PharmaSUG China 2023 - Paper AA-150

Unlocking Efficiency Potential: Harnessing Python for Statistical Programming Excellence in Pharmaceutical Industry

Kuangye Fang, Fosun Pharma

ABSTRACT

In the ever-evolving pharmaceutical industry, the role of statistical programmers is critical in ensuring the accuracy and reliability of clinical trial data analysis results. As the volume and complexity of data continue to increase, there is a growing need for efficient and effective tools to streamline workflows and improve work quality.

The primary objective of this research is to demonstrate the development of a Python application specifically designed to address a common issue encountered by statistical programmers, i.e, the identification of unexpected page break issues in Rich Text Format (RTF) documents. Unexpected page breaks often lead to formatting issues, which need statistical programmers to adjust the programs again and again. By automating this identification process, the proposed Python app offers a solution to optimize our work efficiency.

In conclusion, this paper shows you how Python can be leveraged as a powerful programming language to improve daily work efficiency and quality for statistical programmers in clinical trials, and highlights the importance of embracing technology to optimize workflows, increase productivity, and ensure the reliability of statistical analysis in clinical trials in the pharmaceutical industry.

INTRODUCTION

In this paper, we introduced the development process of an application designed to identify page break issue in RTF (Rich Text Format) files and generate a comprehensive report in excel file format using Python.

This application leverages the power of Python and several key packages to achieve its functionality. The initial step utilizes the "win32com.client" package in python to scan the RTF file and identify any page break issues. This package provides a set of functionalities to interact with the Microsoft Office suite and access the necessary information within the RTF file. By utilizing the functionality of "win32com.client," we can efficiently extract page break information from the RTF file.

Once the page break issues are identified, the "openpyxl" package comes into play. This powerful package enables us to format and organize the page break information as collected in previous step. With "openpyxl" package, we can create a new Excel file which provides a clear and structured summary of the page break issues found in the RTF file. This step ensures that the generated report is user friendly and easy for further use.

To enhance user experience and facilitate interaction with the application, we employ the "PyQt5" package. This package enables us to design a graphical user interface (GUI) that presents the application's functionalities in an intuitive and user-friendly manner. Through the utilization of "PyQt5," users can easily navigate the application, input the target RTF file, and obtain the Excel report with just a few clicks. The integration of a GUI further simplifies the usage of this application and enhances its overall usability.

