PharmaSUG China 2025 - Paper AP-102

An efficient way to combine rtf documents with navigation bars using SAS and python

Yingying Zhuo, Weineng Zhou, Chris Lin, and Kunwan Chen, Clinflash Healthcare Technology (Jiaxing) Co.,Ltd

ABSTRACT

A simple and efficient way to combine rtf documents (i.e., tables, figures, and listings in clinical trial) improve the efficiency and happiness of programmers' daily work. Navigation bars are always welcome to reviewers. Here is sharing an efficient programming providing navigation bars to make the rtf documents combination much easier.

INTRODUCTION

This document will introduce a new method to combine rtf file documents which is created by the author and has been applied successfully for many times, hoping this article will be useful for readers to saving time when combining rtf documents.

First of all, we'll show how to set up code in SAS to produce a navigation bar for the later combination.

Secondly, we'll combine these TFLs by using python.

Finally, we will add the details of TOC (table of contents) to word file.

GENERATING SINGLE TFL IN SAS

Before creating a single TFL document, you need to add Microsoft Word style in coding the "ods rtf" statement.

```
ods rtf file="test.rtf" wordstyle="{\s1 Heading 1 \s2 Heading 2 \s3 Heading
3;}";
```

"s1" is a nickname linking to an already defined Microsoft Word style Heading 1, "s2" for Heading 2, "s3" for Heading 3 and these cannot be other self-defined names.

With this step, you can set up the title in most SAS statement when creating a text file.

For example:

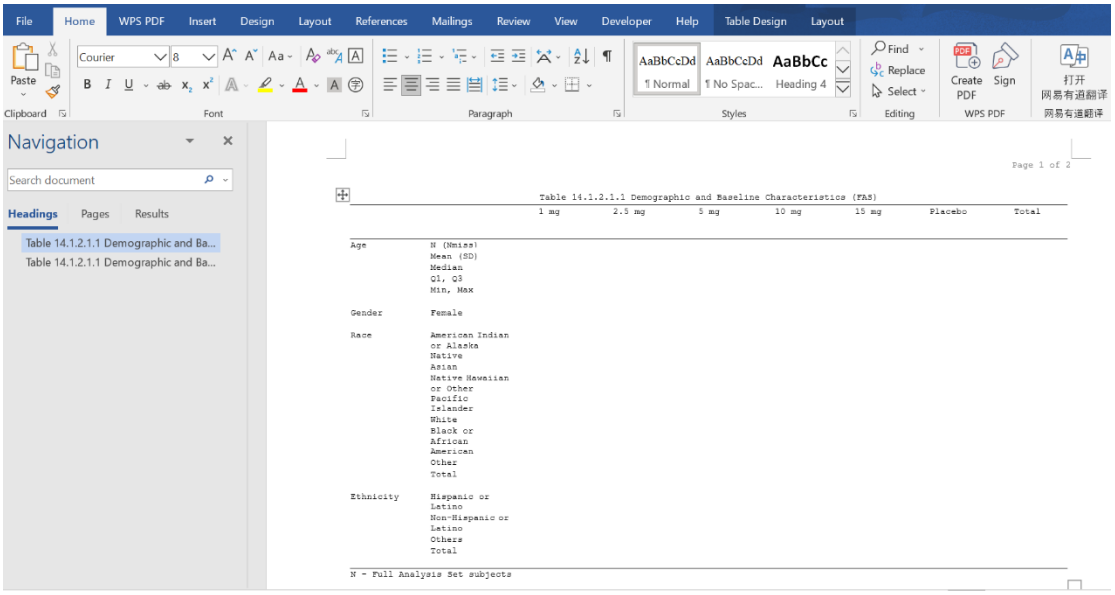
```
proc odstext;
p "{\pard\s1 Table of Contents \par}";
...;
run;

ods text="{\pard\s1\qc Table 1. Demographic characteristic  \par}";
```

TABLES AND LISTINGS

Tables or listings may be long and split across pages. So, it's important to use "pretext" option in proc report when generating navigation tags. This will make sure those cross-page tables have the same title on every page.

```
proc report data=final missing center nowd headline headskip
  style(report)={pretext="\pard\s1\qc {&rtftitle1.}"};
```

Display 1 is sample display or screen capture.



