

## The “Unveiled” secret of “Windows Batch Scripting”

Nicolas Rouillé, Cytel, Geneva, Switzerland

Laura Phelan, Cytel, Paris, France

### ABSTRACT

A batch file is a file that contains a sequence, or batch, of commands. Batch files are useful for storing sets of commands that are always executed together because you can simply enter the name of the batch file, or double click on it, instead of entering each command individually. For example entering a copy or rename command.

For example, the Windows Batch Script language can be used with the interpreter cmd.exe, which is the default interpreter on Windows® platforms: NT, XP, Vista, 7 and later.

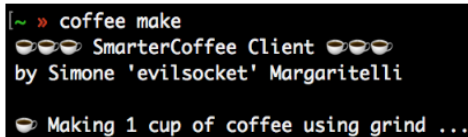
Taking the example of a repetitive task, such as the run of several SAS programs at once, we would like to introduce the basic of Windows Batch Scripting language. The example will show the following:

- Select a set of programs by dragging the selected programs into the icon representing the batch program
- Get automatically the name of the files “passed” to the script and generate a loop
- Within the loop:
  - Batch submit each selected SAS program
  - Prompt the user with the execution status
  - Promptly inform the user if the program generated some errors or warning by analyzing the SAS log
- Prompt the user when the batch is terminated with some metrics e.g. elapsed time

Other examples include creating automatic backups of your study task folders, running P21® in batch, creating a SAS® program standard layout (e.g. with standard header with automatically filled).

### INTRODUCTION

There once was a programmer, Simone Margaritelli, a researcher at Zimperium, who hacked his coffee machine to brew coffee using the command line:



```
[~ >> coffee make
☕ SmarterCoffee Client ☕
by Simone 'evilsocket' Margaritelli
☕ Making 1 cup of coffee using grind ...
```

Another “lazy” programmer wrote some scripts to get his jobs done: Nihad Abbasov, contributing as “Narkoz” on GitHub ; he created some scripts with funny names, including one which sent automatic emails with message like “not feeling well/working from home” if no “interactive sessions on the company server at 8:45am” with his login, even selecting random reasons from a pre-defined list.

While we may not have the audacity of the latter programmer, we hope to show how the use of batch commands can make everyday tasks one keystroke away.

### WHAT IS WINDOWS BATCH SCRIPTING?

From Wikipedia, the free encyclopedia:

*“A batch file is a kind of script file in DOS, OS/2 and Microsoft Windows. It consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file. A batch file may contain any command the interpreter accepts interactively and use constructs that enable conditional branching and looping within the batch file, such as IF, FOR, and GOTO labels. The term “batch” is from batch processing, meaning “non-interactive execution”, though a batch file may not process a batch of multiple data.”*

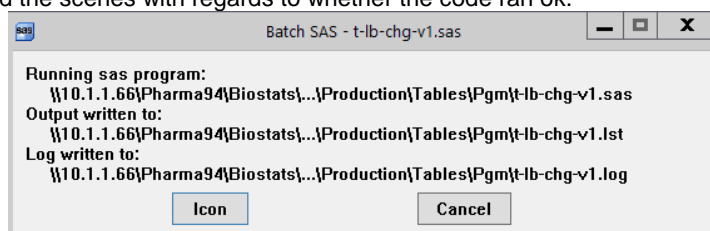
In other words, a batch file is a text file that “batches” (bundles or packages) into a single file, a set of commands that would otherwise have to be presented to the system interactively from a keyboard one at a time, so that multiple actions can be automated on your computer via a simple mouse-click.

# PhUSE EU Connect 2018

## BUILDING A SCRIPT WITH PRACTICAL EXAMPLES

### BATCH RUNNING A SET OF PROGRAMS

We are all familiar with the “Batch Submit with SAS 9.x...” option, which can be handy in its own right but all the action is a little bit behind the scenes with regards to whether the code ran ok.



Users have to manually (groan) open the log, and manually (double groan) search for errors, warnings or other suspicious notes like uninitialized or not referenced. Plus, and this may be more subjective, the log is stored in the same folder as the program which may not follow your company’s standard practice – requiring yet another “manual” action to move the log?

Running in batch SAS programs could also avoid dependency from one’s user profile, which would have been loaded during an interactive SAS session – typically templates inappropriately stored in a sasuser library ; to execute in batch the SAS programs becomes then particularly relevant for a production run.

This is one scenario where Batch scripting can come to the rescue by running one SAS program or a selection of SAS programs, storing the logs in the correct folder, searching the logs for keywords and giving the user a summary of what it found in the logs.

But before we get to that, you may want to back up what is in your folder – in case things go unexpectedly wrong! Why not use batch to manage this as well? Let’s start by a quick introduction to some of the basic script commands.

### CRASH COURSE ON SOME BATCH SCRIPTING COMMANDS

Script	Action
Echo	display as text anything written after echo
@echo off	removes all the paths where the actions are being performed – makes execution/messages easier to read
Pause	batch “pauses” after execution so user can see what happens (unlike the SAS batch run, for example..)
echo.	may be used to display a blank line, between user messages for more elegant presentation, for example
:top (script actions...) goto top	Example of a loop
set	creates a variable, e.g. set name=Laura, and to resolve what the name is surround it with % %, e.g. echo %name% would display Laura
/a	to get mathematical interpretation
== or EQU	is equals, e.g. if %you%==1 echo %v1% - if the variable <b>you</b> equals <b>1</b> then the variable v1 will display
NEQ	is not equals
rem	use to ignore everything on the line following the rem, including quotes, redirection symbols, and other commands
%~d1%~p1	active folder

### MAKING A BACKUP FOLDER SCRIPT

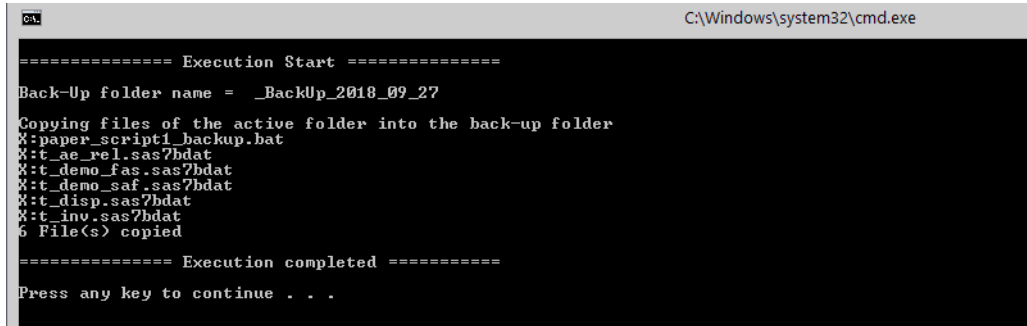
So, let’s get back to backing up our folder (see Script example 1). Based on the above, we can define a subfolder including the date in a specific format as the name, create that subfolder via `mkdir`, and copy our files to there via `xcopy *.*`. Now, we are fussy, and we like our subfolders to be chronologically sorted so we prefer the YYYY\_MM\_DD naming convention compared to the %date which resolves as “day MM/DD/YYYY”. Not unlike the SAS substr function with concatenation, we can select and reorder the date parts using the minus sign to read from the end of the date variable to get for example the year: %date:~-4,4%=2018.

