Paper DH11

Ingenious Raw Data Tracking in SDTM Programs: A New Perspective

Bhanoji Duppada, Vita Clinical Solutions, Waltham, MA, USA
Sumanth Goud Bheemagani, Merck & Co. Inc., North Wales, PA, USA.

ABSTRACT

This paper introduces an innovative approach to streamline the process of tracking raw datasets within the context of Study Data Tabulation Model (SDTM) programs, addressing challenges like the traditional need for complex SAS programming. The SDTM mapping process involves ensuring consistency between annotated Case Report Forms (CRFs) and tracking the raw datasets in SDTM programs demanding significant manual effort and time. However, this paper presents a novel method that expedites the identification of raw datasets within SAS programs through a semi-automated procedure. By harnessing the potential of MS DOS prompt, batch scripts, and Microsoft Excel, this approach not only offers a practical solution to organize and present raw dataset information more efficiently but also tackles the complexities of complex SAS programming and study folder restrictions. Embracing our approach introduces novel data cleaning practices and enhanced data analysis in excel.

SIMPLIFYING COMPLEXITY

Our proposed approach shatters the shackles of complex SAS programming, ushering in an era of user-friendly operations. It empowers users to efficiently track raw dataset names, SDTM variables, and keywords, bypassing the need for resource-intensive program modifications.

UNLEASHING FLEXIBILITY

Employing text file inputs and ingenious batch scripts, our method breathes new life into raw dataset tracking. It liberates the process from the confines of study folders, enabling execution from any accessible location, while expanding possibilities.

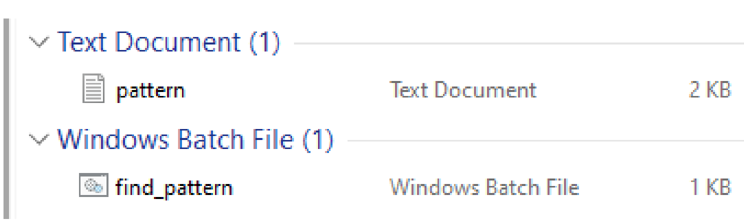
THE PROCESS UNVEILED:

Central to our method is a multi-step process designed for maximum efficiency:

1. **Pattern File Creation:** A "pattern.txt" file is generated, cataloging raw dataset names.
2. **Batch Script Crafting:** A ".bat" script is meticulously crafted to scan SDTM program files for dataset occurrences.
3. **Script Execution:** The batch script is executed, unearthing files containing dataset names, with results seamlessly copied to the "Source_tracker.csv" in the same location of batch script.
4. **Excel Refinement:** Refine the CSV file using Excel or VBA techniques to copy and filter data, add, and populate auxiliary columns, and remove unnecessary values. Utilize Excel's array features and the IF function to get the desired output.
5. **Harnessing Excel Functions:** Utilizing Excel's capabilities, including TEXTJOIN, UNIQUE, and IF functions, yields refined dataset details and counts.

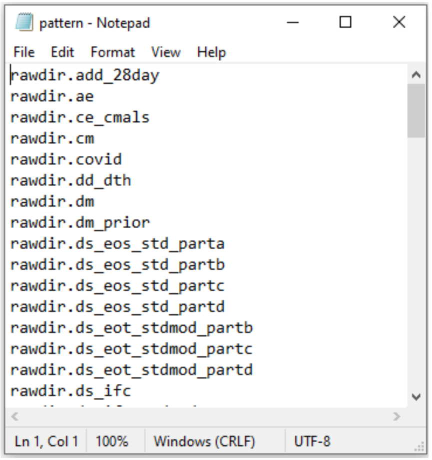
PATTERN FILE CREATION

Create a text file named "pattern.txt" in the same directory as the program and batch files. This file should list all the raw dataset names that will be searched within the program files.



The text file should display the data in a manner similar to that shown in Figure 1.

Figure 1.



BATCH SCRIPT CRAFTING

Generate a text file and paste the provided code into it. Save this file with a '.bat' extension as demonstrated in the accompanying image. Ensure that this batch file is in the same directory as the program files and the pattern file.

BATCH SCRIPT

```
@echo off
REM Create or clear the file at the beginning
> Source_tracker.csv echo File Search Results
for /F "tokens=*" %%i in (pattern.txt) do (
if not "%%i"=="" (
echo Files containing %%i>>Source_tracker.csv
echo Files containing %%i
findstr /I /M /C:%%i /S *.sas >> Source_tracker.csv
findstr /I /M /C:%%i /S *.sas
)
)
echo Check "Source_tracker.csv" for the results
cmd/k
```

AUTOMATED PATTERN SEARCH IN SAS FILES

The script performs an automated search for specified patterns within SAS program files. It operates by iterating over a list of search terms and scanning through .sas files in the current directory and its subdirectories. The results are compiled into a CSV file for easy review.