Display 1. Titles generated in tables and listings

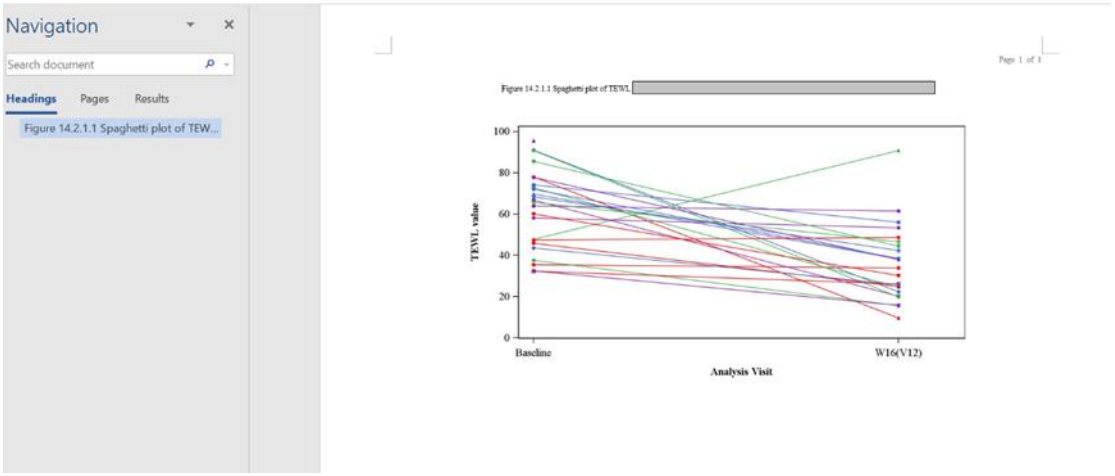
You can see in upper picture that there are duplicate tags, we will keep the first tag in our python program.

FIGURES

In figures part, we use the “ods text” statement as follows:

```
...
ods text="{\pard\s1\qc &rtftitle1. \par}";
proc sgrender data=final template=lineplot;
run;
...
```

Here is the result.

