LSDTopoTools: open-source software for topographic analysis

Introduction

LSDTopoTools is software for performing topographic analysis of landscapes, with applications in hydrology, ecology, soil science, and geomorphology.

Firstly, we should tell you that LSD stands for *Land Surface Dynamics*, and is named after the Land Surface Dynamics research cluster in the School of GeoSciences at the University of Edinburgh.

Today we'll use the code to extract channel networks, get maps of slope and curvature across the landscape, and do some chi profile analysis.

The philosophy behind LSDTopoTools

LSDTopoTools is a software package designed to analyse landscapes for applications in geomorphology, hydrology, ecology and allied fields. It is not intended as a substitute for a GIS, but rather is designed to be a research and analysis tool that produces reproducible data. The motivations behind its development were:

1. To serve as a framework for implementing the latest developments in topographic analysis.
2. To serve as a framework for developing new topographic analysis techniques.
3. To serve as a framework for numerical modelling of landscapes (for hydrology, geomorphology and ecology).
4. To improve the speed and performance of topographic analysis versus other tools (e.g., commercial GIS software).
5. To enable reproducible topographic analysis in the research context.

The toolbox is organised around objects, which are used to store and manipulate specific kinds of data, and driver functions, which users write to interface with the objects.

Getting started

Installing LSDTopoTools

The preferred way of installing LSDTopoTools is using Docker, software for creating and managing containers. This method should work for anyone with Windows 10 Enterprise, Mac, or Ubuntu/Debian. If you have Windows 10 Home, please see our section on Windows Subsystem for Linux below.

To set up **LSDTopoTools** using Docker:

1. Download and install Docker for Windows (only works with Windows 10 enterprise), Docker for Mac, or Docker for Ubuntu or Debian. On MacOS we recommend installing docker using brew: `brew cask install docker`
2. Ensure that you allocate enough memory to your docker instance. This can be done in the `Advanced` section of the Docker preferences menu. We have tested all of todays examples with 8Gb of memory.
3. Create an `LSDTopoTools` directory on your host operating system that you will share with the **LSDTopoTools** docker containers.
   a. We'll assume this is in `C:\LSDTopoTools` on Windows
   b. OR `\LSDTopoTools` on MacOS and Linux.
4. Pull the full **LSDTopoTools** container and run it with a linked volume:
   a. For windows:

```
$ docker run --rm -it -v C:/LSDTopoTools:/LSDTopoTools
lsdtopotools/lsdtt_alpine_docker
```

   b. For MacOS or Linux:

```
$ sudo docker run --rm -it -v /LSDTopoTools:/LSDTopoTools
lsdtopotools/lsdtt_alpine_docker
```

   c. Or if you have a different directory to LSDTopoTools data on your host machine:

```
$ docker run --rm -it -v /PATH/TO/YOUR/DATA:/LSDTopoTools
lsdtopotools/lsdtt_alpine_docker
```

5. Once you run this, you will need to run the script:

```
# Start_LSDTT.sh
```

Windows Subsystem for Linux

Some users have had difficulties getting Docker to install on Windows as it can require changing a setting in the BIOS. If this is the case, you can try to install the code natively using Windows Subsystem for Linux. This is a piece of software from Microsoft that allows you to run a small Linux environment (in our case Ubuntu) from your Windows operating system. We will download this software from the Microsoft Store, which should be preinstalled on any Windows 10 system.

1. Firstly, download the Ubuntu app from Microsoft Store. If you can't find it, then you can also get it from this link. You might need to restart your computer as prompted following this step.

WARNING	The download here is around ~200 Mb, so make sure you have enough space on your system for this. You don't need to register to the store in order to download the app.

1. Launch the Ubuntu app by clicking on it from the Start menu. You will then be prompted to create a new user account and password - **you can choose whatever you want for this, but make sure you remember it! The password is important for installing packages in the next steps.**
2. The first time you launch, make sure you update and upgrade your standard packages by typing:

```
sudo apt update && sudo apt upgrade
```

3. Now install the dependencies for LSDTopoTools:

```
sudo apt install -y build-essential git gdal-bin python-gdal libfftw3-dev cmake
```

4. Then make a new directory for the LSDTopoTools code and download it from GitHub:

```
mkdir LSDTopoTools && cd LSDTopoTools
git clone https://github.com/LSDtopotools/LSDTopoTools2.git
```