FUNCTIONALITY BREAKDOWN

1. **Initialization:** The script starts by disabling the display of each command in the command console to maintain a clean output window.
2. **Result Compilation:** A CSV file named "Source_tracker.csv" is created at the script's start. If this file already exists, it is overwritten to ensure fresh results are captured.
3. **Pattern Iteration:** The script reads each search term from "pattern.txt", ensuring no line is overlooked.
4. **Search Execution:** For each non-empty search term, the script:
    - Appends a descriptor line to "Source_tracker.csv", indicating the beginning of results for a new search term.
    - Echoes this descriptor line to the console, providing real-time progress feedback.
    - Executes a case-insensitive search for the term in all .sas files within the current and nested directories, appending the matching file names to "Source_tracker.csv".
    - Outputs the same search results to the console, offering immediate visibility of matches.
5. **Completion Notification:** Upon completing the searches, the script prompts the user to check "Source_tracker.csv" for a compiled list of results.

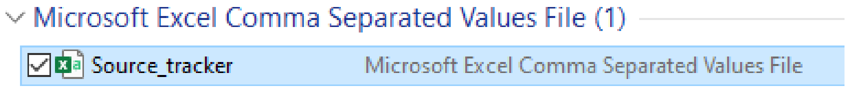
6. **Session Persistence**: Finally, the script prevents the command console from closing automatically, allowing users to review the final message or continue with other commands.

USAGE CONTEXT

This tool aids statistical programmers in efficiently identifying SAS programs that contain specific code snippets or data references, streamlining code reviews and quality control processes.

SCRIPT EXECUTION

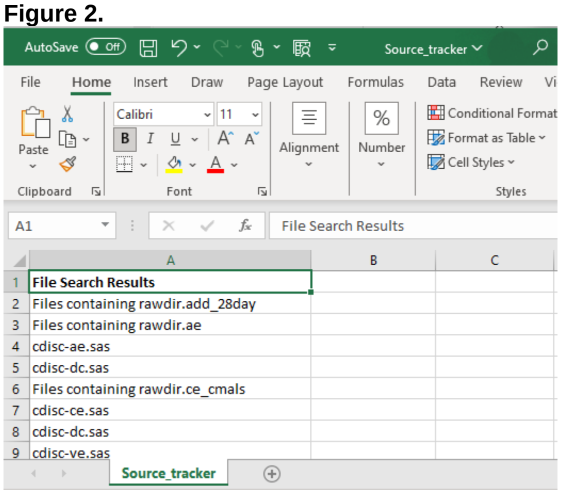
Double click the batch file to execute and the csv file named Source_tracker will be created.



EXCEL REFINEMENT

The CSV file should display the data in a manner similar to that shown in Figure 2. In Column A, raw dataset names are enumerated, with the names of any corresponding SAS files listed beneath each if the datasets are utilized within those programs.

Figure 2.



Apply basic Excel data cleaning methods to refine Column A. This process includes copying and filtering data, creating auxiliary columns, filling in new or supplemental columns, and removing superfluous values. Employ Excel's array functionalities and using IF function and find and replace the string “Files containing” to missing to achieve the desired final output as shown below in Figure 3.

Column A should be split to 2 columns so that column A contains the sas file names and column B should contain the raw dataset names.

Figure 3.

A	B
File Search Results	
cdisc-ae.sas	rawdir.ae
cdisc-dc.sas	rawdir.ae
cdisc-ce.sas	rawdir.ce_cmals
cdisc-dc.sas	rawdir.ce_cmals
cdisc-ve.sas	rawdir.ce_cmals
cdisc-ae.sas	rawdir.cm
cdisc-cm.sas	rawdir.cm
cdisc-dc.sas	rawdir.cm
cdisc-mh.sas	rawdir.cm
cdisc-dc.sas	rawdir.covid
cdisc-se.sas	rawdir.covid
cdisc-ve.sas	rawdir.covid
cdisc-dc.sas	rawdir.dd_dth
cdisc-dd.sas	rawdir.dd_dth
cdisc-ds.sas	rawdir.dd_dth

HARNESSING EXCEL FUNCTIONS

To track the raw dataset names used in sas programs copy the unique raw dataset names in the column C, use the formula **=TEXTJOIN(", ", TRUE, UNIQUE(IF(C1=$B$1:$B$245,$A$1:$A$245,"")))** In column D and apply it to all the cells in the column by dragging the formula so the program names were displayed in column D, if the raw dataset name were used inside them.
use the formula **=IF(D1<>"",LEN(D1)-LEN(SUBSTITUTE(D1,",",""))+1,"")** In column E and apply it to all the cells in the column by dragging the formula so the program names count were displayed in column E.

