PharmaSUG 2017 - Paper HT07

Single File Deliverables: Next Steps

William Coar, Axio Research, Seattle, WA

ABSTRACT

Creating Tables, Listings, and Figures (TLFs) continues to be part of daily tasks in a statistical programming group in the Pharmaceutical Industry. Trends continue to move toward electronic review and distribution, and the end users of the TLFs are often requesting a single file rather than sending each output individually.

In the recent years, the use of item stores has been introduced as a viable option to create a single file deliverable. In 2016, a hands on training was offered to demonstrate their usefulness in our industry setting. During this workshop (Combining TLFs into a Single File Deliverable, PharmaSUG 2016) attendees gained experience with creating, replaying, and restructuring item stores to obtain a single file containing a set of TLFs. Single File Deliverables: Next Steps will build upon what was presented in 2016 working towards more complex implementation and automation.

After a brief refresher, Next Steps will focus on combining and restructuring item stores to obtain a single and well-structured document with hyperlinks/bookmarks for navigation. The use of by-group processing will also be introduced. Throughout the workshop it will become apparent that automation is extremely desirable. Basic automation steps will be introduced to demonstrate an application utilizing macro programming.

This Hands-on-Training will build upon what was introduced in 2016. It will assume that users have a (very) basic understanding of Proc Report, the SG Procedures, item stores, and macros. The use of ODS is required in this application using SAS® 9.4 in a Windows environment.

INTRODUCTION

Although the setting for the application presented may be quite specific, it is not uncommon in the pharmaceutical industry. Individual programs generate one or more TLFs through an Output Delivery System (ODS) destination such as RTF or PDF. These programs are developed over time, and submitted in batch mode for the production version of the reports. The final procedure for all summary tables and listings is Proc Report. Figures result from one of the SG procedures.

In this setting, all outputs tend to be in individual files. As trends move towards electronic review and distribution, a single file structured with hyperlinks and/or bookmarks is desirable. Many options have been proposed that post-process individual RTF or PDF files ([1], [2], [3], [4],[5],[6]). The focus of the 2016 hands-on-training was to develop a basic understanding and learn how to implement the technique of using ODS Document and item stores to obtain a single file deliverable. The focus of this Hands-On-Training is expand upon what was presented in 2016.

A brief review will cover the replay process presented at PharmaSUG 2013 [7]. A series of examples that create, replay, and modify item stores using Proc Report and Proc SGrender will follow. Item stores resulting from by-group processing will be introduced. The final examples will demonstrate that automation is feasible without complication, and in fact preferable. The paper will conclude with a brief summary.

REVIEW OF ITEM STORES & THE REPLAY APPROACH

This two-step process in Figure 1 was proposed in [7] for the creation of a single document that contains a set of TLFs with a hyperlinked table of contents and/or bookmarks. The first step of the process uses ODS Document to create item stores for each individual TLF. Recall that for each individual TLF, there is an existing block of ODS statements to create an RTF or PDF. If the additional ODS Document block is added, accompanied by (minor) program updates for style points, an item store will also be created when each program is executed. ***Item stores hold the data and instructions from the procedure used to create the report***. Just as one obtains a PDF or RTF, the program also contains an additional file: the item store.

The second step uses Proc Document to combine, restructure, and replay these item stores into a single document within an ODS destination. Since all the item stores are replayed within a single ODS destination, the result is a single file that contains all the reports that encompass the TLFs.

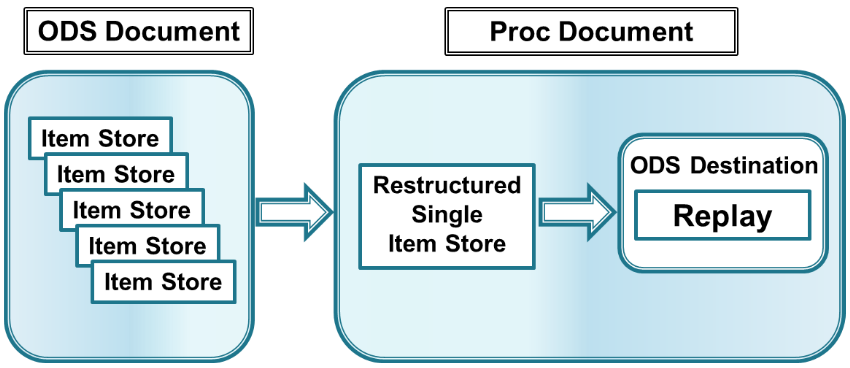


Figure 1 Replay Approach

The discussion presented here focuses on item stores created from Proc Report and SG procedures. For a more general discussion of item stores, see [10].

REVIEW OF CREATING ITEM STORES

While it is common (and reasonable to assume) for Proc Report to be used for tables and listings, it may not be common for figures to be created using Proc SGRender. These techniques apply to all SG procedures, but the author has found there are known issues with (not) displaying titles/footnotes with some SG procedures when the item store is replayed to the PDF destination. While this may seem like a rigid constraint, it is manageable since some SG procedures have an option to output the Graph Template Language (GTL).

Tables and Listings

As stated above, it is easy to create an item store that houses the data and instructions from Proc Report (or Proc SGRender) that can be replayed at a later time. The initial step is to create an ODS Document sandwich that includes all of the desired procedure. **Sample Code 1** below that would create a demographics summary table highlights the statements required around a typical Proc Report.

After adding the ODS Document statements (ie, the ODS Document sandwich), the next step is to add the CONTENTS= option within Proc Report to specify the desired text for the hyperlink or bookmark. The next step is slightly more tedious. The input dataset to Proc Report needs a flag variable that is constant for every record in the data. This flag variable is defined as an ORDER variable with a NOPRINT option. Finally, a BREAK BEFORE statement is added with a necessary contents='' option. Assuming the input dataset has a variable named flag which is set to 1 for every record, **Sample Code 1** shows the necessary updates to Proc Report to obtain more meaningful bookmarks and hyperlinks.

```
ODS Document name=istore.rdemog (write);

ODS rtf file ="rdemog.rtf" style=AxioConslt;

proc report data=tns_rpt center headline headskip nowindows missing contents='Table 1:
Demographics';

       column flag sv1 sv2 row1 _1 _2 _3;

       define flag /order order=data noprint ;

Other define statements

break before flag / page contents='';

run;

ODS RTF close;
```

Sample Code 1 Updates to Proc Report

The result of the ODS Document sandwich is an item store named ***rdemog*** that is stored in the location defined by the libname ***istore***. Notice the use of the option (write). This instructs SAS to recreate the item store if it already exists. Without the use of (write), SAS will automatically add to the existing item store, which may happen if there are multiple executions of the SAS code.

