7/12/2019

Ryan Stocks

RE100, ANU

How to use PHES searching software

User guide

Contents

[Input 2](#_Toc106134640)

[Storage Location 2](#_Toc106134641)

[Digital Elevation Model 2](#_Toc106134642)

[Shapefile Exclusions (Filters) 2](#_Toc106134643)

[Existing Reservoirs 2](#_Toc106134644)

[Shapefile 2](#_Toc106134645)

[CSV Data 2](#_Toc106134646)

[Names 3](#_Toc106134647)

[Generating names csv from dbf 3](#_Toc106134648)

[Variables 3](#_Toc106134649)

[Generating Shapefile Tiles 6](#_Toc106134650)

[Single Run 6](#_Toc106134651)

[Bulk Run 6](#_Toc106134652)

[Tasks File 6](#_Toc106134653)

[Processes File 7](#_Toc106134654)

[Debug 7](#_Toc106134655)

# Input

## Storage Location

In the home directory of the *PHES\_Searching* folder there should be a file called storage\_location. In this file there should be a single line describing the path to the input/output folder in the format “*file\_storage\_location = <path to storage location>;”*. For example, to use the same *PHES\_Searching* folder, use “*file\_storage\_location = ../PHES\_Searching/;”* or for the folder on the pv network drive at ANU use “*file\_storage\_location = ../../../../../p/PHES\_Searching/;”.*

This location will be referred to as *<storage location>* hereafter.

To use a network location, first ensure the computer is connected to the server (Can be accessed through windows explorer). Then in either linux bash or ubuntu (either can be installed on a windows computer), navigate to the home directory (Type “*cd ..”* as many times as needed), then mount the server to the p drive using *“sudo mount -t drvfs \\\\piso.cecs.anu.edu.au\\pv mnt/p/”* (Adjust as needed for an alternate system).

## Digital Elevation Model

Place all required DEM files (1 degree by 1 degree with 1 arcsecond resolution supported, other resolutions unknown) in the *<storage location>/input/DEMs* folder with filename *<latitude>\_<longitude>\_1arc\_v3.tif* (e.g. n00\_e000\_1arc\_v3.tif).

## Shapefile Exclusions (Filters)

If required, place all shapefile exclusions (filters) in the *<storage location>/input* folder and add them as filters in the variables file (See below). For faster runtimes on bulk runs, see generating shapefile tiles below.

## Existing Reservoirs

**NOTE: This is only required for brownfield searching, steps below can be ignored for purely off-river searching.**

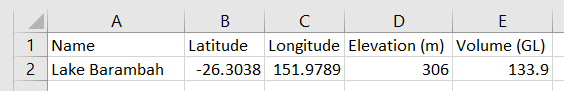
### Shapefile

Place the shapefile containing all existing reservoirs to be checked in the *<storage location>/input/existing\_reservoirs* folder and add its filename as the existing\_reservoirs\_shp in the variables file (See below).

### CSV Data

Place the csv spreadsheet containing information about the reservoirs to be tested in the *<storage location>/input/existing\_reservoirs* folder and add its filename as the existing\_reservoirs\_csv in the variables file (See below).

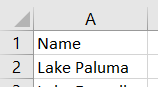
A header row must be present, and columns must match the columns in the image below.



### Names

Place the spreadsheet containing the reservoir names corresponding to the polygons in the shapefile in the order they occur in the shape files (All polygons must be included, details on generating this file from the dbf are outlined below) in the *<storage location>/input/existing\_reservoirs* folder and add its filename as the existing\_reservoirs\_shp\_names in the variables file (See below).

A header row must be present, and columns must match the columns in the image below.



#### Generating names csv from dbf

The csv with a list of names can be generated by using excel to open the dbf file attached to the shapefile of existing reservoirs, finding the column that corresponds to the name and copy and pasting it into a new file (As seen above).

