User Script libraries

Peter Mason, March 2018

Introduction

Perhaps some of you know that you can lump two or more scalar batch-scripts into one file and, with a little attention to the second line of the file, TSG will know how to handle it. It will present you with a list of script methods to choose from when you create a batch-script scalar from the file. TSG has been able to do this for years when given an external file, so what's all the fuss then? It's about organising things. A user-script library is handled differently to an ad-hoc file of scripts. Given the right attention, a user script library file can be hooked up with TSG to work very much like TSG's own built-in scripts, with easy access, uniquely-identified script methods, version tracking, and documentation links.

Who is it for?

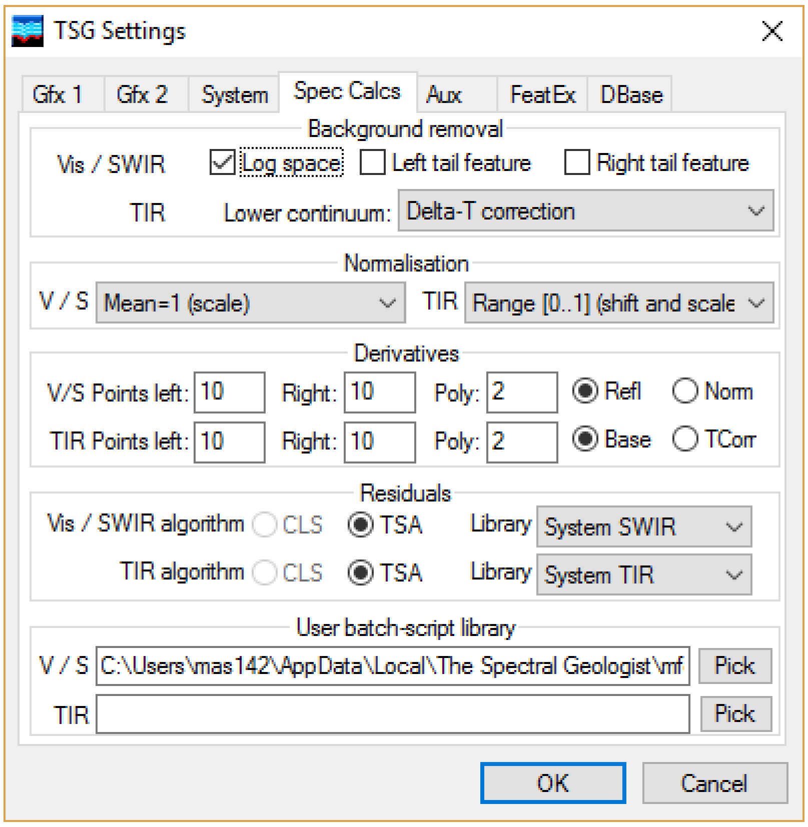
The average TSG user is not interested in creating scalar batch scripts. If that's you then you may have some doubts about reading on, but do read on. The first part of this document describes how to use script libraries. The TSG distribution now includes a few user script libraries and it is likely that more libraries will become available in the future.
If you already create scalar batch-scripts or are an advanced user who might be willing to give it a try then also see the latter part of this manual, which describes creating your own user script libraries. The user batch-script library system offers a way to have standard algorithm collections. If you share scalar calculation ideas with other TSG users then it could be helpful. Even if you don't collaborate with other TSG users, you might find that it helps to keep things tidy. Remember that a batch script does not have to be complicated – it can contain just one method (e.g., a PFit). i.e., You can code normal TSG scalars as single-method batch scripts and so have them "in the system".

Part 1: Using script libraries

Overview

- You can set a system-wide **default** library for VNIRSWIR datasets and another for TIR datasets.
    - However one library can actually contain a mixture of scripts, e.g., VNIRSWIR and TIR scripts.
    - At this time the system can only be configured with these two user script libraries. The system may be expanded in the future.
- Each TSG dataset has a user script library associated with it.
    - The dataset inherits the system default library to begin with.
- The dataset's script library is used by TSG's Scalar Construction Wizard when a "Batch" scalar is created.
    - Select the **User** radio button when creating the scalar. The library's *compatible* scripts are then shown in the **Script** list.

