Using the channel profile analysis tool, built by the University of Edinburgh Land Surface Dynamics group

Simon M. Mudd, Mikaël Attal, David T. Milodowski, Stuart W.D. Grieve and Declan A. Valters
School of GeoSciences, University of Edinburgh
Contact: simon.m.mudd _at_ ed.ac.uk

Table of Contents

1. Quick guide

If you already know more or less what you are doing, but need a quick reminder, here are the steps involved:

a. Prepare your .chan file either using LSDTopoToolbox, (which is projected to be released in 2014 but before then you can request a copy of the binaries from Simon Mudd), or from your own script.
b. If the programs aren't complied, make them with: `chi_m_over_n_analysis.make` and `chi_get_profiles.make`
c. Run the program `chi_m_over_n_analysis.exe` to determine the best fit *m/n* ratio. Note: this step will take some time (a few minutes to several days depending on your paramter values).
d. If the *m/n* analysis yeilds ambiguous results, use the script `movern_sensitivty_driver_generation.py` to generate driver files for a sensitivity analysis and repeat the m/n analysis. In fact, we recommend doing this for all natural channels since in many places a stream power incision model doesn't do a great job of explaining channel geometry and if you only use one set of parameter values you might just get a set that gives a spuriously 'clean' result.
e. Run the program `chi_get_profiles.exe` to get the transformed profiles in chi-elevation space.
f. Visualise the data with Python scripts.

2. Overview

This document gives instructions on how to use the segment fitting tool for channel profile analysis developed by the Land Surface Dynamics group at the University of Edinburgh. The tool is used to examine the geometry of channels using the integral method of channel profile analysis. For background to the method, and a description of the algorithms, we refer the reader to Mudd et al. (2013, draft manuscript). For background into the strengths of the integral method of channel profile analysis, the user should read Perron and Royden (2013, ESPL):

http://mit.edu/perron/www/files/PerronRoyden13.pdf

This document guides the user through the installation process, and explains how to use the model. You will need a c++ compiler for this tutorial. If you have no idea what a c++ compiler is, see the appendix. Visualisation of the model results is performed using Python scripts. We recommend installing Python(x,y) (https://code.google.com/p/pythonxy/) and running the scripts within Spyder (which is installed with python(x,y). Both the recommended compiler and Python(x,y) are open source: you do not need to buy any 3rd party software (e.g., Matlab, ArcMap) to run our topographic analysis!

3. Warning

This code is for research purposes and is under continuous development, so we cannot guarantee a bug-free experience! If you find a bug, please send an email to simon.m.mudd _at_ ed.ac.uk

4. Steps involved to perform channel analysis

Performing the channel analysis involves 4 steps, including visualization of the data

a. Preparing the data.
b. Performing a statistical analysis to constrain the most likely *m/n* ratio (if you don't know what that is, read Mudd et al (2013, in prep)).
c. Extract the chi profiles
d. Visualize the data.

a. Preparing the channel data

The segment fitting algorithm works on a 'channel' file (we use the extension .chan to denote a channel file).

If you have the full LSDTopoToolbox there are functions for this but at the moment we are not releasing the full toolbox since it has some algorithms that we haven't published yet. You can email simon.m.mudd _at_ ed.ac.uk for the binaries until the code is released (probably in 2014). If you do have the full package read the chi_analysis_getting_channel.docx document and skip to step 3b.

Users who don't have the full LSDTopoToolbox package will need to write a script to convert profile data of channels and channel networks into channel files. The channel file starts with six lines of header information that is used to reference the channel to a DEM. If the channel is not generated from a DEM these six rows can contain placeholder values. The six rows are

```
Nrows <- number of rows
Ncols <- number of columns
Xllcorner <- location in the x coordinate of the lower left corner
Yllcorner <- location in the y coordinate of the lower left corner
Node_spacing <- the spacing of nodes in the DEM
NoDataVal <- the value used to indicate no data
```

This header information is *not* used in the segment analysis; it is only preserved for channel data to have some spatial reference so that scripts can be written to merge data from the channel files with DEM data.

The rest of the channel file consists of rows with 9 columns.

- The first column is the channel number (we use c++ style zero indexing so the main stem has channel number 0).
- The second column is the channel number of the receiver channel (the channel into which this channel flows). The main stem channel flows into itself, and currently the code can only handle simple geometries where tributaries flow into the main stem channel *only*, so this column is always 0.
- The third column is the node number on the receiver channel (which, recall, ***must be the main stem***) into which the tributary flows. The main stem is defined to flow into itself. Suppose the main stem has 75 nodes. The third column would then be 74 for the main stem (because of zero indexing: the first node in the main stem channel is node 0). Nodes are organized from upstream down, so the most upstream node in the main stem channel is

node zero. Suppose tributary 1 entered the main stem on the 65th node of the main stem. The third column for tributary 1 would be 64 (again, due to 0 indexing).

- The 4th column is a 'node reference' which is specific to the University of Edinburgh's DEM analysis code, which is still under development and not being released with the channel profile tool. Users can input a placeholder here (e.g. -99). This number is not used in the analysis.
- The 5th column is the row in a DEM the node occupies: it can be used to refer the channel back to a DEM, but is not used in the analysis.
- The 6th column is the column in a DEM the node occupies: it can be used to refer the channel back to a DEM, but is not used in the analysis
- The 7th column is the flow distance from the outlet of the node. It should be in metres.
- The 8th column is the elevation of the node. It should be in metres.
- The 9th column is the drainage area of the node. It should be in metres squared.

