

VERBOSE LOGGING

1

Overview	2
Logged Information	2
Verbose Batch Execution	4
Windows	4
UNIX	7
Details	8
Example	9
Input File	9
Output File	9
Log File	11

OVERVIEW

Batch processing support to run S-PLUS scripts non-interactively has been a key feature of S-PLUS, and this feature has been available for a number of releases. Often this is performed to execute S-PLUS scripts as part of a production process.

For example, in the medical and pharmaceutical industries, it is vital that data management and statistical tasks be validated and documented. All steps involved in experimentation and validation must be documented and preserved in order to demonstrate to oversight bodies that the results are complete, accurate, valid, and reproducible. In addition to source files and generated output, log files are typically retained for these purposes.

The standard batch processing can record standard input, output, and errors to one or more files. This is the same output that the user would see executing the commands in the script at the command line.

In production use, it is valuable to include detailed information on specific tasks, such as when the commands were executed, what files were created, and the circumstances under which errors occurred. The information should be available in a text file that is readily understood and has tags to indicate the type of information in each line. This allows tools such as Perl to automatically extract information from the file.

The verbose logging function in S-PLUS provides this type of information regarding the execution of an S-PLUS script.

Logged Information

There are many features of the verbose log file:

- The log file has a structured format with the first seven columns reserved for line numbers and a line description tag. A colon in the eighth column separates these tags from the rest of the text; lines without colons in the eighth column are continuation of the previous tagged line. This makes parsing of the log file by Perl or other languages easy.
- There is a header with information about the user, the machine, the input and output files.
- If a file gets sourced in by the input file, it is logged (e.g., **S.init** is sourced in at the start of every S-PLUS session).

- Each database that gets attached is logged.
- Each line from the input file appears in the log file, identified by its line number.
- Each complete input expression is timed.
- A summary is logged when data is read in. The summary includes the disk location of the file, the resulting data frame size, the types of columns in the data frame, and missing value counts. Data frame reading and writing are also logged.
- Errors are logged but (by default) processing of the script continues.
- On completion of the script, a summary of the entire processing is written.

VERBOSE BATCH EXECUTION

Verbose batch execution can be invoked in a number of different ways, depending upon the platform you're using. We discuss the different methods by platform below.

Windows

In Windows, batch execution can be done using three unique sources:

- GUI , via the **BATCH** dialog
- Command Prompt, entering commands
- S-PLUS command line, with the **Target** field of an S-PLUS shortcut.

GUI

On Windows, the S-PLUS program group menu (accessed by navigating to **Start ► Programs ► S-PLUS 6.2**) includes a menu item for the **BATCH** executable. This menu item launches the **BATCH** processing dialog, which is used to specify the S-PLUS script to execute, and has options specific to batch operation. This dialog is shown in Figure 1.1.

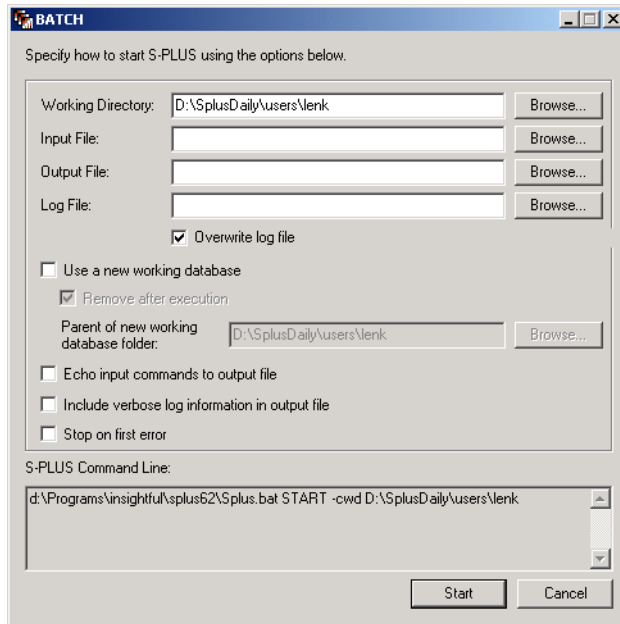


Figure 1.1: The **BATCH** dialog, which executes the batch script in Windows.

Working Directory The directory where S-PLUS reads and writes files. This directory typically contains a **.Data** directory with S-PLUS objects; see the description for **Use a new working database** for details.

