Dealing With Difficult Data

Do you have something to say about this article?
Visit the *Journal* pages of the ISACA® website *(www.isaca.org/journal)*, find the article and click on the Comments link to share your thoughts.

Applying data analytics (DA) techniques to the information systems (IS) internal audit process has moved from a desired activity to a required one. Data analytics can be applied to a wide variety of IS audit activities—annual risk assessment, analytical review of engagement to support planning, IS control testing and findings follow-up.[1, 2, 3, 4] Applying DA can help achieve better risk coverage and more cost-effective testing.

The failure of DA programs in internal audit is more frequent than desired. Reasons for lack of success are related to the root causes of:[5]

- Lack of support at the top
- Inability to access appropriate data in a timely manner
- Lack of talent
- Absence of success criteria, process and procedure
- Lack of embrace by the non-DA members of the department

Ensuring that the previously mentioned pitfalls do not derail an enterprise DA program requires a different degree of political, process and economic challenges to overcome among organizations. Enterprises that have established strong data governance programs that define standards for data structure and consistency can avoid many of the issues outlined in this article.

Michael T. Hoesing, CISA, ACDA, CDP, CFSA, CIA, CISSP, CMA, CPA
Rejoined the First Data internal audit team in July 2015, leading the build-out of the data analytics team. Hoesing's recent experiences include training and consulting as a master trainer for ACL Services Ltd., IS audit leader at the internal audit department of First National Bank of Nebraska, senior manager of operational and systems risk management for PricewaterhouseCoopers, audit director for American Express and First Data, and other positions in public accounting with McGladrey and other regional firms. Hoesing is also an adjunct instructor at the University of Nebraska at Omaha (USA), teaching the area's longest running IS audit class to graduate students.

Process breakdowns can reduce the impact of a DA program. Failure to validate the data can lead to inappropriate analysis and might be caused by:

- Lack of balancing and reconciling procedures to ensure data completeness and accuracy
- Not reviewing the sources from which the data are obtained to ensure that the data align with the audit objectives
- Not ensuring that tools and queries that are used to obtain the data are accurately applied to the data in scope

The inability to execute is another reason a DA program might fail or underdeliver. Not being able to complete a requested analysis diminishes the positive impacts of a DA program. One key roadblock to execution is the poor quality of data available for DA. Data can be more difficult to work with if they contain structural characteristics that some typical desktop tools might not be able to accommodate. Microsoft Excel is a common delivery vessel for providing data to auditors. The flexibility of Excel sets the stage for inconsistent content, which results in files that can be difficult to use with conventional DA tools.

Data quality challenges may be present at the file level, e.g., overall file size. Many audit data analysts are using desktop or laptop machines with limitations in memory resources, and sometimes larger files cannot be opened. Some data analysis software has limitations on the size of a record they can process. A file may arrive for analysis that is encoded using a format such as Unicode. That format may require a special version of software unavailable to the analyst. Analysts may be working in an environment that produces files from various platforms and operating systems that structure data differently.

Excel, while making it easy to convey data to auditors, can present data quality problems. Excel allows the flexible presentation of text by allowing line feeds and alignment tabs within a cell that may cause interpretation problems in analysis during import. Analysis tools frequently expect Excel

to contain the column titles in only row 1; titles or other narratives involving more than the first row of the sheet can make importing to analysis software difficult. Recipients of analysis output may have limitations on the types of files they may further process. The ability to change the type (e.g., CSV, XML) of the Excel content broadens the audience for the results. For example, when processing multiple Excel workbooks for different subsidiaries, inconsistent sheet names may hinder automated processes to import multiple files. Changing the sheet name within a workbook to a standard naming convention can facilitate automated loading of data. If an Excel sheet uses cell color to indicate content, that color cannot be interpreted or processed by analysis tools. Converting Excel cell colors into text strings can enable further processing.

Other source file issues that may need to be overcome before analysis can continue include:

- Device control characters such as tabs and line feed may cause interpretation issues during import for other file types, not just Excel.
- Report files deserve their own treatise, but a general approach, such as that discussed in this article, will provide help on many varieties of differently structured report files.
- Compressed data usually need to be uncompressed using tools and software that may be part of the original compression process before being analyzed with other downstream analysis software.

