Paper CT07

Patient Narratives – Speeding Up Narrative Generation Using ODS Document Stores

Nev Cope
AstraZeneca, Cambridge, UK

ABSTRACT

Patient narratives are a critical part of many clinical trials, and generating them programmatically with SAS® for hundreds of trial subjects can take a long time. What if, instead of generating a narrative that processes many sentences and tables repeatedly for each subject, we could generate the sentence/table for all subjects just once and then manipulate all the outputs using ODS Document Stores?

Well – you can! This paper will demonstrate techniques using PROC DOCUMENT to manipulate a collection of SAS outputs from one structure into another, speeding up the generation of a large numbers of narrative reports.

INTRODUCTION

Section 12.3.2 of the ICH-E3 guidelines state that deaths, serious adverse events or other significant adverse events should be documented in a patient narrative (within either the text or section 14.3.3 of the study report).

Just as with the usual tables, figures and listings generated by study programmers for section 14 of the clinical study report, the programs used to generate narratives typically get copied from one study to another for many years.

This can lead to both programming stagnation (with the mantra “but it’s what we always do”) and execution inefficiencies (with the mantra “this program takes SO long to run”)

Both of these outcomes had been reached within the AstraZeneca programming teams, and as part of the Data Automation group I was tasked with investigating ways to improve narrative generation across different therapeutic areas.

Please note, all patient data displayed in this paper is simulated.

WHAT ARE NARRATIVES?

The actual layout of a narrative can vary between the different Therapeutic Areas (TAs). Each of the 3 TAs that I was working with had specified their requirements for narratives via a Standard Narrative Participant Template document.

Generally, a narrative consists of a mixture of tables and sentences, arranged sequentially on a page with either one subject per document, or a single document containing all subjects and a contents page.

For the Cardiovascular, Renal and Metabolism (CVRM) TA the standard template consisted of a straightforward 13 tables, beginning with demography and an overview of Adverse Events (AEs), before going on to detail a subject’s medical history and study drug administration, and ending with laboratory data and other safety information (Figure 1 below)

Study ID: D########## <<Study ID>> (<<Study Name>>) Page <<x>> of <<y>>
Subject ID: E######## <<Subject ID>>
Narrative category: Deaths=Myocardial Infarction; All serious Adverse Events=Diarrhoea and Nausea
<<Narrative Category>> = <<List of AE(s) triggering the category>> <<Repeat for each category triggered>>

<Table 1>General information

Study ID:	D####X#####
Subject ID:	E#######
Age/Sex/Race/Ethnicity:	65 years/Male/Asian/Not Hispanic
<Planned/Randomized> Treatment[b]:	Roxadustat
Actual Treatment[b]:	Roxadustat
Date (Day[a]) of Randomization:	DD MMM YYYY (###) or Not applicable
Study Treatment[b]:	Roxadustat 200mg BID
Date of First (Day[a]) / Date of Last (Day[a]) Study Treatment[b]:	DD MMM YYYY (###) / DD MMM YYYY (###)
Date (Day[a]) of Death:	DD MMM YYYY (###) or Not applicable
Country:	France
BMI (kg/m^2):	28.15
Smoking status:	Former smoker

[a] Relative to date of randomization
[b] Study Treatment – Investigational product and/or planned additional treatment given to a subject during the course of the study <it is the responsibility of the study team to align the definition with the protocol>

<Table 2>
Qualifying Event and Risk Factors (Optional for studies)
< The Qualifying Event or Risk Factor could be one of the below examples or something completely different and should not repeat what is already found in the narratives unless considered a true risk factor (ie. Age >= 65). In many cases, this information is part of the inclusion criteria. >

Qualifying Event or Risk Factor	Description
INDEX Event #	<supporting information>
Medical History Term #	<supporting information>
Inclusion Criteria #	<supporting information>

<Table 3>
Important Study Events

Death	<Preferred Term> Cardiovascular Death
Serious adverse events (SAEs)	<Preferred Term> Peripheral arterial occlusive disease <Preferred Term> Cardio-respiratory arrest
AEs leading to permanent discontinuation of study treatment[a]	<Preferred Term> Latent autoimmune diabetes in adults
Adverse events of special interest (AESIs)	<Preferred Term> Not applicable

[a] Study Treatment – Investigational product and/or planned additional treatment given to a subject during the course of the study <it is the responsibility of the study team to align the definition with the protocol>

<Table 4>
Medical History

Start Date (Day[a])	Stop Date (Day[a])	Preferred Term	Current/Past
AUG 2014	SEP 2014	Pain	Current
FEB 2015	ongoing	Fatigue	Current

<Table 5>
Specific Medical and Surgical History<Medical History is displayed whether or not it is present>

Specific Medical and Surgical History	Description
Medical History Term	<Yes/No> | <Start Date(Day[a])> | <Location of Event> | <Findings>: <Result>

[a] Relative to date of randomization

<Table 6>
Prior/Concomitant Medication

Start Date (Day[a])/ Stop Date (Day[a])	Medication Name	ATC Classification	Medication Dose	Route	Frequency
DD MMM YYYY (###)/ DD MMM YYYY (###)	HYDROCHLOROTHIAZIDE	Thiazides, plain			
DD MMM YYYY (###)/ ongoing	DICLOFENAC	Antiinflam preps, non-steroids for topical use			
DD MMM YYYY (###)/ DD MMM YYYY (###)	CEFUROXIME	Second-generation cephalosporins			

[a] Relative to date of randomization

Figure 1 - CVRM Narrative Template Excerpt

The Respiratory and Immunology (R&I) TA was completely different, consisting of only 2 initial tables (containing a subject header and list of AEs), followed by 22 sentences - sometimes grouped into paragraphs (Figure 2 below).