When creating a ToC after these modifications, the resulting output will have hyperlinks/bookmarks that say “Table 1: Demographics” rather than a generic “Table 1”. The ToC will continue to have a level label “The Report Procedure”. However, this will be ignored later when the Read and Restructure phase of the Replay Approach is reviewed.

The exercises of this Hands-on-Training are meant to allow the user to gain experience in these areas for outputs generated from Proc Report. To this point, creation of item stores for Proc Report (tables and listings) will be reviewed in Exercise1,

The objectives of Exercise 2 are to review the internal structure of an item store and to provide an example of replaying the item store. Recall that the item store is a SAS file with its own internal structure only readable by SAS. Thus, to see what is inside of the item store, we must rely on a SAS procedure: Proc Document. **Sample Code 2** below provides the SAS code necessary to see what is inside of the newly created item store.

```
proc document name=istore.rdemog (read);
       list / levels=all;
run;
quit;
```

Sample Code 2 Item store from Proc Report

The results are shown in **Output 1**, where we see the item store created when using Proc Report consists of directories and a table. While this may not tell us much about what is ***really*** inside of the item store (you need to replay it for that), it provides us with key information about its structure: it consists of directories and a table. In fact, this structure from Proc Report is very predictable, which will become important in the later examples of this workshop when we seek to combine individual item stores.

Listing of: \Istore.Rdemog\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table

Output 1 Item store from Proc Report

To regenerate the report, a programmer needs to actually replay the item store. To do so requires the use of Proc Document with a REPLAY statement, as seen in **Sample Code 3**.

```
ODS PDF file ="rdemog_replay.pdf" style=AxioConslt;

proc document name=istore.rdemog (read);
       Replay Report#1l;
run;
quit;

ODS PDF close;
```

Sample Code 3 Replay a report from an item store

Since the Proc Document and REPLAY statement are within an ODS PDF sandwich using the same ODS style, the resulting PDF (named Example_2.pdf for demonstration) will be identical to the original PDF assuming no other options have changed.

Tip #1: Item stores are not specific to ODS destination or ODS style. Item stores can be replayed into different ODS destinations using different styles. These files may look different, but the contents displayed will be the same.

Tip #2: The (read) option is used in the Proc Document so the item store itself remains unchanged by Proc Document. Other options such as (write) and (update) are used to modify an existing item store using Proc Document.

While a Table of Contents (ToC) and/or bookmarks may not be useful for a single table, it is reasonable to introduce them at this time. Sample ODS RTF and ODS PDF statements below in **Sample Code 4** show the use of options that will create the desired hyperlinked ToC in RTF and bookmarks in PDF.

```
ODS RTF file ="rdemog_replay.rtf" style=AxioConslt contents=on toc_data;

ODS pdf file ==rdemog_replay.pdf" style=AxioConslt pdftoc=2 contents=on;
```

Sample Code 4: ODS statements with options for ToC

When initially opened in Word, the resulting RTF will contain what appears to be an empty ToC on the first page, with the table starting on page 2. The ToC exists, but it needs to be populated. There are a number of ways to do so. One method is to select the ToC by using Control->A, then hit the F9 key. Other methods exist, such as right clicking on a ToC and selecting “update field”.

The page numbering may also need a minor update. The page on the first table may continue to say Page 1 even though it is not the first page of the document. If this is desired, the user simply needs to update the attributes of the page numbering to be *continue from previous section*.

When opening the PDF, the user will find there are 2 levels in the bookmarks. The second level actually points to the table. Since the file opens with the bookmarks pane open, it may not be necessary to have bookmarks and a ToC. A ToC within PDF does not work well when figures are in the document, thus in general, should be avoided.

Tip #3: Due to inherent differences between RTF and PDF, you may notice differences in spacing and line breaks when an item store is replayed to both destinations. While the outputs may look slightly different, the contents displayed are the same.

Figures

As noted above, the application presented relies on Proc SGRender to create figures. Once the GTL code is obtained, it is easily modified to allow the program to first create a template from GTL, and then create the figure by using Proc SGRender. This also allows for opportunities to provide custom figures without extensive knowledge of GTL.

Tip #4: The SG Procedures and replaying item stores from an SG procedure don’t always play well with PDF. When using PDF, there are known issues where titles and footnotes are not displayed. Furthermore, use of the ODS escape ~LASTPAGE will result in a PDF file without a graph.

Tip #5: The size of a figure in PDF may be reduced depending on the number of titles and footnotes when using SAS Version 9.4. The author suggests using the DESIGN size options in GTL to maximize the size of the graph for RTF destination. Once replayed in PDF, the size of the image may be more reasonable.

As with tables and listings, modifications to SAS code are required, but manageable. The first is to obtain GTL for the figure in preparation to use Proc SGRender. *In doing so, the GTL must be stored in a permanent template store so it is available at a later point in time when the item store is recalled.* **Sample Code 5** shows the necessary update to the GTL code so that the GTL is stored in a permanent template.

```
ods path (prepend) istore.templat (update);

proc template;

   define statgraph ....;
```

Sample Code 5 Storing GTL in a permanent template store

There are a few noteworthy additions to the SAS code in **Sample Code 5.** The first is the use of (prepend) in the ODS path statement, which allows the user to simply add a new location to the beginning of the existing ODS search path. The second is the use of (update). This instructs SAS to save the subsequent graph template in a permanent catalog. If the catalog exists, SAS will simply update it. If the catalog does not exist, then SAS will create it. Be sure to use naming conventions that allow unique graph template names (such as program names). All will be stored in the same permanent catalog in a folder defined by the libname istore.

Tip #6: While it may be acceptable to use dashes (-) in file names for SAS programs, dashes are not acceptable for naming item stores or templates.

Remember, the primary reason for storing the GTL in a permanent catalog is that it needs to exist later when the item store is replayed.