This article elaborates on these data quality issues and suggests techniques to overcome them.

File-level and Record-level Issues

Considering data arrive at the highest order of aggregation in files and techniques applied at the file level which may impact most records, the discussion will start at the file and record levels. Corrections at these levels usually must be done first, before the data can be brought into conventional analysis tools.

File Size Is Too Large

It is valuable to review data natively, using either the software with which the data were created (e.g., Excel) or using editors (e.g., Notepad++), before using the data in analysis tools. Some DA software tries to load an entire file for processing into memory. A very large file may exceed the memory limitations of the machine. If the data file is consistent in structure, reviewing a subset of records can help to confirm structure, such as delimiters and character qualifiers.

> **“It is valuable to review data natively, using either the software with which the data were created or using editors, before using the data in analysis tools.”**

Microsoft PowerShell is a Windows automation and scripting platform that is built on the .NET Framework.[6] The following PowerShell script streams in a file, regardless of its total size, and then extracts a subset of 100 records:

```
Get-Content “\\path\sourcefile.txt” | Select
-First 100

|Out-File –FilePath “\\path\sourcefile_
first_100_records.txt” -Encoding ASCII
```

The script commandlets and parameters work as follows:

1. Get-Content commandlet locates and opens the file that is specified between the quotes, i.e., *“\\path\sourcefile.txt”.*
2. | (the first pipe, or vertical bar) forwards the result of the previous commandlet Get-Content into the succeeding commandlet Select.
3. *-First* (the parameter to the commandlet Select) specifies how many records to obtain (100).

4. | (the second pipe) forwards the result of the two commandlets (100 records) to the commandlet Out-File, which creates a new file.
5. *-FilePath* (the parameter to the commandlet Out-File) specifies the location and name of the new file, i.e., *"\\path\sourcefile_first_100_records.txt"*.
6. *-Encoding* parameter specifies the new file type *(ACSII)*. Note: The PowerShell default file type when creating a result file is Unicode.

Long Record Lengths

A file may contain some records that contain a large number of bytes. The quantity of bytes may exceed the capacity limits that analysis tools enforce. For example, the 2016 version of Excel cannot have a character field with more than 32,767 bytes. Microsoft ACL for Windows 11.4.2 is a data analytics and analysis product and has a maximum-bytes-per-record limit of 32,767, although it can process files with as many records as the operating system allows. The following PowerShell script determines if any file record exceeds that limit. This script writes the first 25 characters of the excessive record to the screen and works best if the file contains a few offending records. If the screen output is empty, then the file has all of its records within the limit.

```
$data = Get-Content "<yourpath><yourfilename>"
foreach($line in $data)
{
      if ($line.Length -gt 32767) {
          Write-Host "A match"
          Write-Host $line.substring(1,25)
          }
}
```

The script commandlets, *parameters* and variables work as follows:

- $data—Variable to hold the contents of a file
- foreach—Cycles through the lines in the data file
- **if**—Logical test
- $line—Variable to hold the contents of each line
- *.Length*—Identifies the attribute of the number of bytes for each line processed
- *-gt*—Logical test for greater than
- Write-Host—Directs output to the screen
- *.substring*—Attribute of a segment of each line processed

Unicode File Encoding

Sometimes, Unicode files must be processed with special versions of software. It may be easier to work with the data in a tool that is designed to handle ASCII-encoded files. Converting a Unicode file to ASCII can be accomplished at the Windows command line, with the command:

TYPE:

C:\> TYPE *unicodefile* > *asciifile*

PowerShell achieves the same result with the following command (PowerShell creates its output as a Unicode file, unless the -encoding switch is used to produce an ASCII file):

PS> Get-Content *sourceunicodefile* | Out-File-Encoding ASCII *asciioutputfile*

Both commands create an additional copy of the data; however, if storage space is available, a copy of the source file should be made before

performing any of the enhancement techniques in this and succeeding sections to preserve the chain of evidence. Also, any derived file in this or the following sections should be balanced and reconciled to the system of record and have other validation techniques (e.g., malformed key fields, empty fields, omissions, duplicates, values out of range) applied as needed.