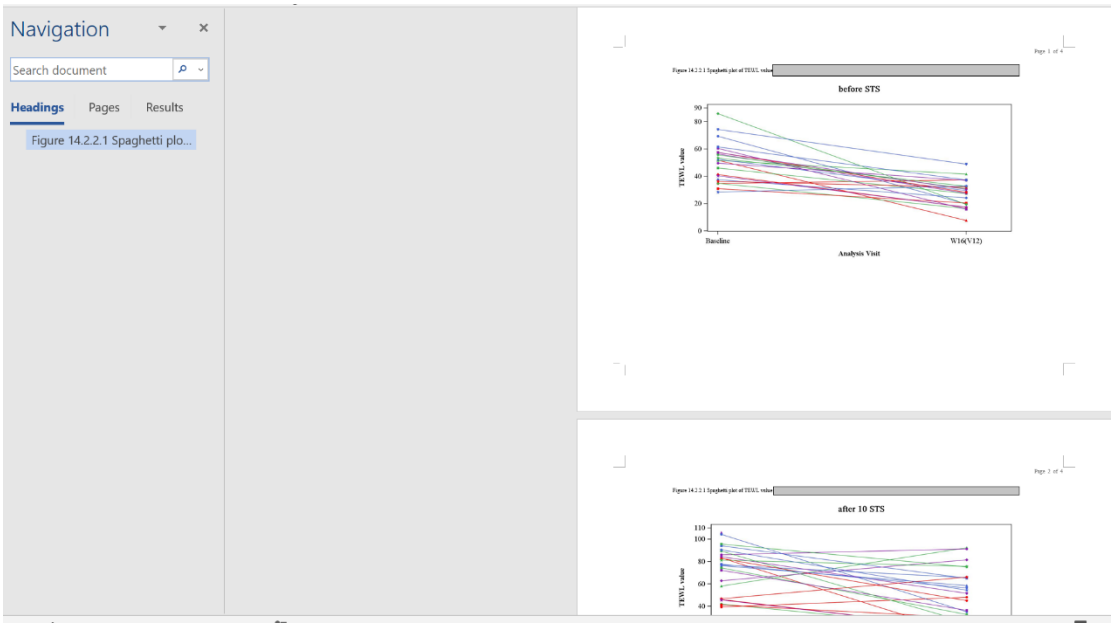


Display 2. Titles generated in figures

Note that the code above applies when only one page exists in one document. When there are multiple pages, at this time we need to control in SAS that the output style is s1 (heading1) only on the first page, and subsequent pages only produce titles and do not link to navigation tabs.

```
%macro plotloop;
```

```
    %do i= 1 %to 4;
    ods rtf startpage=now;
          %if &i=1 %then %do;
                ods text="{\pard\s1\qc &rtftitle1. \par}";
          %end;
          %else %do;
                ods text="{\pard\qc &rtftitle1. \par}";
          %end;
          proc sgrender data=final&i. template=lineplot&i.;
          run;
    %end;
%mend;

%plotloop;
```



Display 3. Retention of the first navigation tab in the figure

COMBINATION

After each rtf document has been processed, the next step is to combine them and remove the duplicate tags of table and listing.

The bookmark code (in rtf) is shown below:

The first bookmark

```
\pard}}{\upr{\*\bkmkstart IDX}{\*\ud{\*\bkmkstart IDX}}}{\*\bkmkend
IDX}\pard\b0\i0\chcbpat8\qc\f1\fs16\cf1\pard\s1\qc {Table 14.3.3.2.2.1
Change from Baseline in Parameters of Blood Chemistry (SS) -
MAD}{}\cf0\chcbpat0
```

The second bookmark

```
\pard}}{\upr{\*\bkmkstart IDX1}{\*\ud{\*\bkmkstart IDX1}}}{\*\bkmkend
IDX1}\pard\b0\i0\chcbpat8\qc\f1\fs16\cf1\pard\s1\qc {Table 14.3.3.2.2.1
Change from Baseline in Parameters of Blood Chemistry (SS) -
MAD}{}\cf0\chcbpat0
```

The third bookmark

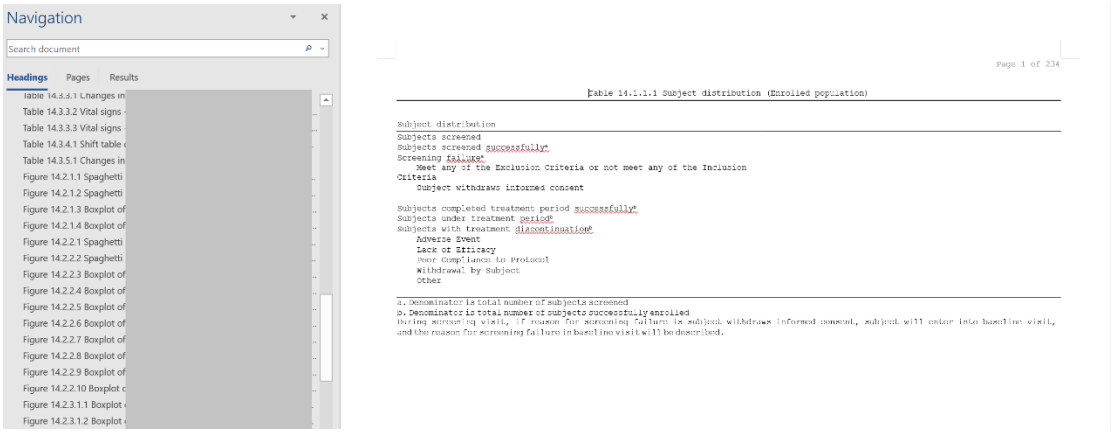
```
\pard}}{\upr{\*\bkmkstart IDX2}{\*\ud{\*\bkmkstart IDX2}}}{\*\bkmkend
IDX2}\pard\b0\i0\chcbpat8\qc\f1\fs16\cf1\pard\s1\qc {Table 14.3.3.2.2.1
Change from Baseline in Parameters of Blood Chemistry (SS) -
MAD}{}\cf0\chcbpat0
```