As with Proc Report, a minor update is needed to obtain informative hyperlinks and bookmarks. **Sample Code 6** provides example code of the updates required to create an item store for figures created using Proc SGRender.

```
ODS PDF file="fcumenrl.pdf"  style=AxioConslt nogfootnote nogtitle notoc ;

ODS Document name=istore.fcumenrl (write);

proc sgrender data=rpt template=fcumenrl description="Figure 1: Cumulative
Enrollment";

run;

ODS Document close;
```

Sample Code 6 Updates to Proc SGRender

Note that with Proc SGRender, the DISCRIPTION= option is used rather than CONTENTS= as in Proc Report. Also see that an item store is created simply by making an ODS Document sandwich.

The same code provided in **Sample Code 2** can be used to see what is inside of the item store resulting from Proc SGRender, with a minor update to change the name of the item store. Once this code is executed, you will see the structure of an item store created using Proc SGRender consisting of one graph.

More on Figures

In the pharmaceutical industry, it is common to provide figures of laboratory parameters or vital sign parameters over time. The resulting figure is typically in a single RTF (or PDF) file with one page per parameter. Each page may have customization with respect to the parameter, such as labeling of the y-axis. A “one-at-a-time” approach can be used to obtain such a figure where a macro is called once for each parameter. Since the macro calls are all within an ODS RTF (or PDF) sandwich, they all end up in a single file.

This concept extends to ODS Documents and item stores as well. To this point, there has only been a single table or graph in each item store. However, this is not a limitation. Item stores can contain multiple tables and/or graphs. The following example considers a lab figure over time where there are multiple lab parameters of interest (4 for demonstration). Consider a single item store created in manner where a graph is created for each parameter within a single ODS sandwich. This results in an item store with multiple graphs. Using techniques in **Sample Code 2** pointing to **name=istore.flabchem (read)**, we see the item store looks like:

Listing of: \Istore.Flabchem\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\SGRender#1	Dir
2	\SGRender#1\SGRender#1	Graph
3	\SGRender#2	Dir
4	\SGRender#2\SGRender#1	Graph
5	\SGRender#3	Dir
6	\SGRender#3\SGRender#1	Graph
7	\SGRender#4	Dir
8	\SGRender#4\SGRender#1	Graph
…	…	…

Output 2 Item store from Proc SGRender with 4 Images

Note here that this item store consists of *directories* and *graphs* whereas those from Proc Report consisted of *directories* and *tables*. In both cases, the structure of the item store is predictable. This becomes important during the read and re-structure process of creating the single document.

The item store in **Output 2** was generated for a custom lab figure to assess mean(SD) over time. Although the figure was created in a single RTF file, there were actually 21 images, one for each of 21 laboratory parameters of interest. For illustration, only 4 were included in **Output 2**.

For simpler figures that fit on a single page, the item store would only have a single directory and graph, both labeled Sgrender#1\SGRender#1. For graphs that might have one image per page, we find a predictable structure in the item store: a directory and graph for each image labeled sequentially.

Based on the above examples, it is clear that the proposed process of working with item stores deals with *directories*, *reports*, and *graphs*. Although not immediately obvious based solely on **Output 2,** it is seen that SAS (predictably) increments as additional directories and reports (tables/figures) are added. ***This incrementing is essential when automating the creation of a single restructured report.***

To this point, creation of item stores for Proc SGRender (figures) has been reviewed. Modifications required for obtaining a hyperlinked ToC and informative hyperlinks/bookmarks have been presented. Exercise 3 in the Hands-on-Training is meant to allow the user to gain experience in viewing item stores that have multiple graphs within them as well as replaying only select components of the item store.

READ, RESTRUCTURE, AND REPLAY ITEM STORES

Now that the user has experience with creating, reading, and replaying item stores, the next step in the application is referred to as **read and restructure**. We recall that a single item store can have multiple tables and graphs. While simply reading in multiple tables and graphs into a single item store may give you a single file deliverable, it will not yield the desired structure in a Table of Contents or Bookmarks. The structure we wish to obtain allows us to have the TLFs in the single file grouped in a meaningful way. For example, we may wish to have all of the Adverse Events displayed together, followed by the display of all Laboratory reports.

Based on the preceding examples, replaying the contents of the item store should be straightforward using the REPLAY statement. The following discussion focuses on reading and restructuring to obtain a *single item store with the contents of the entire deliverable*. When this is replayed within an ODS sandwich to either RTF or PDF destination, the result is a single document that has all the TLFs with a hyperlinked Table of Contents and bookmarks of the desired structure.

The details of reading in and restructuring item stores in general can be found in Lawhorn [5]. A more applicable discussion can be found in [7], [8], [9] and [12], the predecessors of this Hand-On-Training.

Exercise 4 is designed to demonstrate the use of COPY, MOVE, DELETE, and RENAME statements. In this workshop it is done through interactive execution of a single Proc Document call. Similar to other SAS procedures

```
proc document name=work.app4(write);
    copy  \istore1.raeov_closed\Report#1 to ^;
    copy \istore1.rae_closed\Report#1 to ^;          Step 1
    list     / levels=all;
    run;
   *****************************************************;
    move \report#2\report#1 to Report#1;
    delete Report#2;                                  Step 2
    list     / levels=all;
    run;
    *****************************************************;
    copy \istore1.rlabcwrst_closed\Report#1 to ^;
    list     / levels=all;                            Step 3
    run;
    *****************************************************;
    copy \istore1.flabchem_closed\SGRender#1 to ^;
    list     / levels=all;                            Step 4
    run;
    *****************************************************;
    rename sgrender to report;
    list     / levels=all;                            Step 5
    run;
    *****************************************************;
    move \Report#4\SGrender#1 to Report#3;
    delete Report#4;                                  Step 6
    list     / levels=all;
    run;
quit;
```

Sample Code 7: Interactive Execution of Proc Document

(such as Proc SQL), Proc Document will keep executing until SAS finds a Quit statement. This allows a user to interactively work through the read and restructuring step by step to gain a better understanding of the technique.

The Sample Code above shows that the user can view (results) while the read and restructure process is underway. Each step is described below.

Step 1: The first step reads in two item stores, both from Proc Report (we know this by the file naming convention).The combined item store now consists of directories labeled Report#1 and Report#2. Note that these are simply labels. Observations 1 and 4 tell us there are two parent directories. Each parent directory contains a single table.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#2	Dir
5	\Report#2\Report#1	Dir
6	\Report#2\Report#1\Report#1	Table

Output Step 1

As is, the structure of the Table of Contents will have 2 sections since there are two parent directories. Since these two tables are related, it is desirable to have them in the same section of the ToC. This is done through Step 2.