UNIX/Linux End-of-record Markers

Sometimes, a file has a line feed (LF) in the middle of a record that can disrupt data imports. LF is used in Linux/Unix systems to indicate an end-of-record marker. This situation can cause the import process to try to create two records when only one record is appropriate when the analysis software is processing a Windows file.

If the file is in a Windows format, the standard end-of-record marker is the two-byte carriage return and line feed (x0D and x0A). The process to remove the individual line feed (LF) involves two steps:

1. Find and replace all of the LF bytes into single blank spaces. This process also changes all of the end-of-record markers to CR (carriage return with a blank space).
2. Find and replace all of the CR instances with CRLF.

The text editor Notepad++ can be used to create a revised copy of the data after the source file backups are made. First, change the line feeds (**figure 1**):

1. In the View configuration, select Show Symbols then select Show all Characters. The carriage returns and line feeds are in reverse video.
2. Select the Search menu, then select Replace.
3. In the Find what box, enter \n. This specifies that the line feeds are to be located.
4. In the Replace with box, enter a single space using the keyboard spacebar.
5. Select the Extended radio button to enable working with the device-control character symbols (\n for line feed (LF) and \r for carriage return (CR)) rather than the literals.

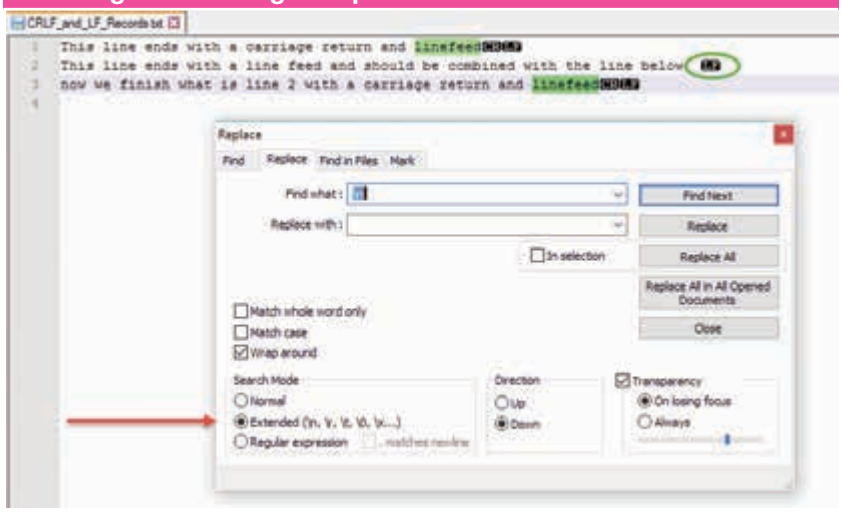
> **“Any derived file in this or the following sections should be balanced and reconciled to the system of record and have other validation techniques applied as needed.”**

6. Select Replace All. All line feeds are replaced with a space. The carriage returns remain.

After all the LF device-control characters are removed, the file contains one line per record. The three lines in **figure 1** are now two lines in **figure 2**, and each line is one record. Each line is terminated with a single character for carriage return (CR). To replace the CR with CRLF (**figure 2**):

1. In the View configuration, with the Replace panel open, select the Extended radio button.
2. In the Find what box, enter \r. This specifies that the single carriage returns are to be located.

Figure 1—Using Notepad++ to Remove All Line Feeds



Source: M. Hoesing. Reprinted with permission.

3. In the Replace with box, enter \r\n.
4. Select Replace All. All carriage returns are replaced with carriage returns and line feeds (CRLF), which is the Windows end-of-record designation.

The two lines in **figure 2** are two records that each conclude with the Windows end-of-record designation CRLF.