Protocol: D##########
<Safety/All> Population

<USUBJID> Narrative

<Table 1>

Protocol: DXXXXXXXXXX	**Subject Enrolment Code:** E########
Country: France	**Age/Sex/Race/Ethnicity:** 48 years/Male/White/Not Hispanic or Latino
Investigational Product and Dosing Regimen:	**Adverse Event Preferred Term:** <preferred term> (<category>) Angioedema (SAE) Anaphylactic reaction (Anaphylactic reactions) Hypersensitivity (SAE) Anaphylactic reaction (Anaphylactic reactions) <Sort by STUDYID, USUBJID, AESTDTC, AEENRTPT, AEENDTC AEDECOD and only display the Adverse Events triggered by a narrative category with the categories displayed in parenthesis following the adverse event>

<Table 2>

Adverse Events <Provide a complete chronological list of only adverse events triggering a narrative>

Preferred Term	Event Start Date	Event Stop Date	Related (Yes/No)	Seriousness (Yes/No)	Action Taken with IP	Outcome	Severity
Adverse Event 1	07-Nov-2017	09-Nov-2017	No	Yes	Dose not changed	Recovered/ resolved	Moderate
Adverse Event 2	28-Nov-2017	02-Dec-2017	No	Yes	Dose not changed	Recovered/ resolved	Severe
Adverse Event 3	05-Jan-2018	05-Jan-2018	No	Yes	Dose not changed	Recovered/ resolved	Mild
Adverse Event 4	15-Feb-2018	16-Feb-2018	No	Yes	Not applicable	Recovered/ resolved	Moderate

Paragraph 1 includes sentence 1
<Sentence 1>
Subject <Subject ID>, a <Age>-<Age units>-old <race> <gender> from <Country>, with <Disease Under Study (DUS)> diagnosed on <Date DUS First Diagnosed>, was screened in <Study ID> study on <Date of Informed Consent>.

Paragraph 2 includes sentence 2
<For sentence 2, display the code for either 2a or 2b depending on the data for each subject>
<Sentence 2>
The subject
- <if randomised to blinded treatment, display the following: 'was randomised to '>
- <if randomised to open label treatment, display the following: 'initiated treatment with '>

<Sentence 2a><If the randomisation date and the first IP dose date are the same, display the following (similar to event-based narrative):> <Investigational product dose and dosing regimen> and received the first dose on <dosing start date>.

<Sentence 2b><If the randomisation date and the first IP dose date are different, display the following:> <Investigational product dose and dosing regimen> on <date randomised> and received the first dose on <dosing start date>.

Paragraph 3 includes sentence 3
<Sentence 3>
The subject's medical history included <derived term> <continue to list all (current, past and other) medical history followed by a comma without start/stop dates unless otherwise specified. Prior to the last value, display 'and' and the last medical history value> < If there is no medical history, display 'The subject had no medical history,'>.

Paragraph 4 includes sentence 4
<Sentence 4>
The subject's surgical history included <derived term>, <continue to list all surgical history followed by a comma without start dates. Prior to the last value, display 'and' and the last surgical history value>. <If there is no past surgical history, display 'The subject had no surgical history.'

Figure 2 - R&I Narrative Template Excerpt

Finally, the Oncology TA narrative template is more extensive and was a combination of the CVRM and R&I structures, with a mixture of 19 tables and 41 sentences.

LEGACY PROGRAMMING LIBRARIES

The sequential nature of the output structure may seem straightforward to begin with, but on closer scrutiny of the templates complications emerge. In some cases the template stipulates that groups of sentences be repeated within a section describing a particular AE. For example, in the R&I template, each of the Adverse Events that are to be discussed is detailed in a block of text consisting of sentences 8 to 22 that is then repeated for each AE.

The programming complexity was further increased by having to mix tabular and textual elements on a page, as well as potentially report subject data both individually (one output document per subject) and in a combined format (all subjects together in one document, with a table of contents).

The 3 different TA code libraries were maintained independently, and often study-specific modifications were added. Over time these libraries had become unwieldy, slow and very hard to transfer between studies. They were also complex to debug when inevitable issues occurred.

In the code I reviewed, the results tended to be stored in separate data sets, one for each table or sentence. Then various programs took these data sets and used hard-coded proc reports to sequentially generate each output element while writing to an ODS destination, one subject at a time. If there was a large number of subjects being reported, then the programs could take a long time to run. Some teams I spoke to suggested it could take over 20 minutes to generate their study narratives, sometimes more, with any programming errors along the way causing the whole process to crash midway though.

While the methods used for data generation seemed logical, the output generation step seemed overly complicated and inefficient.

In an effort to speed up the final reporting stage, I investigated how the different output elements could be created concurrently with the underlying data, and then somehow manipulated into a different order later.

This led me to a much underused area of the SAS Output Delivery System – the Document Store.

ODS DOCUMENT STORES

In the early days of SAS you were very limited in the output you could create. From a clinical study reporting perspective, you could output either flat text files, bitmap image files or vector graphic files…and that was it.