Step 2: The MOVE and DELETE command are used to move the essential information for the second table (observations 5 and 6) to the parent directory Report#1. The remaining record (observation 4) is no longer needed so it is removed with the DELETE command. The resulting combined item store is see below.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table

Output Step 2

At this point, the combined item store has one parent directory (Report#1) with two tables.

Step 3: Suppose there is interest in reading in a laboratory summary table. As done in steps 1 and 2, this is done through a simple COPY command. The structure of the item store is shown below.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table

Output Step 3

Note the sequential incrementing. Since this is the third item store with the name Report, SAS automatically labels the parent folder Report#3. Given that this lab summary table is different than adverse events, it may be desirable to group lab data together, and keep them in a section separate from the adverse events. As long as any subsequent lab tables or figures are contained in the parent folder Report#3, all laboratory outputs will be in the same section of the final document.

Step 4: When reading in a single lab figure (or one single image within an item store with multiple images), the COPY command is again used. Note however, that the resulting figure is in its own parent directory SGRender#1.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table
9	\SGRender#1	Dir
10	\SGRender#1\SGRender#1	Graph

Output Step 4

Step 5: To move the graph in SGRender#1 to the parent folder Report#3, first RENAME SGRender to Report. The result of the RENAME statement is shown below.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table
9	\Report#4	Dir
10	\Report#4\SGRender#1	Graph

Output Step 5

Notice that SAS automatically increments to Report#4 with the RENAME statement.

Step 6: At this point, the familiar MOVE and DELETE statements move the graph from Report#4 to parent directory Report#3. The directory for Report#4 (observation 9) is deleted. The end resulting is now a combined item store with two sections: adverse events and laboratory data.

Listing of: \Work.App4\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#3	Dir
7	\Report#3\Report#1	Dir
8	\Report#3\Report#1\Report#1	Table
9	\Report#3\SGRender#1	Graph

Output Step 6

While it may be possible to skip the RENAME statement and just COPY or MOVE the graph to Report#3, it may be desirable to use the RENAME statement for consistency of COPY/MOVE/DELETE blocks. The RENAME helps especially in automation as the key blocks of code are COPY/RENAME/MOVE/DELETE commands.

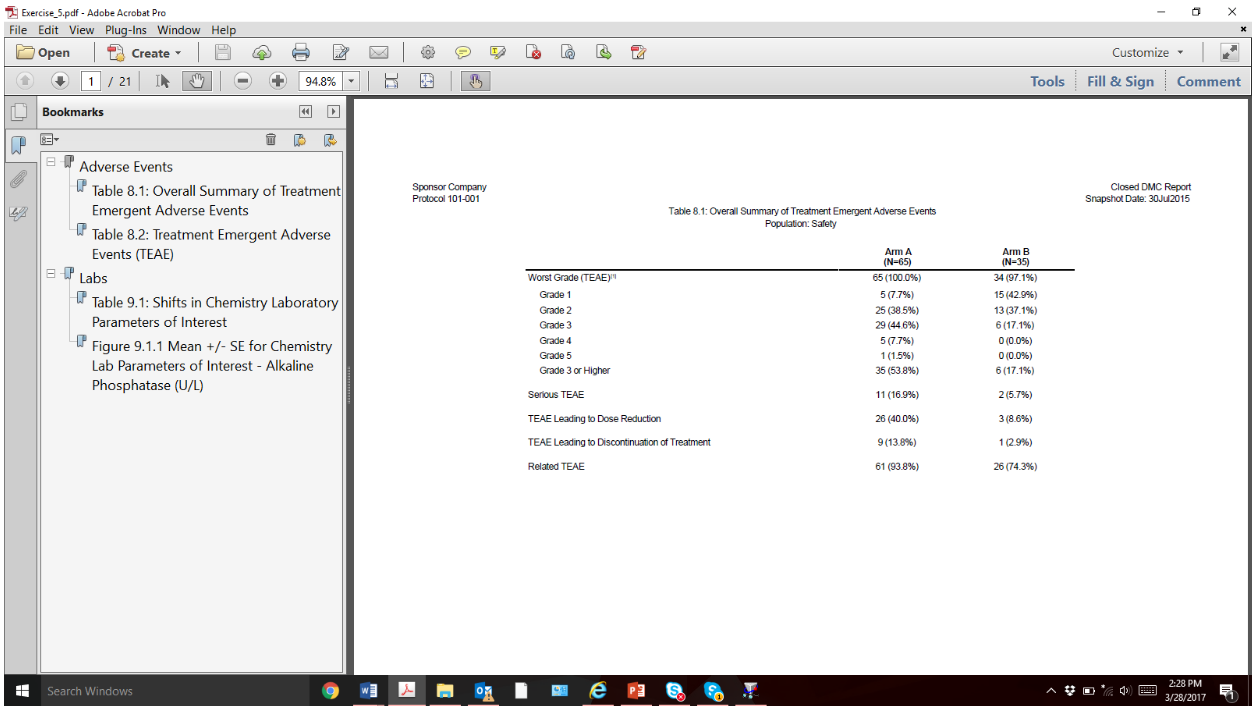
The current combined item store has two sections as desired, but no section labels. These can be added within the blocks of code in Sample Code 7. Alternatively, this can be done through a separate Proc Document call to UPDATE our combined item store as seen in Sample Code 8.

```
proc document name=work.app4(update);
        setlabel ^\Report#1 "Adverse Events";
        setlabel ^\Report#3 "Labs";
run;
quit;
```

Sample Code 8

The key is to use the (update) option rather that the (write) option. SETLABEL commands in the above example demonstrate adding labels to the parent directories Report#1 and Report#3. These labels will be what is displayed in the section headers in the table of contents and bookmarks.

The final output (in PDF) shows the restructured item store replays with the desired properties.