# Variables

All variables that can be adjusted are in the variables file in the home directory of *<storage location>* and are briefly outlined in the table below. Each line is in the format <variable> = <value>; and // is treated as a comment (Similar to C++ formatting).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Variable | Description | Notes | Required for | | | | | | | |
| Search Driver | Shapefile Tiling | Screening | Pairing | Pretty Set | Constructor | Greenfield | Brownfield |
| tasks\_file | String containing filename for text file with list of cells/reservoirs to run. | See Bulk Run below for details | ✓ | ✓ |  |  |  |  |  |  |
| processes\_file | String containing filename for text file with list of programs to run. | See Bulk Run below for details | ✓ |  |  |  |  |  |  |  |
| existing\_reservoirs\_csv | String containing filename for csv file with existing reservoir information. | See Existing Reservoirs input above |  |  |  |  |  |  |  | ✓ |
| existing\_reservoirs\_shp | String containing filename for shapefile with existing reservoir polygons. | See Existing Reservoirs input above |  |  |  |  |  |  |  | ✓ |
| existing\_reservoirs\_shp\_names | String containing filename for csv file with names for existing reservoirs in the shapefile. | See Existing Reservoirs input above |  |  |  |  |  |  |  | ✓ |
| border | Integer number of cells to import as a “border” when screening. (I.e. maximum number of arcseconds for reservoir to extend outside of ‘home’ DEM) | Must be consistent for screening/pairing/pretty set/constructor |  |  | ✓ | ✓ | ✓ | ✓ |  |  |
| dambatter | Decimal representing the inverse gradient of dam wall (i.e. a value of 3.0 means a dam wall that goes up 1m every 3m horizontally) |  |  |  | ✓ |  | ✓ | ✓ |  |  |
| cwidth | Decimal representing the width of the top of the dam wall in metres. |  |  |  | ✓ |  | ✓ | ✓ |  |  |
| freeboard | Decimal representing height of top of dam wall above top water level. |  |  |  | ✓ |  | ✓ | ✓ |  |  |
| min\_watershed \_area | Decimal representing the minimum watershed area in hectares to be considered as a pour point. |  |  |  | ✓ |  |  |  |  |  |
| contour\_height | Integer height difference in metres to define pour points on streams. |  |  |  | ✓ |  |  |  |  |  |
| min\_reservoir \_volume | Decimal defining the minimum potential volume to be considered as a potential reservoir. |  |  |  | ✓ |  |  |  |  |  |
| min\_reservoir \_water\_rock | Decimal defining the minimum best-case water to rock to be considered as a potential reservoir. |  |  |  | ✓ |  |  |  |  |  |
| min\_max\_dam \_height | Minimum potential dam wall height before entering an excluded area to be considered as a potential reservoir. |  |  |  | ✓ |  |  |  |  |  |
| filter | Filters for exclusion of reservoirs, for example world protected areas. Special values of *filter = use\_world\_urban;* and *filter = use\_tiled\_filter*; will use the world urban filter if available and the generated tiled filter respectively. Other filenames defined as filters will exclude reservoirs from overlapping those polygons.  Filters defined using *filter\_to\_tile* will be tiled using the shapefile tiling program for faster runtimes on bulk runs. | Duplicate definitions will add more filters |  | ✓ | ✓ |  |  |  |  |  |
| dam\_wall\_heights | Comma separated list of integer dam wall heights to test in screening (Used for interpolation later) | More dam wall heights will give more precise results but also require more storage and time |  |  | ✓ | ✓ | ✓ | ✓ |  |  |
| min\_head | Integer representing the minimum head (Height separation) between two reservoirs to be considered a potential pair. |  |  |  |  | ✓ |  |  |  |  |
| max\_head | Integer representing the maximum head between two reservoirs to be considered a potential pair. |  |  |  |  | ✓ |  |  |  |  |
| min\_pair\_water \_rock | Decimal representing the minimum water to rock for two reservoirs to be considered a potential pair. |  |  |  |  | ✓ |  |  |  |  |
| min\_slope | Decimal representing the minimum edge to edge slope for two reservoirs to be considered a potential pair. |  |  |  |  | ✓ | ✓ | ✓ |  |  |
| min\_pp\_slope | Decimal representing the minimum pourpoint to pourpoint slope for two reservoirs to be considered a potential pair. |  |  |  |  | ✓ |  |  |  |  |
| max\_lowers\_per \_upper | Maximum number of lower reservoirs to keep per upper reservoir. | Used to save on storage, keeps highest predicted figure of merits |  |  |  | ✓ |  |  |  |  |
| tolerance\_on \_FOM | Percentage error allowed above maximum figure of merit for potential rough pairs. |  |  |  |  | ✓ |  |  |  |  |
| gravity | Force of gravity in (9.8) |  |  |  |  | ✓ | ✓ | ✓ |  |  |
| generation \_efficiency | Decimal representing percentage efficiency of the turbines. |  |  |  |  | ✓ | ✓ | ✓ |  |  |
| usable\_volume | Decimal representing the percentage of the volume of the dam that is usable. |  |  |  |  | ✓ | ✓ | ✓ |  |  |
| water\_density | Density of water in (1000.0) |  |  |  |  | ✓ | ✓ | ✓ |  |  |
| powerhouse\_coeff | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| power\_exp | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| head\_exp | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| power\_slope \_factor | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| slope\_int | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| head\_coeff | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| power\_offset | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| tunnel\_fixed | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| dam\_cost | Magic | FOM |  |  |  | ✓ | ✓ | ✓ |  |  |
| volume\_accuracy | Decimal representing maximum percentage error on final volume |  |  |  |  |  | ✓ | ✓ |  |  |
| dam\_wall\_height \_resolution | Decimal representing resolution in metres of dam wall heights to test |  |  |  |  |  | ✓ | ✓ |  |  |
| minimum\_dam \_height | Minimum final dam height in metres to be considered a reservoir. | Used to avoid reservoirs in existing depressions |  |  |  |  | ✓ | ✓ |  |  |
| good\_colour | Colour of a pin on a pair with the best class in the kml. | Format: Opacity, Blue, Green, Red |  |  |  |  |  | ✓ |  |  |
| bad\_colour | Colour of a pin on a pair with the worst class in the kml. | Format: Opacity, Blue, Green, Red |  |  |  |  |  | ✓ |  |  |
| upper\_colour | Colour of an upper reservoir in the kml. | Format: Hexadecimal in order Opacity, Blue, Green, Red (e.g. #88F0AA14) |  |  |  |  |  | ✓ |  |  |
| lower\_colour | Colour of a lower reservoir in the kml. | Format: Hexadecimal in order Opacity, Blue, Green, Red (e.g. #88F0AA14) |  |  |  |  |  | ✓ |  |  |
| test | Reservoir sizings to test in the format *<Capacity in GWh>, <Hours of storage>*. | Duplicate definitions will add more tests (Output as separate files) |  |  |  | ✓ | ✓ | ✓ |  |  |
| Class Cutoff | Class cut-offs are defined in the format of a single capital letter, then the power cost and storage cost, e.g.  A = 533, 47; | Duplicate definitions will add more classes |  |  |  | ✓ | ✓ | ✓ |  |  |