Many of these columns are not used in the analysis but are there to allow the user to refer the channel file back to a DEM. The columns used in the segment fitting analysis are the 1st, 2nd, 3rd, 7th, 8th and 9th columns, the other columns can be occupied by placeholder values. Columns are separated by spaces so rows will have the format

```
Chan_number receiver_chan receiver_node node_index row col flow_dist elev drainage_area
```

Here are the first few lines of an example file:

```
2907
3473
548517
4.40339e+06
10
-9999
0 0 1793 1619544 735 720 59819.3  408.117 30000
0 0 1793 1622700 736 719 59805.1  406.679 30300
0 0 1793 1625857 737 718 59791  404.598 31000
0 0 1793 1629014 738 717 59776.8  402.726 43900
0 0 1793 1632173 739 717 59766.8  400.542 45900
0 0 1793 1635333 740 717 59756.8  399.258 47100
```

b. Performing a statistical analysis to constrain the most likely *m/n* ratio

Once the profile data has been converted into a .chan file, the data can then be processed to determine the most likely *m/n* ratio for individual channels and also via the collinearity test (see Mudd et al (2013, draft manuscript)).

i. Compiling the code

The code for running the statistical analysis to find the most likely *m/n* ratio can be compiled by calling the makefile `chi_m_over_n_analysis.make`. If you are using a windows machine and have installed Cygwin (see appendix) you need to ensure that you have installed the make utility. To make the file navigate the folder that contains it and run

```
make -f chi_m_over_n_analysis.make
```

This will create the program chi_`m_over_n_analysis.exe`.

ii. Running the code

This program is run with 2 arguments to the command line. The first argument is the path name of the path where the .chan file is located, along with a driver file that contains the parameters of the analysis. All data will be printed to files in this path. The second argument is the name of the driver file. We typically use a .driver extension for the driver file but this is not a requirement. For example, we call the program with

```
./chi_m_over_n_analysis.exe
/home/smudd/papers/Segment_fitting/Data_and_code_repository/Data/PA/ chi_parameters.driver
```

The '`./`' leading '`chi_m_over_n_analysis.make`' is only necessary on a Linux system. The driver file contains a number of parameters for running the analysis. This file is used on several different programs so not all parameters are used by `chi_m_over_n_analysis.exe`. The parameters must be listed in the right order and there cannot be any extra information between parameters (e.g., a string describing them). The parameters are:

- 1st row: the prefix of the channel file (that is, the bit without the .chan). So, if the channel file is some_channels.chan the first row of the driver file would be some_channels
- 2nd row: Not used by `chi_m_over_n_analysis.exe`.
- 3rd row: Not used by `chi_m_over_n_analysis.exe`. For reference it is the junction index of the junction that is the downstream starting point of the analysis.
- 4th row: Not used by `chi_m_over_n_analysis.exe`.
- 5th row: Not used by `chi_m_over_n_analysis.exe`.
- 6th row: A reference drainage area for integrating to chi space (A_0)
- 7th row: The minimum number of pixels in a segment. See Mudd et al (2013, draft manuscript) for guidance. Values between 10-20 are recommended. The computational time required is a highly nonlinear inverse function of this parameter. 20 might lead to a run lasting a few minutes, whereas 5 might take many hours (or even days).
- 8th row: The standard deviation of error (σ in the Mudd et al paper) on the DEM, and also some error from geomorphic noise (i.e., a boulder in the channel). For SRTM this should be something like 10-30 m. The larger this number, the fewer segments you will get (see Mudd et al 2013 draft manuscript).
- 9th row: The starting *m/n* value to test if it is the most likely.
- 10th row: The change in *m/n* value that you want to loop over (suppose the starting *m/n* is 0.2 and the change in *m/n* is 0.05, then the next *m/n* value after 0.2 is 0.25).
- 11th row: The number of *m/n* values you want to loop through.
- 12th row: The 'target nodes', which is the maximum number of nodes you want to run through the partitioning algorithm at a time. Recommended values are 80-140. The computational time is nonlinearly related to this parameter, 80 might take several minutes whereas 140 will take many hours, and 200 will take months.
- 13th row: The number of iteration on the Monte Carlo routine that finds the statistics for every node in the channel network rather than a subset of nodes.
- 14th row: Not used `chi_m_over_n_analysis.exe`.