Adverse Events
- Table 8.1: Overall Summary of Treatment Emergent Adverse Events
- Table 8.2: Treatment Emergent Adverse Events (TEAE)

Labs
- Table 9.1: Shifts in Chemistry Laboratory Parameters of Interest
- Figure 9.1.1 Mean +/- SE for Chemistry Lab Parameters of Interest - Alkaline Phosphatase (U/L)

Sponsor Company
Protocol 101-001

Closed DMC Report
Snapshot Date: 30Jul2015

Table 8.1: Overall Summary of Treatment Emergent Adverse Events
Population: Safety

	Arm A (N=65)	Arm B (N=35)
Worst Grade (TEAE)[1]	65 (100.0%)	34 (97.1%)
Grade 1	5 (7.7%)	15 (42.9%)
Grade 2	25 (38.5%)	13 (37.1%)
Grade 3	29 (44.6%)	6 (17.1%)
Grade 4	5 (7.7%)	0 (0.0%)
Grade 5	1 (1.5%)	0 (0.0%)
Grade 3 or Higher	35 (53.8%)	6 (17.1%)
Serious TEAE	11 (16.9%)	2 (5.7%)
TEAE Leading to Dose Reduction	26 (40.0%)	3 (8.6%)
TEAE Leading to Discontinuation of Treatment	9 (13.8%)	1 (2.9%)
Related TEAE	61 (93.8%)	26 (74.3%)

Output 3 Final Deliverable

Exercises 1 through 4 serve as review of the read and restructure technique. Thus far, there has been no mention of by-group processing.

BY-GROUP PROCESSING

In the event that by-group processing occurs with Proc Report, the resulting item stores have a different structure, though they still essentially contain tables and directories. However, these item stores have intermediate directories associated with each BY group.

Exercises 5 and 6 are designed to investigate the structure of item stores from by-group processing associated with site listings. For data review purposes (and sometimes regulatory requirements from Office of Scientific Investigations), it may be desirable to have a base set of listings run by site. However, the desired deliverable is a PDF that groups all the data within a site together. By this we mean that all listings consisting of only Site 1 data are grouped together so that data within Site 1 are easily reviewed. Based on what was demonstrated in earlier examples, this can be done by having a combined item store where all listings for a specific site are within the same parent directory.

Both Exercise 5 and Exercise 6 assume that the original item store was created using by-group processing in Proc Report. Creation of the item store itself is done in the same manner as Exercise 1.

Example 5 begins with techniques used in Exercise 2 which can be applied here to see the structure of an item store. We immediately find the internal structure of the item store is different when by-group processing is utilized.

Listing of: \Istore1.Ldemog_adhoc\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\ByGroup1#1	Dir
3	\Report#1\ByGroup1#1\Report#1	Dir
4	\Report#1\ByGroup1#1\Report#1\Report#1	Table
5	\Report#1\ByGroup2#1	Dir
6	\Report#1\ByGroup2#1\Report#1	Dir
7	\Report#1\ByGroup2#1\Report#1\Report#1	Table
8	\Report#1\ByGroup3#1	Dir
9	\Report#1\ByGroup3#1\Report#1	Dir
10	\Report#1\ByGroup3#1\Report#1\Report#1	Table

Output 4 Item Store with BY-GROUP processing

Replaying the entire listing (including all BY groups) is straightforward using Sample Code 3 with a minor modification of eliminating the Report#1. In fact, Sample Code 3 can be simplified to just use Replay;.

Tip #7: If the entire contents of an item store is to be replayed, simplify the code to say Replay;. There is no real need to specify a Report number or figure number if all contents are to be replayed.

In the event that only a single BY-GROUP is desired in a replay, there are two options. In our case, we desire to replay the data for Site 2. The first option is to "predict" what BY group number contains the desired. By (convenient) design, we know that Site 2 will always be the second by group as demonstrated in Sample Code 9

```
proc document name=istore1.ldemog_adhoc(read);
    replay \Report#1\ByGroup2#1;
run;

quit;
```

Sample Code 9

Alternatively, a WHERE statement can be used to select records where the site variable='Site 2'. The use of a WHERE statement works well, and it is not restricted to BY-GROUP processing. A WHERE statement can be used in a general setting to replay part of any item store provided there is understanding of how the item store was created. Replaying the same listing of Site 2 using a WHERE statement is shown in Sample Code 10

```
proc document name=istore1.ldemog_adhoc(read);
   replay \Report#1\ByGroup2#1;
run;

quit;
```

Sample Code 10

Exercise 5 demonstrates that we can easily identify and replay site specific information. However, in the case of our example request, we wish for site listings (single PDF) with site-specific listings to be grouped together. Exercise 6 demonstrates that this can be done with the same COPY/RENAME/MOVE/DELETE commands shown in previous examples.

Exercise 6 uses the same step-by-step approach used in Exercise 4 to demonstrate how the COPY/RENAME/MOVE/DELETE statements are used to obtain a single file with the desired structure. The initial

```
proc document name=work.app61(write);
copy \istore1.ldemog_adhoc\Report#1\ByGroup1#1 to ^;
rename \ByGroup1#1 to Report;
copy \istore1.lbasec_adhoc\Report#1\ByGroup1#1 to ^;
rename \ByGroup1#1 to Report;
move report#2\Report#1 to Report#1;
delete report#2;
list / levels=all;
run;
```