System default user script libraries


TSG Settings

Gfx 1 | Gfx 2 | System | Spec Calcs | Aux | FeatEx | DBase

Background removal
Vis / SWIR [✓ Log space] ☐ Left tail feature ☐ Right tail feature
TIR Lower continuum: Delta-T correction

Normalisation
V / S Mean=1 (scale) TIR Range [0..1] (shift and scale

Derivatives
V/S Points left: 10 Right: 10 Poly: 2 [● Refl] ○ Norm
TIR Points left: 10 Right: 10 Poly: 2 [● Base] ○ TCorr

Residuals
Vis / SWIR algorithm ○ CLS [● TSA] Library System SWIR
TIR algorithm ○ CLS [● TSA] Library System TIR

User batch-script library
V / S C:\Users\mas142\AppData\Local\The Spectral Geologist\mf Pick
TIR Pick

OK Cancel

Start TSG and use the **File -> Settings** menu ***while no dataset is open***. Switch to the **Spec Calcs** tab.

At the bottom you will find a section named User batch-script library. It has two rows of controls. The V / S row is for VNIRSWIR datasets and the TIR row for TIR datasets. Use the **Pick** Button(s) to select your user script library file(s).

Where are user script library files located?

I give up, you tell me. Just kidding. Sort of.
User script libraries that are bundled with TSG are installed alongside the TSG8 executable. This is in a subdirectory in your "App Data Local" area. For example, my Windows login is **`mas142`** so my TSG8 is installed here:

```
C:\Users\mas142\AppData\Local\The Spectral Geologist\
```

Unlike the "Program files" directory that TSG7 used, you can access this directory. If you have other user script libraries then you might consider copying them here. (Before long you'll get used to the clumsy directory name.) If you'd rather not then I recommend creating an easily-located directory for your user scripts. Perhaps you could even use the directory where you keep all your aux-match datasets[1].

What user script libraries come with TSG8? (March 2018)

- **mfemscripts2016.txt**
  A collection of "multiple feature extraction method" scripts managed by Carsten Laukamp, CSIRO. These are at least medium-complexity scripts that carry a level of interpretation (i.e., they target specific minerals). Some documentation is included.
- **legacy_tsg7_scripts.txt**
  The scripts that were recently booted out of TSG's built-in "system" collection.
- **ms_tir_scripts.txt**
  A collection of TIR scripts by Martin Schodlok.

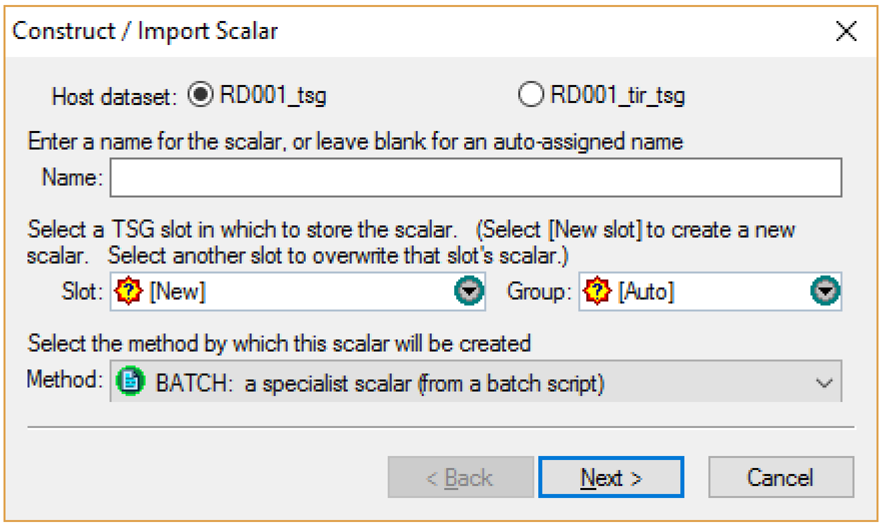
[1] Just kidding again. That idea never took off.