## PhUSE EU Connect 2018

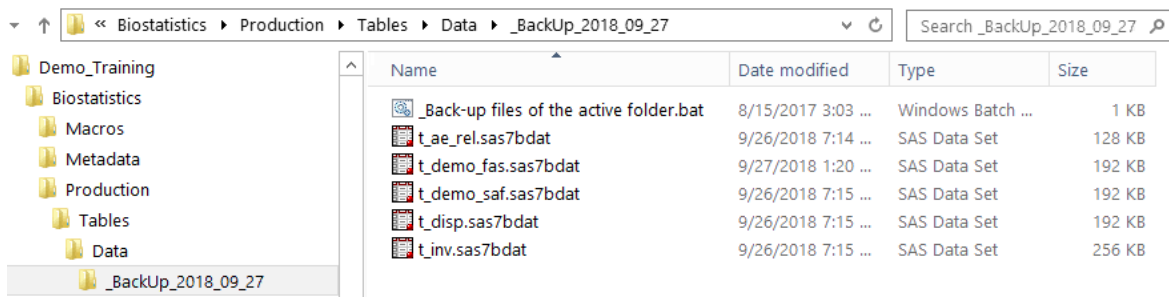
### Script example 1: Backup folder contents

```
ECHO Back-Up folder name = _BackUp %date:~-4,4% %date:~-10,2% %date:~7,2%
mkdir _BackUp_%date:~-4,4%_%date:~-10,2%_%date:~7,2%
ECHO.
ECHO Copying files of the active folder into the back-up folder
xcopy *.* _BackUp_%date:~-4,4%_%date:~-10,2%_%date:~7,2% /y
```

And in so doing, by running our script, we have backed up our files and are now covered in case of accidental file overwrite or removal.



```
=====  
Execution Start  
=====  
Back-Up folder name = _BackUp_2018_09_27  
Copying files of the active folder into the back-up folder  
X:paper_script1_backup.bat  
X:t_ae_rel.sas7bdat  
X:t_demo_fas.sas7bdat  
X:t_demo_saf.sas7bdat  
X:t_disp.sas7bdat  
X:t_inv.sas7bdat  
6 File(s) copied  
=====  
Execution completed  
=====  
Press any key to continue . . .
```



Name	Date modified	Type	Size
Back-up files of the active folder.bat	8/15/2017 3:03 ...	Windows Batch ...	1 KB
t_ae_rel.sas7bdat	9/26/2018 7:14 ...	SAS Data Set	128 KB
t_demo_fas.sas7bdat	9/27/2018 1:20 ...	SAS Data Set	192 KB
t_demo_saf.sas7bdat	9/26/2018 7:15 ...	SAS Data Set	192 KB
t_disp.sas7bdat	9/26/2018 7:15 ...	SAS Data Set	192 KB
t_inv.sas7bdat	9/26/2018 7:15 ...	SAS Data Set	256 KB

### CLEANING SUBFOLDERS SCRIPT

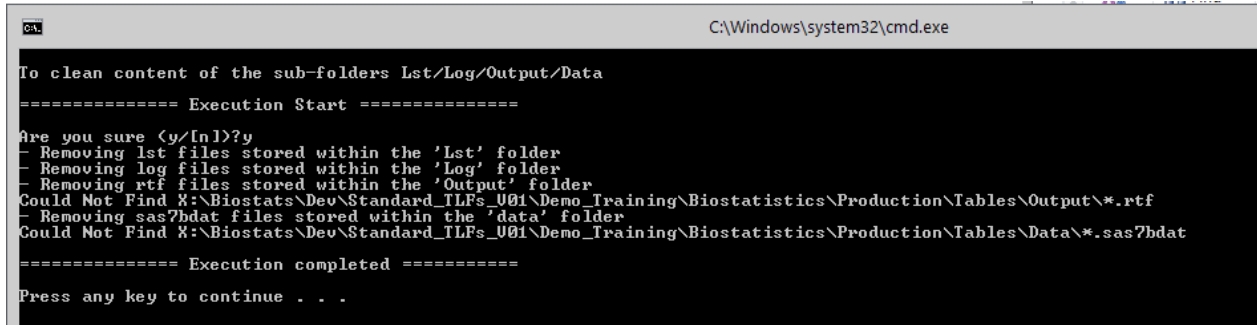
Like we said, we are fussy. We also like to keep things clean, and before we test running any programs with this new script, we'd like remove all leftover outputs and logs, etc., without manually going from folder to folder, selecting files and deleting them - and risk removing the wrong files in the process. So we made a batch script to do it for us.

```
@ECHO off
ECHO.
ECHO.To clean content of the sub-folders Lst/Log/Output/Data
ECHO.
ECHO ===== Execution Start =====
ECHO.
SET /P AREYOUSURE=Are you sure (y/[n])?
IF /I "%AREYOUSURE%" NEQ "y" GOTO END
ECHO - Removing lst files stored within the 'Lst' folder
del "%~dp0\..\Lst\*.lst" /f /q
ECHO - Removing log files stored within the 'Log' folder
del "%~dp0\..\Log\*.log" /f /q
ECHO - Removing rtf files stored within the 'Output' folder
del "%~dp0\..\Output\*.rtf" /f /q
ECHO - Removing sas7bdat files stored within the 'data' folder
del "%~dp0\..\Data\*.sas7bdat" /f /q
ECHO.
:END
ECHO ===== Execution completed =====
ECHO.
PAUSE
```

First we SET or create the variable AREYOUSURE, because we've had feedback in the past that people were not so sure they wanted to delete all their work, understandably. The /P is a sort of switch which allows the developer to set