Sample Code 11

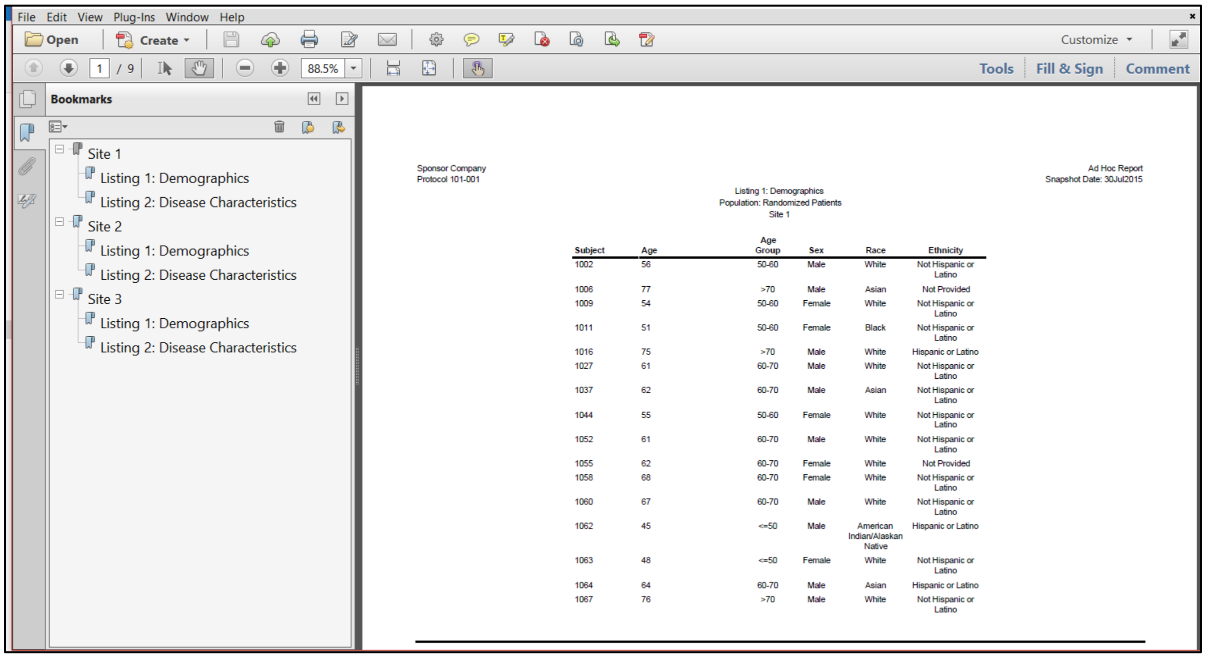
code for Exercise 6 is shown in Sample Code 11.In the interest of keeping this exhausting paper from becoming too lengthy, only some output from the iterative execution is shown below.

Remember, the directory names/numbers are simply labels. The copy command only selects directories and rows associated with directory Report#1\ByGroup#1. This copies observations (2, 3, and 4) in Output 4 into the working item store. What isn't immediately obvious from the above code is the fact that the parent directly is labeled ByGroup1#1. The RENAME statement immediately following the COPY statement renames the parent folder to Report#1. The result of these initial steps is an item store with structure identical to Output 1. These are the first steps in moving towards the familiar setting of parent directories labeled "Report".

The second COPY statement again copies contents of a different item store (Listing of Baseline Characteristic) for the same BY group into the working item store. It is assumed that site associated with the BY group number within each listing is consistent across all listings. This is not an unreasonable assumption as defensive programming can be put in place to guarantee this. If there are no data for a given site, it is probably desirable to have a page that suggests this anyway.

The second RENAME statement similarly renames the parent directory to be "Report", but given the incrementing in SAS this directory name is Report#2. We are now in the same familiar setting as demonstrated in previous exercises. We continue to use COPY\RENAME\MOVE\DELETE blocks. As long as all listings for a given site are within the same parent folder, all listings of site specific data will be replayed together.

Sample Code 11 demonstrates grouping of 2 listings for Site 1. It is easy to imagine this can be done for all sites using similar blocks of code as seen in Output 5. Furthermore, it is easy to see that more listings can be added other than the 2 in this example.



Sponsor Company
Protocol 101-001

Ad Hoc Report
Snapshot Date: 30Jul2015

Listing 1: Demographics
Population: Randomized Patients
Site 1

Subject	Age	Age Group	Sex	Race	Ethnicity
1002	56	50-60	Male	White	Not Hispanic or Latino
1006	77	>70	Male	Asian	Not Provided
1009	54	50-60	Female	White	Not Hispanic or Latino
1011	51	50-60	Female	Black	Not Hispanic or Latino
1016	75	>70	Male	White	Hispanic or Latino
1027	61	60-70	Male	White	Not Hispanic or Latino
1037	62	60-70	Male	Asian	Not Hispanic or Latino
1044	55	50-60	Female	White	Not Hispanic or Latino
1052	61	60-70	Male	White	Not Hispanic or Latino
1055	62	60-70	Female	White	Not Provided
1058	68	60-70	Female	White	Not Hispanic or Latino
1060	67	60-70	Male	White	Not Hispanic or Latino
1062	45	<=50	Male	American Indian/Alaskan Native	Hispanic or Latino
1063	48	<=50	Female	White	Not Hispanic or Latino
1064	64	60-70	Male	Asian	Hispanic or Latino
1067	76	>70	Male	White	Not Hispanic or Latino

Output 5: Site Listings

In Exercises 5 and 6, the user finds that the read and restructure approaches from earlier exercises can be used with BY-GROUP processing as well once there is an understanding of the underlying structure of the item store from BY-GROUP processing as well as an understanding of the desired structure of the combined item store. As noted earlier, the approach demonstrated assumes that all sites have data in all listings and all listings are sorted by site. This ensures that BYGROUP1#1 will always be associated with the first set, BYROUP1#2 with the second site, and so on. This is not a limitation to the use of using item stores for BYGROUP processing, but it certainly simplifies automation.

Tip #8: BY-GROUP information can be added to the bookmark simply by adding the #BYVALn key word to the CONTENTS= option in proc report.

AUTOMATION

Read and restructure concepts were introduced prior to BY-GROUP processing to show they are easily adapted to the setting of BY-GROUP processing. A similar approach is taken as automation is introduced. Only the basics are discussed in this workshop, though the user should be able to see adaptions to other settings are feasible.

The most logical approach to automation is through SAS macro language. This workshop will assume some basic understanding of macro programming, but it is not critical to successfully complete the exercises since macro code is provided.

The examples thus far have not included a large number of TLFs. Exercise 7 allows the user to work with complete code in a slightly more complex setting. The focus of Exercise 7 is to emphasize *incrementing* and *retaining*. Incrementing is associated with automatic incrementing every time SAS finds an item store with directory named **Report**. Retaining is associated with keeping track of the desired parent directory so that similar tables and graphs can be grouped together.

Techniques presented in Example 4 are repeated for Example 7. The only difference is that there are more tables and listings, and there are more sections to the final report. Thus, SAS code is omitted here.

The resulting combined item store is shown in Output 6.