Setting a dataset’s script library

This works very much like the above, only you go into **File -> Settings** while a dataset is open. (You are now adjusting the dataset’s settings rather than the system global ones.)

- If the dataset consists of a VNIRSWIR + TIR pair then both the V / S and TIR script control sets will be shown, otherwise just one set will.
- If the dataset has no script selected yet then it inherits the system default. You will see the system default script selected here. If you change the selection now then you are only changing it for the current dataset.

Creating a scalar from a user script library


Construct / Import Scalar

Host dataset: [✓ RD001_tsg] RD001_tir_tsg

Enter a name for the scalar, or leave blank for an auto-assigned name

Name:

Select a TSG slot in which to store the scalar. (Select [New slot] to create a new scalar. Select another slot to overwrite that slot's scalar.)

Slot: [New] Group: [Auto]

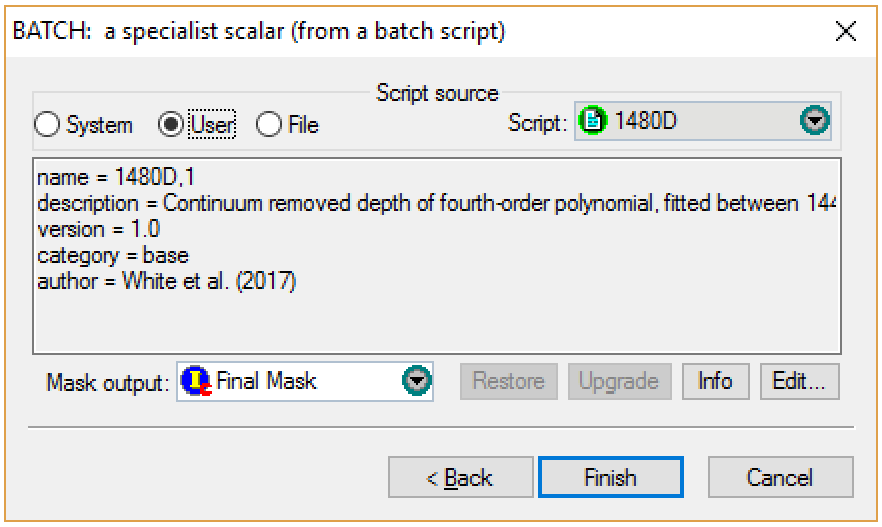
Select the method by which this scalar will be created

Method: BATCH: a specialist scalar (from a batch script)

Open a dataset and use the **Edit -> New scalar** menu to bring up the Scalar Construction Wizard.

If you have a VNIRSWIR + TIR dataset pair then (as always) be mindful of the Host dataset radio buttons along the top. (Pick the right dataset to host the new scalar.)

Select the **BATCH** item in the **Method** list. **Next**.


BATCH: a specialist scalar (from a batch script)

Script source

System [✓ User] File Script: 1480D

```
name = 1480D,1
description = Continuum removed depth of fourth-order polynomial, fitted between 14
version = 1.0
category = base
author = White et al. (2017)
```

Mask output: Final Mask

Select the **User** radio button in the Script source area at the top, then look at the **Script** list. It will show all scripts in the library that are compatible with the dataset. Select one.

- TSG shows some metadata describing the script.
- If the script author included a documentation link then the **Info** button will be live. Click it to view the script’s documentation. (TSG will normally bring up your web browser for this.)
- If you would like to see and perhaps edit the script’s “code” then click the **Edit** button. TSG will bring up a separate edit window. Any changes you make to the script code will only apply to the scalar that you are creating now, not the script library.
- If there is a “final_mask” scalar in the dataset then (as usual) it will be selected automatically.
- Click **Finish** to calculate the scalar and close the Scalar Construction Wizard.

The other buttons

- The **Restore** button is only enabled if you change the script via Edit. Click it to restore the script to the original code.
    - If you bring up an existing script scalar for modification then the Restore button will be enabled if the scalar was constructed from an edited script. (TSG can track things for a scalar that was created from a user script library.) Click it to restore the script to the original code in the script library.
- The **Upgrade** button is always disabled when creating a new scalar.
    - If you bring up an existing script scalar for modification then the Upgrade button will be enabled if the script author has since released a new version of the script concerned. Click it to upgrade the scalar to the new version.

