SESUG Paper 052-2024

A SAS Macro to parse Clinical Conductor Timesheets

Kennedy Gachigi, MS, Research Scientist, OrthoCarolina Research Institute

ABSTRACT

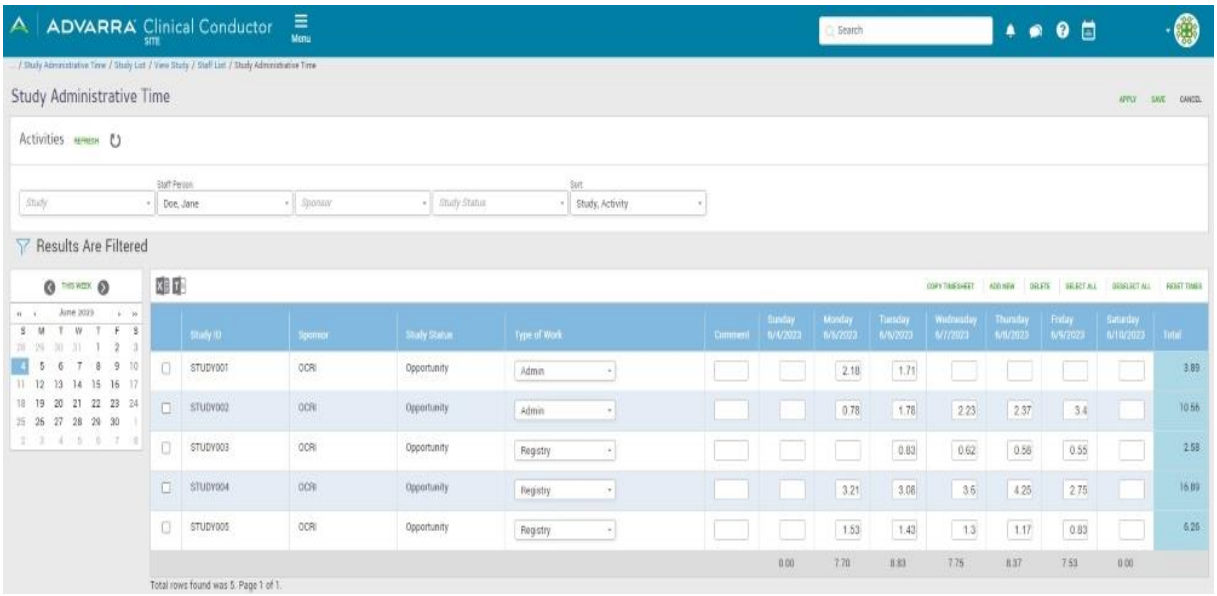
Most time management software packages have inbuilt reporting mechanisms to process administrative reports such as payroll. Organizations are constantly adopting new software to help improve business processes and reduce costs. The setting for this paper is a research institute that adopts a central clinical research management system (CRMS), to track all their clinical trial data as well as time spent on projects by each employee.

This paper describes a macro that is useful in generating monthly timesheet reports from clinical conductor, a new clinical trial management system by Advarra that does not have a specific inbuilt report to summarize monthly data. The paper is for intermediate level programmers with an understanding of the macro and SQL environments in SAS.