- 15th row: The vertical drop over which slope-area analysis will be performed.
- 16th row: The horizontal interval over which slope area analysis will be performed
- 17th row: The maximum change in drainage area of an interval for slope-area analysis as a fraction of the area at the midpoint of an interval.
- 18th row: The ‘target skip’, which is the average number of nodes the routine skips when it is trying to compute the best segments. If the DEM is 90m resolution, for example, the resolution the algorithm will work at is ~300 meters. Here is an example file:

```
UniUplift09b_mk4_t6
0.0001
300
0
0.02
1000
20
20
0.2
0.025
26
120
250
0.95
20
500
0.2
2
```

And here is the cheat sheet (also included in driver_cheat_sheet.txt):

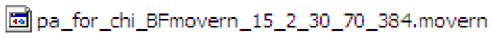
e_bathdem	<- filename prefix
0.0001	<- minimum slope, don't change
300	<- N contributing pixels for a channel. Could reduce to, say 100 or even 50.
1332	<- junction number, this will change
0.05	<- area fraction for channel pruning. 1= mainstem only, low numbers= more tributaries
1000	<- A_0 for chi analysis: probably don't need to change
20	<- minimum segment length. Should be between 5-20.
10.5	<- sigma: some estimate of uncertainty in elevation data. Smaller = more segments
0.15	<- starting m/n for best for m/n testing
0.025	<- increment of m/n for best for m/n testing
21	<- number of m/n values tested for m/n testing
90	<- target length of nodes to be analysed for segments. Should be between 80-150
250	<- number of iterations for Monte Carlo analysis. 250 seems okay
0.95	<- Not used anymore!
20	<- Vertical interval for sampling for S-A analysis. Should be scaled to DEM resolution
500	<- Horizontal interval for sampling for S-A analysis. Should be scaled to DEM resolution.
0.2	<- An area thinning fraction for S-A analysis. 0.2 is probably about right.
2	<- The mean number of data nodes you skip for each node of segment

Once this program has run, it will print out a file with the filename prefix and en extension of .movern.

NOTE: This program is computationally expensive. Increasing the target length of nodes to be analysed and reducing the minimum segment length increases the computational time required in a highly nonlinear fashion. Increasing the skip value can reduce computational time required. You can expect the computation to take several minutes (e.g. minimum segment length ~20, target nodes ~100, skip set so main stem has 300-500 nodes analysed) to many hours (e.g. minimum segment length of 5, target nodes of 120-140, skip set such that thousands of nodes are analysed).

iii. The .movern file

The .movern file is produced by the statistical analysis of the channel network in order to find the most likely *m/n* ratio. The filename contains information about parameter values; these are parsed by the visualisation algorithms. The format of the filename is the **filename prefix**, followed by _BFmovern_, followed by the **sigma** values, the **skip** value, the minimum **segment length** value the **target nodes** value and the **junction number**, all separated by the underscore symbol ('_'). The file then has the extension .movern.


pa_for_chi_BFmovern_15_2_30_70_384.movern

The format of the file is:

- 1st row: In the first column of the first row there is a placeholder value, -99, followed by the *m/n* ratios tested each followed by a space.
- 2nd row: In the first column is a placeholder value, -99, followed by the mean *AICc* (from n_iterations iterations) for each tested *m/n* ratio for the collinearity test. These are separated by spaces.
- 3rd row: In the first column is a place holder value of -99, followed by the standard deviation of the *AICc* for the collinearity test. When fits are extremely poor, the likelihood approaches zero. Calculating the *AICc* involves taking the logarithm of the likelihood, to avoid this, the code assigns a very small number to 0 likelihoods. This results in a high, but not infinite, value of *AICc.* These poor fits will have a standard deviation of zero.
- Even rows thereafter: The first column is the channel number. The following columns are the mean *AICc* values for that channel.
- Odd rows thereafter: The first column is the channel number. The following columns are the standard deviations of the *AICc* values for that channel.

An example file looks like:

```
-99 0.3 0.325 0.35 0.375 0.4 0.425 0.45 0.475 0.5 0.525 0.55 0.575 0.6
-99 4008 4008 4008 4008 4008 4008 4008 3814.91 2244.75 3970.91 4008 4008 4008
-99 0 0 0 0 0 0 0 342.679 320.434 165.212 0 0 0
0 2004 2004 2004 2004 2004 1983.16 896.708 262.442 150.257 388.449 912.111 1726.88 2001.94
0 0 0 0 0 0 117.541 87.9564 27.0629 22.415 71.0617 214.265 338.284 32.5587
1 2041.46 2040.94 2040.53 2034.99 1822.29 506.372 802.273 1397.62 2044.06 2045.3 2047.11 2047.9 2049.58
1 3.2877 3.28297 3.33219 62.2154 368.387 171.395 269.268 360.844 4.82244 5.21089 5.84698 6.65561 7.23849
```

c. Optional: performing a sensitivity analysis on the best fit *m/n* ratio

For structurally or tectonically complex landscapes, it can be difficult to constrain the *m/n* ratio. In such cases, it is wise to perform a sensitivity analysis of the best fit *m/n* ratio. To facilitate this, we provide a python script, `movern_sensitivty_driver_generation.py`, that generates a number of driver files with the parameters minimum_segment_length, sigma, mean_skip and target_nodes that vary systematically. To run this script you will need to change the data directory and the filename of the original driver file within the script. You can then run the script in Spyder (which is installed with the Python(x,y) distribution) or from the command line

```
Python  movern_sensitivty_driver_generation.py
```

Note that if you run from the command line you will need to navigate to the folder that contains the script.