Listing of: \Work.App81\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Report#1	Dir
2	\Report#1\Report#1	Dir
3	\Report#1\Report#1\Report#1	Table
4	\Report#1\Report#2	Dir
5	\Report#1\Report#2\Report#1	Table
6	\Report#1\Report#3	Dir
7	\Report#1\Report#3\Report#1	Table
8	\Report#4	Dir
9	\Report#4\Report#1	Dir
10	\Report#4\Report#1\Report#1	Table
11	\Report#4\Report#2	Dir
12	\Report#4\Report#2\Report#1	Table
13	\Report#6	Dir
14	\Report#6\Report#1	Dir
15	\Report#6\Report#1\Report#1	Table
16	\Report#6\Report#2	Dir
17	\Report#6\Report#2\Report#1	Table
18	\Report#6\Report#3	Dir
19	\Report#6\Report#3\Report#1	Table
20	\Report#6\Report#4	Dir
21	\Report#6\Report#4\Report#1	Table
22	\Report#6\Report#5	Dir
23	\Report#6\Report#5\Report#1	Table

Output 6 Combined Item Store for Automation

The code provided in Example 7 should show the user that he/she will surely approach >1KLOC if the resulting single file has many TLFs.

There are a few things to note as we conceptualize a macro. First, it is clear that there are 3 sections in this single document as the sections are defined by parent directories Report#1, Report#4, and Report#6. We also know that the first section has 3 tables, the second section has 2 tables, and the last section has 5 tables. Without a loss of generality, we refer to tables here although they can be either a table or listing since an item store considers either a "table".

Knowing the number of sections, the number of outputs in each section, and what those reports are within each section, a natural approach is to consider do-loop processing using lists associated with each iteration of the loop.

Table 1 Inputs for Macro

Section number	Number in Section	Section label	Report elements
1	3	Disposition	Rdisp_closed Rdemog_closed Riwrs_closed
2	2	Demographics & Baseline Characteristics	Rdemog_closed Rbasec_closed
3	5	Adverse Events	Raeov_closed Rae_closed Rsae_closed Raedisc_closed Lsae_closed

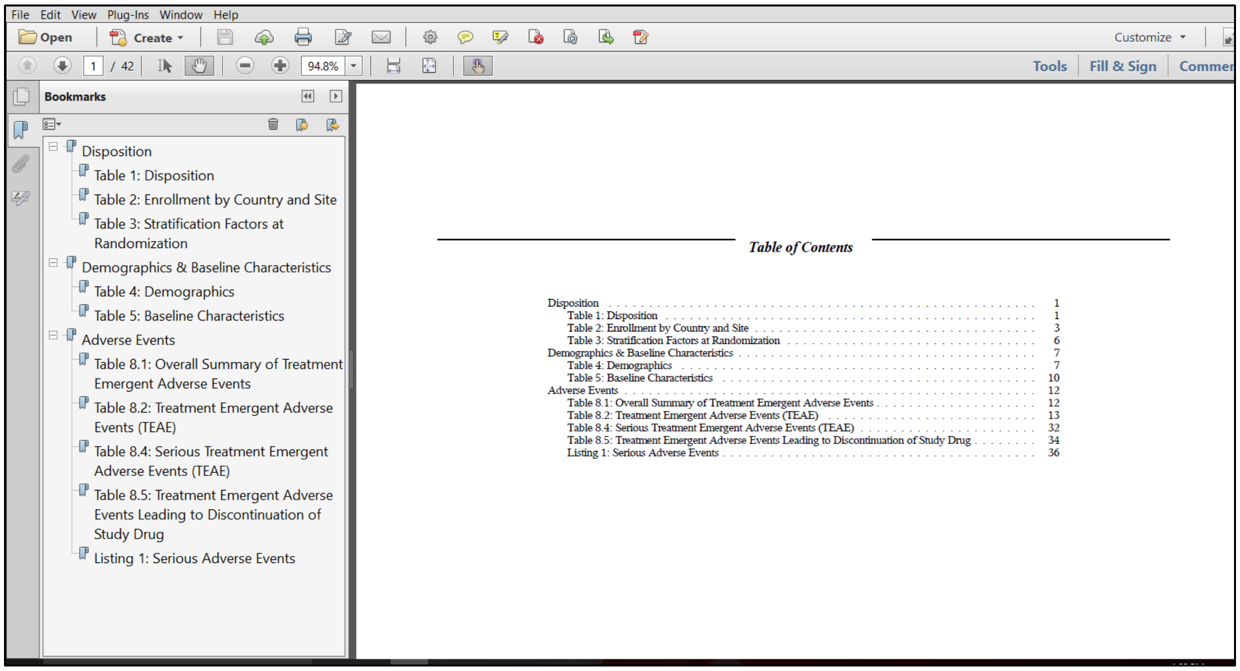
The first step in the macro is to define macro variables to hold counts and lists. This can be done in many ways. In a more advanced setting, Table 1 could exist in the form of a SAS dataset, where the counts and lists can be generated within the macro itself. However, for this workshop, we manually define the macro variables needed.

```
%let nsections=3;
%let ninsection1=3;
%let ninsection2=2;
%let ninsection3=5;

%let isection1=rdisp_closed renrol_closed riwrs_closed;
%let isection2=rdemog_closed rbasec_closed;
%let isection3=raeov_closed rae_closed rsae_closed raedisc_closed lsae_closed ;

%let sectionttl1=Disposition;
%let sectionttl2=Demographics & Baseline Characteristics;
%let sectionttl3=Adverse Events;
```

Sample Code 12: Macro variables for looping

As you can see, the macro variables essentially contain all the information presented in Table 1. We have macro variables for looping within and over each section. There are lists to tell us what item stores go within each section. Last, we have macro variables to indicate section names as they are to appear in the table of contents.

Given the macro variables in Sample Code 12, the macro to create the combined item store is relatively stratigh forward. The macro used for Exercise 8 is found in Appendix 1. The full code is provided simply for discussion within the workshop as it relates to read and restructure of item stores. A screen shot of the final output is show in Output 7.