INTRODUCTION

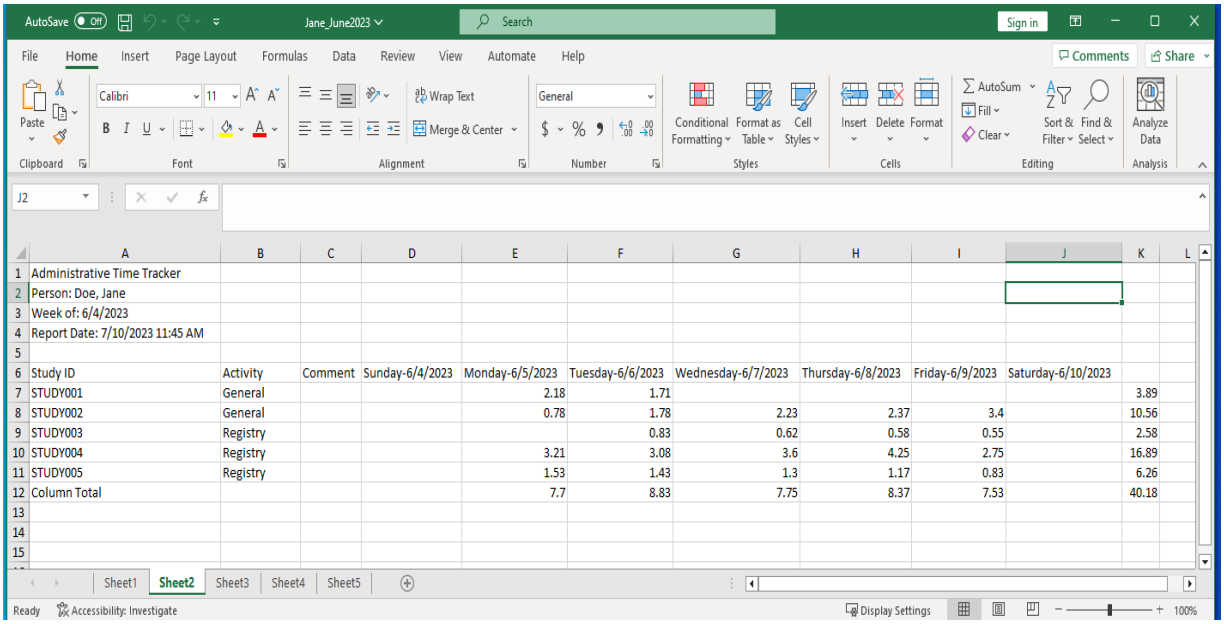
To stay competitive in today's rapidly changing business environment, organizations must be willing and able to evolve their business strategy. In a setting where programmers are part of a research team, their expertise can be harnessed to improve business processes. A transition from one way of doing things to another often brings changes to periodic reports. Our case study is an organization moving from one vendor's customized reports to Clinical Conductor, a new clinical research management system (CRMS) by Advarra. The move is to enable the organization to have all their clinical trial data as well as financial tracking information in one place. As noted by Mullen *et al*[1], adoption of Advarra's CRMS helped them create a standardized experience to manage clinical research from protocol creation to project closeout.

Time tracking is an important utility within this CRMS. The time template previously used was an excel workbook customized to enable easy summaries of the total time spent by each employee in each study. Clinical Conductor reports must be exported as raw excel files that have the weekly information in separate sheets for each employee. Display 1 is a sample page of Clinical Conductor website:



Display 1. Sample page of Clinical conductor timesheet for a week

This data exported to excel is an unstructured spreadsheet with 5 sheets for the month as shown in display 2 below:



	A	B	C	D	E	F	G	H	I	J	K	L
1	Administrative Time Tracker											
2	Person: Doe, Jane											
3	Week of: 6/4/2023											
4	Report Date: 7/10/2023 11:45 AM											
5												
6	Study ID	Activity	Comment	Sunday-6/4/2023	Monday-6/5/2023	Tuesday-6/6/2023	Wednesday-6/7/2023	Thursday-6/8/2023	Friday-6/9/2023	Saturday-6/10/2023		
7	STUDY001	General			2.18	1.71					3.89	
8	STUDY002	General			0.78	1.78	2.23	2.37	3.4		10.56	
9	STUDY003	Registry				0.83	0.62	0.58	0.55		2.58	
10	STUDY004	Registry			3.21	3.08	3.6	4.25	2.75		16.89	
11	STUDY005	Registry			1.53	1.43	1.3	1.17	0.83		6.26	
12	Column Total				7.7	8.83	7.75	8.37	7.53		40.18	
13												
14												
15												

Display 2. Sample export from Clinical Conductor for the month of June 2023

Note that the actual data starts in row 7. Additional processing of columns and rows is required to compute the total time spent monthly by each employee on a specific study. This can be a daunting task without the help of statistical software. In this scenario, a programming language becomes superior since it enables curating the data without errors.

SAS is a powerful tool for such problems as we'll see in this paper. We nest 2 macros; the first one generates one monthly report for each employee from the 5 weekly timesheets and the second one generates the report for all the employees in the organization in one spreadsheet.