The driver files will be numbered (e.g., my_driver.1.driver, my_driver.2.driver, etc.) and you can run them with a command like:

```
./chi_m_over_n_analysis.exe
/home/smudd/papers/Segment_fitting/Data_and_code_repository/Data/Ape
nnines/ my_driver.1.driver
```

Or if you want to run them with no hangup and 'nice'

```
nohup nice ./chi_m_over_n_analysis.exe
/home/smudd/papers/Segment_fitting/Data_and_code_repository/Data/Ape
nnines/ my_driver.1.driver &
```

And then just keep running them in succession until you use up all of your CPUs (luckily at Edinburgh we have quite a few)!

d. Extracting the chi profiles

The next stage of the analysis is to extract the chi profiles. To compile the program to extract the chi profiles, you need to usethe makefile `chi_get_profiles.make`. The program is compiled with `make –f  chi_get_profiles.make`

The makefile compiles a program called `chi_get_profiles.exe`. This is called, like `chi_m_over_n_analysis.exe`, with two arguments: the path name and the driver name. The driver file has exactly the same format as the driver file for `chi_get_profiles.exe`. A chi profile will be produced for each *m/n* value outlined by these elements in the driver file:

- 7th row: The starting (lowest) *m/n* value to test if it is the most likely.
- 8th row: The change in *m/n* value that you want to loop over (suppose the starting *m/n* is 0.2 and the change in *m/n* is 0.05, then the next *m/n* value after 0.2 is 0.25).
- 9th row : The number of *m/n* values you want to loop through.

Users may wish to modify these values in the driver file from the original values to explore only those values which have 'plausible' values of the *m/n* ratio (see Mudd et al (2013 draft manuscript) for guidance). For each *m/n* ratio tested, the code produces a file with the extension .tree and the string within the filename "_fullProfileMC_forced_". This filename also contains the *m/n* value so for example a filename might be called:

`pa_basin_fullProfileMC_forced_`**`0.3_5_2_20_100_3124`**`.tree`.

The numbers in the filename are arranged in the following order: ***m/n* ratio**, **sigma value**, **mean skip**, **minimum segment length** and **target nodes**. The final number before the extension (here, 3124) is copied from the 3rd row of the driver file: it is the '**junction number**'. Users can assign different numbers to different basins to facilitate automation of data analysis.

i. The .tree file

The .tree file has as many rows as there are nodes in the channel network. There will be more nodes in the .tree file than in the .chan file because the code extends all tributaries to the outlet. Each row has 23 columns. The columns are:

- 1st column: The channel number (like in .chan file)
- 2nd column: The receiver channel (like in .chan file)
- 3rd column: The node on receiver channel (like in .chan file)
- 4th column: The node index (like in .chan file)
- 5th column: The row of the node (like in .chan file)
- 6th column: The column of the node (like in the .chan file)
- 7th column: The flow distance of the node
- 8th column: The chi coordinate of the node
- 9th column: The elevation of the node
- 10th column: The drainage area of the node
- 11th column: The number of data points used to calculate node statistics. Because of the skipping algorithm (see Mudd et al (2013 draft manuscript)) not all nodes are analysed each iteration.
- 12th column: The mean $M\chi$ value for the node.
- 13th column: The standard deviation of the $M\chi$ value for the node.
- 14th column: The standard error of the $M\chi$ value for the node.
- 15th column: The mean $B\chi$ value for the node.
- 16th column: The standard deviation of the $B\chi$ value for the node.
- 17th column: The standard error of the $B\chi$ value for the node.
- 18th column: The mean *DW* value for the node.
- 19th column: The standard deviation of the *DW* value for the node.
- 20th column: The standard error of the *DW* value for the node.
- 21st column: The mean fitted elevation for the node.
- 22nd column: The standard deviation of the fitted elevation for the node.
- 23rd column: The standard error of the fitted elevation for the node.

If you are familiar with c++, here is the code for printing the .tree file:

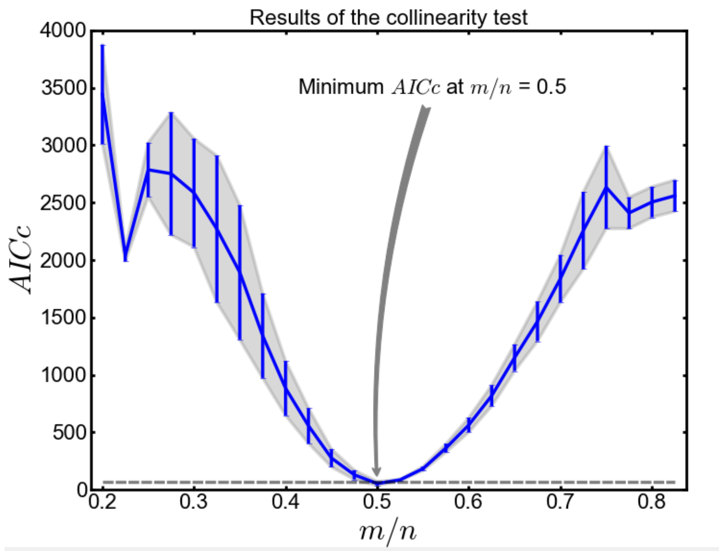
```
channel_profile_out << channel_number << " " << receiver_channel[channel_number] << " "
         << node_on_receiver_channel[channel_number] << " "
         << node[i] << " " << row[i] << " " << col[i] << " " << flow_distance[i] << " "
         << chi[i] << " " << elevation[i] << " " << drainage_area[i] << " "
         << n_data_points_uic[i] << " "
         << m_mean[i] << " " << m_standard_deviation[i] << " " << m_standard_error[i] << " "
         << b_mean[i] << " " << b_standard_deviation[i] << " " << b_standard_error[i] << " "
         << DW_mean[i] << " " << DW_standard_deviation[i] << " " << DW_standard_error[i] << " "
         << fitted_elev_mean[i] << " " << fitted_elev_standard_deviation[i] << " "
         << fitted_elev_standard_error[i] << " " <<  endl;
```

e. Visualising the data

We have also provided python scripts for visualising the data (that is 'visualizing' for you yanks).