## PhUSE EU Connect 2018

the value of a variable to a line of input entered by the user, i.e. the user needs to answer the question "Are you sure (y/n)?" with a y or n before any files are removed. The /I may be used to ignore case sensitivity here. If they answer "n", the script will go to the end (GOTO END) and not perform any action. Otherwise, it moves to the folders of interest and deletes the file types we have specified in those folders, e.g. logs in the Log folder, lsts in the Lsts and so on. The "%~dp0" variable when referenced within a Windows batch file will expand to the drive letter and path of that batch file, so when the script is run from a specific folder, in our case we run from a Pgm sub-folder, del "%~dp0\..\Lst\\*.lst" /f /q moves up and then down a level to the Lst sub-folder and removes all lst files it finds there, ignoring any read-only setting (/f) and without prompting (/q for "quiet mode").



```

C:\Windows\system32\cmd.exe

To clean content of the sub-folders Lst/Log/Output/Data
==== Execution Start =====
Are you sure (y/[n])?y
- Removing lst files stored within the 'Lst' folder
- Removing log files stored within the 'Log' folder
- Removing rtf files stored within the 'Output' folder
Could Not Find X:\Biostats\Dev\Standard_TLFs_U01\Demo_Training\Biostatistics\Production\Tables\Output\*.rtf
- Removing sas7bdat files stored within the 'data' folder
Could Not Find X:\Biostats\Dev\Standard_TLFs_U01\Demo_Training\Biostatistics\Production\Tables\Data\*.sas7bdat
==== Execution completed =====
Press any key to continue . . .
```

### DRAG AND DROP PROGRAM(S) RUN BATCH SCRIPT

Stepping the pace up a bit, to run a program or set of programs in batch, saving the log and lst in their appropriate folders, we could use the following:

```
@ECHO off
ECHO ===== batch BEGIN =====
ECHO.
ECHO Batch execution started at %Time%
ECHO Detection of the active folder = %~d1%~p1

if exist "%~d1%~p1\log" (set LogFolder=%~d1%~p1\log) else if exist "%~d1%~p1..\log" (set
LogFolder=%~d1%~p1..\log) else (set LogFolder=%~d1%~p1)
if exist "%~d1%~p1\lst" (set LstFolder=%~d1%~p1\lst) else if exist "%~d1%~p1..\lst" (set
LstFolder=%~d1%~p1..\lst) else (set LstFolder=%~d1%~p1)

set /A NbProgsRan=0

FOR %%A IN (*) DO (

set /A NbProgsRan+=1

ECHO '%%~nA%%~xA' Execution is ongoing
"D:\SAS94\SASHome\SASFoundation\9.4\sas.exe" -sysin "%~d1%~p1%%~nA%%~xA" -log
"%LogFolder%\%%~nA.log" -print "%LstFolder%\%%~nA.lst" -icon -nosplash -rsasuser
ECHO '%%~nA%%~xA' Execution completed at %Time%
ECHO.
ECHO.

ECHO.
)
ECHO.
ECHO ===== batch END =====
PAUSE
```

First the log and lst folders need to be set: if they are subfolders in the folder from where the program is run, %~d1%~p1\log or in another subfolder at the same level as the program %~d1%~p1..\log, or neither, in which case they are stored with the program.

Every program dropped into the batch script, FOR %%A IN (\*) DO ( is then run  
"D:\SAS94\SASHome\SASFoundation\9.4\sas.exe" -sysin "%~d1%~p1%%~nA%%~xA".  
with their corresponding log/lst saved in the LogFolder/Lstfolder  
-log "%LogFolder%\%%~nA.log" -print "%LstFolder%\%%~nA.lst"

## PhUSE EU Connect 2018

```
===== batch BEGIN =====
Batch execution started at 11:24:05.86
Detection of the active folder = X:\Biostats\Dev\Standard_TLFs\001\Demo_Training\Biostatistics\Production\Tables\Pgm\
't-ae_DEMO-fmtppt.sas' Execution is ongoing
't-ae_DEMO-fmtppt.sas' Execution completed at 11:24:08.85
===== batch END =====

Press any key to continue . . .
```

We'd also however like the script to now tell us what happened in the logs. We can do this by adding a log scan for messages of interest,

```
set /A TotalNbOfLogERROR=0
set /A TotalNbOfLogWARNING=0
...
ECHO --- Log scanning - messages are displayed by decreasing order of criticality:
findstr /A:4F /n "ERROR fatal" "%LogFolder%\%%~nA.log"
findstr /A:2F /n "WARNING uninitialized unequal" "%LogFolder%\%%~nA.log"
findstr /A:1F /n /c:"MERGE statement has more than" "%LogFolder%\%%~nA.log"
ECHO.
)
and summarize the number found using,
rem --- to compute the cumulative number of ERROR or WARNING message(s) in the log files
if exist "%LogFolder%\%%~nA.log" (
    for /f "tokens=*" %%j in ('findstr "ERROR" "%LogFolder%\%%~nA.log"') do @SET /A
    TotalNbOfLogERROR += 1
    for /f "tokens=*" %%j in ('findstr "WARNING" "%LogFolder%\%%~nA.log"') do @SET /A
    TotalNbOfLogWARNING += 1
)
```

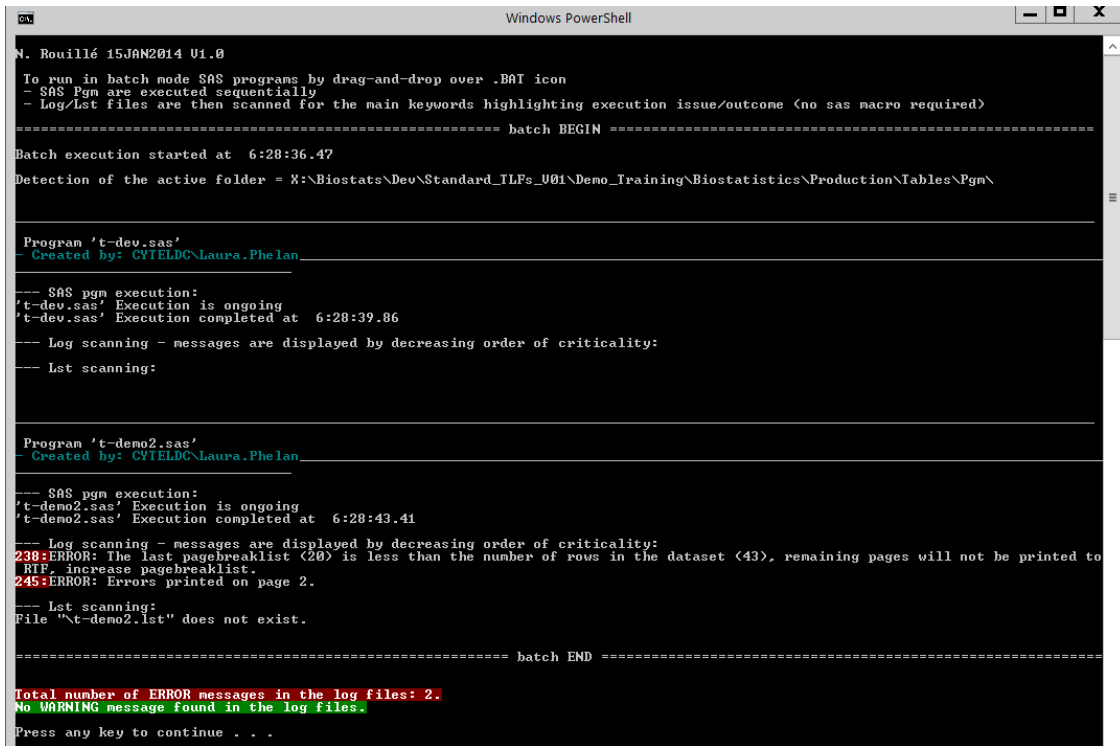
By adjusting color we can highlight for users any important messages found, we liked red for errors or warnings found, and green if there were none, following the color codes:

