Paper AS03

Scrutinize Tables Listings Figures (TLFs) Prior to Customer Delivery

Vijetha Kode, Merck & Co., Inc., North Wales, PA, USA

ABSTRACT

Quality, accuracy, and traceability of statistical analysis results are essential in clinical trials. Decisions are made every day based on statistical results from our programs. Even though validation is part of the report's development, it is always good practice to spot check and review the Tables, Listings and Figures (TLFs) outputs prior to delivering them to customers. This paper focuses on the importance of carefully reviewing TLFs prior to customer delivery as a lead programmer. This careful review can be programmatically conducted in different ways such as spot checking the counts using different procedures, cross checking the counts across the relevant TLFs and other approaches detailed in the paper. Efficient ways to generate TLFs are also described including adding additional checks in analysis dataset programs, assigning macro variables for recurring population filters, titles, and others elucidated in paper.

INTRODUCTION

In clinical trials, quality of the deliverables created through analysis programs using SAS or R relies on the principle of independent validation by peers. Typical concept of validation requires independent programming by the peers, using the same input source files, and produces the same results as the developer. The independent and efficient programming validation helps producing the quality reports by finding most issues but there is still possibility for the quality issues due to various reasons such as:

- Both programmers taking the same assumptions, liberties based on their prior study working experience.
- Inconsistency of the counts between the TLFs across the study.
- Different programmers involved in development/validation of set of TLFs and understanding the specifications differently.
- Inconsistencies within the specification document or programming instructions for relevant TLFs.

Hence it is always good practice to spot check & review the TLFs prior to customer delivery. This paper leverages on efficient ways to generate TLFs and different approaches for reviewing them.

SCRUTINIZE THE KEY TLFs BY DIFFERENT APPROACHES IN ONCOLOGY STUDIES

- Spot check the key TLFs by writing small chunks of code as described in table 1. It may take good amount of time to follow this approach for the first study, but we can use the same code with minimal study specific changes for subsequent studies.
- Cross check counts across TLFs such as Adverse Event Summary with specific AE (Adverse Events) counts, deaths, SAEs (Serious Adverse Events), overall event counts across efficacy reports and many others.
- Check the cutoff date, MedDRA (Medical Dictionary for Regulatory Activities) and WHO drug dictionary, lab and AE CTCAE (Common Terminology Criteria for Adverse Events) versions used in the data.
- Select a random treated participant from SDTM (Study Data Tabulation Mode) dataset and check both safety and efficacy values in the listings for transparency.
- Cross checking the counts across the TLFs (Listing vs Table, Table vs Figure).
- Make sure all the TLFs are generated as per the mockup and specifications.
- Crosscheck number of participants across SDTM datasets and corresponding ADaM datasets such as CM, ADCM and others.

TABLE 1

Type (Oncology)	Details
Baseline Report **ADaM dataset used:** ADSL	**One way to check baseline counts is by using proc freq as shown below** Use variables in freq procedure as needed for baseline report **proc freq** data=ADSL(where=(randfl="Y")); tables sex*trt01p (any variable you wanted to check*trtxxp)/list missing nocol nopercent; run;
Disposition Report **ADaM dataset used:** ADSL	**One way to check disposition counts for study and treatment is by using proc freq as shown below** Use disposition variables in freq procedure as needed for disposition report **proc freq** data=ADSL(where=(randfl="Y")); tables eosstt*dcsreas*dcsreasp eotstt*dctreas*dctreasp/list missing nocol nopercent;

	run;
Exposure Reports **ADaM dataset used:** ADEXSUM, ADSL	**Generally, there are two reports in exposure:** **Exposure by duration and Exposure summary.** • **Exposure by duration, one way to check is by using proc sql** **proc sql**; create table exp_chk as select paramcd, trta, trtan, trtan as coded, trta as decode, usubjid, sum(aval/**30.4367**) as avalm from adexsum where paramcd = 'xxxxx' and usubjid in (select usubjid from adsl where saffl='Y') group by paramcd, trta, usubjid; **quit**; **proc sql**; select decode, count(distinct usubjid) as cnt, sum(avalm) as sumaval from exp_chk where avalm > 0/*give any number you wanted to check*/ group by decode; **quit**; • **Exposure Summary, spot check counts by proc summary** **proc summary** data = adexsum(where=(saffl='Y' and paramcd='xxxxx')); class trt01a; var aval; output out = fin3 n = n mean = mean median = median std = sd min = min max = max; run;

