User manual: IHEWA Python tools (v0.1)

Contents

[1. Introduction 1](#_Toc57321761)

[2. Installation 3](#_Toc57321762)

[2.1. Python set up 3](#_Toc57321763)

[2.1.1. Install Anaconda3 3](#_Toc57321764)

[2.1.2. Install Python package dependency 3](#_Toc57321765)

[2.1.3. Install GDAL and setting the GDAL path 4](#_Toc57321766)

[2.2. IHEWA packages installation 9](#_Toc57321767)

[2.2.1. IHEWAcollect 9](#_Toc57321768)

[2.2.2. IHEWAdataanalysis and IHEWAengine 9](#_Toc57321769)

[2.2.3. Uninstall packages 12](#_Toc57321770)

[3. Usage 13](#_Toc57321771)

[3.1. IHEWAcollect 13](#_Toc57321772)

[3.2. IHEWAdataanalysis 16](#_Toc57321773)

[3.3. IHEWAengine 16](#_Toc57321774)

# Introduction

**IHEWA** stands for **IHE** Delft **W**ater **A**ccounting, a cluster of open-source tools in Python programming language for Water Accounting plus (WA+) framework, which is developed by Water Accounting group at IHE Delft Institute for Water Education. The WA+ process involves collecting data and information (*Inputs*), analyzing and assessing data quality, calculating water accounts, and reporting results in sheets, maps, indicators… (*Outputs*) (Figure 1).

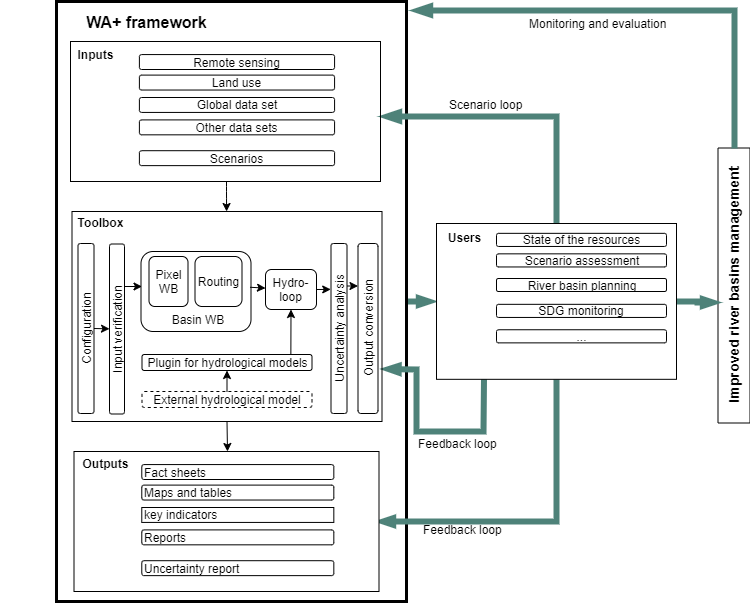


Figure 1: WA+ framework and the link with the water accounting analysts (users)

IHEWA Python tools are developed to assist the water accounting analysts (users) in implementing the WA+ framework. These tools are developed in separate packages so it can also be used for purposes other than water accounting. For example, the IHEWA package for collecting remote sensing data can be installed and used without installation of other IHEWA packages. Below is the overview of the current IHEWA packages:

* **IHEWAcollect:** The first step in WA+ process is collecting inputs which includes remote sensing data. Many remote sensing products are open-access but often provided using different protocols (HTTPS, FTP, API, …). The **IHEWAcollect** package is developed to simplify the process of downloading several datasets in bulk.
  + Github repository: https://github.com/wateraccounting/IHEWAcollect
* **IHEWAdataanalysis:** The next step after collecting remote sensing data is to analyze the data and select suitable products for WA+ report. For example, water balance analysis can be done to assess the quality of the available datasets for the main water balance components such as precipitation (P), actual evapotranspiration (ET) and total water storage change (dS), each of which can be estimated using different products. The **IHEWAdataanalysis** package provides template scripts for such data analyses. At the moment, IHEWAdataanalysis has one template to automatically generate basin average time-series, scatterplot, correlation heat map of possible P-ET-dS combinations from available datasets. Comparing different combinations of P-ET-dS will help water accounting analysts decide better which combination of datasets is suitable for the region of interest. **IHEWAdatanalysis** can be further developed to incorporate other data analyses.
  + Github repository: https://github.com/wateraccounting/IHEWAdataanalysis
* **IHEWAengine:** The last step in WA+ process is to estimate other variables that are not available from remote sensing products, and then synthesize water accounts. For this purpose, many different hydrological models can be potentially used and adapted to WA+ requirements. **IHEWAengine** package is developed with flexible modular structure which provides the WA+ ‘engine’ with a protocol to communicate with different built-in models (developed for WA+) or plug-ins of other common hydrological models (e.g. SWAT+, WEAP, MODFLOW).
  + Github repository: https://github.com/wateraccounting/IHEWAengine

This manual guides the users (water accounting analysts) on how to install Python environment (Anaconda distribution, core packages, projects version control) and IHEWA packages. For each package, an example dataset (test case) is available and users will be guided on how to use the package with the example dataset.

This manual is for Windows users (recommended Operating System: Windows 10 64-bit). For Linux user, all the steps can be found in the Dockerfile in the Github repository.

# Installation

## Python set up

IHEWA Python tools is developed in Python 3.7 environment. We recommend Anaconda3 distribution of Python for simple Python environment and packages manager.

### Install Anaconda3

We suggest to install Anaconda3 (<https://www.anaconda.com/>). It is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

After installing Anaconda3, a Python 3.7 environment must be created since the base environment of the latest Anaconda release is 3.8. First, start Anaconda prompt with ‘Run as administrator’ (Figure 2).

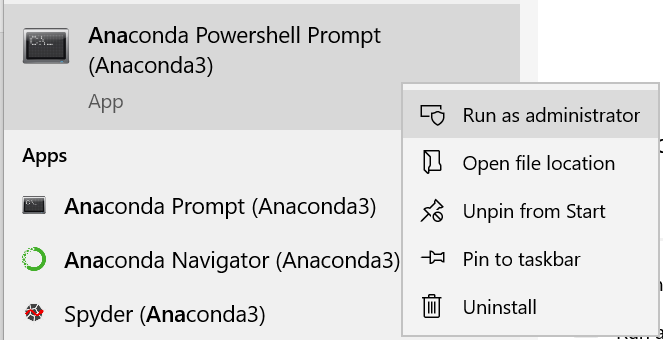


Figure 2: Anaconda prompt

In Anaconda prompt, type in the command:

**conda create –name ihewa python=3.7**

To activate this environment, use the command:

**conda activate ihewa**

### Install Python package dependency

The following table is the list of most important packages required. The full list can be found from “requirements.txt” file in the package repository. To install these packages, open Anaconda prompt and activate the working environment (Python 3.7). Then, run the installation command.

Table 1: Some required packages for IHEWAcollect

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Installation command** |
| beautifulsoup4 | Scrape information from web pages | conda install beautifulsoup4 |
| joblib | Lightweight pipelining in Python | conda install joblib |
| jupyterlab | Web-based user interface for Project Jupyter | conda install jupyterlab |
| numpy | The fundamental package for scientific computing with Python | conda install numpy |
| pandas | Easy to use open source data analysis and manipulation too | conda install pandas |
| matplotlib | Creating static, animated, and interactive visualizations | conda install matplotlib |
| xarray | work with labelled multi-dimensional arrays | conda install xarray |
| paramiko | Implementation of the SSHv2 protocol | conda install paramiko |
| psutil | Process and system utilities | conda install psutil |
| pyproj | Cartographic projections and coordinate transformations library | conda install pyproj |
| pycurl | Fetch objects identified by a URL | conda install pycurl |
| PyYAML | YAML parser and emitter | conda install pyyaml |
| requests | Elegant and simple HTTP library | conda install requests |
| pytest | testing framework Python codes | conda install pytest |
| cairo | 2D graphics library with support for multiple output devices | conda install cairo |
| cairosvg | Convert your SVG files to PDF and PNG | pip install cairosvg |
| gdal | Geospatial Data Abstraction Library is a computer software library for reading and writing raster and vector geospatial data formats | conda install gdal |

### Install GDAL and setting the GDAL path

GDAL is used as the core engine to dealing with geospatial data. If you have not installed GDAL before or if you start from a clean installation of Anaconda, then GDAL should be installed before python can use it. Follow the following steps to install GDAL.

1. Download GDAL binary files

GDAL can be found from different places.

* The source code is located at

<https://github.com/OSGeo/gdal>.

* The unofficial windows binary file is hosted on

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal>.

* The Tamas Szekeres’ windows binaries can be downloaded from

<http://www.gisinternals.com/release.php>.

The following steps are to install GDAL from the binaries downloaded from Tamas Szekeres’ window binaries solution.

1. Before download the binary files, check which version you need. This is concerned with the MSVC version used to compile python and the windows version.

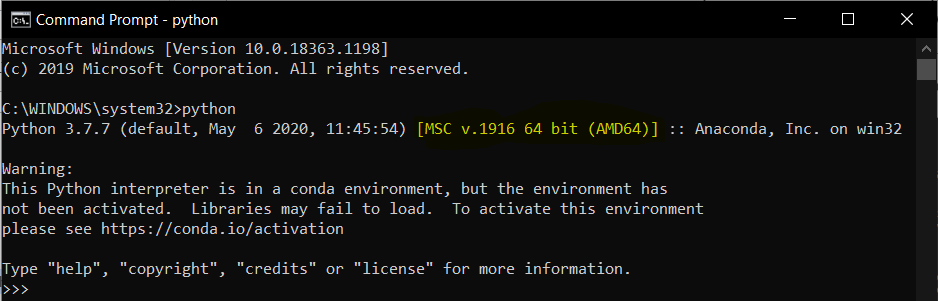
Open command window or Anaconda command and “python” to start python.