Input File Contains the S-PLUS script commands to be run. If you enter just a relative file name, the **BATCH** utility looks for that file relative to the **Working Directory**.

Output File The name of the file that contains the output generated by the S-PLUS script. If you enter just a relative file name, the **BATCH** utility creates the file relative to the **Working Directory**.

Log File The name of the file that contains the log information generated by the S-PLUS script. If you enter just a relative file name, the **BATCH** utility creates the file relative to the **Working Directory**.

Overwrite log file Select this if you want to overwrite the contents of an existing log file.

Use a new working database Enables you to run the script under a newly created **.Data** directory. This directory is created under the directory specified in the **Parent of new working database folder** specified in the field below (typically the **Working Directory**), and is empty except a new `.Random.seed`.

Remove after execution Removes the new working database after the S-PLUS script is finished. This option is only available if you have checked the **Use a new working database** checkbox, and it does not remove any files if you are using the working database specified in the **Working Directory**.

Echo input commands to output file Combines the input commands with the output lines in the output file.

Include verbose logging information in output file Allows you to have the verbose log information mixed in with the script output. This is very useful for debugging and support.

S-PLUS Command Line Contains a copy of the command that executes the batch process when the **Start** button is pressed. The user can copy this command and paste it into a ***.bat** file or create a shortcut for use later.

The **S-PLUS Command Line** is the executable that runs the batch script and contains options, as in this example:

```
"C:\Program Files\Insightful\splus62\cmd\Splus.bat" START  
-project "C:\Program Files\Insightful\splus62\users\lenk"  
-input D:\SplusDaily\users\lenk\Script1.ssc  
-output D:\SplusDaily\users\lenk\Script1.txt  
-logfile D:\SplusDaily\users\lenk\Script1.log -echo -verbose
```

Note that when you enter filenames in the **Working Directory**, **Input File**, **Output File**, or **Log File** fields in the GUI, the content of the **S-PLUS Command Line** is updated with this information.

Command Prompt

To run a Windows batch file from a Command Prompt, the syntax is

```
> Splus SBATCH [options]
```

where [options] are those described in the section Command Line in UNIX (below), except that the -j option is not enabled on Windows.

S-PLUS Command Line

You can also run the batch file using the **Target** field in an S-PLUS command line, as shown in Figure 1.2.

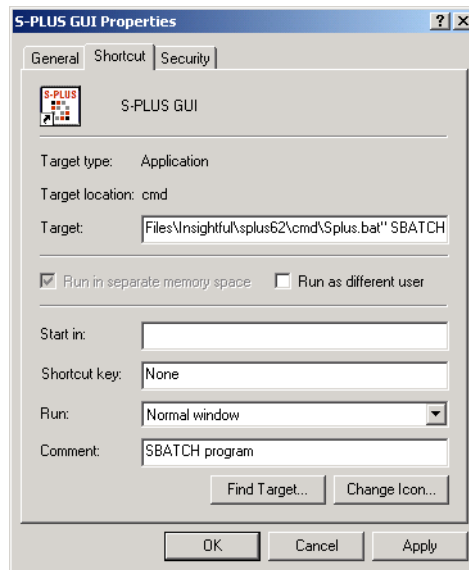


Figure 1.2: Running a batch script in an S-PLUS command line.

You must supply a complete path to the **Splus.bat** script, including the **.bat** file extension. For example:

```
“C:\Program Files\Insightful\Splus62\Splus.bat” SBATCH  
-echo
```

UNIX

Command Line

In UNIX, use `Splus SBATCH` to execute a script under verbose logging. This command has a number of optional arguments:

```
% Splus SBATCH -help
```

```
Syntax : SBATCH [-flags] inputfile
```

```
Optional flags are:
```

<code>-help</code>	Print this message and quit
<code>-j</code>	Enable java (not supported on Win32)
<code>-cwd directory</code>	Change directory to this directory before anything else
<code>-input inputfile</code>	Read input from inputfile (or specify inputfile as last argument)
<code>-output file</code>	Send output to file (default <basename inputfile>.txt)
<code>-logfile file</code>	Write session log to file (default <basename inputfile>.slg) By <basename inputfile> we mean inputfile stripped of possible period and extension
<code>-appendlog</code>	Append to log file (default is to overwrite)
<code>-nologfile</code>	Do not make a session log
<code>-work directory</code>	Use directory for working database (where=1), making it if needed. If work is not specified we will make new randomly named directory for the working database.