Part 2: Making your own libraries

This document will not discuss TSG's batch-script syntax or how to go about coding scalar batch-scripts. Our starting point is a collection of batch-script files, each describing a single scalar's calculation. From there the job is to collect all the scripts in a single text file (leaving out the first 2 lines of each script), set the first two lines of the file, and add metadata to each script.

An example script

Here's an example of the sort of script file that you might have now (note long lines have been wrapped in this document):

```
TSG Specialist Scalar Command Set
Commands = 1
name = Hm-Go_Distr, 6
p1 = profile, layer=ref, stat=depth, bkrem=div, fit=3, wcentre=913, wradius=137
p2=ratio, wnum=450, wdenom=1650, biggest=1
p3= expr, param1=p2, mod1=set, param2=p1, arithop=mult
p4= expr, param1=p3, const2=0.025, arithop=lgt, nullhandling=out
p5= pfit, layer=ref, wunits=nm, wmin=776, wmax=1050, bktype=hull, bksub=div,
order=4, product=0, bktype=hull, bksub=div
return=expr, param1=p4, param2=p5, arithop=mult
```

And here is how it might appear within a script library:

```
TSG Specialist Scalar Command Set
Commands = 23
…
name = Hm-Go_Distr, 6
description = Continuum removed wavelength of the 900 nm absorption minimum
calculated using a fitted 4th order polynomial between 776-1150nm.
date = Thu Nov 12 22:18:35 2015
Version = 1.0
Category = published
scalargroup = mineralogy:iron oxides
uuid = AF696A4B-EE55-4EEC-B646-2AB0309CAE42
replaces = 7118dc56-6be7-4322-a08f-bba55ca5c7c5
author = CSIRO (Tom Cudahy and Erick Ramanaidou, 1997)
doclink = $TSGAPP/mfemcripts2016.htm#Hm-Go_Distr
p1 = profile, layer=ref, stat=depth, bkrem=div, fit=3, wcentre=913, wradius=137
p2=ratio, wnum=450, wdenom=1650, biggest=1
p3= expr, param1=p2, mod1=set, param2=p1, arithop=mult
p4= expr, param1=p3, const2=0.025, arithop=lgt, nullhandling=out
p5= pfit, layer=ref, wunits=nm, wmin=776, wmax=1050, bktype=hull, bksub=div,
order=4, product=0, bktype=hull, bksub=div
return=expr, param1=p4, param2=p5, arithop=mult
…
```

As you may notice, the script code itself (lines `P1` to `return`) is unchanged but there are more metadata fields in the library.

The library's commands= line

All TSG batch-script files start with two lines like this:

```
TSG Specialist Scalar Command Set
Commands = N
```

(Where **N** is a number greater than 0.)
In the library example above, the second line is Commands = 23. Why 23? This tells TSG that the library contains 23 scripts. It's that simple.

Script metadata fields

Most of these fields are optional but the more fields you can provide the better.
Some fields have an ugly thing called a "UUID", which is a "Universally Unique Identifier". UUIDs will get their own chat session later.

Name

e.g., name = Hm-Go_Distr, 6
This field is **essential**.
It has two parts, the name of the script and the number of methods (P1 to return) in the script. The name should be a good one. Scalars made from the script will be named after it (by default). The number of methods must be correct.

Description

A **single line** of text describing the script. Up to 256 characters long.
This field is optional but highly recommended. Users will see it when they select the script for calculation. Tell them something.