Excel Issues

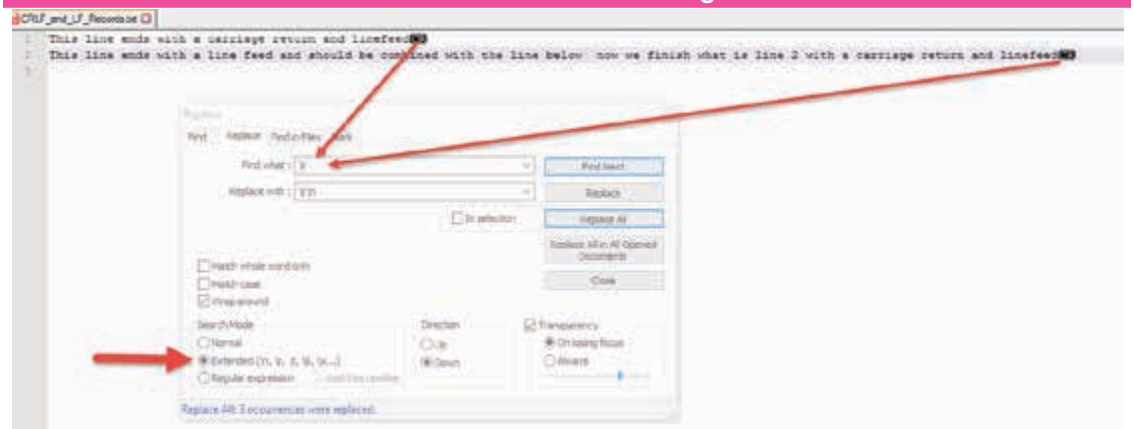
Excel is a popular and affordable desktop tool, and many applications that auditors wish to analyze have capabilities to export their data to Excel. The Excel file structure is a convenient way to gather audit evidence; however, it is important to understand the challenges this format can present to enable correction.

Device-control Formatting in Character Fields
Device-control characters, such as line feeds in an Excel worksheet within a text cell, may cause analysis software to break an Excel line into multiple records. In most cases, a line in an Excel sheet represents a complete record; breaking an Excel line in the middle can cause misinterpretation during import to an analysis tool. Also, device control characters inside a text field will cause the ACL VERIFY command to report an error in the imported table (**figure 3**). If the ACL solution is scripted, stop the script upon any VERIFY errors. If this situation is not considered to be a critical error worthy of stopping processing, it can be corrected in Excel prior to import:

1. If a worksheet has column B with line feeds created by using Alt+Enter, then insert a new column in front of B using the formula =CLEAN(C1).
2. Copy the formula down column C for every row of column B.
3. Select the data needed in the worksheet, including the new column and excluding the damaged column (column B, which contains line feeds).
4. Paste special values to a new worksheet.

The new worksheet will not have the device-control characters (e.g., tab, CR and LF) embedded within the cell content; therefore, downstream analysis software will not try to break a single record into multiple records during an import step, because the Excel structure will identify the termination of a record utilizing its proprietary line numbering rather than CRLF.

Figure 2—Using Notepad++ to Add CRLF Windows End-of-record Designation



Source: M. Hoesing. Reprinted with permission.

Multiple Header Lines
The expectation of most analysis software of an Excel worksheet is that row 1, optionally, contains the column or field titles. If rows exist above the first row of column titles, the rows above the column-title rows should be removed before import.

Removing these initial rows from one Excel worksheet can be achieved manually. However, with multiple workbooks and the same occurrence of non-title header rows repeated in each workbook, it may be more efficient to run a PowerShell script dynamically, substituting the Excel workbook names successively. This can be achieved within an ACL script by:

- Inventorying the Excel file names to import with a DIRECTORY command and redirecting the results to an ACL table
- Processing each file name in that table one at a time using a subscript and a unitary counter with the LOCATE RECORD command to sequentially address each file name
- Creating the Powershell script with the current file name within the subscript using ACL LIST UNFORMATTED commands to create a text file with a PS1 extension that can be batch processed by Powershell
- Running the Powershell .PSI batch script within ACL using the EXECUTE command to produce the clean version of the Excel file. A copy of the Excel file is made and cleaned to preserve the original file.
- Using the ACL IMPORT command to construct your ACL table layout and data
- Repeating until all listed files have been processed. Utilize the ACL command DO SCRIPT WHILE.