# Generating Shapefile Tiles

Ensure filters to tile are set as variables using *filter\_to\_tile = <filename>;* in the variables file at *<storage location>.* Running shapefile\_tiling (use *./bin/shapefile\_tiling* in bash) will then generate shapefile tiles (Subsets of the polygons) for each of the cells in the task list.

# Single Run

To run a single process, use either *./bin/<process> <lon> <lat>* for greenfield searching of a DEM square or *./bin/<process> “<existing reservoir name>”* for a brownfield site. DEMs and existing reservoirs must be run in the order screening -> pairing -> pretty\_set -> constructor.

# Bulk Run

**NOTE: Before a bulk run, delete <storage location>/driver\_files and <storage location>/debug. If in <storage location> directory, can use *make clean* command instead (If make file present).**

To start a bulk run, ensure variables and tasks/processes files (See below) are set correctly, then use *./bin/search\_driver* to start one driver or *make run n=<num processes>* to start several drivers in parallel. The drivers will automatically go through the tasks and processes, storing any error information (See debug below) and will output done when all tasks are complete.

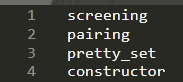
## Tasks File

The tasks file (filename specified in variables) contains a list of all the tasks to complete (DEM squares or existing reservoir names). This is a text file with one task per line in the format *<lon> <lat>* for a DEM square or *“<reservoir name>”* for an existing reservoir. For example:



## Processes File

The processes file (filename specified in variables) contains a list of all the processes to complete (Out of screening, pairing, pretty\_set and constructor). This is a text file with one process per line. For example:



# Debug