Bounds

General format: bounds = [min=x1 [,clipmin=yn]] [,max=x2 [,clipmax=yn]]
e.g., bounds = min=0.0001, clipmin=n, max=0.275, clipmax=y
This field is optional.
You can use it to set lower *and / or* upper acceptability bounds for the script's results. x1 and x2 are numbers; yn is either "y" (yes) or "n" (no).
Examples:

- Min=0.2: If the script's result is less than 0.2 then it is changed to NULL (clipmin wasn't given).
- Min=0.2, clipmin=n: Same as above.
- Min=0.2, clipmin=y: If the script's result is less than 0.2 then it is clipped to 0.2.
- Min=0.2, clipmin=y, max=10.0: If the script's result is less than 0.2 then it is clipped to 0.2, and if it is greater than 10 then it is changed to NULL.
- Min=0.2, clipmin=y, max=10.0, clipmax=y: If the script's result is less than 0.2 then it is clipped to 0.2, and if it is greater than 10 then it is clipped to 10.
- max=10.0: If the script's result is greater than 10 then it is changed to NULL.

Uuid

e.g., uuid = AF696A4B-EE55-4EEC-B646-2AB0309CAE42
Consider this field is **essential** in a user script library.
A script's UUID uniquely identifies it and is used for change and version tracking. UUIDs are discussed more below.

Category

e.g., category = base
This field is **essential**. Your choices are: **user published base utility unvalidated**
This field controls the "folder" where the script appears in the Scalar Construction Wizard's script list.

Version

e.g., version = 2.1
This field is optional but recommended. A 16-character string is accepted. Scripts with version numbers look impressive.

Author

e.g., author = Tommy Edison
This field is optional but recommended. 64 characters. It shows the script's author(s). Users will see it when they select the script for calculation.

Scalargroup

e.g., scalargroup = mineralogy:iron oxides
This field is optional. Scalars created from the script will appear in this scalar group. Your choices are restricted to the scalar groups that are defined within TSG. See TSG's **Edit -> Scalar names and groups** dialog.

Replaces

e.g., replaces = 7118dc56-6be7-4322-a08f-bba55ca5c7c5
This field is optional – use it when necessary.
It gives the UUID of the (older version) script that the current script replaces.
This is how versioning is done. If you come up with a modification to a script in your library then you don't modify the script's code in the library. Instead, you add a whole new script for the new version. Having done that, you include a replaces= field to connect the new script to the old script that it replaces, and you add a replacedby= field to the old script to connect it to the new script.

Replacedby

e.g., replacedby = af696a4b-ee55-4eec-b646-2ab0309cae42
This field is optional – use it when necessary.
It gives the UUID of the (newer version) script that replaces the current script.
See "Replaces" above.

Seealso

e.g., seealso = 5bea6a9a-5c9a-4980-84c5-e4d0690d728c
This field is optional.
It gives the UUID of some other script that has some relationship with the current script. Currently it is only used cosmetically in the Scalar Construction Wizard.

Date

e.g., date = Thu Nov 12 22:18:35 2015
This field is optional.

It gives the date when the script was authored. Currently it is only used cosmetically in the Scalar Construction Wizard.

Doclink

e.g., http://mega.script.server.com/intermediate_725B.htm#platinum_grade_estimator
This field is optional, but it's bound to be appreciated if you can supply it. It is hooked up to the **Info** button in the script Scalar Construction Wizard.
It gives the URL[2] to the script's documentation. It would normally be an http:// or https:// web link as a file:// link is too specific to be useful for general TSG users.
A couple of TSG-specific symbols can be used here:

- **$TSGWEB** refers to the TSG website and is a "shortcut" for:
  https://research.csiro.au/thespectralgeologist
- **$TSGAPP** is a "shortcut" for the TSG installation directory.
  E.g., $TSGAPP/mfemcripts2016.htm#Hm-Go_Distr
    - You might find this one useful if you copy TSG script files (and their help files) to the TSG installation directory, as discussed earlier.

UUIDs

UUIDs are a necessary part of user script libraries. The format used in TSG's scripts consists of 36 characters. It has 5 parts separated by "minus" characters and looks like this:

af696a4b-ee55-4eec-b646-2ab0309cae42

Case doesn't matter – the letters can be upper- or lower-case.
UUIDs are used to uniquely identify scripts. Each UUID that you get from a proper UUID generator is unique. There are websites that will generate UUIDs for you, e.g., https://www.uuidgenerator.net/ Otherwise, contact me (peter.mason@csiro.au) and I can supply them.

[2] "Uniform Resource Locator" or "Unapproachable Raving Lunatic" depending on context