5. This will create a directory called `~/LSDTopoTools/LSDTopoTools2` in your home. Navigate to this directory and run the start up script:

```
cd LSDTopoTools2
bash lsdtt2_setup.sh
```

6. This script will set up the code: we then need one more command to add the binary to your local path. First run:

```
cd bin
pwd
```

7. This will print out a directory path. Copy this directory path, let's call it `/path/to/your/bin` for the sake of this example, and run:

```
export PATH=/path/to/your/bin:$PATH
```

8. This will link the `bin` folder to the system path and make the code executable from anywhere in the Linux subsystem.

Accessing data from Windows

Once installed in the linux subsystem, `Microsoft` actually strongly warns against accessing to Ubuntu file system from `windows` as it can damaged it quite easily. However the opposite is fairy easy: the easiest way to do that is to store the workshop data in your `C` drive, accessible from the linux subsystem with:

+

```
cd /mnt/c/
```

+

Get the workshop data

You can find all of the data you need for the workshop on GitHub. Let's clone this data (either on your Docker container or your Ubuntu subsystem). First, navigate to your LSDTopoTools directory (on Docker this is `/LSDTopoTools/` or on the Ubuntu app `~/LSDTopoTools/`) and then clone the Git repository:

```
git clone https://github.com/LSDtopotools/LSD-workshop.git
```

This will create a directory called `LSD-workshop`. Navigate into this directory and take a look at the files within it. We have provided two main datasets for you to work with: a 1 m DEM from the Pozo catchment, a small catchment on Santa Cruz Island, one of the Californian Channel Islands; and a 10 m DEM of the entire island. You can find these as subfolders within the `data` subdirectory. Let's first of all take a look at the 1 m data:

```
$ cd LSD-workshop/data/Pozo_1m/
$ ls
LSDTT_basic_metrics.param
LSDTT_chi_analysis.param
LSDTT_channel_extraction.param
Pozo_DTM.bil
Pozo_DTM.hdr
Pozo_DTM_hs.bil
Pozo_DTM_hs.hdr
```

The DEM here is in `ENVI bil` format: it consists of two files, `Pozo_DTM.bil` and `Pozo_DTM.hdr`. We also provide three parameter files which all have the extension `.param`. These will be used to run the code throughout the workshop.

Channel extraction

The first part of the workshop will involve testing some different techniques for extracting channel networks from these topographic data. Our channel extraction tool bundles four methods of channel extraction. These are:

- A rudimentary extraction using a drainage area threshold.
- The Dreich method (Clubb et al., 2014).
- The Pelletier (2013) method.
- A geometric method combining elements of Geonet (Passalacqua et al., 2010) and Pelletier (2013) methods that we developed for Grieve et al. (2016) and Clubb et al. (2016) We call this the "Wiener" method (after the wiener filter used to preprocess the data).

These methods are run based on a common interface via the program `lsdtt-channel-extraction`.

The parameter files

Like most of **LSDTopoTools**, you run this program by directing it to a parameter file. The parameter file has a series of keywords. Our convention is to place the parameter file in the same directory as your data.

NOTE	The parameter file has a specific format, but the filename can be anything you want. We tend to use the extensions `.param` and `.driver` for these files, but you could use the extension `.MyDogSpot` if that tickled your fancy.

The parameter file has keywords followed by the `:` character. After that there is a space and the value.

Channel extraction parameter file format

1. Lines beginning with # are comments.
2. Keywords or phrases are followed by a colon (:).
3. The order of the keywords do not matter.
4. Keywords are not case sensitive, but must match expected keywords.
5. If a keyword is not found, a default value is assigned.

Example channel extraction parameter file

Below is an example parameter file. This file is included in the shared folder with the DEM data.

```
# Parameters for channel extraction
# Comments are preceded by the hash symbol
# Documentation can be found here:
# https://lsdtopotools.github.io/LSDTT_documentation/LSDTT_channel_extraction.html

# These are parameters for the file i/o
# IMPORTANT: You MUST make the write directory: the code will not work if it doesn't
exist.
read path: ./
write path: ./
read fname: Pozo_DTM
write fname: Pozo_DTM
channel heads fname: NULL

# Parameter for filling the DEM
min_slope_for_fill: 0.0001

# Parameters for selecting channels and basins
threshold_contributing_pixels: 1000
connected_components_threshold: 100
print_area_threshold_channels: true
print_wiener_channels: true
print_pelletier_channels: false
print_dreich_channels: false

write_hillshade: true
print_stream_order_raster: true
```

For more information on all the different options available see the full channel extraction documentation.