Disposition
Table 1: Disposition
Table 2: Enrollment by Country and Site
Table 3: Stratification Factors at Randomization
Demographics & Baseline Characteristics
Table 4: Demographics
Table 5: Baseline Characteristics
Adverse Events
Table 8.1: Overall Summary of Treatment Emergent Adverse Events
Table 8.2: Treatment Emergent Adverse Events (TEAE)
Table 8.4: Serious Treatment Emergent Adverse Events (TEAE)
Table 8.5: Treatment Emergent Adverse Events Leading to Discontinuation of Study Drug
Listing 1: Serious Adverse Events

Table of Contents

Disposition 1
Table 1: Disposition 1
Table 2: Enrollment by Country and Site 3
Table 3: Stratification Factors at Randomization 6
Demographics & Baseline Characteristics 7
Table 4: Demographics 7
Table 5: Baseline Characteristics 10
Adverse Events 12
Table 8.1: Overall Summary of Treatment Emergent Adverse Events 12
Table 8.2: Treatment Emergent Adverse Events (TEAE) 13
Table 8.4: Serious Treatment Emergent Adverse Events (TEAE) 32
Table 8.5: Treatment Emergent Adverse Events Leading to Discontinuation of Study Drug 34
Listing 1: Serious Adverse Events 36

Output 7: Single File Deliverable with Table of Contents and Bookmarks

Of note in the final deliverable is a Table of Contents. This was easily obtained using the option contents=on (see Sample Code 4). This example for automation consists only of tables and listings. When there are no figures and the desired ODS destination is PDF, SAS can generate a Table of Contents. This can be useful, especially for printing should it be necessary. For electronic review, hyperlinked bookmarks are often sufficient.

In the demonstration of Example 8, there were no figures. Addition of figures requires minor updates to the macro. Recall from Sample Code 7, this is easily done with the addition of a RENAME statement if the output does not come from Proc Report. Keep in mind that SAS will not be able to provide a Table of Contents in PDF automatically. There will be no errors in the log. However, all of the figures will appear to overlap on the first page of the document. Third party add-ins to Adobe are available (such as Mapsoft TOC Builder ~$100). While this does require a minute or two for the user to update the final product in PDF to add the ToC, it is no more work that would be required should RTF have been the ODS destination.

Tip #9: Adding a ToC to the final PDF will change the page numbering. It may be helpful to change the starting value of the PDF using the option PAGENO=.

CONCLUDING REMARKS

The techniques presented allow a user to create, read, restructure, and replay item stores. They apply to both RTF and PDF, and allow for a very custom single file that includes a hyperlinked Table of Contents and/or bookmarks. With exception to updating a ToC in an RTF file, all steps to create the single file deliverable are done within SAS. All programs can continue to be run in batch mode, and automation of creating the single file deliverable is possible (and desirable). Creation of the single file can be done iteratively, and can be done rather quickly (a few minutes).

These techniques have successfully been implemented on many projects at a CRO for the past 4 years. They continue to be used and have proven to be efficient in creating the single file deliverables. Implementing of such techniques is possible in a production setting, though the author admits that it does require some updates to standard libraries of code, primarily the code for the reporting and graphical procedures. Furthermore, transitioning to Proc SGrender and GTL may be intimidating, but this can be mitigated.

We hope that attendees of this Hands-on-Training leave with the confidence to introduce item stores as a feasible approach to creating a single file deliverable. Some techniques for taking the next steps are presented, and demonstrate that the use of item stores can be implemented for non-standard tables (such as site listings), and may in fact be a natural fit (such as for site listings). Macro processing can be used for automation without the need for complexity.

REFERENCES

[1] Shannon, D. "To ODS RTF and Beyond", SUGI 27, Paper 1-27

[2] Osowski, S., Fritchey, T. "Hyperlinks and Bookmarks with ODS RTF", Pharmasug 2006, Paper TT21

[3] Gilmore, J. "Using Dynamic Data Exchange with Microsoft Word", SUGI 22, Paper 308

[4] Gupta, A, "Watermarking and Combining Multiple RTF Outputs in SAS", PharmaSUG 2012, Paper CC06

[5] Anbazhagan, S, and Patel, S., "A SAS Macro Utility to Append SAS-Generated RTF Outputs and Create the Table of Contents", PharmaSUG 2012, Paper AD12

[6] Coar, W. "Appending Reports: A Review and Fresh Look", WUSS 2012, Paper FP-70

[7] Coar, W. "TLFs: Replaying Rather Than Appending" PharmaSUG 2013, Paper 2343-2014

[8] Coar, W. "Automation of Appending TLFs" PharmaSUG 2014, Paper AD22

[9] SAS 9.2 Output Delivery System: User's Guide, Understanding Item Stores, Template Stores, and Directories, Available at: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a003112776.htm

[10] Lawhorn , B. 2011, Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File, SAS® Global Forum

[11] Coar, W. "Generation of Subset Listing" WUSS 2013, Paper FP-93

[12] Coar, W. "Combining TLFs into a Single File Deliverable", PharmaSUG 2016

ACKNOWLEDGMENTS

The author would like to thank the programmers and statisticians at Axio for their discussions in developing the techniques contributing to this paper. The author would also like to acknowledge the many programmers publishing information on the internet, and apologizes to those not properly referenced above, citing the inherent nature of internet searching which makes it difficult to track so many individual ideas.

CONTACT INFORMATION

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

William Coar, PhD
Biostatistician/Director of Statistical Consulting
Axio Research, LLC
Seattle, WA 98121
Email: williamc@axioresearch.com

APPENDIX 1

Macro provided for Exercise 8 that demonstrates looping over and through each section. The macro assumes lists exist in the form of macro variables as in Sample Code 12. In a more advanced setting, the information in Table 1 can be stored as a SAS dataset. This SAS dataset can then be used to automate the creation of the necessary macro variables.

```
%macro buildit;

proc document name=work.app8(write);

* Initialize the cumulative incrementing macro variable to zero.;

%let Thisi=0;
```

```
* Loop through the number of sections.;
%do b=1 %to &NSECTIONS;

    * With each section, loop through the number of item stores within that section.;
    %do c=1 %to &&NINSECTION&B;

        * Identify the individual item store in each iteration.;
        %let thisst=%scan(&&ISECTION&B,&C,' ');

        * If it is the first item store in the section, then only COPY and retain the directly
        number (THISP). Otherwise perform COPY/MOVE/DELETE. Always increment the cumulative item
        store number (THISI).;

        %if &C=1 %then %do;
            %LET THISP=%EVAL(&THISI+1);
            copy \istore1.&THISST.\Report#1 to ^;
            %LET THISI=%EVAL(&THISI+1);
        %end;
        %else %do;
            copy \istore1.&THISST.\Report#1 to ^;
            %LET THISI=%EVAL(&THISI+1);
            move Report#&THISI.\Report#1 to Report#&THISP.;
            delete Report#&THISI.;
        %end;

    %end;

    * Set the label for the higher level bookmark.;
    setlabel Report#&THISP. "&&SECTIONTTL&B";

%end;

run;
quit;
%mend;
```