Figure 3: Get the MSC and Windows version

The version number from the screenshot is v.1916 and the operating system is 64 bit (AMD64). MSC v.1916 corresponds to MSVC 2017 (see <https://en.wikipedia.org/wiki/Microsoft_Visual_C%2B%2B>).

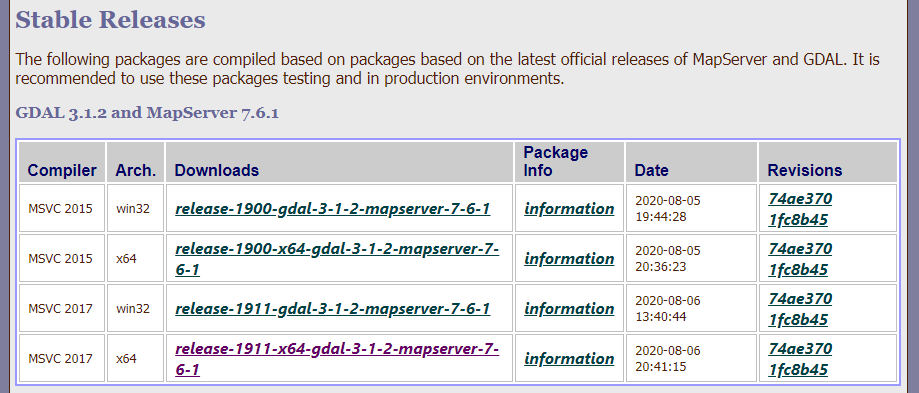
1. Find the corresponding binaries from the link <http://www.gisinternals.com/release.php>. For the above case, the corresponding link is as shown below in the screenshot.

Figure 4: Select GDAL version corresponding to your Python setup

Selecting a release will take you to a new screen promoting a handful of EXE and MSI files. First, download and install the core components of GDAL which is the MSI file corresponding to your python setup. For the above case, the corresponding MSI file is as shown in the screenshot below.

Figure 5: GDAL MIS file corresponding to your Python setup

1. Install the GDAL core MSI file

After downloaded the installation msi file, double click it and the installation will start.

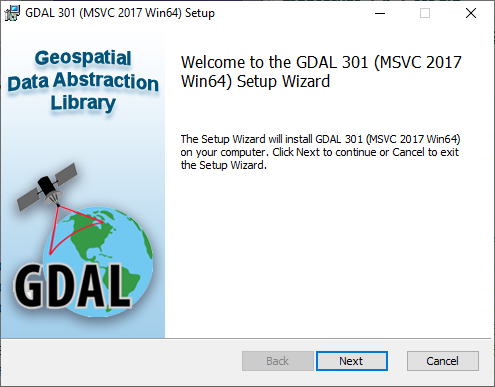


Figure 6: GDAL installation window1

Select “Complete” option.

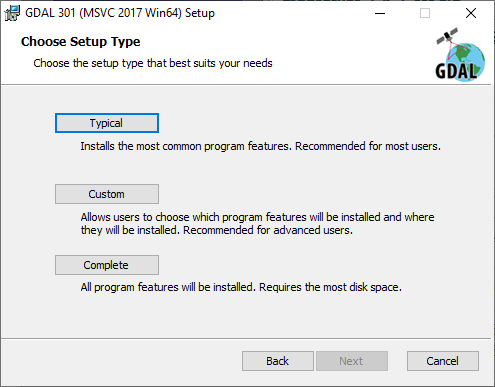


Figure 7GDAL installation window2

1. Setting the GDAL path

After installing GDAL, the next step is to set system path of GDAL, where the system can find the executable files and the data.

If you install GDAl from windows binaries, the default installation path is “C:\Program Files\GDAL”

* + 1. Open “Control Panel”,
    2. go to “System”,
    3. find the “Advanced system settings” on the left side panel,
    4. click it then a system properties window will pop up
    5. Under “Advanced” tab, click the “Environment Variables” button to open the “Environmental Variables” window
    6. Select “path” in the users variable and click on ‘Edit…” to open the “Edit environmental variable” window
    7. Click new and add the GDAL path “C:\Program Files\GDAL”

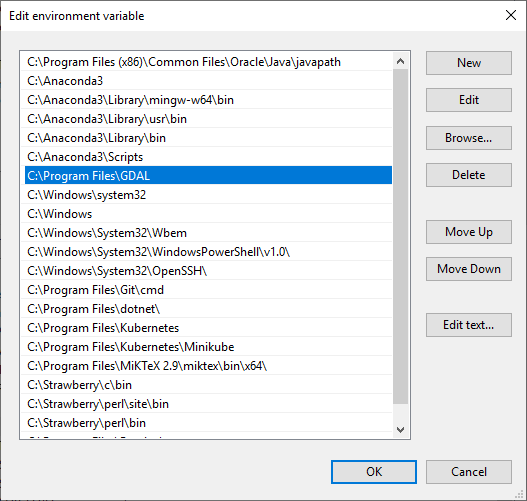


Figure 8: Set GDAL path

* + 1. Close the above window and click “Edit…” under the ‘System Variable” to open the “Edit System variable” window
    2. Add a path variable called “GDAL\_DATA” and set the value to “C:\Program Files\GDAL\gdal-data”

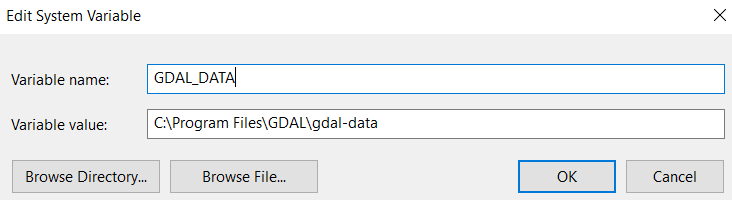


Figure 9: GDAL data system path

* + 1. Add the same way GDAL\_DRIVER\_PATH

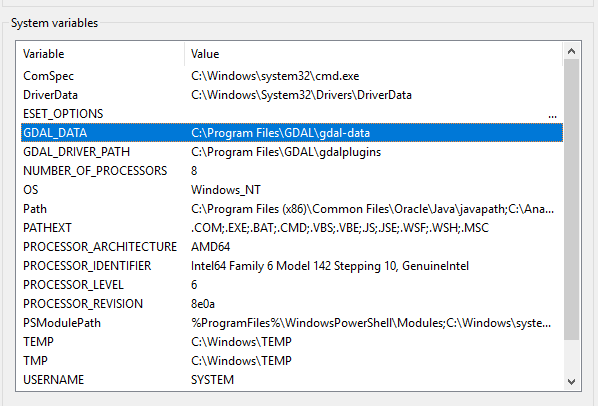


Figure 10: Set GDA\_DATA

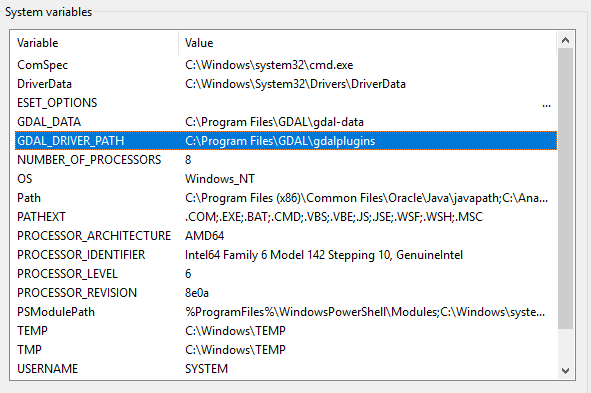


Figure 11: Set GDAL\_DRIVER\_PATH

1. Validate GDAL installation

To validate the installation, open cmd and type “gdalinfo --version”

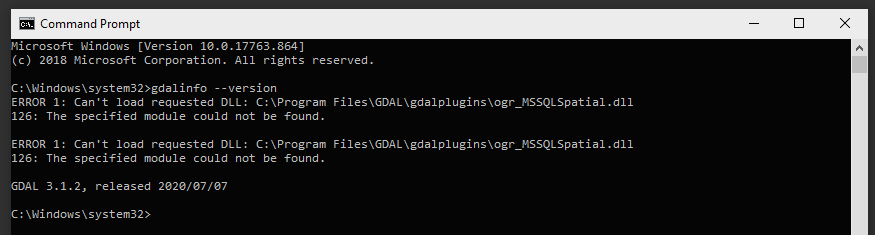


Figure 12: Validate GDAL installation

If you see the gdal version information, the installation is successful

1. Install the python wrapper of GDAL
   * 1. Download python wrapper from the same website. The installation MSI file should fit the python version on your system.
     2. Install the wrapper.



Figure 13: Python GDAL wrapper

* + 1. Validate gdal import. Open the cmd or Anaconda prompt and start python. Inside the python running environment import gdal.