Running the code

To run `lsdtt-channel-extraction`, first navigate to the directory with your data in it:

```
$ cd /LSDTopoTools/LSD-workshop/data/Pozo_1m/
```

All you need to do now is call the program with the parameter file as an argument:

```
$ lsdtt-channel-extraction LSDTT_channel_extraction.param
```

This will automatically run the code for you and produce several different datasets. Now call `ls` to look at the new files that have been created:

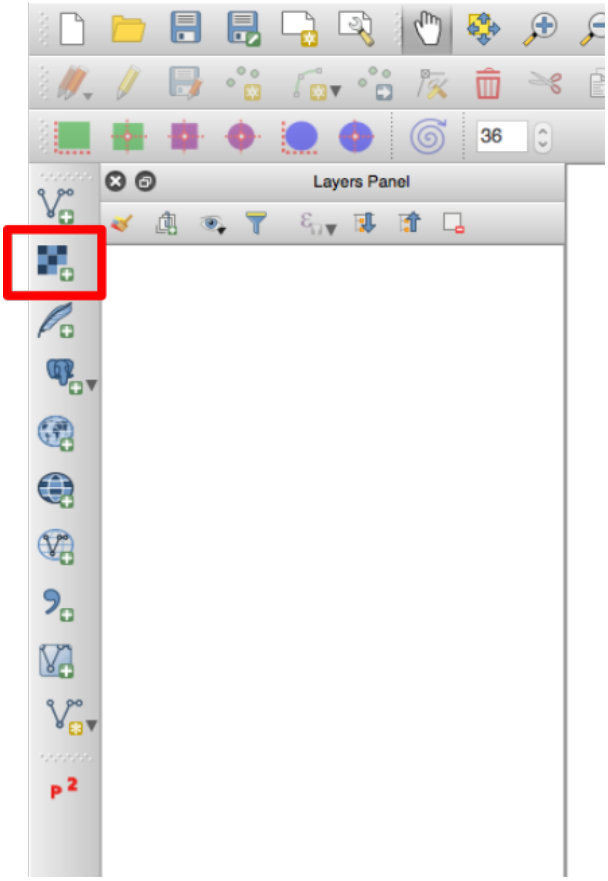
```
$ ls
LSDTT_basic_metrics.param       Pozo_DTM_AT_CN.csv              Pozo_DTM_W_SO.hdr
LSDTT_channel_extraction.param  Pozo_DTM_AT_SO.bil              Pozo_DTM_Wsources.csv
LSDTT_chi_analysis.param        Pozo_DTM_AT_SO.hdr              Pozo_DTM__qq.txt
Pozo_DTM.bil                    Pozo_DTM_ATsources.csv          Pozo_DTM_hs.bil
Pozo_DTM.bil.aux.xml            Pozo_DTM_W_CN.csv               Pozo_DTM_hs.hdr
Pozo_DTM.hdr                    Pozo_DTM_W_SO.bil
Pozo_DTM_ingestedParam.param
```

Visualising the results

For the purposes of this workshop, we will simply visualise the results of the code using GIS software. We will show you how to do this with QGIS, but you can use whatever software you like.

We have just run two different channel extraction methods, an area threshold method, whose outputs are denoted with `_AT`, and the Wiener method, denoted with `_W`.

Open QGIS and click on the Add Raster Layer button on the left of the screen, and select `Pozo_DTM_hs.bil`, `Pozo_DTM_AT_SO.bil` and `Pozo_DTM_W_SO.bil`:



This will load each of the channel networks, coded by Strahler stream order and the hillshade of the catchment. We can then modify the display properties of each of our channel layers, and modify the display order of the layers to visualise the differences between our two networks.

Getting slope and curvature maps

Now we'll do some basic topographic analysis using our program **lsdtt-basic-metrics**. This program can extract topographic metrics like slope, aspect and curvature. The program is intended for everyday data processing. A number of these operations are available in GIS software, but we tend to prefer **LSDTopoTools** because the slope and curvature metrics in typical GIS software only uses information from the nearest pixels. We fit a polynomial surface from a neighbourhood of pixels. See Hurst et al., 2012, DOI: 10.1029/2011JF002057 and Grieve et al., 2016, doi:10.5194/esurf-4-627-2016 for the rationale.