0 = Black	8 = Gray
1 = Blue	9 = Light Blue
2 = Green	A = Light Green
3 = Aqua	B = Light Aqua
4 = Red	C = Light Red
5 = Purple	D = Light Purple
6 = Yellow	E = Light Yellow
7 = White	F = Bright White

For example:

```
if "!TotalNbOfLogERROR!" gtr "0" (
    rem Call :Color 4F "FAILED:"
    rem ECHO. Total number of ERROR messages in the log files: !TotalNbOfLogERROR!
    powershell.exe "(Write-Host -NoNewLine -BackgroundColor ""DarkRed"" -ForegroundColor
    ""White"" Total number of ERROR messages in the log files: !TotalNbOfLogERROR!. )"
    echo.
)
```

## PhUSE EU Connect 2018



```
N. Rouillé 15JAN2014 U1.0
To run in batch mode SAS programs by drag-and-drop over .BAT icon
- SAS pgm are executed sequentially
- Log/Lst files are then scanned for the main keywords highlighting execution issue/outcome (no sas macro required)
===== batch BEGIN =====
Batch execution started at 6:28:36.47
Detection of the active folder = X:\Biostats\Dev\Standard_ILFs_U01\Demo_Training\Biostatistics\Production\Tables\Pgm\

Program 't-dev.sas'
- Created by: CYTELDC\Laura.Phelan

--- SAS pgm execution:
't-dev.sas' Execution is ongoing
't-dev.sas' Execution completed at 6:28:39.86
--- Log scanning - messages are displayed by decreasing order of criticality:
--- Lst scanning:

Program 't-demo2.sas'
- Created by: CYTELDC\Laura.Phelan

--- SAS pgm execution:
't-demo2.sas' Execution is ongoing
't-demo2.sas' Execution completed at 6:28:43.41
--- Log scanning - messages are displayed by decreasing order of criticality:
230:ERROR: The last pagebreaklist <20> is less than the number of rows in the dataset <43>, remaining pages will not be printed to
RIP. increase pagebreaklist.
245:ERROR: Errors printed on page 2.
--- Lst scanning:
File "t-demo2.lst" does not exist.
===== batch END =====

Total number of ERROR messages in the log files: 2.
No WARNING message found in the log files.
Press any key to continue . . .
```

The full script can be found in the appendix.

## OTHER EXAMPLES

### PROGRAM CREATION WITH STANDARD HEADER

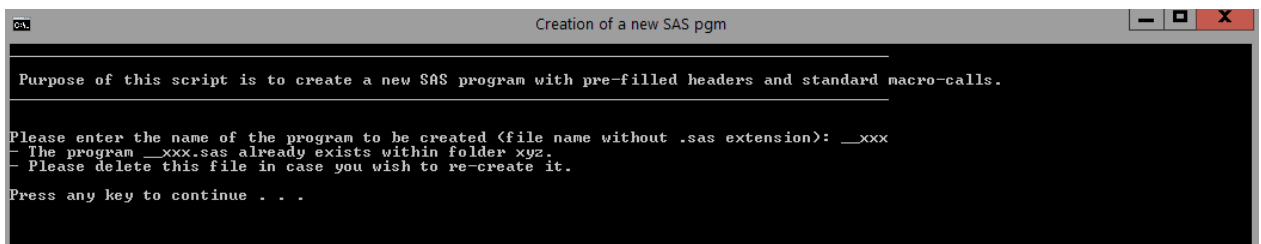
Because we like to follow Good Programming Practice, “A standard header should be used for every program”, we again use batch script to do the work (see the appendix for the full script).

Using again the switch

```
/p SASfile=Please enter the name of the program to be created (file name without .sas extension):,
```

users are prompted to enter a program name. To avoid unintentional overwriting of an existing program, one could add a check if the file already exists, or will receive a warning if the program already exists:

```
if exist %SASfile%.sas goto :FileAlreadyExists.
...
:FileAlreadyExists
ECHO - The program %SASfile%.sas already exists within folder xyz.
ECHO - Please delete this file in case you wish to re-create it.
```



```
Creation of a new SAS pgm

Purpose of this script is to create a new SAS program with pre-filled headers and standard macro-calls.

Please enter the name of the program to be created (file name without .sas extension): xxx
- The program xxx.sas already exists within folder xyz.
- Please delete this file in case you wish to re-create it.
Press any key to continue . . .
```

Details to include in the program header and body can be sent to the sas program (defined in **SASfile**) via all commands starting with @echo and ending with >> "%SASfile%.sas". The directory in which the program is created can be added using %CD% if desired.

Macro calls must be masked with a leading %, e.g.

```
@echo %%getdirf(flvl=4);>> "%SASfile%.sas"
```

Similarly, comments can be included in the body of the program using a leading % symbol, i.e.

```
@echo %%*--- at output level: save lst file if generated;>> "%SASfile%.sas"
```

## PhUSE EU Connect 2018

To ensure the FileAlreadyExists message is not displayed even when a new program is being created, a goto may be deployed to skip it goto :DONE... (:FileAlreadyExists....) :Done.

```

_XXX.SAS x |
dm 'log; clear; lst; clear;';
*****;
* PROJECT/PATH : X:\Biostats\Dev\Standard_TLFs_V01\Demo_Training\Biostatistics\Production\Tables\Pgm\
* CREATION DATE: Wed 09/26/2018 (mm/dd/yyyy)
* PURPOSE      : Programming of output(s) according to project spreadsheet
*****;

%*--- global settings;
%getdirf(flvl=4);
%*--- at output level: save lst file if generated;
%SaveLst;

%*--- end of program -----;

```

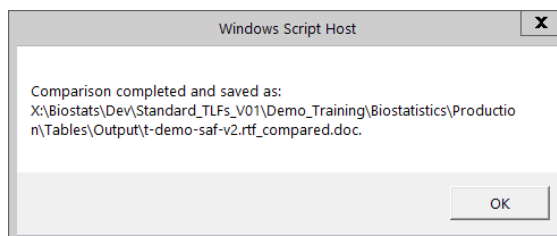
### VBS SCRIPT TO TRACK CHANGES OF RTF/DOC FILES ON-THE-FLY

As a programmer, sometimes you need to make a small adjustment in your ADaM, and rerun the associated outputs. The person on the receiving end, be it a statistician or medical writer or other, may wish to know the exact differences in the outputs they've just reviewed (for example difference between draft and final versions), or even gone as far as including into a study report. Something like this may be appreciated more than the not so user-friendly proc compare?