SAS has been used in the past to build an end-to-end application for timesheet processing as described by Kathryn Pocock and Hugh O'Neill[2] in 1990. To our knowledge SAS code that fills in gaps to build reports from existing software that generate unstructured spreadsheets has not yet been published. This paper presents an example that can be incorporated in such a workflow to improve efficiency.

MACRO REQUIREMENTS

All timesheets must be exported into a folder, and each sheet named the employee's name. We use an example of 2 dummy employees in an organization: Jane and Joe, to run their monthly report for June 2023.

The 2 sample workbooks used in this macro can be found here: https://github.com/KennedyGachigi/SASPaper . Save them on your local computer.

MACRO PROGRAM

After the data is saved in a folder, create 3 macro variables for the date the report is being run, the month and the directory.

```
%let rdate = %sysfunc(date(),YYMMDDN.); * report date;
%let month = June2023; * Change month and year;
%let path = C:\YourLocalFolder;
```

1. The first part of the macro imports all 5 sheets and names them staff1 – staff5.

```
%macro report;

 %do i=1 %to 5;

proc import out = &staff&i
     datafile = "&path\&staff._&month..xlsx"
       dbms = xlsx replace;
       getnames = no;
       datarow = 7; * note: this is the first row of the needed data;
       sheet = "Sheet&i";
run;
```

2. There is a need to dynamically hold the last column in each sheet which has the weekly total for each row. In display 2 above, that is column K, but for a week with fewer days the column changes. Note: from the dictionary.columns table, the macro function %upcase (rather than the regular function upcase) is required to read the data name, which are stored in upper case letters. The call symput function in the subsequent data step is useful when one needs to create a macro variable in a data step.

```
proc sql noprint;
select distinct name into:cols separated by '' from dictionary.columns
  where memname = %upcase("&staff&i") and libname = 'WORK';
quit;

data lastcol;
 cols = "&cols";
 lastcol = substr(reverse(cols), 1, 1); * reverse and get the last column
letter as a column in a dataset;
 call symput('lastcol', lastcol); * create a macro variable that holds the
last letter value of the last column to use in the next data step;
run;
```

3. The following steps are additional processing to create 1 report for each employee from the 5 sheets of data.

```
data &staff&i;
 set &staff&i;
  keep A &lastcol;
  rename A = acc
         &lastcol = total;
  if &lastcol ne 0 and A ne 'Column Total'; * remove column total rows and
accounts with 0 hours;
run;

%end;
```

```
data &staff._0;
 set &staff.1 &staff.2 &staff.3 &staff.4 &staff.5;
run;

* create a report for each account and total hours;
proc sql;
 create table &staff._1 as select distinct acc as Account, sum(total) as
hours from &staff._0 group by acc;
 create table &staff._tot as select sum(total) as total_hours from &staff._0;
quit;

proc sql;
  create table &staff._2 as
   select * from &staff._1 cross join &staff._tot;
quit;

data &staff._report;
 label Account = 'Account'
       hours = 'Hours'
       total_hours = 'Total Hours'
       proportion = 'Proportion'
       percent = '%';

 set &staff._2;
 proportion = round(hours/total_hours, .000001);
 percent = cats(round(proportion*100, .01),'%');
run;

proc export data = &staff._report
     outfile = "&path\EffortFiles_&rdate..xlsx"
       dbms = xlsx label replace;
       sheet = "&staff" ;
run;

%mend report;
```