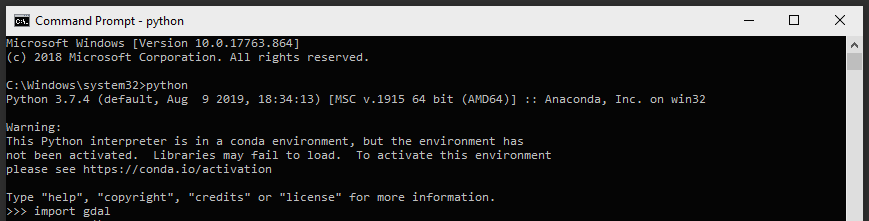


Figure 14: Import gdal

If this failed please look at this and try to correct the paths

https://stackoverflow.com/questions/35500176/importerror-no-module-named-gdal

## IHEWA packages installation

### IHEWAcollect

IHEWAcollect package is hosted on the Python Package Index (PyPI) repository and can be installed using ‘pip’ packages manager

* Install IHEWAcollect package
  1. From PyPi

IHEWAcollect is hosted on PyPi as a python package. To install it from PyPi, type “pip install IHEWAcollect”. All the dependencies will be installed automatically.

* 1. From source code

First clone the source code from github repository. Under the root folder of the repository type “python setup.py install”.

### IHEWAdataanalysis and IHEWAengine

IHEWAdataanalysis and IHEWAengine packages are not yet available on the Python Package Index (PyPI) repository. These packages need to be installed using a setup Python scripts in the package repository. The package repository, therefore, needs to be downloaded first following these steps:

1. Install Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git setup file can be downloaded from <https://git-scm.com/downloads>. Select the latest source release to download and run the setup file with all default options.

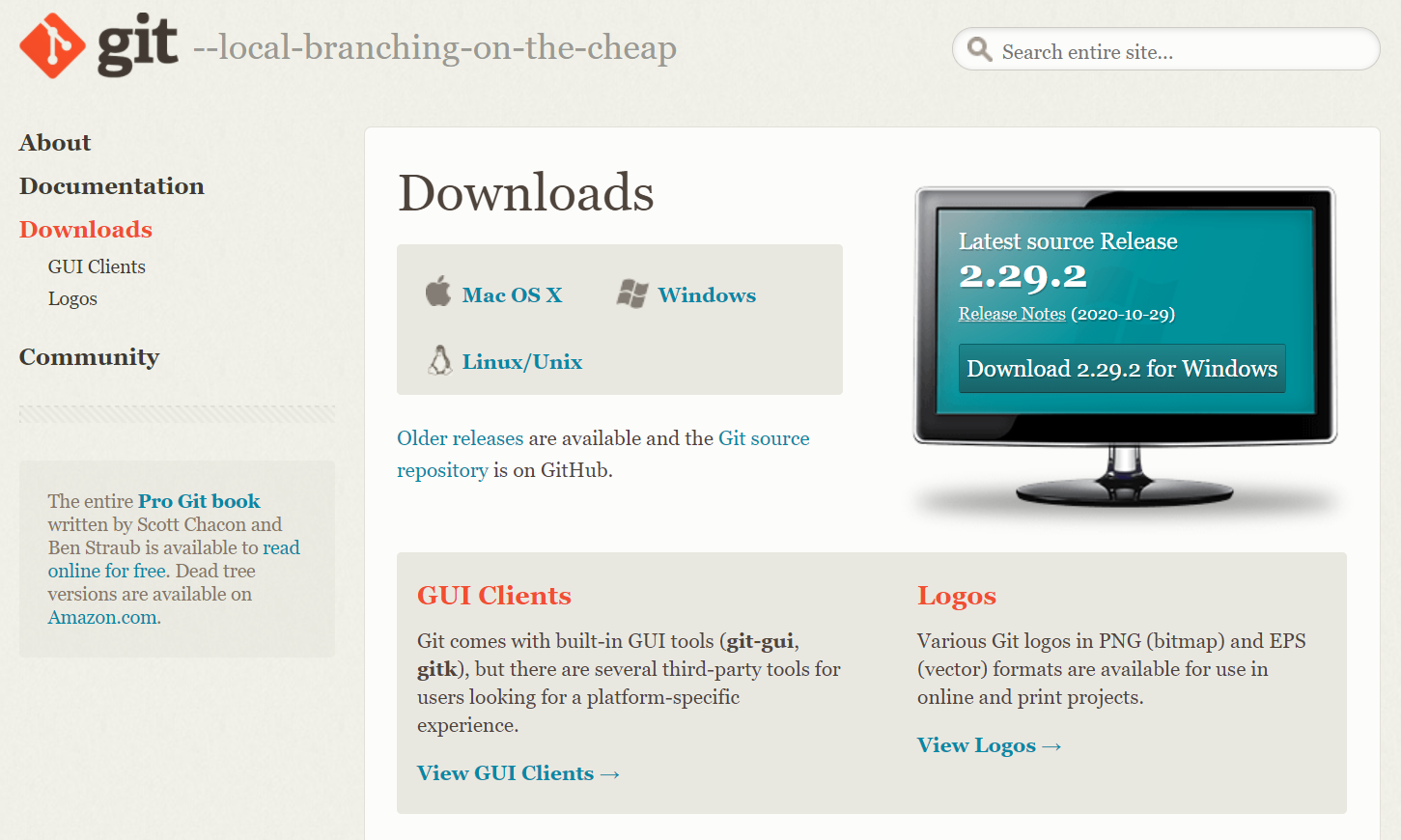


Figure 15: Screen capture from <https://git-scm.com/downloads>

1. Clone package repository

Use Git to clone the whole repository from HTTPS URL:

* For IHEWAdataanalysis: [https://github.com/wateraccounting/IHEWAdataanalysis.git](https://github.com/wateraccounting/IHEWAdataanalysis.gits)
* For IHEWAengine: [https://github.com/wateraccounting/IHEWAengine.git](https://github.com/wateraccounting/IHEWAengine.gite)

The HTTPS URL can also be found on the Github repository of the package (Figure 15):

* For IHEWAdataanalysis: <https://github.com/wateraccounting/IHEWAdataanalysis>
* For IHEWAengine: <https://github.com/wateraccounting/IHEWAengine>

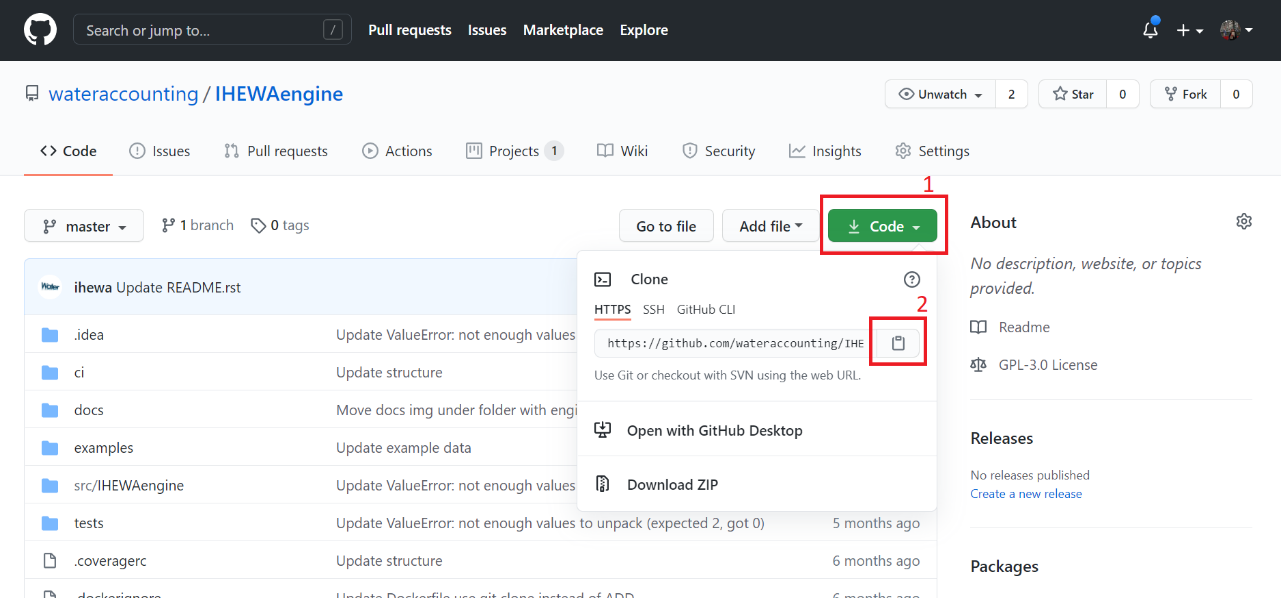


Figure 16: Screen capture from <https://github.com/wateraccounting/IHEWAengine>. Button 1 is to collect the source code. Button 2 is to copy the HTTPS URL.