i. AICc_plotting.py

This script makes a plot of the *AICc* as a function of the *m/n* ratio for each channel as well as for the collinearity test. The mean and standard deviation of the *AICc* is plotted. In addition the *m/n* ratio with the minimum *AICc* value is highlighted, and there is a horizontal dashed line depicting the minimum *AICc* value plus the standard deviation of the minimum *AICc* value. This dashed line can help the user determine which *m/n* ratios are plausible (see Mudd et al (2013, draft manuscript)). Here is an example:



To run the `AICc_plotting.py` script you must modify the path name and filename after line 35 of the script. The file it takes is an .movern file.

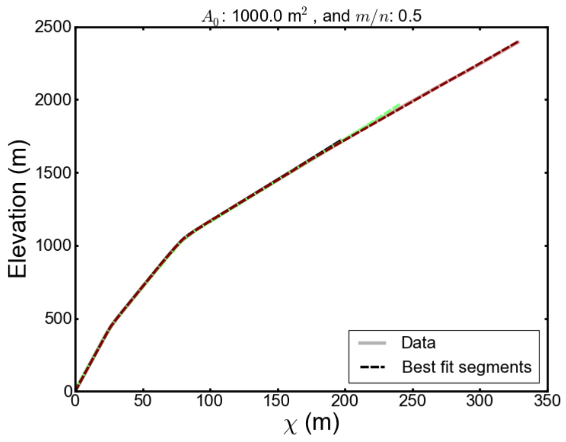
ii. AICc_plotting_multiple.py

This script looks through a directory (you need to change the DataDirectory variable in the script) for any files with *BF_movern_*.movern in them and plots the *AICc* results. The code extracts the parameter values from the filename so each plotted figure has the parameter values in the title. Note that this script plots to file instead of to screen. You can change the kind of output file by changing the parameter ‘`OutputFigureFormat`’. See MatPlotLib (http://matplotlib.org/) documentation for options, but possibilities include ‘jpg’ and ‘pdf’.

iii. chi_visualisation.py

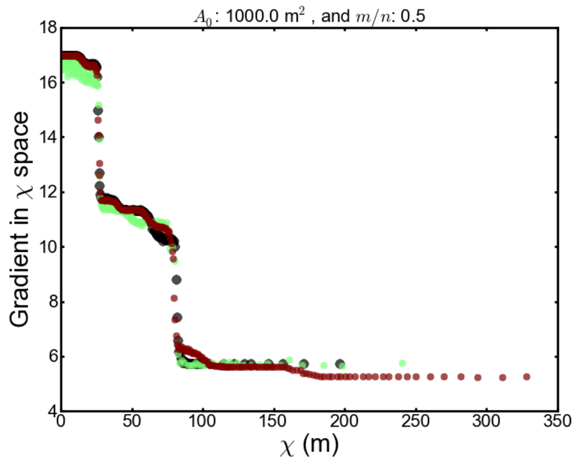
This script makes three figures. First, you must define the path name and the filename after line 39 of the script. This script takes a .tree file. The first figure is a plot of the channels in chi space. The transformed data is in a semi-transparent solid line and the best fit segments are in dashed lines.

Each tributary is plotted with a different colour. Here is an example:

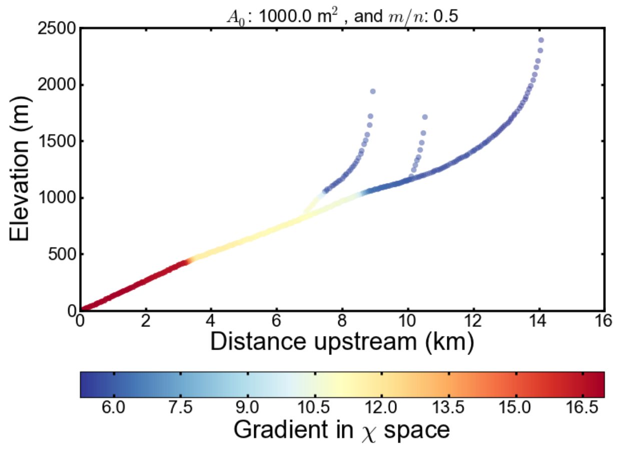


NOTE: These examples are derived from numerical model landscapes and are 'perfect'. Natural channels will be considerably noisier.