EXTRACT PAGE BREAK INFORMATION

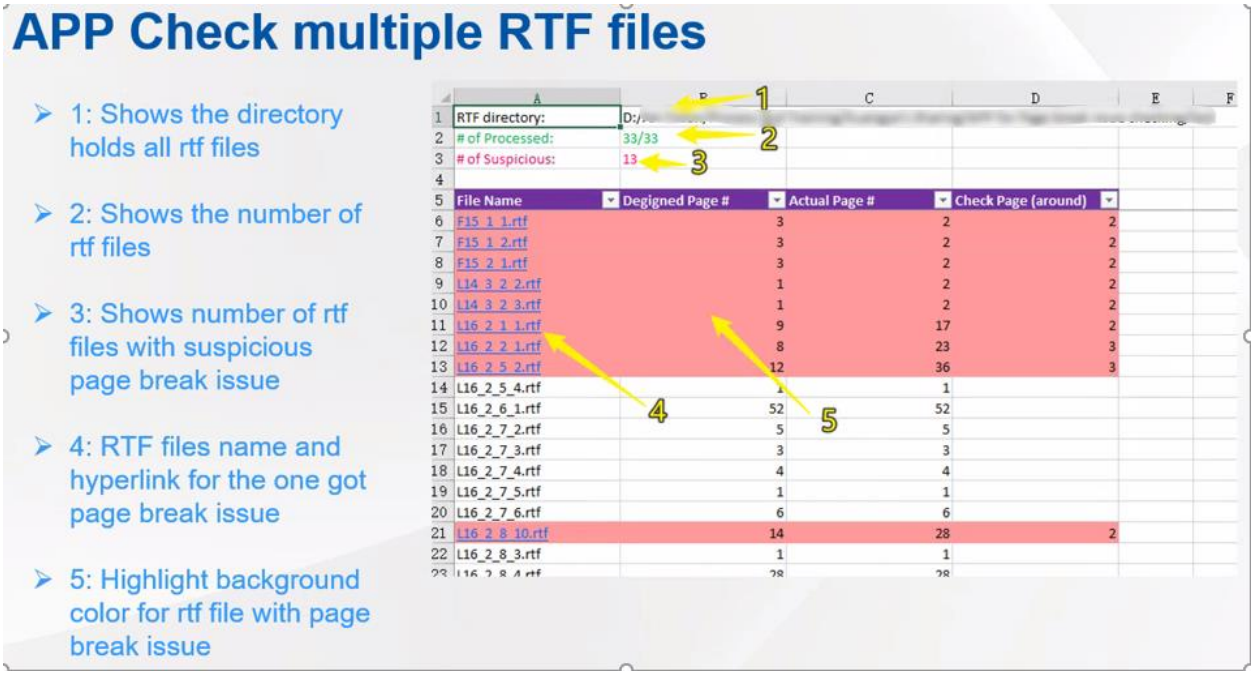
This application utilizes the "win32com.client" package to open an RTF file with Microsoft Word application, check for page break issues, and return information about the number of sections, the number of pages, and the first occurrence of a page break issue as a list.

```
import win32com.client as win32
objWORD = win32.gencache.EnsureDispatch('Word.Application')
info_list = chkRTF_KNL(objWORD, sRTFDIR, sRTFNAME)
#Function to return array of # of section/# of pages/# of first occurrence of pagination issue
def  chkRTF_KNL(self, objWORD, sRTFDIR, sRTFNAME):
    wdActiveEndPageNumber = 3 #Active End Page Number (of a section)
    fullfile = sRTFDIR + '/' +sRTFNAME
    fullfile2 = fullfile.replace('/', '\\')
    doc = objWORD.Documents.Open(fullfile2)

    nsection = doc.Sections.Count
    doc.Repaginate
    npage = doc.Sections(doc.Sections.Count).Range.Information(wdActiveEndPageNumber)
    fpagen = 0
    fsect = 0

    if npage != nsection:
        for sec in doc.Sections:
            nEndPgn = sec.Range.Information(wdActiveEndPageNumber)
            if nEndPgn != sec.Index:
                fpagen = nEndPgn
                fsect = sec.Index
                break

    info_list = [nsection,npage,fpagen,fsect]
    return info_list
```

PRINT PAGE BREAK INFROMATION TO EXCEL FILE

This application utilizes the "openpyxl" package to create an Excel workbook, set up a worksheet, populate cells with data, and apply formatting options. The resulting workbook will have summary information, headers, and formatted data related to RTF files found in the specified directory. Related python code is available in Appendix I.