With the introduction of the ODS in SAS v8.0, suddenly SAS could produce a lot of different file formats, the number of which has continued to grow with each subsequent release.

The ODS allows users to open a “destination” file, for example RTF or PDF, and then from that point on any output generated by a SAS procedure (like PROC PRINT, REPORT, TABULATE, SGPLOT, etc) may be sent to the open destinations and rendered appropriately there. When the destinations are closed, the output file is also closed and written to your file system.

One of the destinations provided by SAS is DOCUMENT. This differs from other ODS destinations in that it is a type of internal SAS storage. It effectively allows you to “capture” SAS output and store it in a native format, to render to an output file at a later point.
This allows you to generate the actual report once, but then create output from it multiple times.

For example, you may generate the results for a report but may need to change the fonts after a review, or the report may have been written to a PDF but you are asked later to provide an RTF instead. Normally you would need to rerun your report code after making these formatting changes, which may be a problem if it was generated with live data (and a snapshot wasn’t taken at the time), but if the output was stored in a document store then a new output can be created directly from the document store entry.

In Figure 3 below, the output from a PROC REPORT is first stored in a SAS document store, and then replayed into multiple output files with different ODS styles.

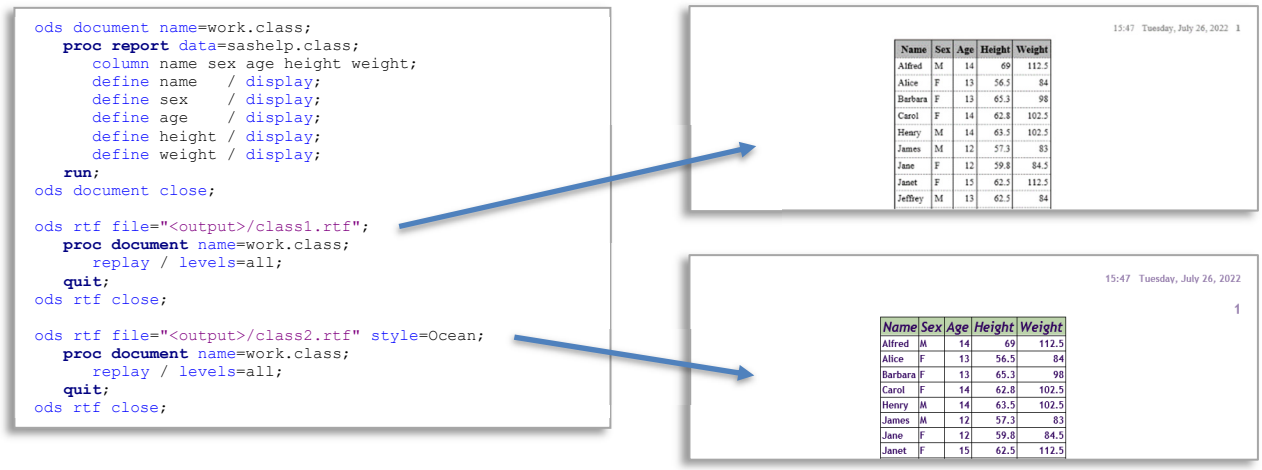


Figure 3 - Recreating output with PROC DOCUMENT

After researching document stores further, two intriguing features came to my attention.

1) Outputs generated using BY-group processing have a separate entry within the store for each BY combination
2) Document stores can store entries within sub-folders

By combining these two features, I designed an approach to narrative generation that allows me to capture output from multiple SAS procedures in SAS document stores, manipulate and reorder these document stores, and then report the results in multiple ways

METHODS

My approach to narrative generation can be broken down into 5 key stages:

1) Create a Master Document Store
2) Prepare the data for each element
3) Output each data element
4) Manipulate document stores
5) Create output file(s)

Each stage is discussed below.

1. CREATE A MASTER DOCUMENT STORE

Here I initialize a fresh ODS document store, and structure it to contain one sub-folder for each subject. As there are definite triggers for deciding when a subject requires a narrative, I already have a list of subjects that can be used to programmatically generate this.

2. PREPARE THE DATA FOR EACH ELEMENT

The elements that form the narratives come in two types: **Table** or **Sentence**.
To prepare the data for tables, I arranged it into a simple structure with one variable per output column, plus a few identifying variables to allow for sorting and by-group processing. Figure 4 below shows an example table data set.