Parameter files

Similar to the channel extraction, we also need to define a parameter file for running the slope and curvature analysis. An example for our DEM is shown below, which again needs to be in the same directory as the DEM:

```
# Parameters for extracting simple surface metrics
# Comments are preceded by the hash symbol

# These are parameters for the file i/o
# IMPORTANT: You MUST make the write directory: the code will not work if it doesn't
exist.
read path: ./
write path: ./
read fname: Pozo_DTM
write fname: Pozo_DTM
channel heads fname: Pozo_DTM_Wsources

# Parameters for surface metrics
surface_fitting_radius: 7
print_slope: true
print_aspect: true
print_curvature: true
print_tangential_curvature: true
```

Run the code

We run the code to get the slope, aspect, and curvature rasters in a similar way to that of the channel extraction. First we navigate to the directory with the data and then use the appropriate parameter file:

```
$ cd /LSDTopoTools/LSD-workshop/data/Pozo_1m/
```

All you need to do now is call the program with the parameter file as an argument:

```
$ lsdtt-basic-metrics LSDTT_basic_metrics.param
```

This will automatically run the code for you and produce several different datasets. We again have lots of new files that have been created:

```
$ ls
LSDTT_basic_metrics.param       Pozo_DTM.hdr
LSDTT_channel_extraction.param  Pozo_DTM_hs.bil
LSDTT_chi_analysis.param        Pozo_DTM_hs.hdr
Pozo_DTM_ASPECT.bil             Pozo_DTM_ingestedParam.param
Pozo_DTM_ASPECT.hdr             Pozo_DTM__qq.txt
Pozo_DTM_AT_CN.csv              Pozo_DTM_SLOPE.bil
Pozo_DTM_AT_SO.bil              Pozo_DTM_SLOPE.hdr
Pozo_DTM_AT_SO.hdr              Pozo_DTM_TANCURV.bil
Pozo_DTM_ATsources.csv          Pozo_DTM_TANCURV.hdr
Pozo_DTM.bil                    Pozo_DTM_W_CN.csv
Pozo_DTM.bil.aux.xml            Pozo_DTM_W_SO.bil
Pozo_DTM_CURV.bil               Pozo_DTM_W_SO.hdr
Pozo_DTM_CURV.hdr               Pozo_DTM_Wsources.csv
```

The slope, curvature, and aspect rasters are named in a self-explanatory fashion, for example `Pozo_DTM_SLOPE.bil` is the slope raster. You can load these into your chosen GIS and take a look at them.

Channel steepness analysis

The final thing we'll do today is to do some channel steepness analysis. For background into channel profile analysis using both slope-area plots and chi, you can refer to our website which has a lot of detail on this. Here, we'll just go through how to actually run the code.

Using the chi mapping tool

This runs in the same way as the channel extraction and the slope and curvature methods. We'll select a series of basins to perform the analysis on, and then print out some csv files and rasters which can be used to visualise channel steepness.

Again, we use a parameter file to run the chi mapping tool. It also has a series of keywords which are used to set which analyses we perform.

We will then visualise the output using QGIS.

Some comments on basin selection

In the **chi mapping tool**, we have several ways to select basins. We feel the default method is best (`find_complete_basins_in_window`). The options are:

- `find_complete_basins_in_window`: This goes through several refining steps. If first checks every basin in the raster and selects basins within a size window between `minimum_basin_size_pixels` and `maximum_basin_size_pixels`. It then takes the resulting list of basins and removes any that are influenced by the edge of the DEM (to ensure drainage area of the basin is correct). Finally, it removes nested basins, so that in the end you have basins of approximately the same size, not influenced by the edge of the DEM, and with no nested basins.

- `find_largest_complete_basins`: This is a somewhat old version that takes all basins draining to edge and works upstream from their outlets to find the largest sub-basin that is not influenced by the edge. To get this to work you **MUST ALSO** set `find_complete_basins_in_window: false`.
- `test_drainage_boundaries`: If either `find_complete_basins_in_window` or `find_largest_complete_basins` are `true` then this is ignored. If not, then it eliminates any basin that is influenced by the edge of the DEM.
- `BaselevelJunctions_file`: If this points to a file that includes a series of integers that refer to junction indices, it will load these indices. If the file doesn't contain a series of integers the most likely result is that it will crash!