Only the s1 in the subsequent bookmark needs to be deleted, and the other of the subsequent bookmark is marked with "bkmkstart IDX*". Then, the subsequent bookmark will be located easily.

The Python code is as follows:

```
if re.search(r'bkmkstart IDX\d+', i_str):
    j_str = i_str.replace(r'\pard\s1\qc', r'\pard\qc').replace(u'\xa0','')
    j = bytes(j_str, encoding = "utf-8")
    out_file.write(j)
else:
    out_file.write(i)
```

Here is python's combined and desensitized documentation.

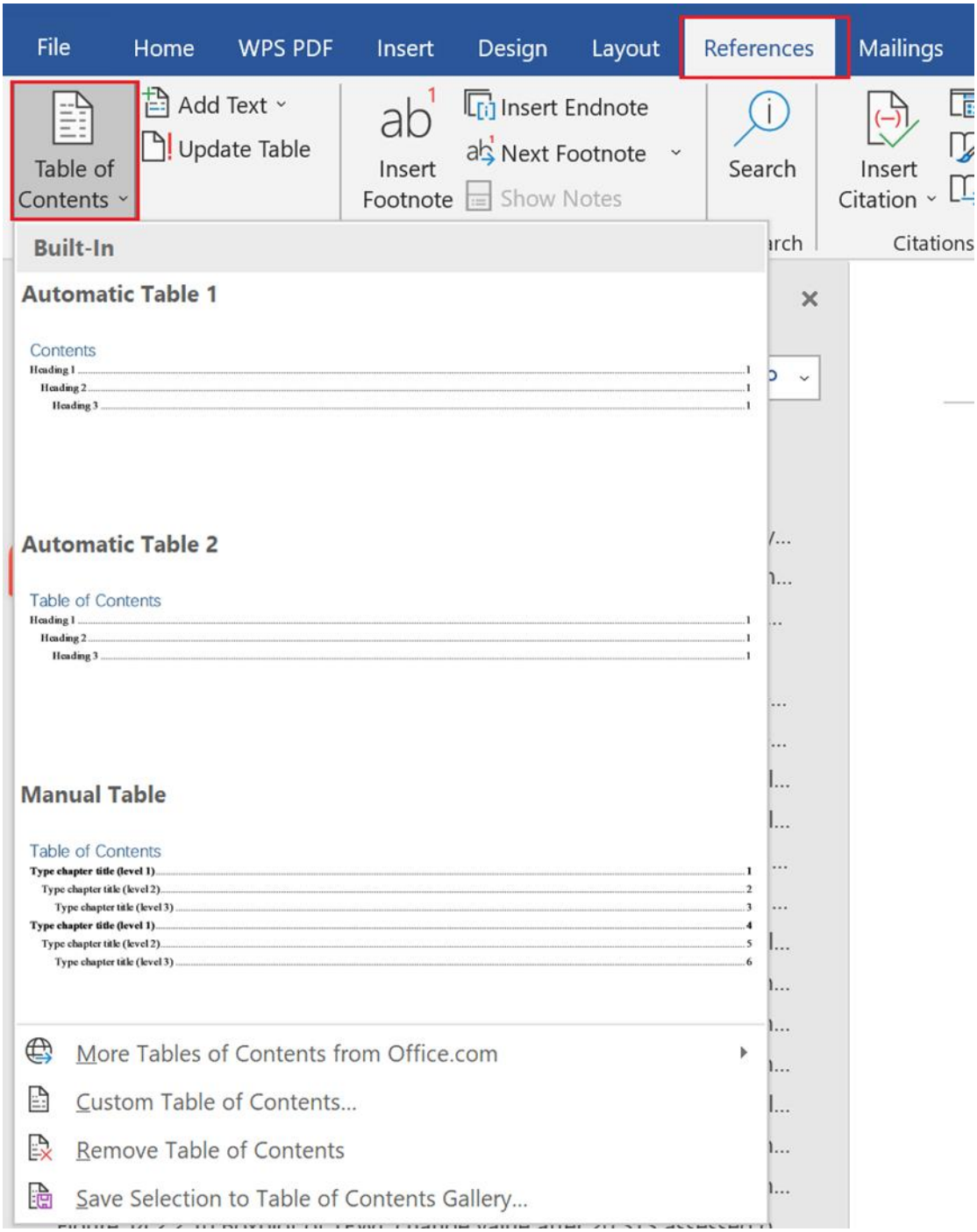


Display 4. Python's combined documentation

Once you put all your rtf documents in one folder and run the python program, it only needs 5 seconds that you can successfully get the combined rtf document. The process is very simple and efficient, and do not need to prepare other auxiliary document like other combining methods. The combined document will be consistent with the original TFLs, which will not change the style or any content of the original file.

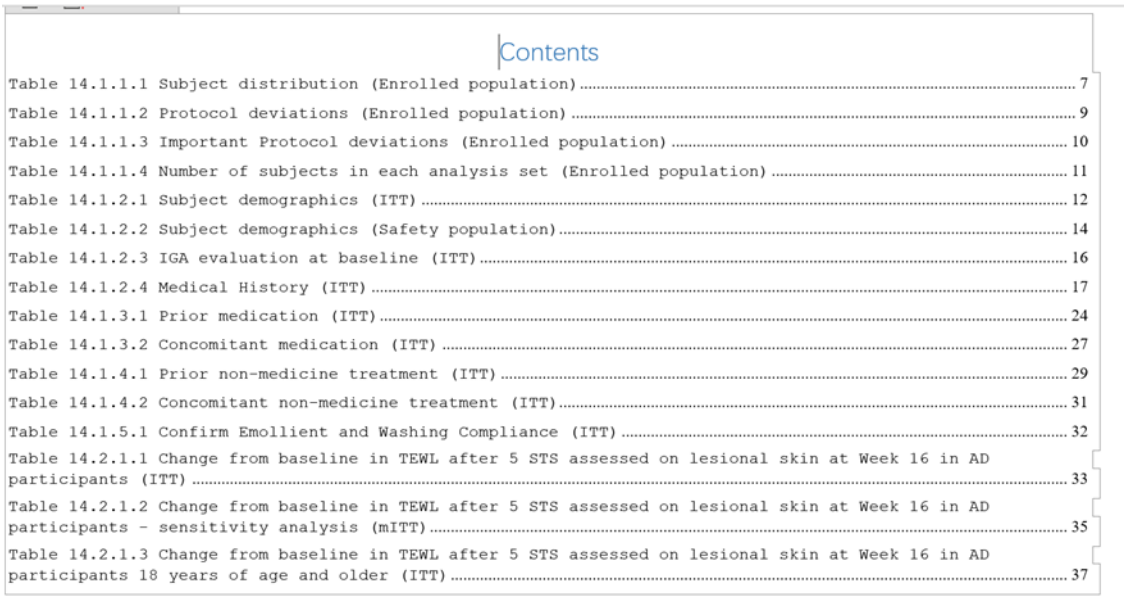
ADDING TABLE OF CONTENTS

After combination, you can add table of contents from Microsoft Word, click References-> Table of Contents->Choose one style from the drop-down list.



Display 5. Adding table of contents using Microsoft Word

Below is the table of contents that was successfully added.



Display 6. Table of contents generated from Microsoft Word

CONCLUSION

In practice, we tried several different methods to merge rtf and add navigation tags, and finally confirmed that this way is the most efficient, the most convenient and the most accurate among the methods. In general, navigation tags are created by adding word heading style in SAS, and individual rtf documents are combined through python processing. The code could be improved a little bit, but the general idea is the correct. The next part includes checking the blank pages in the merged documents (this problem appears when a single rtf document is generated, not related to the merging tool). The python code mentioned above will show below.