The second figure generated by this script is a figure showing the gradient in χ-elevation space as a function of χ. The gradient in χ-elevation is indicative of a combination of erosion rate and erodability, so these plots allow perhaps a clearer idea of the different segments identified by the segment fitting algorithms. The colour scheme from the first figure is carried forward, so that it is easy to identify the characteristics of each tributary. Here is an example:

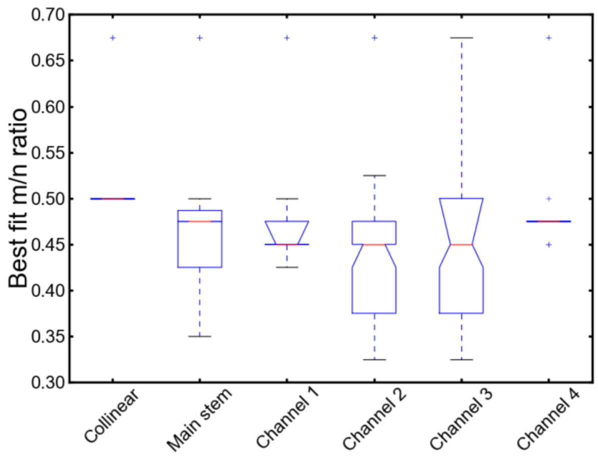


The third figure displays the longitudinal channel profiles (elevation as a function of flow distance), but these channel profiles are coloured by the gradient in χ-elevation space. Here is an example:



iv. bf_movern_sensitivity.py

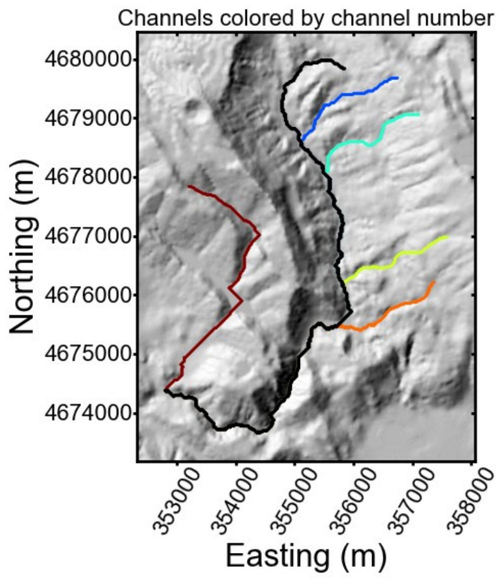
This script looks in the directory DataDirectory for all the files with *BF_movern_*.movern in the filename and then compiles the best fit *m/n* ratio from these files. It then produces box and whisker plots of the best fit *m/n* ratio. The red line is the median *m/n* ratio, the box shows the 25th and 75th percentile *m/n* ratios, the whiskers show the data range, with outliers (as determined by the Matplotlib function boxplot) as '+' symbols. The boxes are notched; these give 85% confidence intervals to the median value after bootstrapping 10,000 times. Here is an example plot:



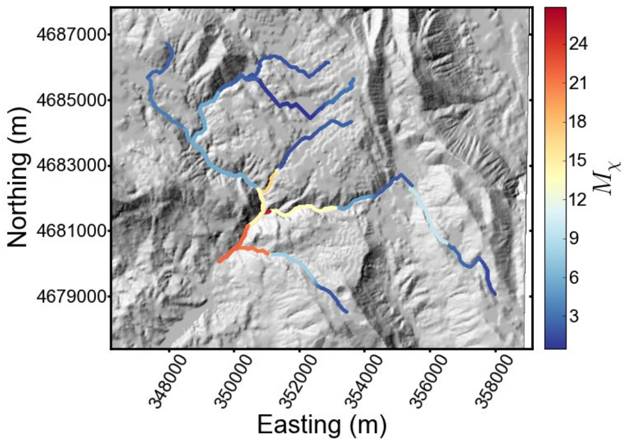
v. raster_plotter_2d_only.py

This script contains two functions. One is for plotting the tributaries superimposed on a hillshade (or any other raster) and another is for plotting the M_χ values superimposed on a raster (we usually do this on the hillshade). For this to work, the .chan file must be referenced to a coordinate system. That means that the row and column information in the .chan file corresponds to the xllcorner, yllcorner and node spacing data in the first few lines of the .chan file. If you have the LSDTopoToolbox this will happen automatically, but if you are writing a script to generate your own .chan files you'll need to ensure that your channel nodes have the correct row and column data. The DEM used for the hillshade **does not** need to be the same as the DEM used for the .chan file (that is, it can have different n_rows and n_columns etc, so you can, in principle, do a chi analysis on a clipped version of a DEM but then plot the results on the full DEM extent. The two functions are:

- coloured_chans_like_graphs: This takes two strings: the filename of the raster and the filename of the .tree file. You have to include the full path name for both of these files. The colouring scheme used for the channels is the same as in the χ–elevation and χ–M_χ plots made by chi_visualisation.py. Here is an example:



- m_values_over_hillshade: Again, this takes two strings: the filename of the raster and the filename of the .tree file. You have to include the full path name for both of these files. The colouring scheme on the M_χ values is the same as in the chi_visualisation.py plot where M_χ is plotted on the channel profile. Here is an example:



To run one or the other of these function you need to scroll to the bottom of the script and comment or uncomment one of the function lines.