Example parameter file

Below is an example parameter file for the Pozo dataset that will extract all the basins and produce chi data for each channel in each basin.

```
# Parameters for performing chi analysis
# Comments are preceded by the hash symbol
# Documentation can be found here:
# https://lsdtopotools.github.io/LSDTopoTools_ChiMudd2014/

# These are parameters for the file i/o
# IMPORTANT: You MUST make the write directory: the code will not work if it doesn't
exist.
read path: ./
write path: ./
read fname: Pozo_DTM
write fname: Pozo_DTM
channel heads fname: Pozo_DTM_Wsources

# Parameter for filling the DEM
min_slope_for_fill: 0.0001

# Parameters for selecting channels and basins
threshold_contributing_pixels: 5000
minimum_basin_size_pixels: 50000
maximum_basin_size_pixels: 600000
test_drainage_boundaries: false
find_largest_complete_basins: false
find_complete_basins_in_window: true

# The data that you want printed to file
write_hillshade: false
print_basin_raster: true

# Chi analysis options
print_chi_data_maps: true
print_basic_M_chi_map_to_csv: true
A_0: 1
m_over_n: 0.45
```

Running the code

Our chi mapping program is called `lsdtt-chi-mapping`. Again, let's navigate to the directory with the Pozo catchment data in it:

```
$ cd /LSDTopoTools/LSD-workshop/data/Pozo_1m/
```

All you need to do now is call the program with the parameter file as an argument:

```
$ lsdtt-chi-mapping LSDTT_chi_analysis.param
```

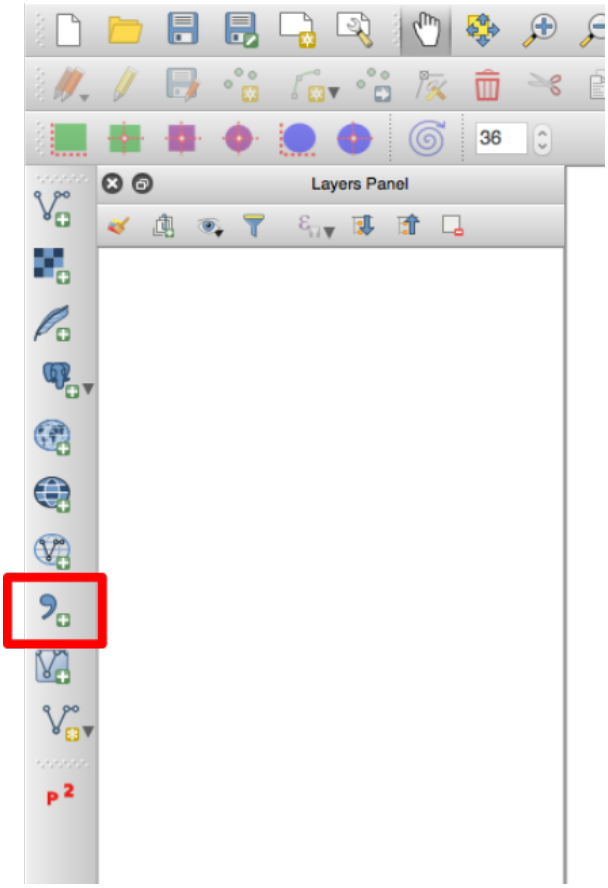
This will automatically run the code for you and produce several different datasets. The key ones that we have generated during the chi analysis are:

1. `Pozo_DTM_AllBasins.bil`: a raster showing the basins that were extracted and analysed.
2. `Pozo_DTM_chi_data_map.csv`: a csv file of the channel network with information about simple parameters such as flow distance, drainage area, and chi coordinate.
3. `Pozo_DTM_MChiBasic.csv`: a csv file with some basic channel steepness information for each point on the channel network.

Visualising the results

The first thing we can do to visualise these results is to load the dataset with the extension `_AllBasins.bil`. This raster gives the locations of all of the basins which have been identified for our analysis. This is always a good initial sanity check of our data. It can be loaded into QGIS in the same manner as each of the previous raster datasets we have loaded.