AE summary and specific reports
 	
ADaM dataset used:
ADSL
and used SDTM AE dataset, so this can verify both ADAE and related TLFs
	AE summary check from SDTM level
	%macro aesum(obs=, out=);
	proc sort data=ae out=ae0;
	by usubjid;
	where not (index(upcase(epoch), "SCREENING") > **0**) and not (index(upcase(epoch), "SECOND") > **0**) and not (index(upcase(epoch), "POST-STUDY") > **0**) and (upcase(aeser)="Y" or (upcase(aeser)="N" and not (index(upcase(epoch), "EXTENDED") > **0**))) and (upcase(AEDECOD) not in ("NEOPLASM PROGRESSION","MALIGNANT NEOPLASM PROGRESSION", 'DISEASE PROGRESSION') or (upcase(AEDECOD) in ("NEOPLASM PROGRESSION","MALIGNANT NEOPLASM PROGRESSION", 'DISEASE PROGRESSION') and upcase(aerel)= "RELATED")) &obs;
	run;
	proc sort data=adsl out=adsl0;
	by usubjid;
	where saffl="Y";
	run;
	data ae_sum_check;
	merge adsl0(in=a) ae0(in=b);
	by usubjid;
	if a and b;
	run;
	proc sort data=ae_sum_check out=&out nodupkey;
	by usubjid;
	run;
	proc freq data=&out;
	tables trt01p/list missing nocol nopercent;
	run;
	%mend aesum;
	/* Overall AE summary*/
	1) %aesum(obs=,out=aesum);
	/*drug related AE*/
	2) %aesum(obs= and upcase(aerel)="RELATED", out=aerel_sum);

	/*toxicity grade 3-5 AE*/ 3) %aesum(obs= and aetoxgr in ('3','4','5'), out=aerel_sum); /*serious related AE*/ 4) %aesum(obs= and aeser="Y", out=ae_sae); /*death*/ 5) %aesum(obs= and aesdth="Y", out=ae_dth); **Cross check AE specific counts with AE summary reports**

Time to onset AE
 	
ADaM dataset used
ADSL, ADAE
	proc sort data=adsl out=adsl0;
	by usubjid;
	where saffl="Y";
	run;
	proc sort data=adae out=adae0;
	by usubjid;
	where saffl="Y" and /*add required variables as needed to filter*/;
	run;
	data ae_sum_check;
	merge adsl0(in=a) adae0(in=c);
	by usubjid;
	if a and c;
	run;
	** tox 3, 5;
	data aetox;
	set ae_sum_check;
	if aetoxgr in ('3','4','5');
	aval=astdt-trtsdt+**1**;
	run;
	proc sort data=aetox;
	by usubjid astdt;
	run;
	data tox_;
	set aetox;
	by usubjid astdt;
	if first.usubjid;
	run;
	proc sort data=tox_;
	by studyid trt01a;
	run;
	proc means data=tox_ mean std median min max q1 q3 noprint;
	by studyid trt01a;
	var aval;
	output out=tox_fin mean=mean std=sd median=median min=min max=max q1=q1 q3=q3;
	run;

Concomitant Medication reports **ADaM datasets used:** ADSL, ADCM	**proc sort** data=adsl out=adsl0; by usubjid; where saffl="Y"; **run**; **proc sort** data=adcm out=adcm0; by usubjid; where saffl="Y" and ontrtfl="Y" and cmgrpid="CM"; **run**; **data** adsl_cm; merge adsl0(in=a) adcm0(in=b); by usubjid; if a and b; **run**; **proc sort** data=adsl_cm out=adzz nodupkey; by usubjid; **run**; /* for overall counts*/ **proc freq** data=adzz; tables trt01p/list missing nocol nopercent;

	run; /*for counts by specific ATC term*/ **proc sort** data=adsl_cm; by atc1 atc2 usubjid; **run**; **data** cm_chk; set adsl_cm; by atc1 atc2 usubjid; if last.usubjid; **run**; **proc freq** data=cm_chk(where=(trt01pn=**1**)); tables trt01p*atc1*atc2/list missing nocol nopercent; **run**; **proc freq** data=cm_chk(where=(trt01pn=**2**)); tables trt01p*atc1*atc2/list missing nocol nopercent; **run**;

Efficacy Reports in Oncology:
 	
Objective Response Rate (ORR)
 	
ADaM Datasets used:
ADSL, ADRS
	Normally there are two types of ORR reports
	Overall ORR summary and ORR Analysis
	
	Overall ORR Summary
	proc sort data=adsl out=adsl0;
	by usubjid;
	where saffl="Y";
	run;
	proc sort data=adrs out=rs;
	by usubjid;
	where paramcd="xxxxx";/*pass paramcd as needed*/
	run;
	data adsl_rs;
	merge adsl0(in=a) rs(in=b);
	by usubjid;
	if a and b;
	run;
	proc freq data=adsl_rs;
	tables avalc/list missing nocol nopercent;
	run;
	
	ORR Analysis (change statistical models according to the protocol)
	data adrs;
	set adrs(where=(randfl="Y" and paramcd="xxxx"));
	if aval in (**2**,**3**) then orr=**0**;
	else if aval ne **.** then orr=**1**;
	run;
	proc sort data=adrs; by trt01pn; **run**;
	proc freq data=adrs;
	by trt01pn;
	tables orr/out=rs0(drop=percent);
	run;
	***95% CI;
	proc freq data=adrs;
	by trt01pn;
	table orr/binomial(exact);
	ods output binomialcls=rs1;
	run;