14.1.6: Demographic and Baseline Characteristics  
Table 14.1.6.2: Safety Analysis Set

	Placebo (N=140)	Trt1 20mg (N=98)	Trt1 60mg (N=194)	Trt2 60mg (N=52)	All Subjects (N=484)
Age (Years)					
n (missing)	140 (0)	98 (0)	194 (0)	52 (0)	484 (0)
Mean (SD)	40.6 (10.1)	38.1 (9.8)	40.2 (10.1)	38.7 (9.7)	39.7 (10.0)
Median	40.5	38.0	41.0	38.0	40.0
Q1 ; Q3	34.0 ; 48.0	31.0 ; 45.0	32.0 ; 48.0	32.5 ; 44.0	33.0 ; 47.0
Min ; Max	16.4 ; 64	19 ; 58	19 ; 62	21 ; 58	18 ; 64
Age Categories, n(%)					
n (missing)	140 (0)	98 (0)	194 (0)	52 (0)	484 (0)
<25	11 ( 7.9)	12 ( 12.2)	22.3 ( 6.7)	5 ( 9.6)	41 ( 8.5)
25-39	53 ( 37.9)	45 ( 45.9)	72 ( 37.1)	22 ( 42.3)	192 ( 39.7)
40-49	49 ( 35.0)	28 ( 28.6)	70 ( 36.1)	14 ( 26.9)	161 ( 33.3)
50-59	24 ( 17.1)	13 ( 13.3)	37 ( 19.1)	11 ( 21.2)	85 ( 17.6)
>=60	3 ( 2.1)	0 ( 0.0)	2 ( 1.0)	0 ( 0.0)	5 ( 1.0)

This example is managed by a VBS code which is having 2 arguments, i.e. the 2 files selected that the user wants to compare. The script is opening in hidden mode the first file with MS Word, and is making use of the track changes facility to open the second file and proceed to the comparison between the 2 files. The differences are highlighted and saved in a new doc file as displayed above; the MS Word session opened temporarily is then closed so the user would only see on the screen one window at the completion of the execution, as following:



The comparison could be done on any file that MS Word is able to open, that is including txt files: so this script can also be used advantageously to compare the 2 versions of one SAS program.

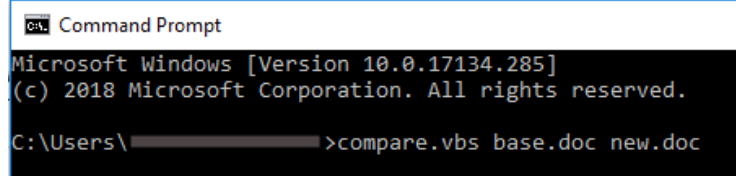
The full script can be found in the appendix.

### CALLING SCRIPTS

One of the selling-point of scripts like these is that they can be called or launched in a variety of ways, at least one of which should satisfy the most fastidious of users. These principles could apply to any script.

## PhUSE EU Connect 2018

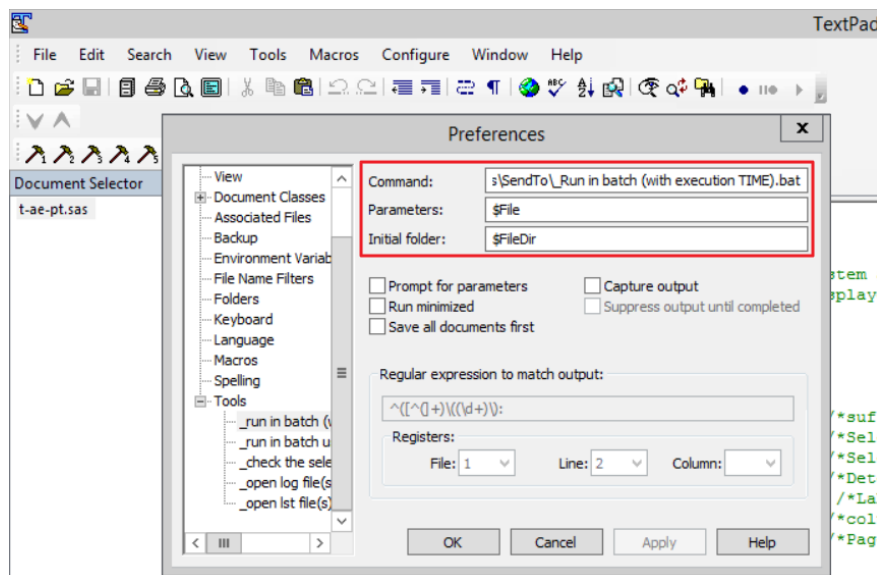
- Scripts can be called from a command line and passing arguments:



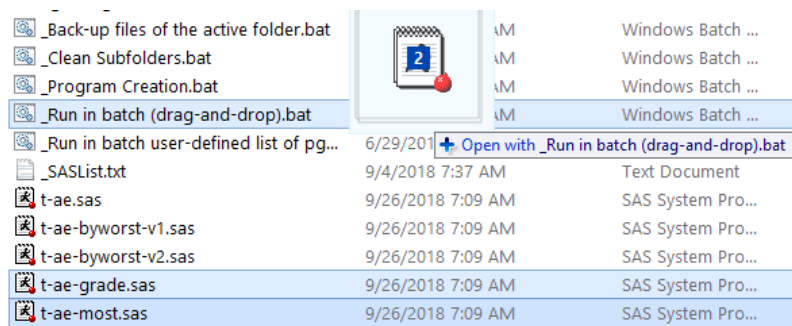
```

C:\Users\[redacted]>compare.vbs base.doc new.doc
  
```

- They can also be called from other software,
  - for example from SAS, by calling the script and passing arguments, which can then be looped in a macro over a selection of files (docs, .rtfs..), though you don't *have to* open software like SAS to use them.
  - Other useful software can live in perfect symbiosis with scripts: they are text editors like UltraEdit® or TextPad®. For example, the execution of one SAS program, the check of its log, the look to its associated lst file, can be done by calling script(s) declared in the 'Preferences' menu as shown hereinafter in this TextPad screenshot:



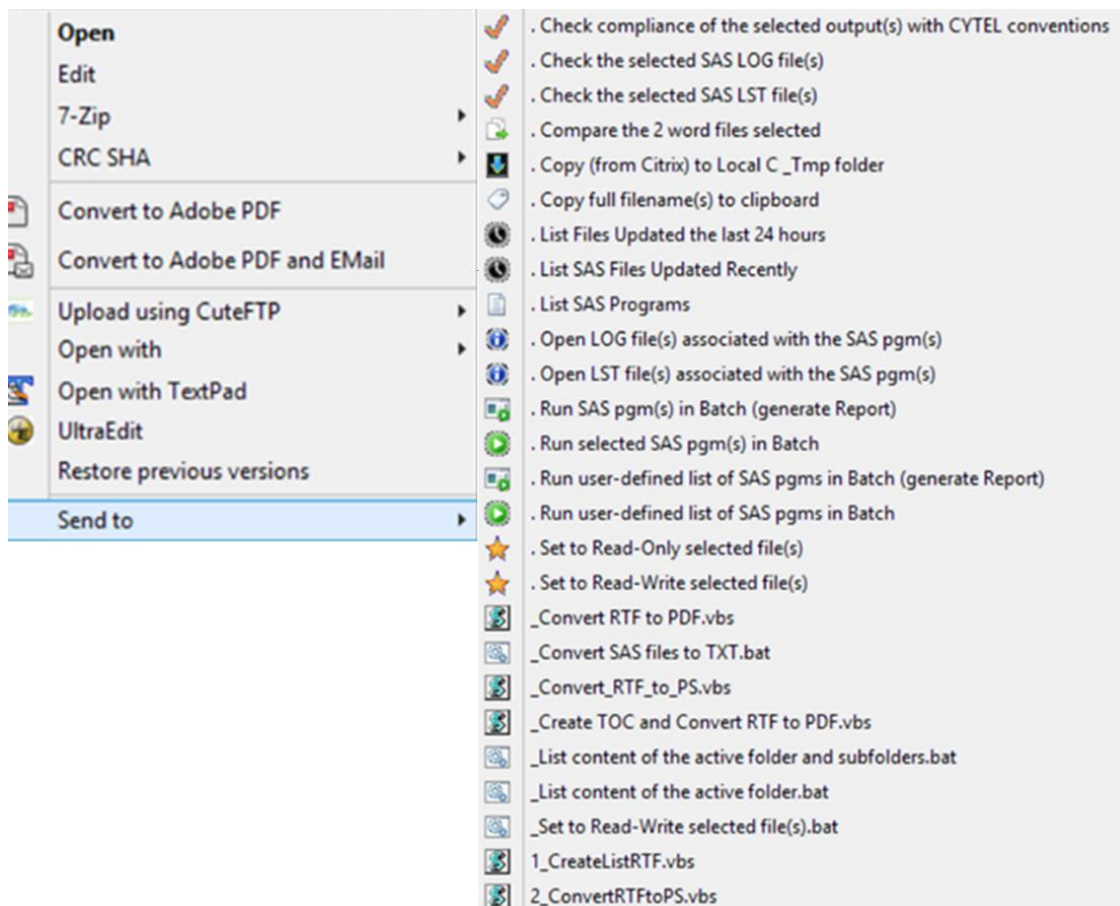
- Taking the run program script mentioned earlier, this is called simply by dragging-and-dropping the arguments (=a selection of files!) over the .bat script, directly from the window explorer interface:



- Another option is via the windows explorer 'send to' menu. Take the compare script we covered, it is copied to the users local profile e.g. [drive]:\Users\[username]\AppData\Roaming\Microsoft\Windows\SendTo. Back at the server, the 2 outputs to compare are selected, right-clicked and the user scrolls to the script action, in this case "Compare the 2 word files selected".



## PhUSE EU Connect 2018



### ADVANTAGES OF USING BATCH SCRIPTS

Among the reasons for getting familiar with and using scripts, is that they are:

- known for their portability and low friction, being
  - versatile, with low dependence upon new release of platforms
  - 'universal', recognized by any windows system
- easy to interact and integrate within other software
- very well documented - lots of resources are available
- commonly learned at school - so they are easy to investigate, and suitable for a broader audience
- particularly interesting to manipulate files: copy, find , rename
- ideal to automate routine tasks and save time

### IS MS-DOS NOW OBSOLETE?

MS-DOS enthusiasts are fairly worrying at every new release of Microsoft platforms that the cmd.exe interpreter would not come by default - as it is for now since ... more than 3 decades! The direction taken by Microsoft is indeed to encourage the use of PowerShell, of VB script, of JavaScript: they are object-oriented, better structured.

For now the coexistence of all is allowing for instance to use PowerShell commands to gather attributes of some files (owner, dates of creation, etc.) within a .bat file containing also DOS commands; you can refer to the appendix listing our 'drag and drop program(s) run batch script' in order to see an illustration of this use.

### OTHER WINDOWS SCRIPTING LANGUAGES

Of course there are other scripting languages out there that need to be carefully considered depending on your environment and strategy; Wikipedia is quoting the following:

- Windows Script Host (.vbs, .js and .wsf) - released by Microsoft in 1998, and consisting of cscript.exe and wscript.exe, runs scripts written in VBScript or JScript. It can run them in windowed mode (with the

## PhUSE EU Connect 2018

wscript.exe host) or in console-based mode (with the cscript.exe host). They have been a part of Windows since Windows 98.

- PowerShell (.ps1) — released in 2006 by Microsoft and can operate with Windows XP (SP2/SP3) and later versions. PowerShell can operate both interactively (from a command-line interface) and also via saved scripts, and has a strong resemblance to Unix® shells.
- Unix-style shell scripting languages can be used if a Unix compatibility tool, such as Cygwin, is installed.
- Cross-platform scripting tools including Perl®, Python®, Ruby®, REXX®, Node.js and PHP® are available for Windows.
- Since 2016, active release and support of Windows Subsystem for Linux (WSL) upon which several Linux distros now run; WSL aims for native Linux compatibility.