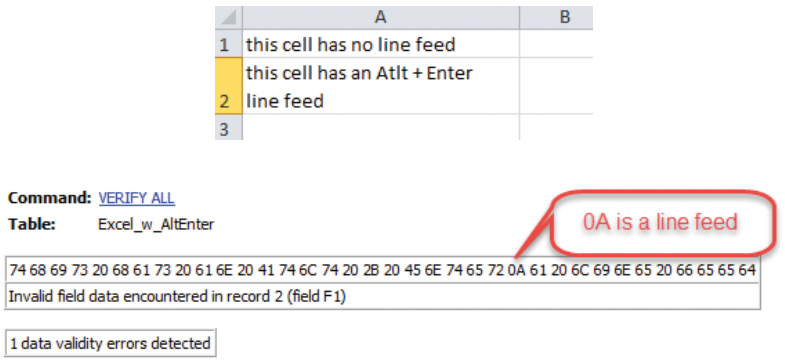
Key PowerShell commandlets to remove the number of extraneous lines at the start of the sheet would include:

- $Workbook = $xl.Workbooks.open($filepath) to open the current Excel workbook
- $ExcelWorkSheet = $xl.WorkSheets.item(1) to open the first sheet inside the workbook
- $ExcelWorkSheet.Cells.Item(1,1).EntireRow.Delete() to remove the first row of the sheet (repetitively). The ACL script requests the user to specify the number of header rows to delete above the title row. Within the Powershell script, this line is repeated as frequently as specified by the user. If there is a large number of leading lines to delete within the sheet, a foreach commandlet may be more appropriate.

Note: Row 1 is deleted repetitively, because row 2 becomes row 1 after the original row 1 is removed.

Retyping a Workbook
In some cases, extraction software cannot create an Excel file format and creates a comma-separated values (CSV) file format. After the CSV file is created, its file structure can be retyped to an Excel XLSX format to allow increased formatting and other features to be used within Excel. Sometimes export processes limit the width of a text cell when exporting directly to Excel, resulting in truncated data. Exporting to a CSV file type does not have as restrictive of a field-width requirement, reducing truncation instances. Use the following PowerShell

Figure 3—Error Report in Imported Table



Source: M. Hoesing. Reprinted with permission.

Enjoying this article?

- **Learn more about, discuss and collaborate on big data in the Knowledge Center.** *www.isaca.org/big-data*

commands to open a CSV file and re-save it as an Excel XLSX format:

```
$excel = New-Object -ComObject "Excel.
Application"
$excel.DisplayAlerts=$False
$excel.Visible=$False
$workbook = $excel.Workbooks.Open(
"<<yourpath>><<yourfilename>>.CSV")
$workbook.
SaveAs("<<yourpath>><<yourfilename>>
.XLSX", 51)
$excel.Quit()
```

The "51" value indicates an Open XML workbook file type.[7]

Renaming a Worksheet
An enterprise may need to normalize Excel worksheet names to run data analytics. For example, a standard Excel workbook with standard worksheet names is distributed by an enterprise's headquarters to dozens of subsidiaries. When the multiple workbooks are returned to the headquarters, it is found that they were modified by the end user, which makes aggregation difficult. The following PowerShell script commandlets can be added to an existing PowerShell script to normalize the worksheet names:

```
$ExcelWorkSheet  = $xl.WorkSheets.item(1)
$ExcelWorkSheet.Name = "Sheet1"
$xl.ActiveWorkbook.Save()
```

The first commandlet loads the first physical worksheet "…item(1)." Worksheets can also be addressed by their name if they are known and consistently structured by the end users.

The second commandlet defines the new name of the loaded worksheet, the final commandlet saves the Excel workbook and the revised worksheet name is saved as part of the workbook.

Turning Excel Cell Colors Into Data
Color added to worksheet cells may contain meaning and be visually appealing, but that meaning or context is not able to be interpreted during import into most analysis software. Usually, the cell colors have consistent meaning throughout the sheet (e.g., green = design, blue = development, orange = testing). Before import, the colors of worksheet cells may be determined and based on their color's numeric value. Additional text columns can be added with contextual descriptions. These descriptions can then be used later in analysis software for aggregation and filtering. PowerShell commandlets that achieve this include:

```
$a = $ExcelWorkSheet.Cells.Item($r,$i -
$NumberofColumns).Interior.ColorIndex

If ($a -eq 43) # Green
{
$ExcelWorkSheet.Cells.Item($r, $i) = 'Design'
```

The first commandlet interrogates the cell color (Interior.ColorIndex) and assigns that value to the variable $a. The next command checks the content of the variable $a (43 is green)[8] and assigns the new column and row cell content the text string ('Design') for that color. Using some nested "while" operators within the PowerShell script, the rows ($r) and the columns ($i) can be cycled through, making the changes progressively throughout the worksheet.