<code>-noclean</code>	Do not remove the new randomly named directory used for a working database
<code>-newworkparent</code>	Directory in which to make the new randomly named working database (value of <code>project directory</code> , whose default =.)
<code>-echo</code>	Echo command lines to output (sets <code>options(echo=T)</code>)
<code>-noecho</code>	Do not set <code>options(echo=T)</code>
<code>-verbose</code>	Echo verbose logging information to output (sets <code>options(verbose=T)</code>)
<code>-quitonerror</code>	Terminate immediately on engine error. (Default: continue through errors)
<code>-background</code>	Run S-PLUS in the background (returning immediately)
<code>-vanilla</code>	Do not run user-defined startup scripts or functions (<code>-work dir</code> is ignored)
<code>name = value</code>	Will be added to environment so <code>getenv("name")</code> returns <code>value</code>

Details

On Windows and UNIX, setting

```
> options(verbose=T)
```

after starting S-PLUS or setting the environment variable `S_VERBOSE = yes` causes most of this information to be printed to the standard output, either the commands on a script window or the output file.

You may use the `logcat()` function to write more logging information. It does nothing if logging is not turned on, but prints one tagged line if it is turned on.

Note that `SBATCH` is designed for production use of verbose logging. To run batch files interactively on Windows or UNIX, use

```
Spplus START
```

More details are available by typing `Spplus START -help`.

EXAMPLE

The following is a short sample batch script, along with the output generated and the log file:

Input File

```
# Import bad file, has blank line at top:
ffDF <- importData("ffBad.txt", separateDelim=T)
# Import correct file
ffDF <- importData("ffGood.txt", separateDelim=T)
summary(ffDF)
# Incorrect model formula:
attach(fuel.frame)
reg1 <- lm(Fuel ~ Wt + Disp., data=ffDF, na.action=na.omit)
# Correct model formula:
reg1 <- lm(Fuel ~ Weight + Disp., data=ffDF,
na.action=na.omit)
reg1
# Write out a data set of results:
exportData(data.frame(fit=fitted(reg1), resid=resid(reg1)),
  file="reg1.sas7bdat")
```

Output File

```
S-PLUS : Copyright (c) 1988, 2003 Insightful Corp.
S : Copyright Lucent Technologies, Inc.
Professional Edition Version 6.2.1 for Microsoft Windows :
2003
Log file will be \\PUSHPULL\d\SPK\SP6\log.txt
Working data will be in d:\spk\sp6\Sc000534.tmp
```

```
> # Import bad file, has blank line at top:
> ffDF <- importData("ffBad.txt", separateDelim=T)
> # Import correct file
> ffDF <- importData("ffGood.txt", separateDelim=T)
> summary(ffDF)
      Weight      Disp.      Mileage      Fuel
  Min.:1845    Min.: 73.0    Min.:18.0
  Min.:2.702703
  1st Qu.:2624    1st Qu.:114.5    1st Qu.:21.0    1st
  Qu.:3.703704
  Median:2903    Median:146.0    Median:23.0
  Median:4.347826
```

Chapter 1 Verbose Logging

```
      Mean:2927      Mean:152.4      Mean:24.4
Mean:4.226319
      3rd Qu.:3310      3rd Qu.:180.0      3rd Qu.:27.0      3rd
Qu.:4.761905
      Max.:3855      Max.:305.0      Max.:37.0
Max.:5.555556
      NA's: 2.0      NA's: 2.0
NA's:2.000000
```

```
      Type
Compact:15
      Large: 3
      Medium:14
      Small:13
      Sporty:11
      Van: 8
```

```
> # Incorrect model formula:
> reg1 <- lm(Fuel ~ Wt + Disp., data=ffDF,
na.action=na.omit)
Problem: Object "Wt" not found
Use traceback() to see the call stack
> # Correct model formula:
> reg1 <- lm(Fuel ~ Weight + Disp., data=ffDF,
na.action=na.omit)
```

```
Call:
lm(formula = Fuel ~ Weight + Disp., data = ffDF, na.action =
na.omit)
```

```
Coefficients:
(Intercept)      Weight      Disp.
    0.484787  0.001239589  0.0008637803
```

```
Degrees of freedom: 61 total; 58 residual
Dropped 3 cases due to missing values
Residual standard error: 0.3870086
```

```
> # Write out a data set of results:
> exportData(data.frame(fit=fitted(reg1),
resid=resid(reg1)),
+           file="reg1.sas7bdat")
[1] 60
```