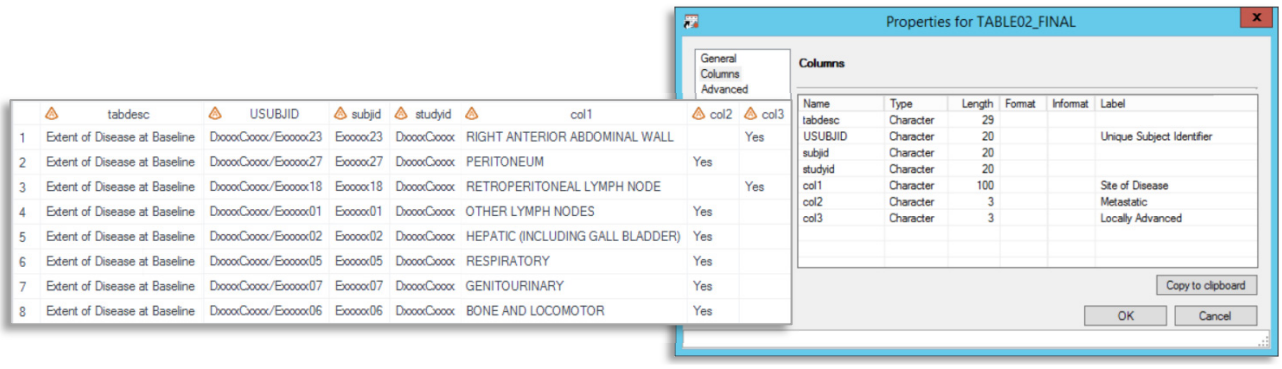


Figure 4 - Example of a data set that is input for a table

For sentences, the data set contains the identifying variables, and then one variable containing the actual text of the sentence. Occasionally there could be multiple variables for when several sentences were strung together into a larger paragraph. Figure 5 below shows an example sentence data set.

tabdesc	USUBJID	subjid	studyid	s01
Medical History	DxxxxCxxxx/Exxxxx23	Exxxxx23	DxxxxCxxxx	Subject Exxxxx23 , a 58 year-old white female from Russian Federation, with Asthma diagn...
Medical History	DxxxxCxxxx/Exxxxx27	Exxxxx27	DxxxxCxxxx	Subject Exxxxx27 , a 52 year-old white male from Russian Federation, with Asthma diagnos...
Medical History	DxxxxCxxxx/Exxxxx18	Exxxxx18	DxxxxCxxxx	Subject Exxxxx18 , a 65 year-old white male from South Africa, with Asthma diagnosed in 1...
Medical History	DxxxxCxxxx/Exxxxx01	Exxxxx01	DxxxxCxxxx	Subject Exxxxx01 , a 39 year-old white male from South Africa, with Asthma diagnosed in 1...
Medical History	DxxxxCxxxx/Exxxxx02	Exxxxx02	DxxxxCxxxx	Subject Exxxxx02 , a 22 year-old black or african american female from South Africa, with A...
Medical History	DxxxxCxxxx/Exxxxx05	Exxxxx05	DxxxxCxxxx	Subject Exxxxx05 , a 63 year-old other race (South African Indian) female from South Africa...
Medical History	DxxxxCxxxx/Exxxxx07	Exxxxx07	DxxxxCxxxx	Subject Exxxxx07 , a 62 year-old black or african american female from South Africa, with A...
Medical History	DxxxxCxxxx/Exxxxx06	Exxxxx06	DxxxxCxxxx	Subject Exxxxx06 , a 47 year-old other race (Colored South African) female from South Afric...

Figure 5 - Example of a data set that is input for a sentence

3. OUTPUT EACH DATA ELEMENT

For both data types (tables and sentences), the resulting data sets conformed to a very specific naming convention and had added metadata (such as the column headers embedded as the variable labels). This allowed the actual output generation to be easily accomplished automatically via a series of macros.
For example, reporting a table involved a call to a macro that specified the input data set, a list of the columns to report and their physical column widths as parameters. The macro would then programmatically generate the PROC REPORT statements and execute them via CALL EXECUTE to create the table output.

For the tabular data PROC REPORT was used for creating output, and for sentences the output was generated with PROC ODSTEXT.

For both procedures, the output was written to a *temporary* ODS document store.

4. MANIPULATE DOCUMENT STORES

Once the output has been captured some manipulation is required to be able to then add it to the Master Document Store (from Stage 1 above).

Output from PROC REPORT was straightforward to manipulate. Because I used by-group processing, I can use the LIST statement in PROC DOCUMENT with the “bygroups” option to give me the subject number that each entry relates to.

The sentence output was harder to manipulate. PROC ODSTEXT does not have a BY statement (it was intended to generate simple blocks of text, for example in PowerPoint or eBooks, rather than report data), so when the results were generated in the previous step it was done so one entry at a time within a loop, with a naming convention on the document store label that included the subject number. PERL regular expressions were then used to extract this from the document path information.

Once each entry in the temporary ODS document store has been assigned a subject number, a simple call to PROC DOCUMENT can be programmatically generated, with RENAME and COPY statements that label them with the element they refer to (table 1, sentence 2, paragraph 3, etc – this is down to the project requirements) and then copy them to the Master Document Store under the correct subject sub-folder.

5. CREATE OUTPUT FILE(S)

The requirements for the final deliverable vary for different study teams. Some require a single RTF document containing all subjects with a contents page, another may want individual RTF files for each subject.

As we have already captured all the output elements and arranged them within an ODS document store, creating different numbers of files and/or layouts is simply a matter of manipulating the document store further as required, and then using PROC DOCUMENT with the REPLAY statement to render the results to whatever ODS destination we want.

Figure 6 below shows the first page of an example narrative output, rendered as a Microsoft Word document.