Figure 1. The page break summary report

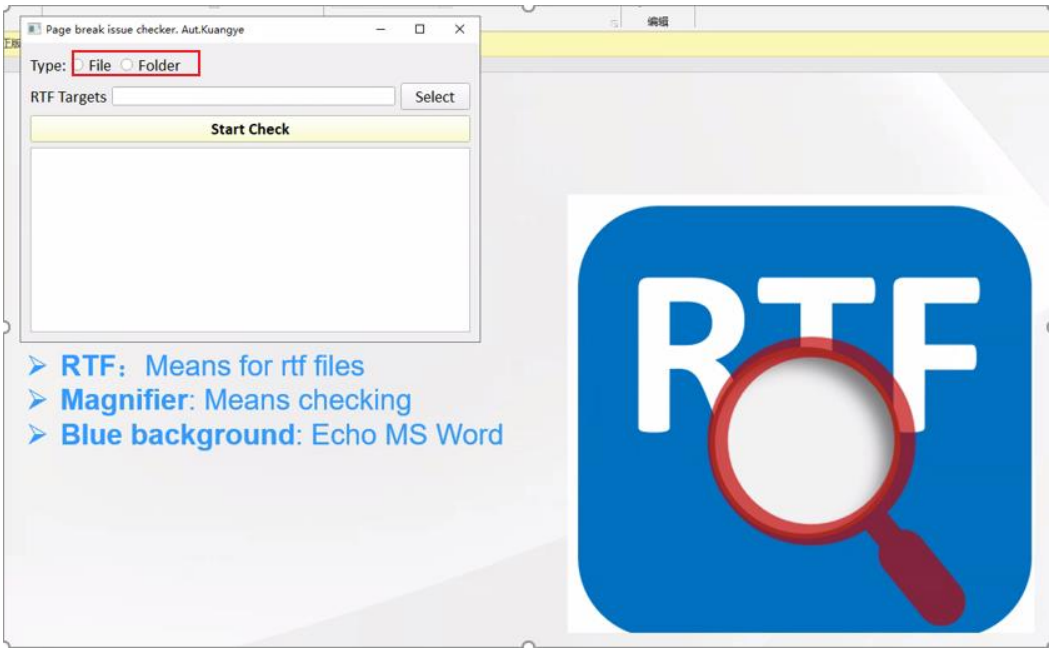


DESIGN GRAPHICAL USER INTERFACE

In the UI layout definition code for the application, it specifies the arrangement and properties of various widgets that will be displayed in the application's window, including labels, radio buttons, text fields, buttons, and a plain text area. The UI layout is defined using a combination of vertical and horizontal layouts to organize the widgets on the form. UI layout definition code is available in Appendix II.

Below is the graphical user interface of this application. This application got two checking mode, one is "File" checking mode to check single RTF file and another is "Folder" checking mode to check all RTF files from a specified directory and return the summary result to one excel file.

Figure 2. Application's Graphical User Interface (GUI)



CONCLUSION

In conclusion, this paper has presented an approach to develop an application using Python for identifying page break issues in RTF files. The development process involved the utilization of three key packages: "win32com.client", "openpyxl", and "PyQt5". Finally, this application can be compiled into executable file with "pyinstaller" package. This application not only allows you to batch check a large number of RTF files at once, quickly identifying page break issues and significantly saving time for output QC (Quality Control) for statistical programmer, but it also can be shared with other statistical programmer colleagues to use, even if they have no knowledge of Python. In summary, this application can improve both individual and team's delivery quality and efficiency for statistical programmer.

CONTACT INFORMATION

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

Kuangye Fang
Scientific Programming Manager
Fosun Pharma
+86 15921860572
fangkuangye@fosunpharma.com

APPENDIX I

Below code in appendix I is used to print page break information to an excel file.

```
from openpyxl.workbook import Workbook
from openpyxl import load_workbook
```

```
from openpyxl.styles import Font, PatternFill, Alignment
from openpyxl.utils import get_column_letter
nrtf = 0
for fpath, dirs, fs in os.walk(inpath):
    for f in fs:
        fi_d = os.path.join(fpath, f)
        ext = os.path.splitext(fi_d)[1][1:]
        if (ext == 'rtf' or ext == 'RTF') and fpath == inpath:
            nrtf += 1
wb = Workbook()
wb['Sheet'].title = 'Pagenition'
ws = wb.active
# summary information
ws['A1'] = "RTF directory:"
ws['B1'] = inpath
ws.merge_cells("B1:Z1")
ws['A2'] = "# of Processed:"
ws['B2'] = r'=concatenate(counta(A6:A65536), "/" + r"', ' + str(nrtf) +  r')'
ws['A3'] = "# of Suspicious:"
ws['B3'] = "=counta(D6:D65536)"
# Headers (starting at line 5)
ws['A5'] = "File Name"
ws['B5'] = "Degigned Page #"
ws['C5'] = "Actual Page #"
ws['D5'] = "Check Page (around)"
ws.freeze_panes = 'A6'
ws.auto_filter.ref = 'A5:D5'
ws['B2'].alignment = Alignment(horizontal='left',vertical='center',wrapText=True)
ws['B3'].alignment = Alignment(horizontal='left',vertical='center',wrapText=True)
for row in range(1,500):
    for col in range(1,5):
        ws.cell(row,col).font = Font(name='Calibri')
        if row == 5:
            ws.cell(5,col).font = Font(name='Calibri',color='FFFFFF',bold=True)
            ws.cell(5,col).fill = PatternFill(fill_type='solid',
                                start_color='7030A0',
                                end_color='7030A0')
```

```
            col_letter = get_column_letter(col)
            ws.column_dimensions[col_letter].width = 20
    ws['A2'].font = Font(name='Calibri',color='00B050')
    ws['B2'].font = Font(name='Calibri',color='00B050')
    ws['A3'].font = Font(name='Calibri',color='FF0070')
    ws['B3'].font = Font(name='Calibri',color='FF0070')

    ws['A' + str(count+5)] = '=HYPERLINK("' + fi_d + '#IDX' + str(info_list[2]-2) + '", "' + f + '")'
    ws['B' + str(count+5)] = info_list[0]
    ws['C' + str(count+5)] = info_list[1]
    ws['D' + str(count+5)] = info_list[2]
    ws['A' + str(count+5)].fill = PatternFill(fill_type='solid',start_color='FF999B',end_color='FF999B')
    ws['B' + str(count+5)].fill = PatternFill(fill_type='solid',start_color='FF999B',end_color='FF999B')
    ws['C' + str(count+5)].fill = PatternFill(fill_type='solid',start_color='FF999B',end_color='FF999B')
    ws['D' + str(count+5)].fill = PatternFill(fill_type='solid',start_color='FF999B',end_color='FF999B')
    ws['A' + str(count+5)].font = Font(name='Calibri',color='0064FF',underline='single')
```