First, right-click in a folder to download the repository, then select ‘**Git Bash Here**’ (Figure 16)

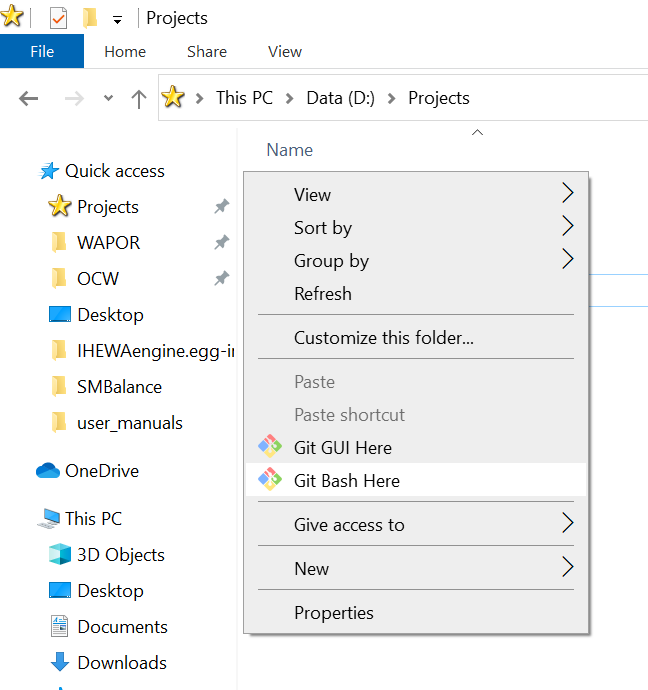


Figure 17: Screen capture of opening Git Bash

In the Git Bash prompt, enter this command: **git clone [HTTPS URL of the package]** (Figure 17)

* For IHEWAdataanalysis: **git clone https://github.com/wateraccounting/IHEWAdataanalysis.git**
* For IHEWAengine: **git clone https://github.com/wateraccounting/IHEWAengine.git**

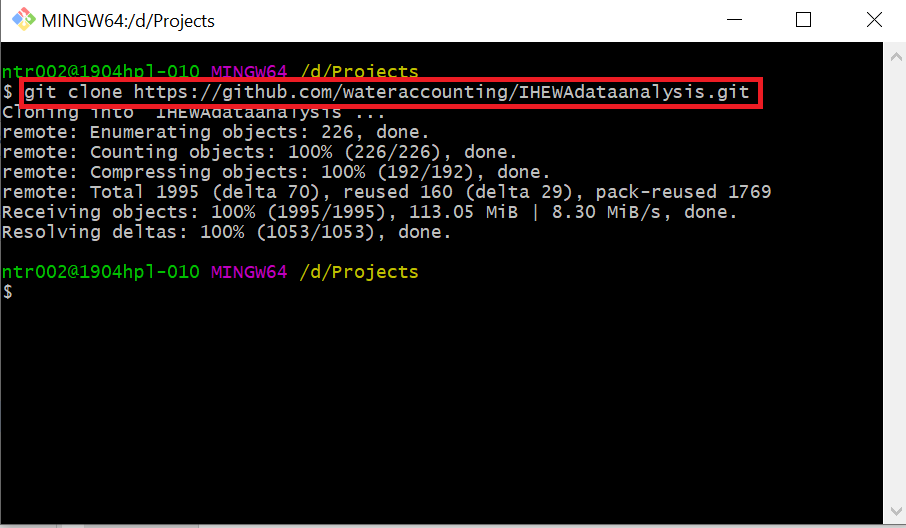


Figure 18: Screen capture of cloning package repository using Git

1. Install package using setup.py

First, run Anaconda prompt as administrator. In Anaconda Prompt window, change working directory to the folder cloned from package repository using cd command. For example, in Figure 17, IHEWAdataanalysis package is cloned in ‘D:\Projects’ folder, then the directory should be changed to ‘D:\Projects\IHEWAdataanalysis’ (Figure 18) using commands:

**D:**

**cd .\Projects\IHEWAdataanalysis\**

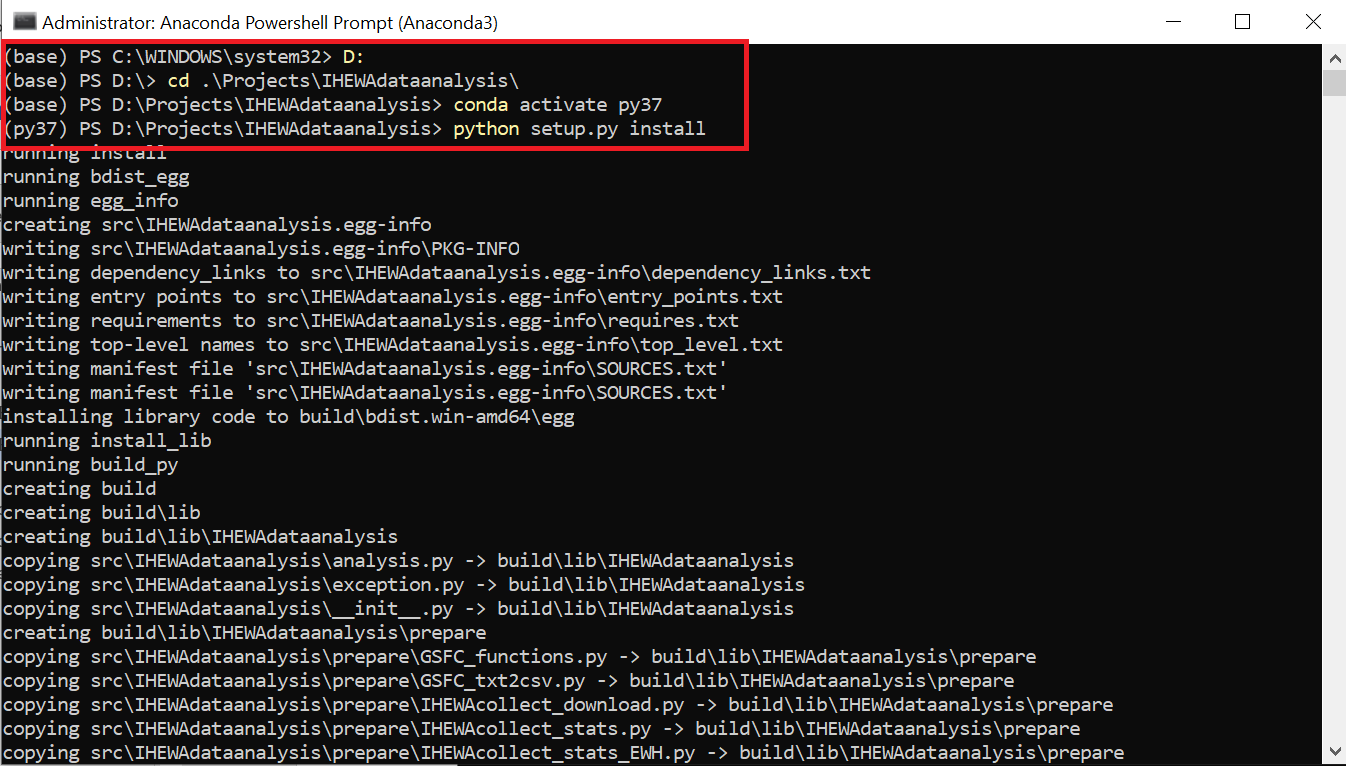


Figure 19: Screen capture of installing IHEWAdataanlysis package using setup file

After that, activate a Python 3.7 environment (set up before). Then, enter the command:

**python setup.py install**

The package is successfully installed once a message ‘*Finished processing*…’ is shown (Figure 19).

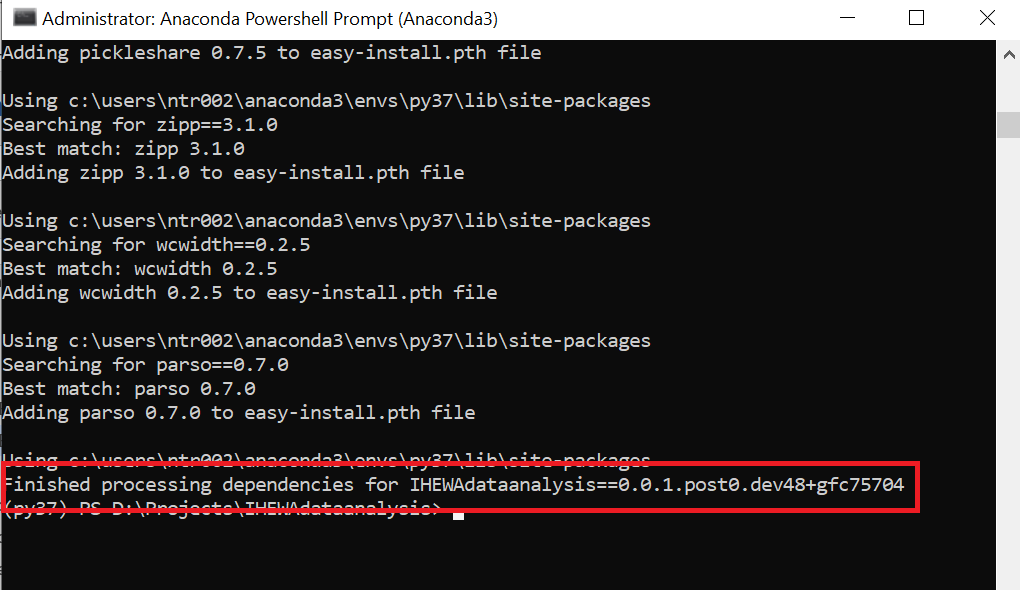
****

Figure 20: Screen capture of successful IHEWAdataanalysis installation

### Uninstall packages

Open Anaconda prompt, activate the Python environment where the package was installed using ‘**conda activate \_env\_**’. Then use the command is “**pip uninstall \_package\_**” to uninstall a package. For example, the step of uninstalling IHEWAengine from py37 environment is given in Figure 20.