Now we are ready to look at the results of our chi analysis, and see the spatial distribution of our chi values across the landscape. To do this, we will import the csv file `Pozo_DTM_chi_data_map.csv` into QGIS. This can be done using the Add Delimited Text File button:



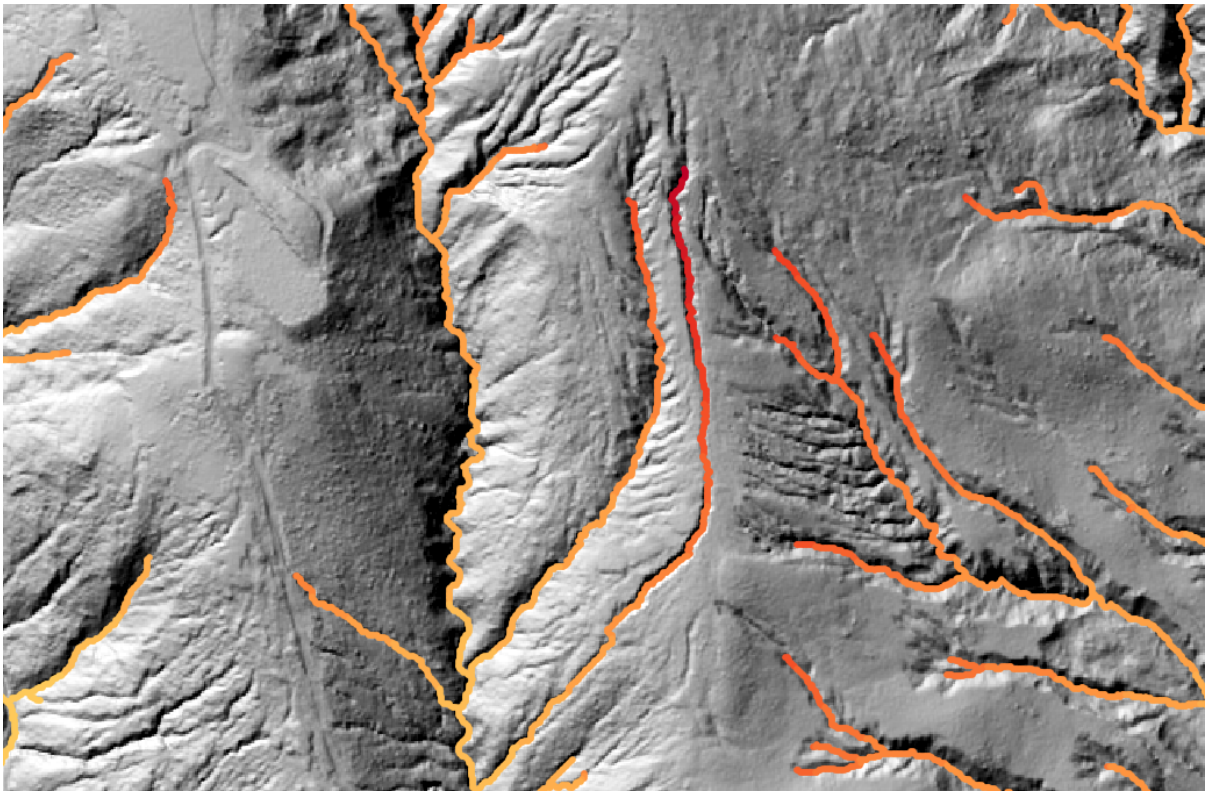
Select the file `Pozo_DTM_chi_data_map.csv` and QGIS should automatically recognise the latitude and longitude columns of the dataset. Click OK, and a dialogue will pop up asking you to select a coordinate system for the data. **Click Cancel** and the data will be projected on-the-fly and should overlie your hillshade and other datasets.

We now want to colour our points based on their chi value to see how this value varies across our

catchments. Right click on the newly created point data layer and select properties, and navigate to the style tab. At the very top of the window, click on **Single symbol** and select **Graduated** instead. We can now select our chi values from the Column dropdown option, choose an appropriate sequential colour ramp to colour our points based on their chi value.

Choose a reasonably large number of classes (around 20) so that we have smooth transitions between colours, and click the classify button to populate the list of classes with our values. At this point if you click apply we will be able to see the colours of our points changing, but because of their dark outlines it is hard to see. Lets turn those off. Click on the change symbol button at the top of the window, and in the dialogue which opens, select simple marker. Now set the outline to transparent, and click OK.

Your data should now look something like this:



and you should be able to explore the spatial variability of chi values across the Pozo catchment.

Python scripts

We're also working on some python scripts for visualising outputs from the software. This repository is messy and not very well ordered yet, but can be cloned here and you can read about our python visualisation tools here