=TEXTJOIN(", ",TRUE,UNIQUE(IF(C1=$B$1:$B$245,$A$1:$A$245,"")))

- This formula creates a list of unique values from **$A$1:$A$245** where the corresponding cell in $B$1:$B$245 matches the value in C1. The unique values are combined into a single string, separated by commas. The range $B$1:$B$245 should be adjusted as needed.
- Adjust the ranges in the above formula, according to the user data and data range in the CSV file.

=IF(D1<>"",LEN(D1)-LEN(SUBSTITUTE(D1,",",""))+1,"")

- This formula calculates the number of items in a comma-separated list in cell D1. If D1 is not empty, it counts the commas and adds one (indicating the number of items); if D1 is empty, it returns an empty string.

The final output should reflect the below as shown in Figure 4.

Figure 4.

D2 =TEXTJOIN(", ",TRUE,UNIQUE(IF(C2=$B$1:$B$245,$A$1:$A$245,"")))

	A	B	C	D	E
1	**File Search Results**		rawdir.add_28day		
2	cdisc-ae.sas	rawdir.ae	rawdir.ae	cdisc-ae.sas, cdisc-dc.sas	2
3	cdisc-dc.sas	rawdir.ae	rawdir.ce_cmals	cdisc-ce.sas, cdisc-dc.sas, cdisc-ve.sas	3
4	cdisc-ce.sas	rawdir.ce_cmals	rawdir.cm	cdisc-ae.sas, cdisc-cm.sas, cdisc-dc.sas, cdisc-mh.sas	4
5	cdisc-dc.sas	rawdir.ce_cmals	rawdir.covid	cdisc-dc.sas, cdisc-se.sas, cdisc-ve.sas	3
6	cdisc-ve.sas	rawdir.ce_cmals	rawdir.dd_dth	cdisc-dc.sas, cdisc-dd.sas, cdisc-ds.sas	3
7	cdisc-ae.sas	rawdir.cm	rawdir.dm	cdisc-dc.sas	1
8	cdisc-cm.sas	rawdir.cm	rawdir.dm_prior	cdisc-dc.sas	1
9	cdisc-dc.sas	rawdir.cm	rawdir.ds_eos_std_parta	cdisc-dc.sas, cdisc-ds.sas, cdisc-se.sas	3
10	cdisc-mh.sas	rawdir.cm	rawdir.ds_eos_std_partb	cdisc-dc.sas, cdisc-ds.sas, cdisc-se.sas	3
11	cdisc-dc.sas	rawdir.covid	rawdir.ds_eos_std_partc	cdisc-dc.sas, cdisc-ds.sas, cdisc-se.sas	3
12	cdisc-se.sas	rawdir.covid	rawdir.ds_eos_std_partd	cdisc-dc.sas, cdisc-ds.sas, cdisc-se.sas	3

To track the sas program names and their corresponding raw dataset names, copy the unique sas program names in the column C use the formula **=TEXTJOIN(", ",TRUE,UNIQUE(IF(C1=$A$1:$A$245,$B$1:$B$245,"")))** In column D and apply it to all the cells in the column by dragging the formula so the raw dataset names were displayed in column D, if the raw dataset name were used inside them.
use the formula **=IF(D1<>"",LEN(D1)-LEN(SUBSTITUTE(D1,",",""))+1,"")** In column E and apply it to all the cells in the column by dragging the formula so the raw dataset names count were displayed in column E.

=TEXTJOIN(", ",TRUE,UNIQUE(IF(C1=$A$1:$A$245,$B$1:$B$245,"")))

- This formula compiles a comma-separated string of unique values from **$B$1:$B$245** that correspond to the value in C1 found anywhere in **$A$1:$A$245**.
- Adjust the ranges in the above formula, according to the user data and data range in the CSV file.

=IF(D1<>"",LEN(D1)-LEN(SUBSTITUTE(D1,",",""))+1,"")

- This is the same as the second formula and serves the same purpose: to count the number of items in a comma-separated list in cell D1, returning an empty string if D1 is empty.

The final output should reflect the below as shown in Figure 5.