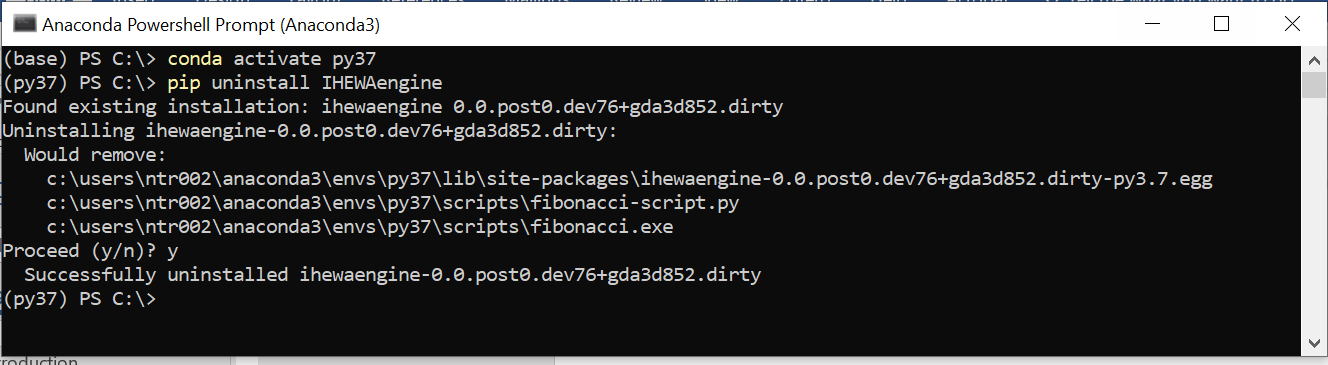


Figure 21: Uninstall IHEWAengine

# Usage

## IHEWAcollect

IHEWAcollect is a collection of tools to download remote sensing data. Most of the remote sensing data require to create an account to be able to access the datasets. Product names, their source and account type for supported products are given in Table 3. The parameters of all the products can be found from <https://github.com/wateraccounting/IHEWAcollect/blob/master/docs/products.rst>.

Table 3: IHEWAcollect supported products

|  |  |  |
| --- | --- | --- |
| Product name | Link | Account type |
| ALEXI | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | IHEWA\_GUESS |
| ASCAT | [https://www.copernicus.eu](https://www.copernicus.eu/) | Copernicus |
| CFSR | [https://www.noaa.gov](https://www.noaa.gov/) | NASA |
| CHIRPS | [https://geog.ucsb.edu](https://geog.ucsb.edu/) | NASA |
| CMRSET | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | IHEWA\_GUESS |
| DEM | [http://earlywarning.usgs.gov](http://earlywarning.usgs.gov/) | NASA |
| ETmonitor | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | IHEWA\_GUESS |
| FEWS | <https://earlywarning.usgs.gov/fews> | NASA |
| GLDAS | <https://ldas.gsfc.nasa.gov/gldas> | NASA |
| GLEAM | [http://www.gleam.eu](http://www.gleam.eu/) | GLEAM |
| GPM | <https://pmm.gsfc.nasa.gov/GPM> | NASA |
| GPM, nc4 | <https://pmm.gsfc.nasa.gov/GPM> | NASA |
| HiHydroSoil | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | IHEWA\_GUESS |
| JRC | [https://global-surface-water.appspot.com](https://global-surface-water.appspot.com/) | NASA |
| MCD12Q1 | <https://lpdaac.usgs.gov/products/mcd12q1v006> | NASA |
| MCD43A3 | <https://lpdaac.usgs.gov/products/mcd43a3v006> | NASA |
| MOD09GQ | <https://lpdaac.usgs.gov/products/mod09gqv006> | NASA |
| MOD10A2 | <https://nsidc.org/data/MOD10A2/versions/6> | NASA |
| MOD11A2 | <https://lpdaac.usgs.gov/products/mod11a2v006> | NASA |
| MOD13Q1 | <https://lpdaac.usgs.gov/products/mod13q1v006> | NASA |
| MOD15A2H | <https://lpdaac.usgs.gov/products/mod15a2hv006> | NASA |
| MOD16A2 | <https://lpdaac.usgs.gov/products/mod16a2v006> | NASA |
| MOD17A2H | <https://lpdaac.usgs.gov/products/mod17a2hv006> | NASA |
| PROBAV | <http://proba-v.vgt.vito.be/en> | VITO |
| RFE | <https://data.noaa.gov/dataset> | NASA |
| SEBS | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | IHEWA\_GUESS |
| SoilGrids | [https://www.isric.org](https://www.isric.org/) |  |
| TRMM | <https://pmm.nasa.gov/trmm> | NASA |
| TWC | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | NASA |
| ALEXI | [https://www.wateraccounting.org](https://www.wateraccounting.org/) | NASA |
| ASCAT | [https://www.copernicus.eu](https://www.copernicus.eu/) | Copernicus |

1. Create account

The first step to use the IHEWAcollect is to create accounts for the dataset you would like to download and save the username and password in the “accounts.yml” file. An example of the structure of “accounts.yml” is given in “accounts.yml-example” file in the GitHub repository (https://github.com/wateraccounting/IHEWAcollect/blob/master/examples/accounts.yml-example).

For example, to use IHEWA\_GUESS account we need to have the following in the “accounts.yml” file.

|  |
| --- |
| IHEWA\_GUESS: |
| username: 'wateraccountingguest' |
| password: 'W@t3r@ccounting' |
| apitoken: null |

Create a folder called “Data” where you would like to save the downloaded data and save the “accounts.yml” file in the folder.

1. Write python script to down the data you need.

Create a python script, and save in the data folder together with the “accounts.yml” file.

The script should have the following line of codes

1. Import packages

# -\*- coding: utf-8 -\*-  
import inspect  
import os  
import IHEWAcollect

1. Write main function. To use IHEWAcollect download datasets, it’s easily to call method “Download”, and pass parameters to the method.

def main(path, test\_args):  
 \_\_  
 for key, value in test\_args.items():  
 print('\n{:>4s}'  
 '{:>20s}{:>6s}{:>20s}{:>20s}{:>20s}\n'  
 '{:->90s}'.format(key,  
 value['product'],  
 value['version'],  
 value['parameter'],  
 value['resolution'],  
 value['variable'],  
 '-'))  
  
 IHEWAcollect.Download(workspace=path,  
 product=value['product'],  
 version=value['version'],  
 parameter=value['parameter'],  
 resolution=value['resolution'],  
 variable=value['variable'],  
 bbox=value['bbox'],  
 period=value['period'],  
 nodata=value['nodata'],  
 is\_status=False,  
 is\_save\_temp=True,  
 is\_save\_remote=True)

1. Define the parameters of each dataset

inspect.getfile(  
 inspect.currentframe()))  
 )  
 # example to download actual-evapotranspiration data for Africa.  
 test\_args = {  
 '1a': {  
 'product': 'ALEXI', # product name  
 'version': 'v1', # version   
 'parameter': 'evapotranspiration', # variable name   
 'resolution': 'daily', # time resolution  
 'variable': 'ETA', # variable code

# the bounding box in degrees of latitude and longitude  
 'bbox': {  
 'w': -5.0,  
 'n': 30.0,  
 'e': 5.0,  
 's': 25.0  
 },

# the start and end of the time period   
 'period': {  
 's': '2005-01-01',  
 'e': '2005-01-02'  
 },

# no data value  
 'nodata': -9999  
 }  
 }  
  
 main(path, test\_args)

To download a product, you need to specify the name of the product, name of the variable, frequency (time resolution of the data), the bounding box of the area in degrees of latitude and longitude and the start and end of the time period for which the data need to be downloaded. These need to be specified as arguments to the main function as shown in the above example.

Once the script Run the example and the accounts file are save in a data folder, then you can run the script in an IDE like Spyder or in the command window.

1. Run the script to download data

To run the script in the command window, start python and type “python *scriptname*.py”, where the “*scriptname”* is the name of the script.

A screen shot of a computer

Description automatically generated

After finish the download data, a “IHEWAcollect” folder will be created UNDER THE DATA FOLDER. Inside it contains 3 sub-folders. The final data is stored inside download folder. The remote folder keeps the data downloaded from server. The temporary folder stores the temporary files during the data processing.

## IHEWAdataanalysis

**IHEWAdataanalysis** can have many templates for data analyses. Currently, the package has one template for comparing main components of water balance (precipitation, actual evapotranspiration, and total water storage change) from different remote sensing products. First, monthly raster data of each product need to be collected, which can be done using **IHEWAcollect**. Then, a monthly basin-wide average time-series and a mean annual raster of each product needs to be prepared in **Step 1**.After that, in **Step 2** and **Step 3,** a configuration file was created to list all available products with the path to the prepared data. In **Step 4,** the information and data from the configuration file (.yml) are passed to the analysis template scripts. All possible combination of P, ET, dS will be generated and compared in heatmap, scatterplot, and maps.

1. Prepare data

Under the “src/IHEWAdataanalysis/prepare” there are python scripts to prepare data from IHEWAcollect download folder. The data preparation steps are:

* 1. Use IHEWAcollect to download remote sensing data
  2. Zonal\_stats is a module to statistical analysis of remote sensing data. The import analysis is mean value of the shape polygon area (sub-basin).

from rasterstats import zonal\_stats

Load shapefile.

ds\_shp\_driver = ogr.GetDriverByName('ESRI Shapefile')  
# 0 means read-only. 1 means writeable.  
ds\_shp = ds\_shp\_driver.Open(file\_shp, 0)

Write csv column names.

# csv column names  
col\_names = []  
# stats\_names = ['count', 'min', 'mean', 'max', 'median']  
for i in range(ds\_shp\_layer\_fea\_n):  
 for stats\_name in stats\_names:  
 col\_names.append('{}\_{}'.format(stats\_name, i))  
fp.write('{dtime},{cols}\n'.format(dtime='date', cols=','.join(col\_names)))

Loop of IHEWAcollect download tif files.

for var in tif.keys():

if date\_month == date.month:

# calculate monthly data

ds\_stats = zonal\_stats(file\_shp, file\_tif\_monthly, stats=' '.join(stats\_names))

* 1. Use gdalwarp to resample and cut tif files.

cmd2 = 'gdalwarp -of GTiff -overwrite -s\_srs epsg:4326 -t\_srs epsg:4326 -r near -tr 0.05 0.05 -cutline {shp} -crop\_to\_cutline {fi} {fo}'

* 1. Create NetCDF file

# create NetCDF file  
nco = netCDF4.Dataset(file\_o, 'w', clobber=True)  
if data is not None:  
 # create dimensions, variables and attributes:  
 nco.createDimension('lon', nlon)  
 nco.createDimension('lat', nlat)  
 nco.createDimension('time', ntime)

nco\_crs = nco.createVariable('crs', 'i4')

nco\_var = nco.createVariable(tif[var]['variable'], 'i2', ('time', 'lat', 'lon'), zlib=True, complevel=9,  
 fill\_value=-9999)

1. Write python script

Create a python script, and name it as “example.py”.