Time to Response **ADaM Datasets used:** ADSL, ADRS, ADTTE	**proc sort** data=adsl out=adsl0; by usubjid; where randfl="Y"; **run**; **proc sql**; create table t2resp as select distinct a.usubjid ,a.aval/**30.4367** as aval ,**1** as orr from adtte as a , adrs where a.paramcd="T2RIRC" and adrs.paramcd="BORCFIRC" and adrs.avalc in ('CR','PR') ; **quit**; **data** t2r_population; merge adsl0 (in=a) t2resp (in=b);

	by usubjid; if a; if orr ne **1** then orr=**0**; **run**; **proc means** data=t2r_population min max median mean stddev nway noprint missing; var aval; class trt01pn orr; output out=_stat(where=(orr=**1**)) min=min max=max median=median mean=mean stddev=stddev; **run**;

Duration of Response
 	
ADaM Datasets used:
ADSL, ADRS, ADTTE
	proc sql;
	create table resp_dur as select distinct a.usubjid ,a.aval/**30.4367** as aval ,a.cnsr as censor from adtte as a , adrs where a.paramcd="RDIRC" and adrs.paramcd="BORCFIRC" and adrs.avalc in ('CR','PR') ;
	quit;
	data dur_population;
	merge adsl0 (in=a) resp_dur (in=b);
	by usubjid;
	if a and b;
	run;
	ODS OUTPUT ProductLimitEstimates=_Est Quartiles=_quart;
	proc lifetest data =dur_population ALPHAQT=**0.05**;
	time aval*censor(**1**);
	strata trt01pn;
	run;
	ODS OUTPUT CLOSE;
	proc means data=dur_population min max mean nway noprint MISSING;
	var aval;
	class trt01pn;
	output out=min_max min=min max=max mean=mean;
	run;
	ODS OUTPUT Quartiles =_quartiles ProductLimitEstimates=_pl;
	Proc lifetest data=dur_population timelist= **6 9 12** ALPHAQT=**0.05**;
	time aval*censor(**1**);
	strata trt01pn;
	run;
	ODS OUTPUT CLOSE;

Progression Free Survival (PFS)
Overall Survival (OS)
 	
ADaM Datasets used:
ADSL, ADTTE
	Progression Free Survival and overall Survival
	
	%macro eff(obs=,out=);
	proc sort data=adsl out=adsl0;
	by usubjid;
	where randfl="Y";
	run;
	proc sort data=adtte out=tte;
	by usubjid;
	where &obs;
	run;
	data &out;
	merge adsl0(in=a keep=usubjid) tte(in=b);
	by usubjid;
	if a and b;
	avalm=aval/**30.4367**;
	run;
	proc freq data=&out;
	tables trt01p/list missing nocol nopercent;
	run;
	proc freq data=&out;
	tables trt01p*cnsr/list missing nocol nopercent;
	run;
	proc lifetest data=&out method=kM alpha=**0.05** outsurv=statkm plots=none;
	time avalm*cnsr(**1**);
	strata trt01pn;
	run;

	proc means data=&out sum noprint nway;
	  var avalm;
	  class trt01pn;
	  output out=zz sum=sumd;
	run;
	proc lifetest data=&out method=kM alpha=**0.05** timelist=**6**,**12**,**18** reduceout outsurv=statkm6 plots=none;
	  time avalm*cnsr(**1**);
	  strata trt01pn;
	run;
	
	%mend eff;
	
	%eff(obs=(upcase(paramcd)=”OS”), out=os_chk);
	%eff(obs=(upcase(paramcd)=”PFSIRC”), out=pfs_chk);

EFFICIENT WAYS TO GENERATE TLFS

1. Assign standard macro variables stated below for recurring filters in startup program and use in TLF programs
    - population selection filter
    - title and subtitle filters
    - observation selection filter and standard footnote filters for AE reports
    - cutoff date
    - MedDRA, CTCAE version and others as needed
2. Try to add checks at ADaM (Analysis Dataset Model) dataset program level to catch important data issues such as uncut last known alive date greater than death date, partial death dates and others.
3. Ensure TLFs can be generated using ADaM datasets without significant derivations in TLFs programs
4. Ensure P21 is run on ADaM datasets before starting TLFs work
5. Create a batch program for all TLFs and run at once
6. Have a log file created for all outputs and develop a separate macro to capture all log files into one output.

CONCLUSION

Overall quality checks of statistical analysis results in clinical trials is a complex process due to large volume of data, complicated derivations, analysis and different structures of data and reports (SDTM, ADaM and TLFs). Even though validation is part of quality programming development, due to various issues and possibilities mentioned above that can affect the reports quality, I emphasize it is important to spot check and review the TLFs as a programming lead, after validation is completed to improve the quality of outputs before customer delivery.

REFERENCES

Quality Control and Validation – More than Just PROC COMPARE.
https://www.lexjansen.com/phuse/2019/pp/PP08.pdf

ACKNOWLEDGMENTS

Naveen Kommuru, Mary Varughese, and Amy Gillespie.

CONTACT INFORMATION

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

Vijetha Kode
Merck & Co., Inc.
351 N Sumneytown Pike
North Wales, PA, 19454, USA
Email: vijetha.kode@merck.com