Study: DxxxxCxxxxx
Safety Population

Exxxxxxx Narrative

Subject Details

Study:	DxxxxCxxxxx
Country:	Spain
E code:	Exxxxxxx
Age/Sex/Race:	52 years/F/White/Not Hispanic or Latino
Baseline Performance Sataus:	0
Consent Date:	01 MAY 2017 (-5)
Randomisation Date:	06 MAY 2017 (1)
Treatment Arm:	AZD9291
Investigational Product (IP):	IP: Newdrug 10mg BD
Days[a] on IP:	901
Study Date[a] (Day) of Start IP:	07 MAY 2017 (2)
Study Date[a] (Day) of Stop IP:	06 NOV 2019 (914)
Study Date[a] (Day) Decision Made to Discontinue IP:	Not applicable
Reason for IP Discontinuation:	Not applicable
Study Date[a] (Day) of Death:	
Cause of Death (Primary):	
Cause of Death (Secondary):	
Serious Adverse Event(s):	Gastroenteritis and Ureterolithiasis
AE Leading to Discontinuation:	Not applicable

NOTE: No Extent of Disease at Baseline Reported for this subject

General Pathology

Study Date[a] (Day) of Initial Diagnosis	Primary Tumor Location	Histology Type	Histology Type Details	Primary Tumor	Regional Lymph Nodes	Distant Metastases	AJCC Stage at Initial Diagnosis[b]
15 FEB 2017 (-154)	LUNG	ADENOCARCINOMA, MALIGNANT		T1B	N2	M0	STAGE IIIA

[a] Relative to start of fisrt dose of study medication
[b] to be adapted based on tumor indication

Prior Anti-Cancer Therapy

Start Date[a] (Day) / Stop Date[a] (Day)	Cancer Therapy Agent Reported Term (Preferred Term)	Therapy Class	Treatment Status	Reason for Therapy	Number of Cycles	Route	Best Response
07 APR 2017 (-103) / 21 JUN 2017 (-28)	PEMETREXED (PEMETREXED)	CYTOTOXIC CHEMOTHERAPY	ADJUVANT	Lung Cancer	4	INTRAVENOUS	NOT APPLICABLE
07 APR 2017 (-103) / 21 JUN 2017 (-28)	NEDAPLATIN (NEDAPLATIN)	CYTOTOXIC CHEMOTHERAPY	ADJUVANT	Lung Cancer	4	INTRAVENOUS	NOT APPLICABLE

[a] Relative to start of fisrt dose of study medication

Subject Exxxxxxx, a 52 year-old white female from Spain who participated in protocol DxxxxCxxxxx, a double blind parallel study in subjects with Non-small Cell Lung Carcinoma. The subject consented on 01 MAY 2017 and was randomized to Newdrug.

Subject reported no past medical history. Concurrent medical condition(s) included Acute sinusitis from 09 JAN 2017, Ventricular extrasystoles from JUL 2016 and Hepatic steatosis from 02 JAN 2017.

The subject was a non-smoker. At study entry (screening) the subject had stage IIIA Non-small Cell Lung Carcinoma at initial diagnosis. The subject had a WHO performance status of 0 at study entry.

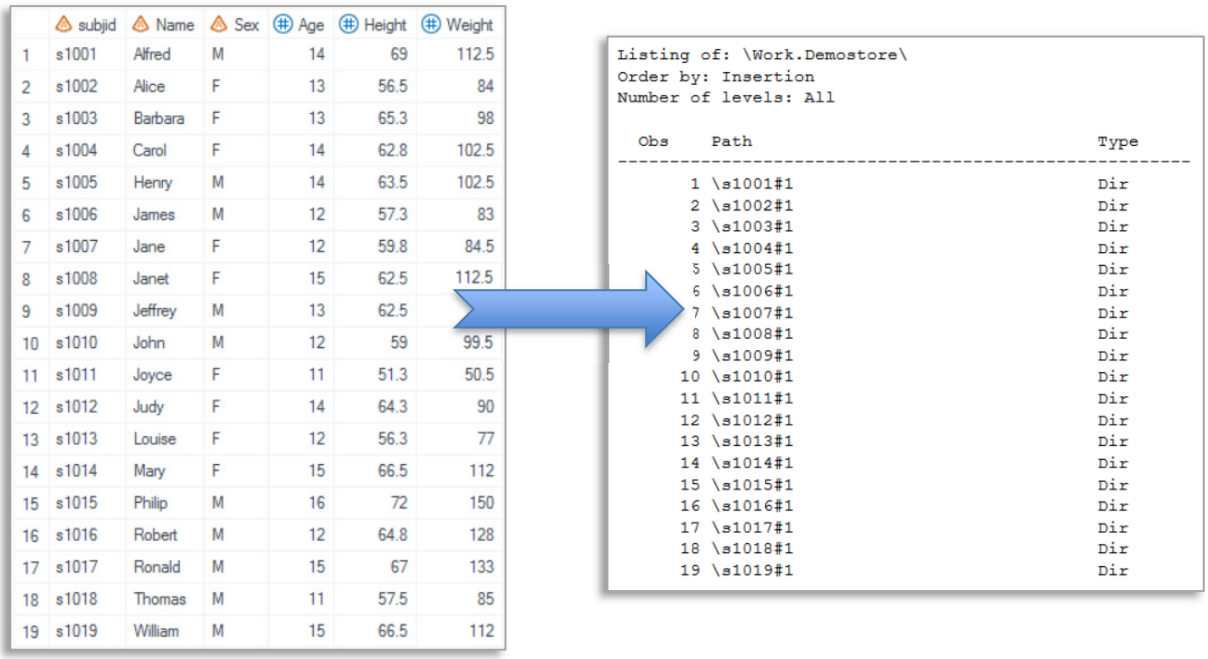
Treatment with Newdrug was started on 07 MAY 2017 and the subject was receiving 10 mg by oral BD. The last dose of Newdrug prior to Binocular Cataract [Cataract] onset was on 20 JAN 2019.