First import packages

# -\*- coding: utf-8 -\*-  
import inspect  
import os  
  
import IHEWAdataanalysis

Second, write main function. To use IHEWAdataanalysis analysis datasets, it’s easily to call method “Analysis”, and pass yaml file to the method.

if \_\_name\_\_ == "\_\_main\_\_":  
 print('\nAnalysis\n=====')  
 path = os.path.join(  
 os.getcwd(),  
 os.path.dirname(  
 inspect.getfile(  
 inspect.currentframe()))  
 )  
  
 analysis = IHEWAdataanalysis.Analysis(path, 'example.yml')  
 # print(analysis.\_Analysis\_\_conf)

1. Write configuration file (.yml)

Create a YAML file, and name it as “example.yml”, which should be placed in the same folder with the python script. Define basin name, example “area1”. Under basin

* “hydrology” block defines the basin area, hydrological year, analysis period and area shapefile.
* “directory” block defines the data path.

areas:  
 area1:  
 hydrology:  
 *# km2 / 1000.0 = Mm2* area: 21.503  
 year: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2]  
 *# csv column* date:  
 - start: '2005-01-01'  
 - end: '2012-12-31'  
 basin:  
 shp:  
 folder: 'data/Shapefile'  
 fname: 'Mindanao-RiverBasin.shp'  
  
 directory:

ETA:  
 ALEXI:  
 tif:  
 folder: 'data/Output/tif'  
 fname: 'ALEXI\_yearly.tif'  
 csv:  
 folder: 'data/Output/csv'  
 fname: 'ETA-ALEXI.csv'  
 column: 'mean\_1'  
 index: 'date'

1. Run the analysis script

In Anaconda prompt window, activate python environment and change working directory to where **example.py** python script is saved. Then, run the command:

**python example.py**

After that, the output graphs and csv files will be saved in the folder **IHEWAdatanalysis** under the working directory (Annex 1).

## IHEWAengine

IHEWAengine packages has 3 main modules: (1) Engine 1 includes pixel-based water balance and routing model, which are used to estimate variables not available from input data (e.g. incremental ET, rainfall ET, runoff, return flow, discharge…), (2) Engine 2 module computes the indicators and values in WA+ sheets from both input and output data of Engine 1, and (3) plugins module is developed to incorporate the results from other hydrological models to WA+ sheets (e.g. SWAT+, MODFLOW, WEAP…).

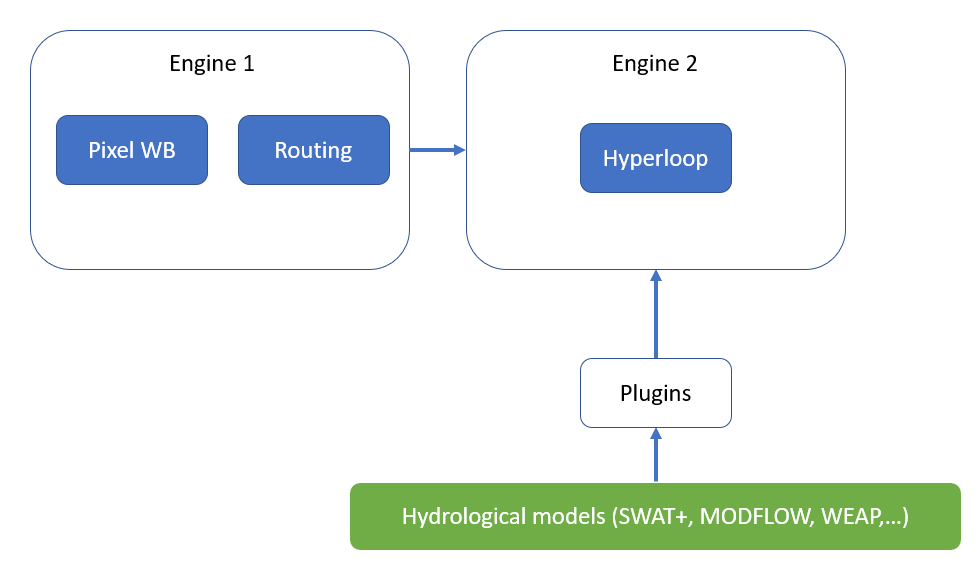


Figure 22: Diagram of IHEWAengine modules

The WA\_Hyperloop [[1]](#footnote-1)package has been integrated in IHEWAengine.engine2 module. This example below shows how to run Hyperloop in IHEWAengine.

1. Collect and place input data in IHEWAdata folder structure

The folders stores engine2 input data are (Annex 2):

* “series”, time series data
* “static”, 2D map
* “hydsim”, hydrological simulation result
* “remote”, time series maps of remote sensing data

The example input data for Hyperloop can be found in the project repository:

<https://github.com/wateraccounting/IHEWAengine/tree/master/examples/data/engine2/Hyperloop>

1. Write a Python script to run IHEWAengine

Create a python script, and save it as “example.py”. The codes in the examples folder “ex\_engine2\_Hyperloop.py” is the details steps. It can be simplified (or replaced) by the follow codes.

First import packages

# -\*- coding: utf-8 -\*-  
import os  
import inspect  
import IHEWAengine

if \_\_name\_\_ == "\_\_main\_\_":  
 print('\nEngine\n=====')  
  
 path = os.path.join(  
 os.getcwd(),  
 os.path.dirname(  
 inspect.getfile(  
 inspect.currentframe()))  
 )  
  
 engine = IHEWAengine.Engine(path, 'example.yml')

1. Write configuration file (.yml)

Create a YAML file, and name it as “example.yml” and save in the same folder with ‘example.py’. Define basin name, example “area1”. Under basin, the following configuration parameters need to be defined:

* “engine2” main block of Engine2 parameters.
* “name” defines engine type to be imported.
* “version” defines engine version.
* “data” block contains the input data.
  + “basins”, information each sub-block is the name of sub-basin.
  + “maps”, define the location of “static”, “hydsim” and “remote” data.

engines:

engine1:

*# name: ETmonitor*

*# name: MODFLOW*

*# name: PCRaster*

*# name: SurfWAT*

*# name: SWATp*

*# name: WaterPix*

*# name: Wflow*

engine2:

name: 'Hyperloop'

version: 'v0.1'

*# output: 'output'*

data:

basins:

Test:

id: 0

steps:

'Reproject data': True

'Create Sheet 4 and 6': True

'Create Sheet 2': True

'Create Sheet 3': True

'Create Sheet 5': True

'Create Sheet 1': True

'Create Sheet 7': True

parameter:

alpha\_min: null

recycling\_ratio: 0.02

dico\_in:

1: []

dico\_out:

1: [0]

discharge\_out\_from\_wp: True

*#discharge\_out\_from\_wp: False*

*#OutletPoints: '\*.shp'*

fraction\_xs:

- 4

- 25

- 4

- 25

lu\_based\_supply\_split: True

grace\_supply\_split: True

water\_year\_start\_month: 1

ndm\_max\_original: True

grace\_split\_alpha\_bounds:

- [0.0, 0.5, 0.9999]

- [0.0001, 1., 1.]

series:

path: 'D:\Github\wateraccounting\IHEWAengine\examples\data\engine2\Hyperloop\series'

file:

GRACE: 'GRACE/GSFC-average\_mmwe.csv'

crops:

1:

- 'Growing Seasons/palm\_perennial.txt'

- 'Palm Oil'

- 'Other crops'

- '-'

- 52.0

2:

- 'Growing Seasons/palm\_perennial.txt'

- 'Palm Oil'

- 'Other crops'

- '-'

- 33.0

3:

- 'Growing Seasons/rice\_irrigated\_java.txt'

- 'Rice - Irrigated'

- 'Cereals'

- '-'

- 52.0

4:

- 'Growing Seasons/rice\_rainfed\_java.txt'

- 'Palm Oil'

- 'Cereals'

- '-'

- 35.0

non\_crop: null

*# non\_crop:*

*# mea: null*

*# milk: null*

*# timber: null*

*# aquaculture: null*

maps:

static:

path: 'D:\Github\wateraccounting\IHEWAengine\examples\data/engine2/Hyperloop/static'

file:

equiped\_sw\_irrigation: 'GMIA-aeisw\_pct\_aei\_v5.asc'

wpl\_tif: 'WPL.tif'

environ\_water\_req: 'EWR.tif'

population\_tif: 'Population.tif'

cattle: 'Cattle.tif'

dem: 'DEM-HydroShed\_m\_3s.tif'

dir: null

lu: 'LU.tif'

full\_basin\_mask: 'LU.tif'

masks:

1:

- 'Full'

- 'SubBasin.tif'

- []

- []

remote:

path: 'D:\Github\wateraccounting\IHEWAengine\examples\data/engine2/Hyperloop/remote'

file:

ndm: 'NDM'

p: 'Precipitation'

et: 'Evaporation'

n: 'RainyDays'

lai: 'LAI'

etref: 'ETref'

hydsim:

path: 'D:\Github\wateraccounting\IHEWAengine\examples\data/engine2/Hyperloop/hydsim'

file:

root\_depth: 'RootDepth.tif'

bf: 'Baseflow'

sr: 'SurfaceRunoff'

tr: 'TotalRunoff'

perc: 'Percolation'

dperc: 'IncrementalPercolation'

supply\_total: 'Supply'

dro: 'IncrementalRunoff'

etb: 'ETblue'

etg: 'ETgreen'

rzsm: 'RootDepthSoilMoisture'