Other Issues

Special case file formats usually work well for the original purpose for which they were designed—Electronic Data Interchange (EDI) files for transmission consistency, report files for expedient transfer to printers and compressed files to conserve storage space. However, these files sometimes need special treatment either before use in, or within, general purpose tools.

Device-control Characters Outside of Excel
Hex editors can reveal and rectify some character codes within a file before they cause trouble. American Standard Code for Information Interchange (ASCII) codes x01 through x1F (1 through 31 decimal) are generally used to control devices and are normally not data. Replacing these device-control characters with a blank space (ASCII code x20, i.e., decimal 32) usually does not harm the data content and may be necessary for the file to properly process

import and verification commands in the data analysis software.

Challenging Report Files
Report files were designed to be pleasing to read. They contain white space, page and column headers, and items that are not data. Turning a report file into data depends on consistent physical placement of data horizontally and vertically on the report. Many import tools rely on this consistency to turn the report into data. Some software tools may have difficulty distinguishing the location of data if it drifts (i.e., left or right, or up or down) within the report.

Inconsistent report files may be approached with purpose-built software.[9] Inconsistent report files may also be brought into other data analysis tools, even if the wizards in those tools cannot handle the complexity of the report structure. These other tools may be able to import the report into one character field that is long enough to obtain the widest line of the report. After a report is loaded into the analysis software, parsing functions and commands can be used to identify the data segments, i.e., SPLIT, SUBSTR, LAST, REVERSE, which are usually used twice—once to approach the character string from the right, and again to restore the original order—within the long all-inclusive character field.

Unpacking a Compressed Certificate Revocation List
If one wishes to compare certificate serial numbers against serial numbers in a certificate revocation list file (.CRL file type), a preliminary step must be performed to decompress the native .CRL file. A machine with OpenSSL installed should have the utility module crlutil installed. That tool can be used to turn the compressed .CRL file into a text file, which can be imported into most data analysis tools. The –S switch (case sensitive) is to show, or display, the file. The –i switch designates the source .CRL file as input to decompress. The resulting file has Linux end-of-record markers (LF only). To unpack a compressed certificate revocation list file:

```
/usr/bin/crlutil -S -i "yourcrlfile.crl"  >
yourdecompressedcrlfile.txt
```

Conclusion

There are many technical options that are used to construct data files. Some of those options make the import and analysis of data challenging. This article listed several of those challenges and provided some ideas on overcoming the impediments and achieving enterprise audit objectives.

After the structural and consistency issues have been solved, the enhanced copy of the data file can be imported into analysis tools. Then, the resulting knowledge, synthesis and insights that are provided by the analysis can enhance the audit risk assessment and testing processes.

Endnotes

1 ISACA®, *Data Analytics—Practical Approach*, USA, 2011, *www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/Data-Analytics-A-Practical-Approach.aspx*
2 Lambrechts, A.; J. Lourens; P. Millar; D. Sparks; "GTAG16: Data Analysis Technologies," The Institute of Internal Auditors (IIA), August 2011, *www.theiia.org/guidance/technology/gtag-16/?sf2002075=1*
3 Stippich Jr., W.; B. Preber; *Data Analytics: Elevating Internal Audit's Value*, IIA, April 2016, *https://bookstore.theiia.org/data-analytics-elevating-internal-audits-value*
4 Verver, J.; *Workbook for a Successful Audit Analytics Program*, High Water Advisors, April 2016, *www.highwateradvisors.com/workbook-for-a-successful-audit-analytics-program*
5 *Op cit,* Lambrechts
6 Microsoft, "What is PowerShell?" 2016, *https://msdn.microsoft.com/en-us/powershell/mt173057.aspx*
7 Microsoft, "XlFileFormat Enumeration (Excel)," *https://msdn.microsoft.com/en-us/library/office/ff198017.aspx*
8 Chisholm, R.; "Adding Color to Excel 2007 Worksheets by Using the ColorIndex Property," Microsoft, February 2008, *https://msdn.microsoft.com/en-us/library/cc296089(v=office.12).aspx*
9 Datawatch Monarch software is designed to turn report files into data, *www.datawatch.com.*