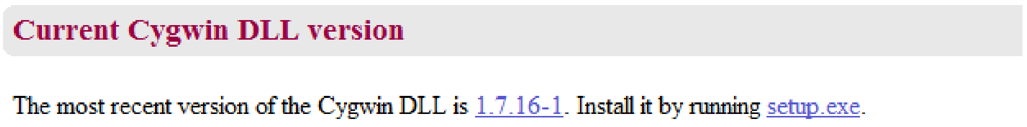
Appendix: Installing a compiler

This section describes how to install the software on your Windows computer

Installing a compiler on a windows machine

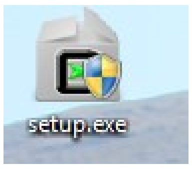
There are several ways to install a compiler on a Windows machine but at Edinburgh when not using Linux we use the Cygwin environment (http://www.cygwin.com/).

1. Go to the Cygwin website and download setup.exe:

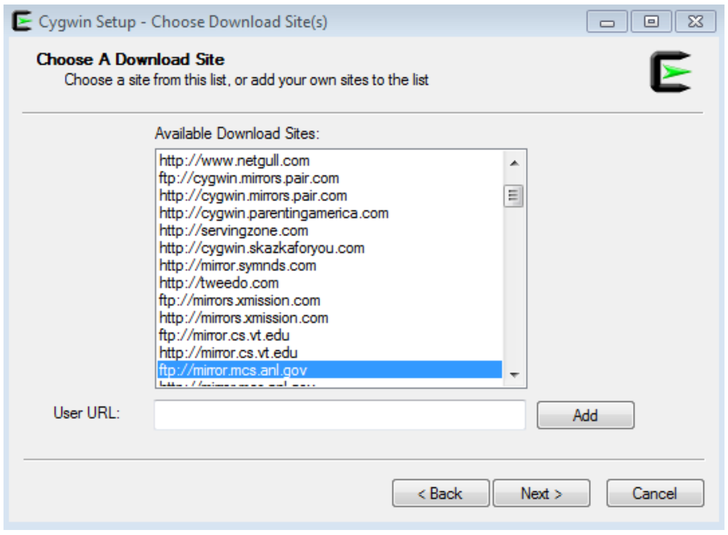


Current Cygwin DLL version

The most recent version of the Cygwin DLL is 1.7.16-1. Install it by running setup.exe.

2. Once you download setup double click to run it: 

3. You will get a bunch of warnings, but just keep clicking next until you get to a screen to choose a 'mirror' site, and just pick a site (it doesn't really matter which one):



Cygwin Setup - Choose Download Site(s)

Choose A Download Site

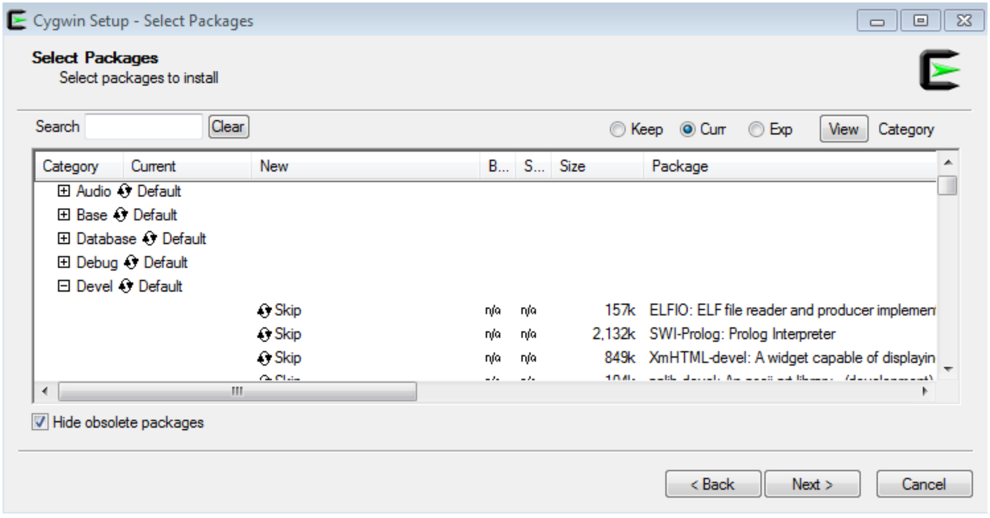
Choose a site from this list, or add your own sites to the list

Available Download Sites:

```
http://www.netgull.com
ftp://cygwin.mirrors.pair.com
http://cygwin.mirrors.pair.com
http://cygwin.parentingamerica.com
http://servingzone.com
http://cygwin.skazkaforyou.com
http://mirror.symnds.com
http://tweedo.com
ftp://mirrors.xmission.com
http://mirrors.xmission.com
ftp://mirror.cs.vt.edu
http://mirror.cs.vt.edu
ftp://mirror.mcs.anl.gov
```

User URL:

4. You will get a window that asks you what packages you want to install. Then expand the 'devel' list of packages:



Cygwin Setup - Select Packages

Select Packages
Select packages to install

Search [ ] Clear ○ Keep ● Curr ○ Exp View Category

Category	Current	New	B...	S...	Size	Package
⊞ Audio ⟳ Default						
⊞ Base ⟳ Default						
⊞ Database ⟳ Default						
⊞ Debug ⟳ Default						
⊟ Devel ⟳ Default						
		⟳ Skip	n/a	n/a	157k	ELFIO: ELF file reader and producer implemen
		⟳ Skip	n/a	n/a	2,132k	SWI-Prolog: Prolog Interpreter
		⟳ Skip	n/a	n/a	849k	XmHTML-devel: A widget capable of displayin

☑ Hide obsolete packages

< Back Next > Cancel

5. You toggle between installing and not installing by clicking these buttons: ⟳. The things that are essential to install are the g++ compiler and the 'make' utility.