1. Run the IHEWAengine script

In Anaconda prompt window, activate python environment and change working directory to where **example.py** python script is saved. Then, run the command:

**python example.py**

After that, the output graphs and csv files will be saved in the folder **IHEWAengine** under the working directory (Annex 2)

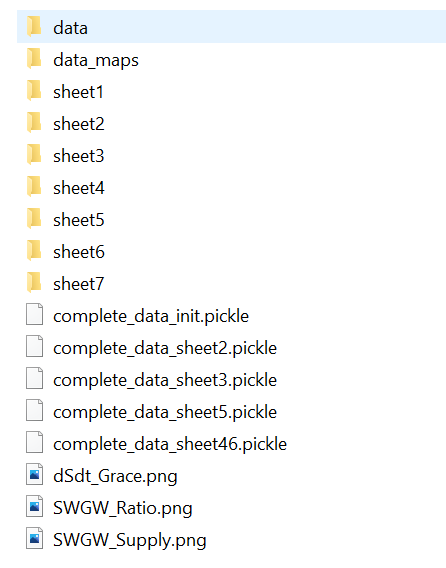
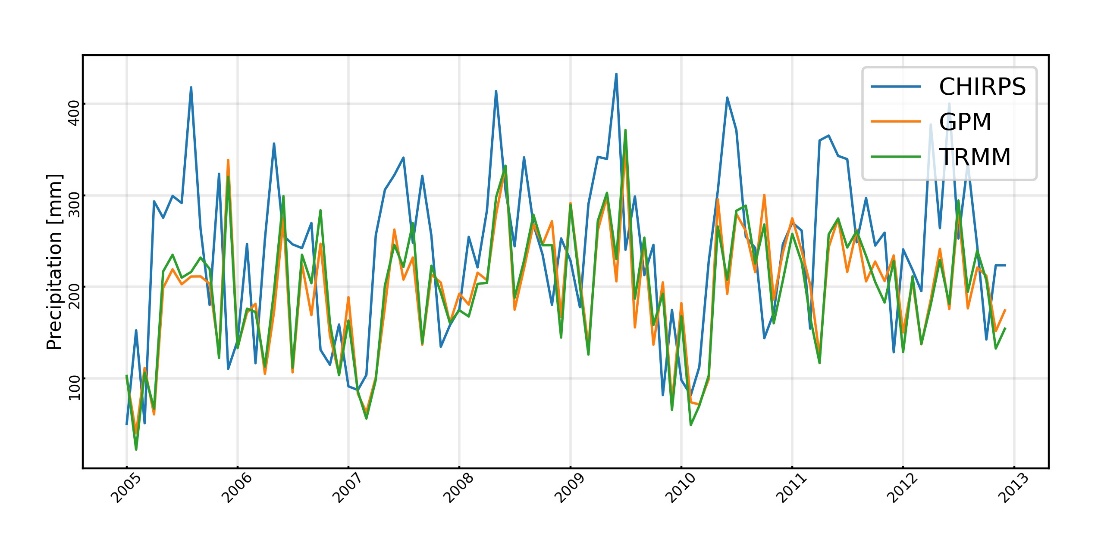