APPENDIX II

Below code in appendix II is used to design the graphical user interface of the application.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <ui version="4.0">
   <class>backup_vdifile</class>
   <widget class="QWidget" name="check_rtffile">
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>0</y>
      <width>528</width>
      <height>343</height>
     </rect>
    </property>
    <property name="windowTitle">
     <string>Form</string>
    </property>
    <layout class="QVBoxLayout" name="verticalLayout_2">
```

```
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_3">
  <item>
   <widget class="QLabel" name="label_2">
    <property name="font">
     <font>
      <family>Calibri</family>
      <pointsize>14</pointsize>
     </font>
    </property>
    <property name="text">
     <string>Type:</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QRadioButton" name="fileBtn">
    <property name="enabled">
     <bool>true</bool>
    </property>
    <property name="font">
     <font>
      <family>Calibri</family>
      <pointsize>14</pointsize>
     </font>
    </property>
    <property name="text">
     <string>File</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QRadioButton" name="folderBtn">
    <property name="font">
     <font>
      <family>Calibri</family>
      <pointsize>14</pointsize>
```

```
    </font>
   </property>
   <property name="text">
    <string>Folder</string>
   </property>
  </widget>
 </item>
 <item>
  <spacer name="horizontalSpacer">
   <property name="orientation">
    <enum>Qt::Horizontal</enum>
   </property>
   <property name="sizeHint" stdset="0">
    <size>
     <width>318</width>
     <height>20</height>
    </size>
   </property>
  </spacer>
 </item>
</layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout">
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_2">
    <property name="leftMargin">
     <number>0</number>
    </property>
    <item>
     <widget class="QLabel" name="label">
      <property name="font">
       <font>
        <family>Calibri</family>
        <pointsize>14</pointsize>
       </font>
      </property>
```

```
  <property name="text">
   <string>RTF Targets</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QLineEdit" name="dirEdit">
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>16777215</height>
   </size>
  </property>
  <property name="sizeIncrement">
   <size>
    <width>0</width>
    <height>0</height>
   </size>
  </property>
  <property name="baseSize">
   <size>
    <width>0</width>
    <height>0</height>
   </size>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="dirBtn">
  <property name="font">
   <font>
    <family>Calibri</family>
    <pointsize>14</pointsize>
   </font>
  </property>
  <property name="text">
   <string>Select</string>
```

```
     </property>
    </widget>
   </item>
  </layout>
 </item>
 <item>
  <layout class="QHBoxLayout" name="horizontalLayout"/>
 </item>
 <item>
  <widget class="QPushButton" name="runBtn">
   <property name="font">
    <font>
     <family>Calibri</family>
     <pointsize>14</pointsize>
     <weight>75</weight>
     <italic>false</italic>
     <bold>true</bold>
    </font>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: rgb(254, 255, 195);</string>
   </property>
   <property name="text">
    <string>Start Check</string>
   </property>
  </widget>
 </item>
</layout>
</item>
<item>
 <widget class="QPlainTextEdit" name="StatusInfo">
  <property name="font">
   <font>
    <family>Calibri</family>
```

```
        <pointsize>12</pointsize>
       </font>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
  <resources/>
  <connections/>
 </ui>
```