Figure 5

A	B	C	D	E
cdisc-ae.sas	rawdir.ae	cdisc-ae.sas	rawdir.ae, rawdir.cm, rawdir.ho_hospt, rawdir.pr_ndt	4
cdisc-dc.sas	rawdir.ae	cdisc-dc.sas	rawdir.ae, rawdir.ce_cmals, rawdir.cm, rawdir.covid, ra	60
cdisc-ce.sas	rawdir.ce_cmals	cdisc-ce.sas	rawdir.ce_cmals	1
cdisc-dc.sas	rawdir.ce_cmals	cdisc-ve.sas	rawdir.ce_cmals, rawdir.covid, rawdir.ds_ifc, rawdir.ds	45
cdisc-ve.sas	rawdir.ce_cmals	cdisc-cm.sas	rawdir.cm	1
cdisc-ae.sas	rawdir.cm	cdisc-mh.sas	rawdir.cm, rawdir.mh, rawdir.pr_ndt, rawdir.sb_sod1	4
cdisc-cm.sas	rawdir.cm	cdisc-se.sas	rawdir.covid, rawdir.ds_eos_std_parta, rawdir.ds_eos_	48
cdisc-dc.sas	rawdir.cm	cdisc-dd.sas	rawdir.dd_dth	1
cdisc-mh.sas	rawdir.cm	cdisc-ds.sas	rawdir.dd_dth, rawdir.ds_eos_std_parta, rawdir.ds_eo	18

ADVANTAGES

- The input search values were provided dynamically through a text file and can be modified easily as needed.
- No need to change or update the batch script after creation.
- No need to learn the excel advanced functions, clean the excel sheet and maintain the data in proper columns as shown previously and copy and paste the formula. So that anybody can easily obtain the results with basic excel knowledge.
- The batch file assumes the sas files and pattern.txt file in the same location of the batch files, so no need to provide the path manually.
- If the results when compared with the annotated CRF, so that discrepancies will be found if any exists.
- The batch file can be adapted in various ways to look for strings, specific keywords, and regular expressions. Additionally, users can input search words in the pattern.txt file according to their requirements based on the SAS programs, maximizing the effectiveness of this process.

LIMITATIONS

- This process will only work when the user has access to the study folder to run the batch script or else the user should copy the sas files. Pattern.txt and batch script to a workspace where the user has access.
- The user should have access to command prompt. Most of the windows used by default have command prompt access.
- Getting the unique name of raw dataset names or program names seems difficult but this can be achieved by using simple CMD commands like DIR /B /A *.*.
- Eliminate irrelevant comments from the SAS programs that contain raw dataset names to ensure the programs are clean for accurate results. Otherwise, these commented raw dataset names might appear as if they are utilized in the programs, leading to incorrect outcomes.
- To accurately monitor the raw datasets, it's essential to map the programs initially using both the library name and raw dataset name. If this isn't done, modifying the batch script to incorporate regular expressions will be required to ensure precise results.

CONCLUSION

In conclusion, this paper introduces an efficient and effective method for tracking raw datasets within SDTM programs, addressing the key challenges of complex SAS programming and limited accessibility. By employing a combination of MS DOS prompt, batch scripts, and Microsoft Excel, our approach significantly simplifies the data tracking process. It provides a user-friendly and flexible solution that enhances data analysis and cleaning practices. This innovative technique offers precise dataset tracking, cross-referencing with annotated CRFs to detect discrepancies, and efficient SDTM mapping validation. However, its success relies on well-structured program files and a judicious exclusion of extraneous comments. Adaptation of "pattern.txt" dataset names is essential to cater to specific needs.

REFERENCES

https://trumpexcel.com/multiple-lookup-values-single-cell-excel/.

CONTACT INFORMATION

Bhanoji Duppada
Vita Clinical Solutions
281 Winter Street, 1st Floor
Waltham, MA 02451
Work Email: bduppada@softworldinc.com

Sumanth Goud Bheemagani
Merck & Co., Inc. USA
351 N Sumneytown Pike,
Upper Gwynedd, North Wales, PA 19454
Work Email: Sumanth.goud.bheemagani@merck.com

APPENDIX

```
@echo off
REM Create or clear the file at the beginning
> Source_tracker.csv echo File Search Results
for /F "tokens=*" %%i in (pattern.txt) do (
if not "%%i"=="" (
echo Files containing %%i>>Source_tracker.csv
echo Files containing %%i
findstr /I /M /C:%%i /S *.sas >> Source_tracker.csv
findstr /I /M /C:%%i /S *.sas
)
)
echo Check "Source_tracker.csv" for the results
cmd/k
```