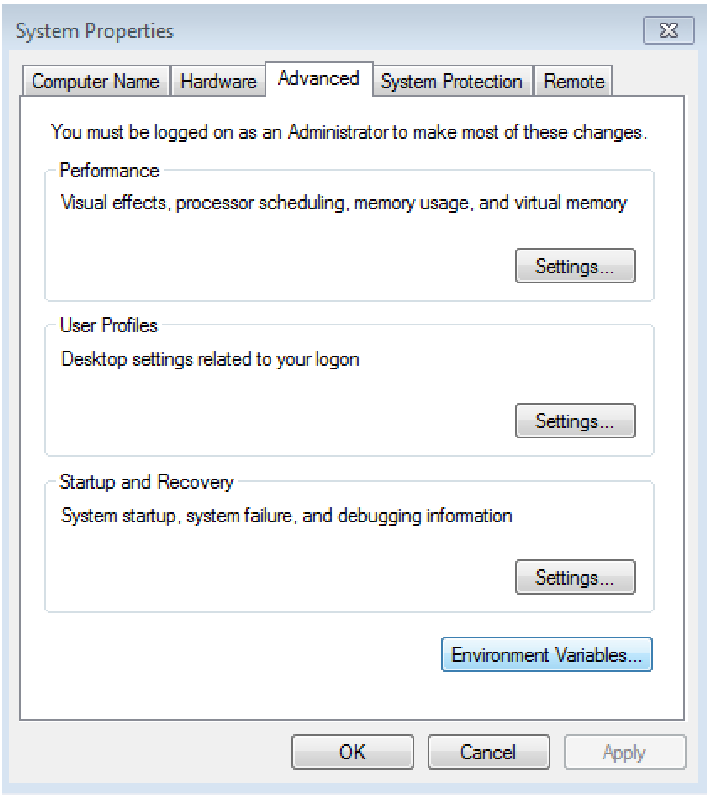
3.4.4-999	⟳ Keep	n/a	☐	7,829k	gcc-g++: C++ compiler
3.82.90-1	⟳ Keep	n/a	☐	442k	make: The GNU version of the 'make' utility

The GNU debugger might also come in handy later.

7.3.50-3	⟳ Keep	n/a	☐	5,600k	gdb: The GNU Debugger

Note that as you learn about these tools you can always run setup again and install more stuff.
6. Once you have got g++ and make, just click next a few times and things will install.
7. Once installation is complete, you need to edit the PATH environment variable on your machine so it isees all the cygwin programs. If you are using windows 7, you can just type 'environment variables' into the search for program bar, and then click on the link to edit

the variables. You will get this screen:



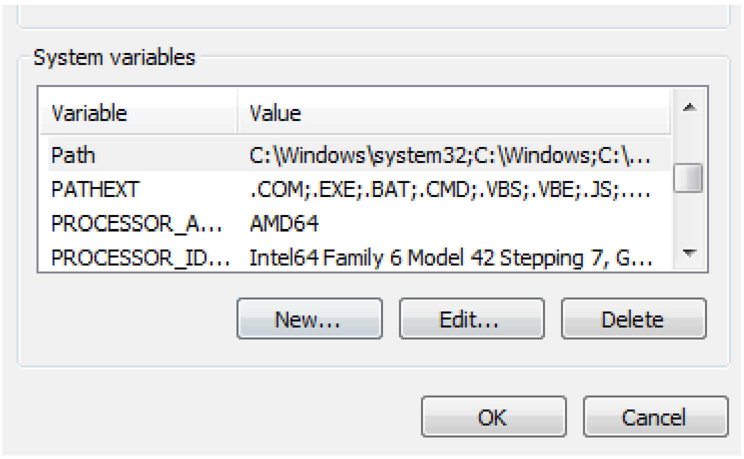
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your logon

Startup and Recovery
System startup, system failure, and debugging information

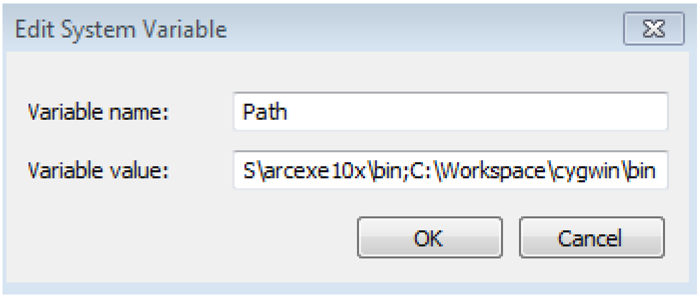
8. Click on the environment variables button and then click on path and then edit



System variables

Variable	Value
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64
PROCESSOR_ID...	Intel64 Family 6 Model 42 Stepping 7, G...

9. Append the path with a semicolon and then the path to the cygwin bin folder:



Edit System Variable

Variable name: Path

Variable value: S\arcexe10x\bin;C:\Workspace\cygwin\bin