Log File

START: S-PLUS Version 6.2 for WIN386 started at Fri Aug 29
15:14:09
2003

NOTE: User=spk Machine=PUSHPULL
Directory=\\PUSHPULL\d\SPK\SP6

NOTE: S-PLUS is installed in directory
\\PUSHPULL\d\splus62

NOTE: Input file is \\HOMER\spk\ff.ssc

NOTE: Output file is \\PUSHPULL\d\SPK\SP6\out.txt

PARSING: \\PUSHPULL\d\splus62\S.init

DATA: Attaching directory
\\PUSHPULL\d\spk\sp6\Sc000534.tmp\.Data in
position 1 with name d:\spk\sp6\Sc000534.tmp

DATA: Attaching directory
\\PUSHPULL\d\splus62\library\nlme3\.Data in
position 6 with name nlme3

DATA: Attaching directory
\\PUSHPULL\d\splus62\library\menu\.Data in
position 7 with name menu

DATA: Attaching directory
\\PUSHPULL\d\splus62\library\sgui\.Data in
position 8 with name sgui

TIME: Task #1 succeeded (Seconds = 1.531 CPU, 13.781
elapsed)

PARSING: \\HOMER\spk\FF.SSC

1 : # Import bad file, has blank line at top:
TIME: Task done (Seconds = 0 CPU, 0 elapsed)

2 : ffDF <- importData("ffBad.txt", separateDelim=T)
NOTE: Imported 62 by 5 data.frame with column names Col1,
Col2, Col3,
Col4, Col5

NOTE: Counts of column types: factor:5

NOTE: There are 5 missing values

NOTE: 1 rows have some missing values

DATA: Storing 62x5 data.frame 'ffDF' on database
d:\spk\sp6\Sc000534.tmp

TIME: Task done (Seconds = 0.484 CPU, 0.734 elapsed)

3 : # Import correct file
TIME: Task done (Seconds = 0 CPU, 0 elapsed)

Chapter 1 Verbose Logging

```
4      : ffDF <- importData("ffGood.txt", separateDelim=T)
      NOTE: Imported 64 by 5 data.frame with column names
      Weight, Disp.,
           Mileage,
      Fuel, Type
      NOTE:   Counts of column types: factor:1, numeric:4
      NOTE:   There are 6 missing values
      NOTE:   4 rows have some missing values
      DATA: Storing 64x5 data.frame 'ffDF' on database
d:\spk\sp6\Sc000534.tmp
      TIME: Task done (Seconds = 0.391 CPU, 0.625 elapsed)

5      : summary(ffDF)
      DATA: Reading 64x5 data.frame 'ffDF' on database
d:\spk\sp6\Sc000534.tmp
      TIME: Task done (Seconds = 0.593 CPU, 0.656 elapsed)

6      : # Incorrect model formula:
      TIME: Task done (Seconds = 0 CPU, 0 elapsed)

7      : reg1 <- lm(Fuel ~ Wt + Disp., data=ffDF,
na.action=na.omit)
      DATA: Reading 64x5 data.frame 'ffDF' on database
d:\spk\sp6\Sc000534.tmp
      ERROR: Problem: Object "Wt" not found
      TIME: Task done (Seconds = 0.157 CPU, 0.281 elapsed)

8      : # Correct model formula:
      TIME: Task done (Seconds = 0 CPU, 0 elapsed)

9      : reg1 <- lm(Fuel ~ Weight + Disp., data=ffDF,
na.action=na.omit)
      DATA: Reading 64x5 data.frame 'ffDF' on database
d:\spk\sp6\Sc000534.tmp
      TIME: Task done (Seconds = 0.203 CPU, 0.265 elapsed)

10     : reg1
      TIME: Task done (Seconds = 0.078 CPU, 0.078 elapsed)

11     : # Write out a data set of results:
      TIME: Task done (Seconds = 0 CPU, 0 elapsed)
```

```
12      : exportData(data.frame(fit=fitted(reg1),  
resid=resid(reg1)),
```

```
13      : file="reg1.sas7bdat")
```

```
      TIME: Task done (Seconds = 0.141 CPU, 0.203 elapsed)
```

```
      NOTE: There were 1 errors and 0 warnings in this session
```

```
      TIME: Session done (Seconds = 7.483 CPU, 36.109 elapsed)
```

```
      QUIT: End session at Fri Aug 29 15:14:45 2003
```