Figure 23: Output folder in the working directory IHEWAengine/ engine2.Hyperloop

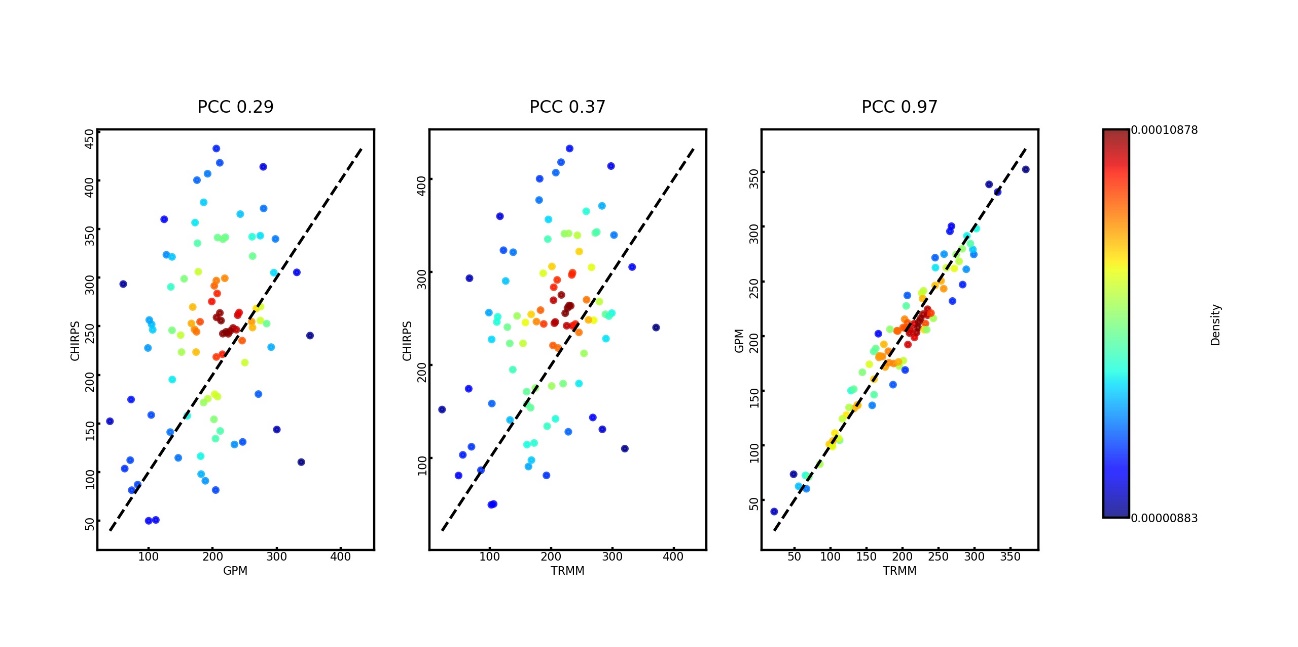
# Annexes

## Outputs of IHEWAdataanalysis

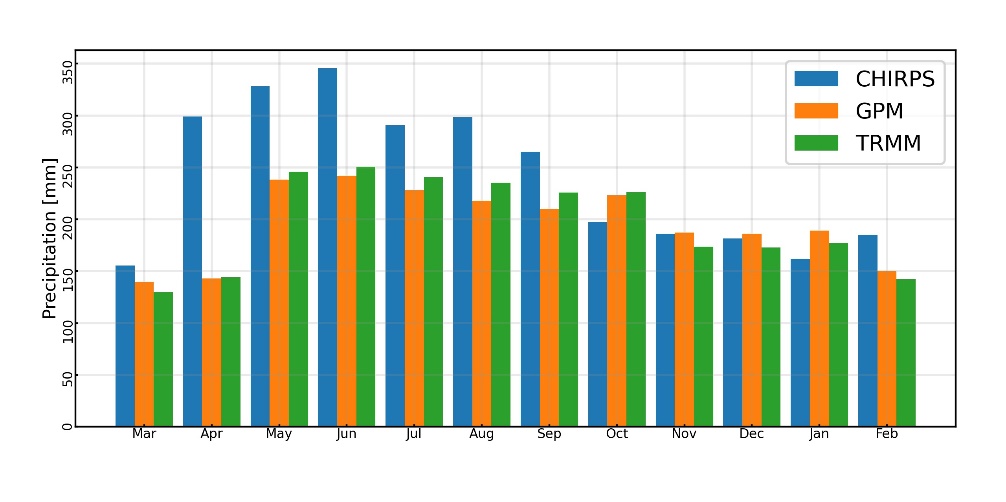
Line chart

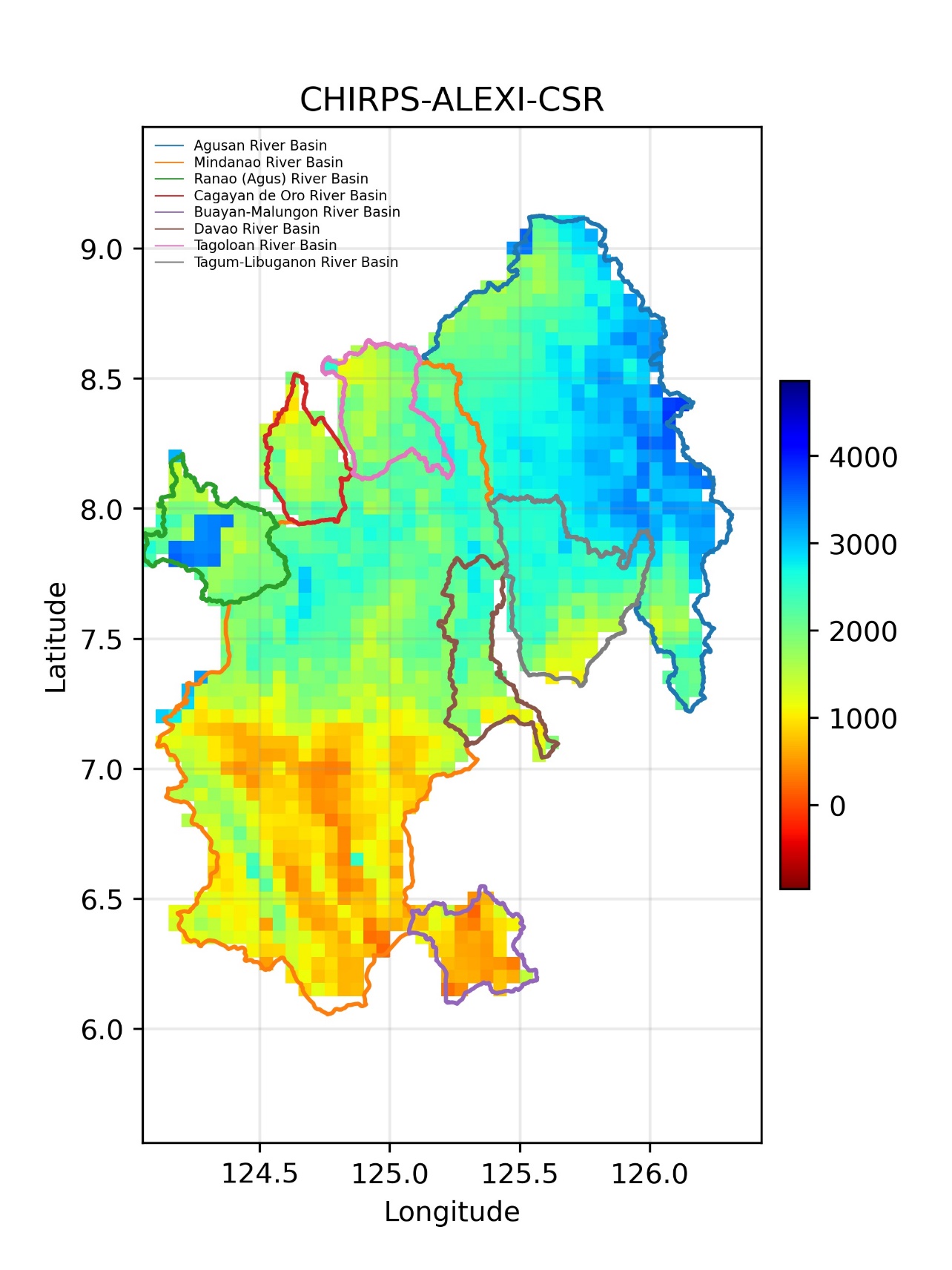


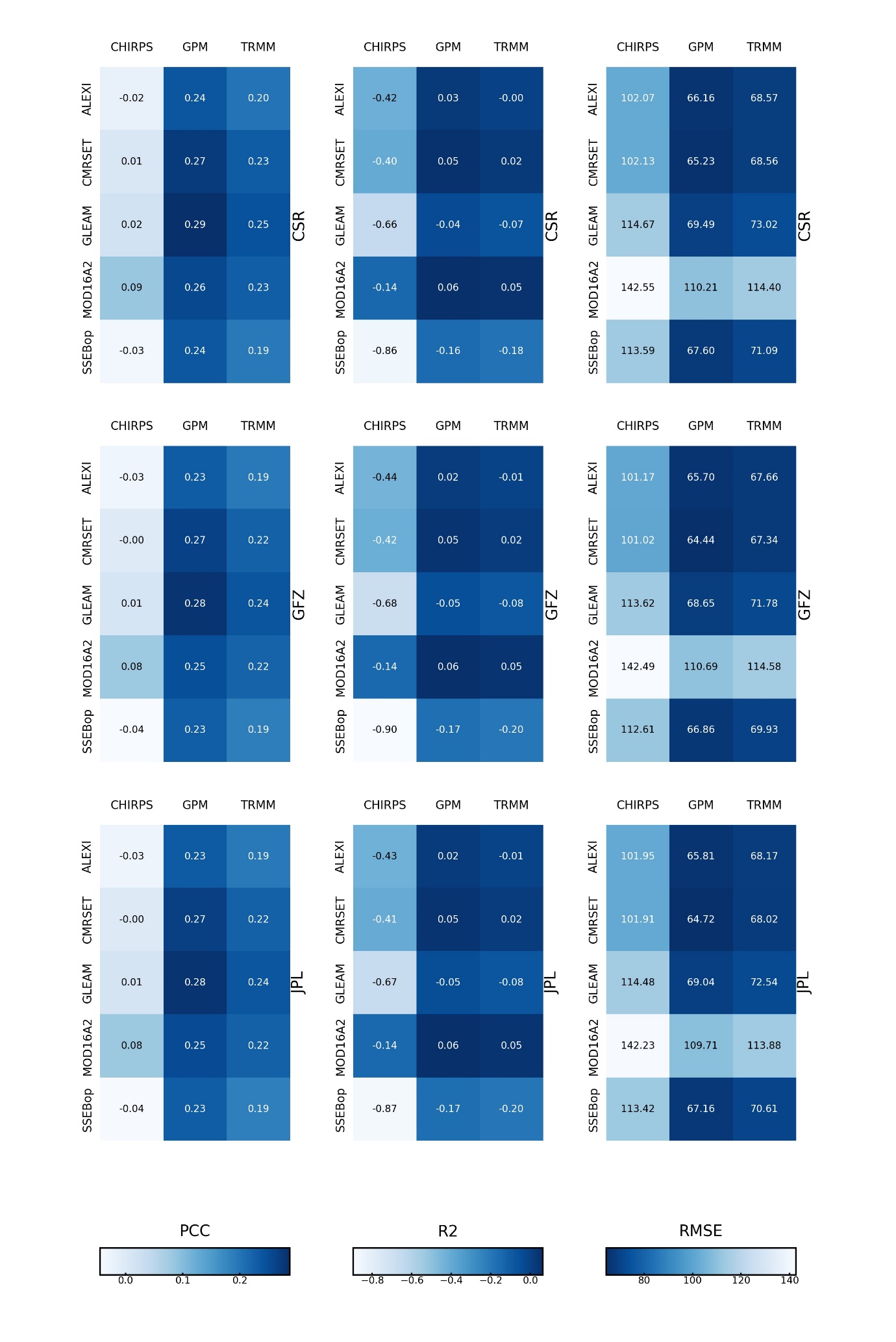
Scatter chart



Bar chart



P-ET-dS map chart

Heat map chart

## Folder tree of IHEWAengine.engine2 working directory

### Input folder

+---hydsim

| |

| +---Baseflow

| |

| +---ETblue

| |

| +---ETgreen

| |

| +---IncrementalPercolation

| |

| +---IncrementalRunoff

| |

| +---Percolation

| |

| +---RootDepthSoilMoisture

| |

| +---Supply

| |

| +---SurfaceRunoff

| |

| \---TotalRunoff

|

+---remote

| +---ETref

| |

| +---Evaporation

| |

| +---LAI

| |

| +---NDM

| |

| +---Precipitation

| |

| \---RainyDays

|

+---series

| +---GRACE

| | GSFC-average\_mmwe.csv

| |

| \---Growing Seasons

| palm\_perennial.txt

| rice\_irrigated\_java.txt

| rice\_rainfed\_java.txt

|

+---static

| Cattle.tif

| DEM-HydroShed\_m\_3s.tif

| EWR.tif

| GMIA-aeisw\_pct\_aei\_v5.asc

| LU.tif

| Population.tif

| SubBasin.tif

| WPL.tif

### Output folder

| \---Example0

| |

| +---data

| | |

| | +---demand

| | |

| | +---fractions

| | | |

| | | +---fractions

| | | |

| | | \---fractions\_dryness

| | |

| | +---i

| | |

| | +---return\_fractions

| | |

| | +---return\_gwgw

| | |

| | +---return\_gwsw

| | |

| | +---return\_swgw

| | |

| | +---return\_swsw

| | |

| | +---supply\_gw

| | |

| | +---supply\_sw

| | |

| | +---supply\_swa

| | |

| | \---t

| |

| +---data\_maps

| | +---bf

| | |

| | +---diff

| | |

| | +---dperc

| | |

| | +---dperc\_corr

| | |

| | +---dro

| | |

| | +---dro\_corr

| | |

| | +---et

| | |

| | +---etb

| | |

| | +---etg

| | |

| | +---etref

| | |

| | +---lai

| | |

| | +---n

| | |

| | +---ndm

| | |

| | +---p

| | |

| | +---perc

| | |

| | +---r

| | |

| | +---rzsm

| | |

| | +---sr

| | |

| | +---sr\_corr

| | |

| | +---supply\_total

| | |

| | +---supply\_total\_corr

| | |

| | +---tr

| | |

| | \---tr\_corr

| |

| +---sheet1

| | |

| | +---sheet1\_monthly

| | |

| | \---sheet1\_yearly

| |

| +---sheet2

| | +---sheet2\_monthly

| | |

| | \---sheet2\_yearly

| |

| +---sheet3

| | +---WP\_Y\_Seasonly\_csvs

| | |

| | +---WP\_Y\_Seasonly\_graphs

| | |

| | +---WP\_Y\_Yearly\_csvs

| | |

| | \---WP\_Y\_Yearly\_graphs

| |

| +---sheet4

| | |

| | +---sheet4\_monthly

| | |

| | \---sheet4\_yearly

| |

| +---sheet5

| | +---sheet5\_monthly

| | |

| | \---sheet5\_yearly

| |

| +---sheet6

| | +---sheet6\_monthly

| | |

| | \---sheet6\_yearly

| |

| \---sheet7

| +---Feed

| |

| +---Fuel

| |

| +---RZstor

| |

| +---sheet7\_monthly

| |

| +---sheet7\_yearly

| |

| +---split\_y

| |

| +---temp\_cattle

| |

| \---temp\_et\_recycle

1. <https://github.com/wateraccounting/WA_Hyperloop> [↑](#footnote-ref-1)