Figure 6 - Example of a final narrative rendered with ODS WORD

WORKED EXAMPLE

The appendix of this paper contains a complete program to demonstrate all the concepts I have discussed. All references to line numbers below refer to this example program. It takes the SAS-supplied SASHELP.CLASS data set and creates a narrative containing 3 tables for each "subject" within a single document.

Figure 7 below shows SASHELP.CLASS with a dummy subject number applied, and then the hierarchy of the Master Document Store created from it at **Line 037**.



	subjid	Name	Sex	Age	Height	Weight
1	s1001	Alfred	M	14	69	112.5
2	s1002	Alice	F	13	56.5	84
3	s1003	Barbara	F	13	65.3	98
4	s1004	Carol	F	14	62.8	102.5
5	s1005	Henry	M	14	63.5	102.5
6	s1006	James	M	12	57.3	83
7	s1007	Jane	F	12	59.8	84.5
8	s1008	Janet	F	15	62.5	112.5
9	s1009	Jeffrey	M	13	62.5	
10	s1010	John	M	12	59	99.5
11	s1011	Joyce	F	11	51.3	50.5
12	s1012	Judy	F	14	64.3	90
13	s1013	Louise	F	12	56.3	77
14	s1014	Mary	F	15	66.5	112
15	s1015	Philip	M	16	72	150
16	s1016	Robert	M	12	64.8	128
17	s1017	Ronald	M	15	67	133
18	s1018	Thomas	M	11	57.5	85
19	s1019	William	M	15	66.5	112

Figure 7 - SASHELP.CLASS and the resulting Master ODS Document Store

Figure 8 below shows the raw SAS Report output from the 3 PROC REPORT calls at **Lines 051**, **062** and **072**.