1. **@echo off**
   Explanation: This command suppresses the output of the commands as they are executed, making the console output cleaner and easier to read.
   - **Source_tracker.csv echo File Search Results**
     Explanation: This line initializes the **Source_tracker.csv** file by creating it (or clearing it if it already exists) and writes the header "File Search Results" into the file.
2. **for /F "tokens=*" %%i in (pattern.txt) do (**
   Explanation: This initiates a loop that reads each line from the **pattern.txt** file. Each line is treated as a separate token and is assigned to the variable **%%i**.
3. **if not "%%i"=="" (**
   Explanation: Inside the loop, this statement checks if the variable **%%i** (which represents the current line or token from **pattern.txt**) is not empty. It ensures that only non-empty lines are processed.
4. **echo Files containing %%i>>Source_tracker.csv**
   Explanation: If the current token **%%i** is not empty, this line appends the text "Files containing [token]" to the **Source_tracker.csv** file, where [token] is the actual content of **%%i**.
5. **echo Files containing %%i**
   Explanation: This echoes the text "Files containing [token]" to the command prompt as a status message, allowing the user to see the progress in real time.
6. **findstr /I /M /C:%%i /S *.sas >> Source_tracker.csv**
   Explanation: This command searches for the case-insensitive string specified in **%%i** within all **.sas** files in the current directory and its subdirectories, then appends the filenames of the files containing the string to **Source_tracker.csv**.
7. **findstr /I /M /C:%%i /S *.sas**
   Explanation: Similarly, this searches for the string **%%i** but this time it displays the filenames of the **.sas** files containing the string to the console.
8. **)**
   Explanation: This closes the **if** statement.
9. **)**
   Explanation: This closes the **for** loop.
10. **echo Check "Source_tracker.csv" for the results**
    Explanation: After the loop has finished processing all tokens, this message is displayed to the user, prompting them to check the **Source_tracker.csv** file for a comprehensive list of the results.
11. **cmd /k**
    Explanation: This command keeps the command prompt window open after the batch script has completed its execution. Without this, the command prompt window would close immediately after finishing, and the user would not be able to see the final messages or any error messages that might have been displayed.

EXPLANATION OF THE EXCEL FORMULA.

The formula **=TEXTJOIN(", ", TRUE, UNIQUE(IF(C2=$B$1:$B$245,$A$1:$A$245,"")))** performs the following functions:

- **TEXTJOIN**: This function concatenates values with a specified delimiter. Here, it uses a comma followed by a space (", ") to separate the values in the final string.
- **TRUE**: The second parameter of TEXTJOIN specifies if empty cells should be ignored. TRUE means that they will not be included in the concatenated result.
- **UNIQUE**: This function extracts unique values from the provided array or range.

- **IF**: The IF function checks each cell in the range $B$1:$B$245 to see if it matches the value in cell C2. If there is a match, the corresponding value from $A$1:$A$245 is returned. Otherwise, an empty string ("") is returned.
- **Result**: The combination of these functions results in a single string that lists unique values from the range $A$1:$A$245 that correspond to the matching value in cell C2 in the range $B$1:$B$245, separated by commas.
- This formula is particularly useful for summarizing data where you need a list of unique entries that correspond to a specific identifier or condition.
- The given Excel formula **=IF(D2<>"",LEN(D2)-LEN(SUBSTITUTE(D2,",",""))+1,"")** is designed to count the number of delimiters (in this case, commas) in a string plus one, under the condition that the cell is not empty. Here's a breakdown of its components:
- **IF(D2<>"", ... , "")**: This is a conditional statement that checks if cell D2 is not empty (**D2<>""**). If D2 is not empty, it executes the formula within the first set of commas. If D2 is empty, it returns an empty string (**""**).
- **LEN(D2)**: This function returns the length of the string in cell D2.
- **SUBSTITUTE(D2,",","")**: This function replaces all commas in the string in D2 with nothing (effectively deleting them), and then the length of this new string is measured.
- **LEN(D2) - LEN(SUBSTITUTE(D2,",",""))**: This calculates the difference in length between the original string and the string without commas, which gives the number of commas in the original string.
- **... + 1**: One is added to the count of commas to adjust for the number of elements in the list separated by commas. For example, if there are two commas, there are three elements in the list.

So, the whole formula checks if D2 is not empty, and if it isn't, it calculates the number of elements separated by commas in the cell. If the cell is empty, it returns an empty string. This can be useful when counting the number of entries in a CSV formatted cell.