilapogu, Ephicacy Consulting Group

ABSTRACT

Many of the day-to-day tasks and responsibilities of the statistical programmer of a pharmaceutical research and development group or contract research organization (CRO) involved include importing/exporting data, deriving variables and creating analysis data sets, and creating clinical study report (CSR) materials such as tables, listings, and figures (TLFs). These outputs are used for submission to regulatory agencies and are generally programmed in SAS®. There is a growing movement and acceptance towards using Python and R in the Clinical SAS® programming world. In our paper, we explore tips and tricks on how to use Python and R in conjunction with SAS® to more effectively and efficiently deliver statistical programming outputs. We intend to provide examples of common SAS® procedures and syntax that are used in the creation of analysis datasets and TLFs and translate them to Python and R to help the beginner Python and/or R programmer familiarize themselves with the alternative way of programming these statistical outputs.

INTRODUCTION

Many of the day-to-day tasks and responsibilities of the statistical programmer of a pharmaceutical research and development group or contract research organization (CRO) include importing/exporting data, deriving variables and creating analysis data sets, and creating clinical study report (CSR) materials such as tables, listings, and figures (TLFs). These outputs are used for submission to regulatory agencies and are generally programmed in SAS®.

There is a growing movement to integrate artificial intelligence (AI) capabilities, such as machine learning and natural language processing, into the clinical programming world. SAS programmers are encouraged to pursue relevant education and training to adapt to the anticipated evolution of the clinical SAS world. Two very useful tools for SAS programmers wishing to expand their AI knowledge are Python and R. Python is a high-level, open source general-purpose language; R software is a free open source software for statistical computing and graphics. Since clinical SAS programmers come from diverse backgrounds and may not necessarily have a background in computer science, learning a new language may seem like a difficult task.

In this paper, we provide examples of SAS procedures and syntax commonly used by the clinical programmer and the equivalent Python and R code. We hope draw to parallels in the syntax of the languages to help allay the concerns clinical SAS programmer may have with learning a new language and to help familiarize themselves with an alternative way of programming.

COMMONLY USED PROCEDURES AND SYTNAX

COMMENTS AND DOCUMENTATION

Documentation of programming is crucial for the clinical SAS programmer. Comments in SAS start with an asterisk, *, and end with a semi-colon, ;. Alternatively, the SAS® programmer can start embed comments within a /* and a */. Comments in Python and R start with the hash character, # , and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal.

SAS	Python	R
*this is a comment; /*this is a comment*/	#this is a comment	#this is a comment

IMPORTING DATA

SAS	Python	R
proc import datafile = "C:\mydata.csv" out=d2 dbms=csv replace; run;	import pandas as pd d2 = pd.read_csv("C:\mydata.csv")	d2 <- read.table("C:\mydata.csv", header=TRUE, sep=",", row.names="id")

PRINTING DATA

SAS	Python	R
proc print data = d1 var1 run;	print("var1") OR print(var1)	head(var1)

FUNCTIONS

SAS	Python	R
var2 = upcase(var1)	var2 = var1.upper()	var2 <- toupper(var1)
var2 = lowcase(var1)	var2 = var1.lower()	var2 <- tolower(var1)
var2 = propcase(var1)	var2 = var1.title()	
var2 = left(var1)	var2 = var1.lstrip()	var2 <- trimws(var1, "l")
var2 = trim(var1)	var2 = var1.rstrip()	var2 <- trimws(var1, "r")
var2 = strip(var1)	var2 = var1.strip()	var2 <- trimws(var1)
put(numeric_var1, best.)	str(numeric_var1)	toString(numeric_var1)

CONCATENATION

SAS	Python	R
x = "hello" y = "world" x || " " || y >> "hello world"	x = "hello" y = "world" x + y >> "hello world"	x = "hello" y = "world" paste(x, y, sep = " ") >> "hello world"

COUNTS

SAS	Python	R

proc freq data=d1;
  tables var1*var2;
run;
	x.value_counts(dropna =
	  False).sort_index()
	
	pd.crosstab(df.var1,
	  df.var2).apply(lambda
	  r: r/r.sum(), axis =
	  1)
		table(d1$var1, d1$var2)
		
		prop.table(table(d1$var1,
		  d1$var2))
		prop.table(table(d1$var1,
		  d1$var2), 1)
		prop.table(table(d1$var1,
		  d1$var2), 2)

DATA MANIPULATION

Subsetting Data

SAS	Python	R

data d2;
  set d1;
  where x= 'hello';
run;
	d2 = [d1.x == "hello"]	d2 <- subset(d1, x ==
		  "hello")

Merging Data

SAS	Python	R

data d3;
  merge d1(in=a) d2(in=b);
  by var1;
/*left join*/ if a;
	import pandas as pd
	
	#left join
	d3 = pd.merge(d1,
		#left join
		d3 <- merge(x=d1, y=d2,
		  by = "var1",
		  all.x=TRUE)

/*right join*/ if b;
/*inner join*/ if a and b;
/*full merge is default*/
run;
	d2, on = 'var1',
	how = 'left')
	
	#right join
	d3 = pd.merge(d1,
	d2, on = 'var1',
	how = 'right')
	
	#inner join
	d3 = pd.merge(d1,
	d2, on = 'var1',
	how = 'inner')
	
	#outer join
	d3 = pd.merge(d1,
	d2, on = 'var1',
	how = 'outer')
		
		#right join
		d3 <- merge(x=d1, y=d2,
		by = "var1",
		all.y=TRUE)
		
		#inner join
		d3 <- merge(x=d1, y=d2,
		by = "var1")
		
		#outer join
		d3 <- merge(x=d1, y=d2,
		by = "var1",
		all=TRUE)

Sorting Data

SAS	Python	R
proc sort data=d1; by var1 var2…..; run;	df.sort(['var1', 'var2'], ascending = True)	d1[order(d1$var1, d1$var2)]

Dropping and Keeping Variables

SAS	Python	R
data d2; set d1; keep var1 var2….; run;	d2 = d1.loc[:, ['var1', 'var2'])	d2 <- d1[c("var1", "var")]
data d2; set d1; drop var3 var4…; run;	d2 = d1.drop(['var1', 'var2'], axis = 1)	d2 = subset(d1, select = -c("var1","var2"))

CONCLUSION

There is a growing movement and acceptance towards using Python and R in the Clinical SAS® programming world. We hope that the examples of common SAS® procedures and syntax and their translation to Python and R provided in this paper can help the beginner Python and/or R programmer adapt to the anticipated changing landscape of the clinical programming world.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Janet J. Li
Pfizer Inc.
janet.li@pfizer.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.