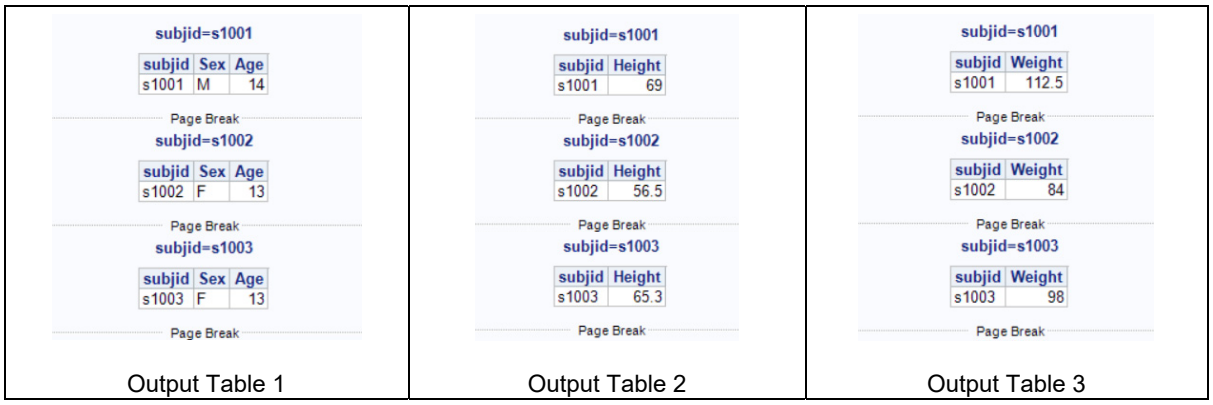


Figure 8 - Example output tables

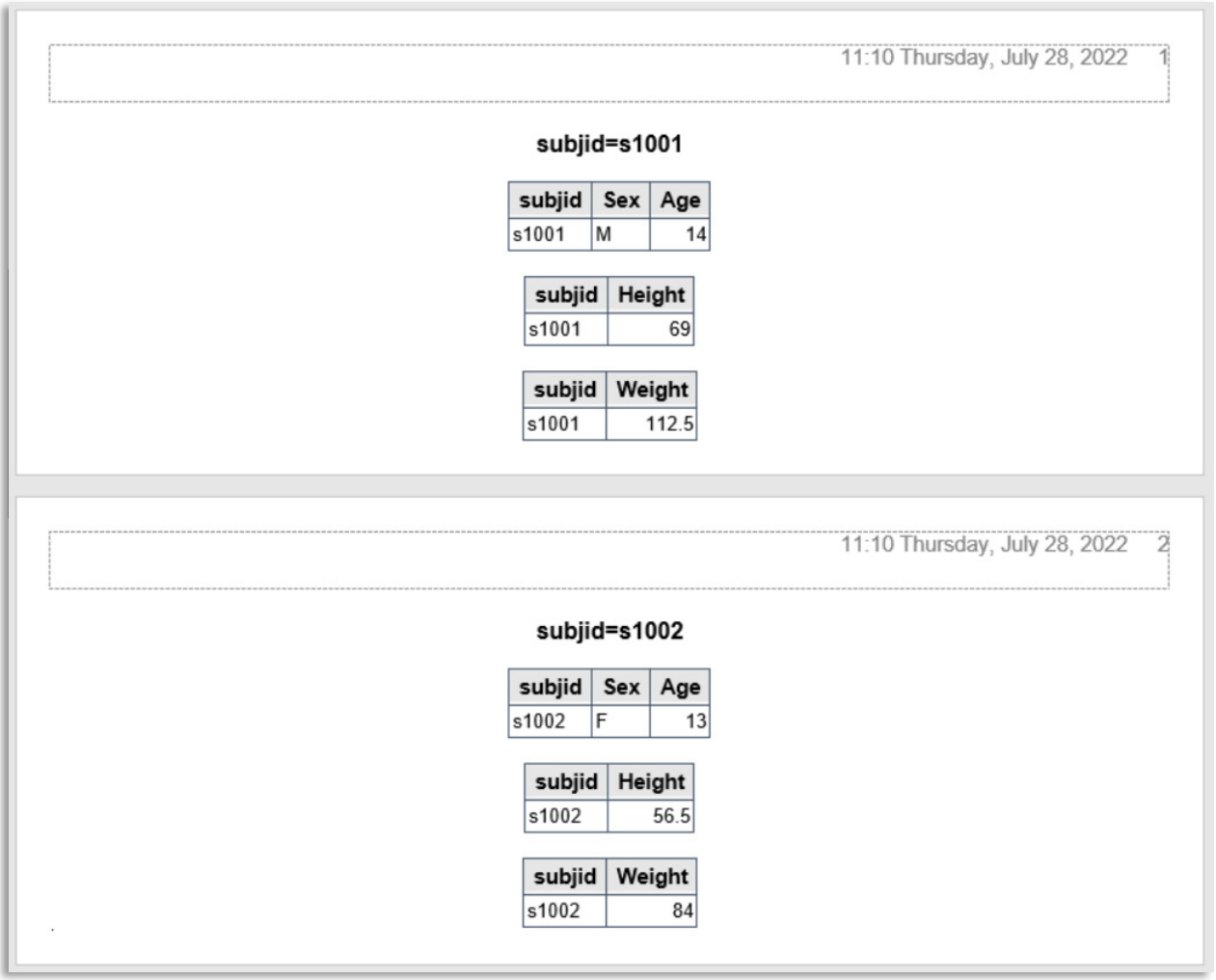
The data set MOVE_TABLE (**Line 088**) creates the PROC DOCUMENT code shown in Figure 9 below. This will rename the entries from the default path with the "Report#1" label to something more meaningful (in this case "table1") and then copy it to the Master Document Store.

```
proc document name=work.temp_table1;
   rename \Report#1\ByGroup1#1\Report#1\Report#1 to table1 ;
     copy \Report#1\ByGroup1#1\Report#1\table1 to \work.demostore\s1001#1 ;

   rename \Report#1\ByGroup2#1\Report#1\Report#1 to table1 ;
     copy \Report#1\ByGroup2#1\Report#1\table1 to \work.demostore\s1002#1 ;

   rename \Report#1\ByGroup3#1\Report#1\Report#1 to table1 ;
     copy \Report#1\ByGroup3#1\Report#1\table1 to \work.demostore\s1003#1 ;
   <MORE STATEMENTS>
quit;
```

Figure 9 - Example document store manipulation

Please note the strange ordering of the %process_table calls at **Lines 118-120** was deliberate, to show that the order in which the ODS DOCUMENT entries are created is unimportant because they would be re-ordered in the master template store later (at **Line 164**)

Finally, the demonstration program creates a single MS WORD document (**Line 186**), as shown in Figure 10 below (with vertical white space removed for clarity)



11:10 Thursday, July 28, 2022 1

subjid=s1001

subjid	Sex	Age
s1001	M	14

subjid	Height
s1001	69

subjid	Weight
s1001	112.5

11:10 Thursday, July 28, 2022 2

subjid=s1002

subjid	Sex	Age
s1002	F	13

subjid	Height
s1002	56.5

subjid	Weight
s1002	84

Figure 10 - Final example output

LESSONS LEARNED

I found the redesigning of the narratives programming a very interesting, and ultimately very rewarding, experience. However, it was not as straightforward as I had anticipated!

After I had completed the first TA (which was all table elements), the next TA had a different approach (virtually all textual elements) that meant I had to write new routines, and ended up becoming a separate macro library. The final TA was a mixture of both approaches, with text and tabular elements and an additional looping mechanism to allow groups of elements to be repeated (e.g. for each AE or event a subject may have had), at which point I consolidated all the macro libraries into a single code base that should satisfy all eventualities moving forward.

The entire process would have been smoother had I worked on combining all TAs together from the beginning – although part of the reason for the fragmentary nature of my work was due to the Project Lead leaving the company early on, causing some disruption. Additionally, some of the TAs were having their specifications reviewed and amended while I was working on the macros and example code, so there was some rework required.

It's also important to reiterate the importance of getting hold of detailed specifications and thoroughly analysing them at the start of a project like this. While adapting the code as new features came to light was reasonably straightforward in my case, being able to properly map out how everything would work together before getting too deep into the programming would have saved me a lot of time.

CONCLUSION

This method of constructing Subject Narratives has proved to be very useful in my redesigning of the process over the last year. I originally performed a Proof of Concept using dummy data, and then moved onto developing the macros for each TA, before consolidating them into a single macro library.

After the initial steep learning curve around understanding ODS Document Stores, and the development of a handful of macros to make the processing of them painless, they proved to be a fast and efficient way to create and manipulate all of my narratives output.

If you find yourself generating large and complex outputs in some sort of loop (repeating by subject, some event or perhaps by a lab parameter) that appears to take a long time at the reporting stage, then it may be worth looking into ODS Document Stores and PROC DOCUMENT to see if they can speed up your workflow.