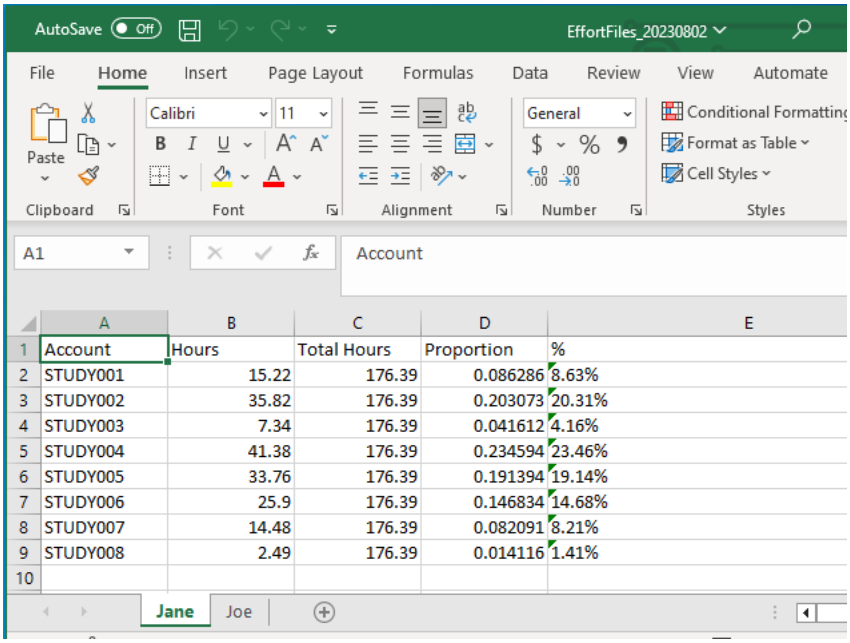
4. We now need another macro to loop through the staff names. In this macro, we nest the %report macro above into the %effort macro. Remember, a macro written inside of another macro does not work well and is harder to debug. Nesting is a better approach.

```
 * options mlogic mprint; * Only need this to debug in case the macro runs
into errors;

%let stafflist = %str(Jane, Joe); * add staff names as needed;

%macro effort;
 %let n = 1;
 %let staff = %scan(&stafflist,&n);
 %do %until(&staff eq ); * this "do until" helps to bypass declaring the
number of employees as a macro variable;

 %report; * call this macro here;

%let n = %eval(&n+1);
 %let staff =%scan(&stafflist,&n);
%end;
```

```
%mend;

%effort;
```

MACRO OUTPUT

After running the macro, the result is one spreadsheet named EffortFiles with a separate tab for each employee summarizing the monthly use of time on each study as shown in display 3.



	A	B	C	D	E
1	Account	Hours	Total Hours	Proportion	%
2	STUDY001	15.22	176.39	0.086286	8.63%
3	STUDY002	35.82	176.39	0.203073	20.31%
4	STUDY003	7.34	176.39	0.041612	4.16%
5	STUDY004	41.38	176.39	0.234594	23.46%
6	STUDY005	33.76	176.39	0.191394	19.14%
7	STUDY006	25.9	176.39	0.146834	14.68%
8	STUDY007	14.48	176.39	0.082091	8.21%
9	STUDY008	2.49	176.39	0.014116	1.41%
10					

Display 3. Final report file for all employees in one spreadsheet

CONCLUSION

Reports generated by clinical research management systems such as Advarra need further refining to make use of the timesheet information. As organizations move towards streamlining fragmented infrastructure, the cost of that is data stored and exported in unstructured files that need further processing. SAS reporting can be utilized in this scenario in departments outside of research. The first step is to include the statistician/programmer in management conversations that involve improving repetitive departmental reports. Organizations that have programmers on staff can utilize their vital reporting skills and get away from manual processes that consume a lot of person-hours without additional costs in software.

REFERENCES

1. Mullen CG, Houlihan JY, Stroo M, Deeter CE, Freel SA, Padget AM, Snyder DC. Leveraging retooled clinical research infrastructure for Clinical Research Management System implementation at a large Academic Medical Center. J Clin Transl Sci. 2023 May 18;7(1):e127. doi: 10.1017/cts.2023.550. PMID: 37313387; PMCID: PMC10260330.
2. Kathryn Pocock, Hugh O'Neill of PAREXEL International Limited. A SAS® SYSTEM FOR TIMESHEET PROCESSING. SAS European Users Group International Conference 1990.

ACKNOWLEDGMENTS

We would like to acknowledge Caleb Michalek, Clinical Trials Manager at OrthoCarolina Research Institute for presenting this problem and continuously utilizing the solution.

CONTACT INFORMATION

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

Kennedy Gachigi

kennedy.gachigi@orthocarolina.com