REFERENCES

Lauren Haworth, Genentech, Inc., South San Francisco, CA “Applying Microsoft Word Styles to ODS RTF Output” https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/043-30.pdf

CONTACT INFORMATION

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

Yingying Zhuo
Clinflash Healthcare Technology (Jiaxing) Co.,Ltd
Yingying.Zhuo@clinflash.com

Weineng Zhou
Clinflash Healthcare Technology (Jiaxing) Co.,Ltd
Weineng.Zhou@clinflash.com

Chris Lin
Clinflash Healthcare Technology (Jiaxing) Co.,Ltd
Chris.Lin@clinflash.com

Kunwan Chen
Clinflash Healthcare Technology (Jiaxing) Co.,Ltd
Kunwan.Chen@clinflash.com

APPENDIX: PYTHON PROGRAM

```
import os
import re
import glob
import time
import pandas as pd
rootpath = os.getcwd()
print(rootpath)

start = time.time()

T_list = []
for file in os.listdir(rootpath):
    file_path = os.path.join(rootpath, file)
    if os.path.isfile(file_path):
        if file[0:1].upper() == "T" and file[-4:] == ".rtf":
            # print(file)
            T_list.append(file)


F_list = []
for file in os.listdir(rootpath):
    file_path = os.path.join(rootpath, file)
    if os.path.isfile(file_path):
        if file[0:1].upper() == "F" and file[-4:] == ".rtf":
            # print(file)
            F_list.append(file)

L_list = []
for file in os.listdir(rootpath):
    file_path = os.path.join(rootpath, file)
    if os.path.isfile(file_path):
        if file[0:1].upper() == "L" and file[-4:] == ".rtf":
            # print(file)
            L_list.append(file)


file_list = T_list + F_list + L_list
# print(file_list)

def makedir(ndir):
  try:
    os.makedirs(ndir)
  except FileExistsError:
    pass

def combinertf(filename, pagedelimit = True):

    """
    filename: the name of final combined rtf document.
    pagedelimit: determine if rtf would start in a new page or new line.
    """
    # filedir='./'
    # filename = 'output.rtf'
    if '/' in filename:
        makedir('/'.join(filename.split('/')[:-1]))
```

```
    else:
        makedir('output')
        filename = 'output/' + filename

    # file_list = readfile(filedir)
    test = filename
    try:
        file_list.remove(test)
    except ValueError:
        pass

    out_file = open(test,'wb')
    start = '{'
    start_byte = bytes(start, encoding = "utf8")
    out_file.write(start_byte)

    for fname in file_list:
        print(fname)
        if  test in fname:  continue
        with open(fname, 'rb') as f1:
            mylist = list(l1 for l1 in f1)
            mylist[0] = mylist[0].strip()[1:]
            mylist[-1] = mylist[-1].strip()[:-1]

            for i in mylist:

                i_str = str(i, encoding = "utf-8")

                if re.search(r'bkmkstart IDX\d+', i_str):
                    # j_str = i_str.replace(r'\outlinelevel2',
r'').replace(u'\xa0','')

                    j_str = i_str.replace(r'\pard\s1\qc',
r'\pard\qc').replace(u'\xa0','')
                    j = bytes(j_str, encoding = "utf-8")
                    out_file.write(j)
                else:
                    out_file.write(i)


            if pagedelimit & (fname != file_list[-1]):
                parpage = "\sect"
                parpage_byte = bytes(parpage, encoding = "utf-8")
                out_file.write(parpage_byte)

    end = '} '
    end_byte = bytes(end, encoding = "utf-8")
    out_file.write(end_byte)
    out_file.close()
    print('Combine finished!')

combinertf(
    filename='output.rtf',
    pagedelimit = True
)
end = time.time()
print("It's used : %.2f second" % (end - start))
```