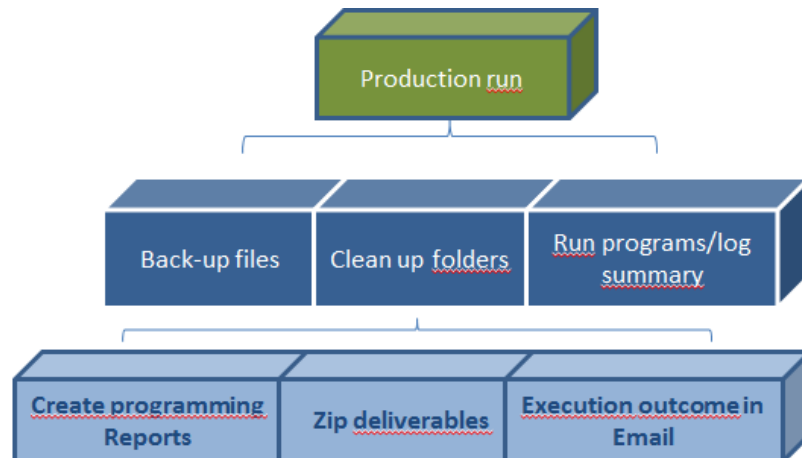
### CONCLUSION

Following the thought process involved as described in this paper, you can use scripts as stepping stones to build a bigger process. Whatever could be the scripting language of your choice, start with considering a script as a unit of task:

- Split the process, identify routine tasks
  - e.g. Back-up / Clean-up / Program Run / Log check / Summary
- Select most appropriate programming language, ideally, trying to do abstraction:
  - from other languages
  - from specialized software (code resources will be more limited)
- Build each process like a wall: modules can be added brick by brick to build a complete 'solution',
- Take care of the user experience
  - in requesting minimal action from his side (e.g. by counting the number of mouse clicks requested to perform one operation) ,
  - and in prompting his input appropriately: for instance confirmation before a critical action is performed.
- Interface your program with other software (text editors, call from SAS with the x command, etc.), so their integration is optimal
- Be minimal in the writing of a program. for maintenance and ease of updates

➔ ... And then, the scripts will follow you in time!

An organized library of scripts can result in an entire solution, e.g. automating all the sequence of a production run:



## PhUSE EU Connect 2018

### REFERENCES

Narkov scripts: <https://fossbytes.com/this-lazy-programmer-wrote-some-scripts-to-secretly-automate-a-lot-of-his-job/>  
Windows batch scripting: [https://en.wikipedia.org/wiki/Batch\\_file](https://en.wikipedia.org/wiki/Batch_file)  
Guide to Windows Batch Scripting: <http://steve-jansen.github.io/guides/windows-batch-scripting/>  
DosTips - The DOS Batch Guide: <https://www.dostips.com/>  
Running Windows or MS-DOS Commands from within SAS:  
<http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#exittemp.htm>  
SAS Batch processing under Windows:  
<https://communities.sas.com/t5/SAS-Communities-Library/Batch-processing-under-Windows/ta-p/475977>

### ACKNOWLEDGMENTS

We would like to also thank Remi Gerin for his “laziness” and “fussiness” leading to several script developments that has made the lives of our programming teams easier.

### CONTACT INFORMATION

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

Nicolas Rouillé  
Cytel Inc.  
Route de Pré-Bois 20  
1215 Geneva – Switzerland  
Email: [nicolas.rouille@cytel.com](mailto:nicolas.rouille@cytel.com)

Laura Phelan  
Cytel France Sarl  
63, Avenue des Champs Elysées  
75008 Paris, France  
Email: [laura.phelan@cytel.com](mailto:laura.phelan@cytel.com)

Web: [www.cytel.com](http://www.cytel.com)

Brand and product names are trademarks of their respective companies

# PhUSE EU Connect 2018

## APPENDIX

### FULL SCRIPT: DRAG AND DROP PROGRAM(S) RUN BATCH SCRIPT

```
1 setlocal enableDelayedExpansion
2 rem color F0
3 title Execution in batch mode of SAS pgm(s)
4 mode con cols=130
5 @ECHO off
6 ECHO.
7 ECHO N. Rouill, 15JAN2014 V1.0
8 ECHO.
9 ECHO. To run in batch mode SAS programs by drag-and-drop over .BAT icon
10 ECHO. - SAS Pgm are executed sequentially
11 ECHO. - Log/Lst files are then scanned for the main keywords highlighting execution issue/outcome (no sas macro required)
12 ECHO.
13 ECHO ===== batch BEGIN =====
14 ECHO.
15 ECHO Batch execution started at %Time%
16 ECHO.
17 ECHO Detection of the active folder = %~d1%-p1
18 rem --- to test if log and lst folders are existing as subfolder or alternatively as parent folder
19 if exist "%~d1%-p1\log" (set LogFolder=%~d1%-p1\log) else if exist "%~d1%-p1..\log" (set LogFolder=%~d1%-p1..\log) else (set LogFolder=%~d1%-p1)
20 if exist "%~d1%-p1\lst" (set LstFolder=%~d1%-p1\lst) else if exist "%~d1%-p1..\lst" (set LstFolder=%~d1%-p1..\lst) else (set LstFolder=%~d1%-p1)
21
22 set /A NbProgsRun=0
23 set /A TotalNbOfLogERROR=0
24 set /A TotalNbOfLogWARNING=0
25
26 FOR %%A IN (*) DO (
27 ECHO.
28 ECHO.
29 ECHO =====
30 ECHO. Program '%~nA%%-xA'
31 powershell.exe "(Write-Host -NoNewLine -ForegroundColor ""DarkCyan"" - Created by:(Get-Acl ""%~d1%-p1%%-nA%%-xA"").Owner)"
32 ECHO =====
33 ECHO --- SAS pgm execution:
34
35 set /A NbProgsRun+=1
36
37 ECHO '%~nA%%-xA' Execution is ongoing
38 "D:\SAS94\SASHome\SASFoundation\9.4\sas.exe" -sysin "%~d1%-p1%%-nA%%-xA" -log "%LogFolder%\%%-nA.log" -print "%LstFolder%\%%-nA.lst" -icon -nosplash -reasuser
39 ECHO '%~nA%%-xA' Execution completed at !time!
40 ECHO.
41 ECHO --- Log scanning - messages are displayed by decreasing order of criticality:
42 findstr /A:4F /n "ERROR fatal" "%LogFolder%\%%-nA.log"
43 findstr /A:2F /n "WARNING uninitialized unequal" "%LogFolder%\%%-nA.log"
44 findstr /A:1F /n /c:"MERGE statement has more than" "%LogFolder%\%%-nA.log"
45 ECHO.
46
47 rem --- to compute the cumulative number of ERROR or WARNING message(s) in the log files
48 if exist "%LogFolder%\%%-nA.log" (
49 for /f "tokens==" %%j in ('findstr "ERROR" "%LogFolder%\%%-nA.log"') do @SET /A TotalNbOfLogERROR += 1
50 for /f "tokens==" %%j in ('findstr "WARNING" "%LogFolder%\%%-nA.log"') do @SET /A TotalNbOfLogWARNING += 1
51 )
52
53 ECHO --- Lst scanning:
54 if exist "%LstFolder%\%%-nA.lst" findstr /A:8F /n /c:"Number of Variables with Differing Attributes:" "%LstFolder%\%%-nA.lst"
55 if exist "%LstFolder%\%%-nA.lst" findstr /A:4F /n /c:" but not in " "%LstFolder%\%%-nA.lst"
56 if exist "%LstFolder%\%%-nA.lst" findstr /A:4F /n /c:"Number of Variables with Conflicting Types:" "%LstFolder%\%%-nA.lst"
57
58 if exist "%LstFolder%\%%-nA.lst" findstr /A:1F /n /c:"NOTE: No unequal values were found. All values compared are exactly equal." "%LstFolder%\%%-nA.lst"
59 if not exist "%LstFolder%\%%-nA.lst" echo File "%LstFolder%\%%-nA.lst" does not exist.
60 ECHO.
61 )
62 ECHO.
63 ECHO ===== batch END =====
64 echo.
65
66 if "!TotalNbOfLogERROR!" gtr "0" (
67 rem Call :Color 4F "FAILED:"
68 rem ECHO. Total number of ERROR messages in the log files: !TotalNbOfLogERROR!
69 powershell.exe "(Write-Host -NoNewLine -BackgroundColor ""DarkRed"" -ForegroundColor ""White"" Total number of ERROR messages in the log files: !TotalNbOfLogERROR!. )"
70 echo.
71 ) else (
72 rem Call :Color 2F "PASSED:"
73 rem ECHO. No ERROR message found in the log files.
74 powershell.exe "(Write-Host -NoNewLine -BackgroundColor ""DarkGreen"" -ForegroundColor ""White"" No ERROR message found in the log files.)"
75 echo.
76 )
77 if "!TotalNbOfLogWARNING!" gtr "0" (
78 powershell.exe "(Write-Host -NoNewLine -BackgroundColor ""DarkRed"" -ForegroundColor ""White"" Total number of WARNING messages in the log files: !TotalNbOfLogWARNING!. )"
79 echo.
80 ) else (
81 powershell.exe "(Write-Host -NoNewLine -BackgroundColor ""DarkGreen"" -ForegroundColor ""White"" No WARNING message found in the log files.)"
82 echo.
83 )
84
85 ECHO.
86 PAUSE
87 exit /B
```