CONTACT INFORMATION

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

Nev Cope
AstraZeneca
Academy House, 136 Hills Road, Cambridge, CB2 8PA

Email: neville.cope@astrazeneca.com

APPENDIX

Demonstration code for all the techniques discussed in this paper is presented below.
Please update the "`%let outfile=`" statement at **Line 001** as appropriate.

```
001  %let outfile=<OUTPUT FOLDER>/demo.docx;
002
003  PROC DATASETS LIB=WORK KILL MEMTYPE=(ITEMSTOR DATA) ;
004  RUN ;
005
006
007  *----------------------------------------*
008  | Stage 1 - Create master document store |
009  *----------------------------------------*;
010
011  *--- Process raw data ---;
012  data class;
013     length subjid $8.;
014     set sashelp.class;
015     subjid=cats("s10", put(_n_, z2.));
016  run;
017  proc sort data=class;
018     by subjid;
019  run;
020
021  data create_master;
022     set class end=eof;
023     length exec $100;
024     if _n_=1 then do;
025        exec="proc document name=work.demostore;";
026        output;
027     end;
028     exec=catx(" ", "make", subjid, ";");
029     output;
030     if eof then do;
031        exec="quit;";
032        output;
033     end;
034  run;
035  data _null_;
036     set create_master;
037     call execute(exec);
038  run;
039
040  *----------------------------------------*
041  | Stage 2 - Prepare data for each element |
042  *----------------------------------------*;
043  *** SASHELP.CLASS is used in this example ***;
044  *** In a real study specific data sets would be created here ***;
045
046  *-------------------------------------*
047  | Stage 3 - Output each data element |
048  *-------------------------------------*;
049
050  *--- Create demo table 1 ---;
051  ods document name=work.temp_table1 (write);
052     proc report data=class;
```

```
053         by subjid;
054         column subjid sex age;
055         define subjid / display;
056         define sex    / display;
057         define age    / display;
058      run;
059   ods document close;
060
061   *--- Create demo table 2 ---;
062   ods document name=work.temp_table2 (write);
063      proc report data=class;
064         by subjid;
065         column subjid height;
066         define subjid / display;
067         define height / display;
068      run;
069   ods document close;
070
071   *--- Create demo table 3 ---;
072   ods document name=work.temp_table3 (write);
073      proc report data=class;
074         by subjid;
075         column subjid weight;
076         define subjid / display;
077         define weight / display;
078      run;
079   ods document close;
080
081   *--- Move table to master document store ---;
082   %macro process_table(tabnum=);
083      ods output properties=tables_by;
084      proc document name=work.temp_table&tabnum.;
085         list / levels=all bygroups;
086      quit;
087
088      data move_table;
089         set tables_by end=eof;
090         length exec $1000;
091         regex=prxparse('/(.+ByGroup\d+\#1\\)(.+\\)(.+)/');
092         istable=prxmatch(regex, path);
093         toppath=prxposn(regex, 1, path);
094         midpath=prxposn(regex, 2, path);
095         curr_name=prxposn(regex, 3, path);
096         *--- build commands ---;
097         if _n_=1 then do;
098            exec="proc document name=work.temp_table&tabnum.;";
099            output;
100         end;
101         newpath=cats("\work.demostore\", subjid, "#1");
102         rename=catx(" ", cats(toppath, midpath, curr_name), "to", "table&tabnum.");
103         copy=catx(" ", cats(toppath, midpath, "table&tabnum."), "to", newpath);
104         exec=catx(" ", "rename", rename, ";"
105                      , "copy", copy, ";"
106                   );
107         output;
108         if eof then do;
109            exec="quit;";output;
110         end;
```

```
   run;

   data _null_;
      set move_table;
      call execute(exec);
   run;
%mend process_table;
%process_table(tabnum=2);
%process_table(tabnum=3);
%process_table(tabnum=1);


*-----------------------------------------*
| Stage 4 - Manipulate ODS document stores |
*-----------------------------------------*;

*--- Read document store contents ---;
ods output properties=results;
proc document name=work.demostore;
   list / levels=all bygroups;
quit;

data order;
   set results end=eof;
   by subjid;
   retain counter;
   if _n_=1 then counter=0;
   if first.subjid then counter+1;
   if eof then call symput("numpats", counter);
   check=prxparse("/\\(s\d+#1)\\table([0-9])#\d/");
   if prxmatch(check, path) then tabnum=prxposn(check, 2, path);
run;

proc sort data=order;
   by subjid tabnum;
run;


*--- Remove some page breaks and reorder ---;
data repagination;
   set order end=eof;
   by subjid;
   if not first.subjid then delpgbrk=1;
   length string $1000;
   *--- build commands ---;
   if _n_=1 then do;
      string="proc document name=work.demostore;";
      output;
   end;
   if delpgbrk=1 then do;
      string=catx(" ", "obpage", path, "/ delete;");
      output;
   end;
   string=catx(" ", "move", path, "to", cats("\", subjid, "#1"), "/ last ;");
   output;
   if eof then do;
      string="run;";output;
   end;
```

```
run;

data _null_;
   set repagination;
   call execute(string);
run;

/*ods output properties=results2;
proc document name=work.demostore;
   list / levels=all bygroups;
quit;
*/

*------------------------------*
| Stage 5 - Create output file |
*------------------------------*;

ods word file="&outfile.";
   proc document name=work.demostore;
      replay / levels=all;
   quit;
ods word close;
```