# PhUSE EU Connect 2018

## FULL SCRIPT: PROGRAM CREATION WITH STANDARD HEADER

```
1 rem color F0
2 title Creation of a new SAS pgm
3 mode con cols=130 lines=14
4 @ECHO off
5 ECHO
6 ECHO.
7 ECHO. Purpose of this script is to create a new SAS program with pre-filled headers and standard macro-calls.
8 ECHO.
9 ECHO.
10 ECHO.
11 set /p SASfile=Please enter the name of the program to be created (file name without .sas extension):
12 if exist %SASfile%.sas goto :FileAlreadyExists
13
14 @echo off
15 @echo dm 'log; clear; lst; clear;'> "%SASfile%.sas"
16 @echo *****;> "%SASfile%.sas"
17 @echo * PROJECT/PATH : %CD%\>> "%SASfile%.sas"
18 @echo * CREATION DATE: %date% (mm/dd/yyyy) >> "%SASfile%.sas"
19 @echo * PURPOSE : Programming of output(s) according to project spreadsheet>> "%SASfile%.sas"
20 @echo *****;> "%SASfile%.sas"
21 @echo.>> "%SASfile%.sas"
22 @echo.>> "%SASfile%.sas"
23 @echo %*--- global settings;>> "%SASfile%.sas"
24 @echo %*getdirf(flvl=4);>> "%SASfile%.sas"
25 @echo %*--- at output level: save lst file if generated;>> "%SASfile%.sas"
26 @echo %*SaveLst;>> "%SASfile%.sas"
27 @echo.>> "%SASfile%.sas"
28 @echo %*--- end of program -----;>> "%SASfile%.sas"
29 ECHO The program %SASfile%.sas has been created within %CD%.
30 goto :DONE
31
32 :FileAlreadyExists
33 ECHO - The program %SASfile%.sas already exists within %CD%.
34 ECHO - Please ensure to manually delete this file in case you would like it to be automatically re-created.
35
36 :DONE
37 ECHO.
38 PAUSE
--
```

## FULL SCRIPT: VBS TO COMPARE ON-THE-FLY

```
1 Option Explicit
2 Sub main()
3 dim objArgs,num,sBaseDoc,sNewDoc,objScript,word,destination
4 Set objArgs = WScript.Arguments
5 num = objArgs.Count
6 if num < 2 then
7     MsgBox "Usage: compare.vbs base.doc new.doc", vbExclamation, "Invalid arguments"
8     WScript.Quit 1
9 end if
10 sBaseDoc=objArgs(0)
11 sNewDoc=objArgs(1)
12 Set objScript = CreateObject("Scripting.FileSystemObject")
13 If objScript.FileExists(sBaseDoc) = False Then
14     MsgBox "File " + sBaseDoc + " does not exist. Cannot compare the documents.", vbExclamation, "File not found"
15     WScript.Quit 1
16 End If
17 If objScript.FileExists(sNewDoc) = False Then
18     MsgBox "File " + sNewDoc + " does not exist. Cannot compare the documents.", vbExclamation, "File not found"
19     WScript.Quit 1
20 End If
21 Set objScript = Nothing
22 On Error Resume Next
23 set word = createobject("Word.Application")
24 If Err.Number <> 0 Then
25     Wscript.Echo "You must have Microsoft Word installed to perform this operation."
26     Wscript.Quit 1
27 End If
28 On Error Goto 0
29 ' Open the new document
30 set destination = word.Documents.Open(sNewDoc)
31 ' Hide it
32 destination.Windows(1).Visible=0
33 ' Compare to the base document
34 destination.Compare(sBaseDoc)
35 ' Show the comparison result
36 'word.ActiveDocument.Windows(1).Visible = 1
37 ' Mark the comparison document as saved to prevent the annoying ' "Save as" dialog from appearing.
38 'word.ActiveDocument.Saved = 1
39 word.ActiveDocument.SaveAs(replace(sBaseDoc,".doc","") & "_compared.doc")
40 ' Close the first document
41 destination.Close
42 'word.visible=True
43 word.quit
44 WScript.Echo "Comparison completed and saved as: " & replace(sBaseDoc,".doc","") & "_compared.doc" & "."
45 End Sub
46
47 Call main
```