awsm Documentation

Micah Sandusky

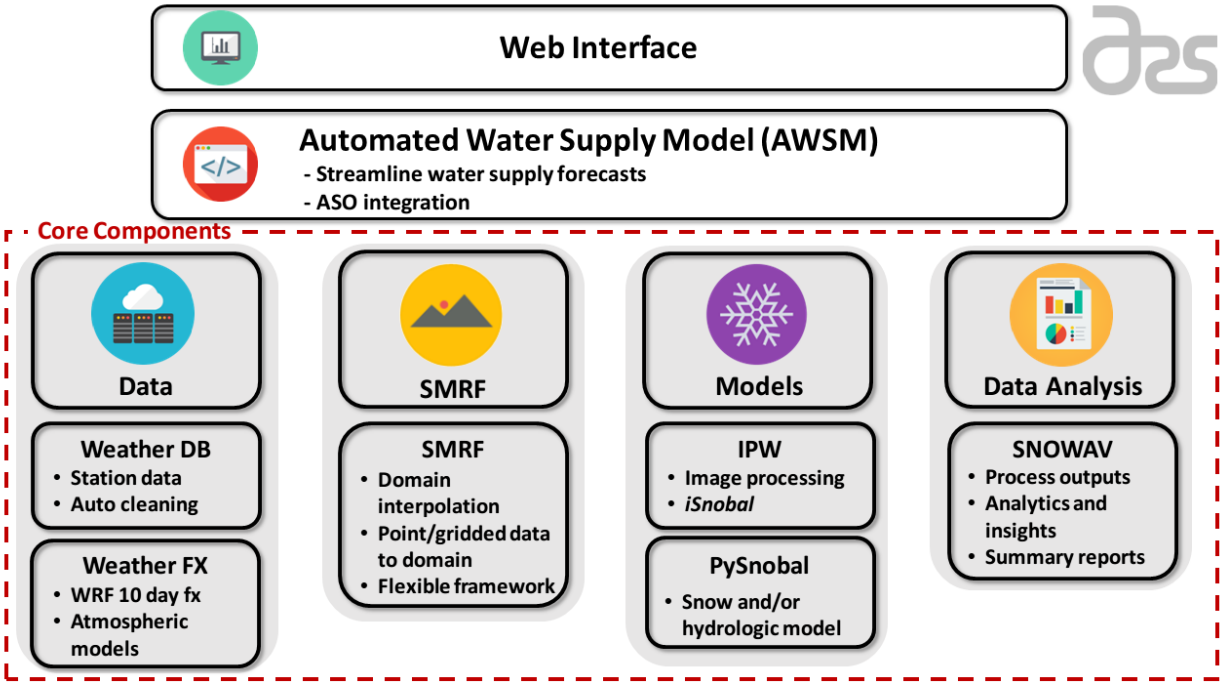
May 30, 2018

Contents

CHAPTER 1

Automated Water Supply Model

Automated Water Supply Model (AWSM) was developed at the USDA Agricultural Research Service (ARS) in Boise, ID. AWSM was designed to streamline the workflow used by the ARS to forecast the water supply of multiple water basins. AWSM standardizes the steps needed to distribute met. data with SMRF, run an energy and mass balance with iSnobal, and process the results, while maintaining the flexibility of each program.



1.1 Quick Start

The fastest way to get up and running with AWSM is to use the docker images that are prebuilt and can deployed cross platform.

To build AWSM natively from source checkout the install instructions here.

1.1.1 Docker

To mount a data volume, so that you can share data between the local filesystem and the docker, the *-v* option must be used. For a more in depth dicussion and tutorial, read about docker volumes. The container has a shared data volume at */data* where the container can access the local filesystem.

When the image is run, it will go into the Python terminal within the image. Within this terminal, AWSM can be imported. The command */bin/bash* can be appended to the end of docker run to enter into the docker terminal for full control. It will start in the */data* location with AWSM code in */code/awsm*.

For Linux:

```
docker run -v <path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

For MacOSX:

```
docker run -v /Users/<path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

For Windows:

```
docker run -v /c/Users/<path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

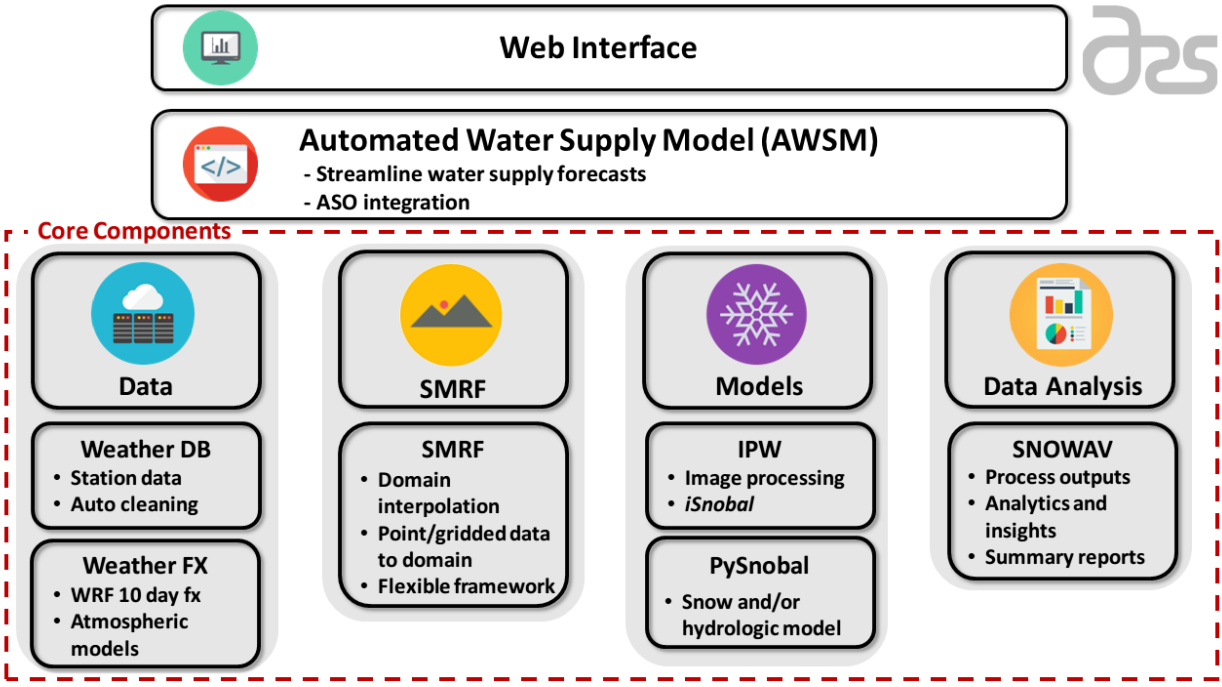
1.2 Running the test

```
docker run -it usdaarsnwrc/awsm /bin/bash
cd /code/smrf
gen_maxus --out_maxus test_data/topo/maxus.nc test_data/topo/dem.ipw
cd /code/awsm
awsm test_data/RME_run/config_pysnobal.ini
```

The output netCDF files will be placed in the */code/awsm/test_data/RME_run/output/rme/devel/wy1998/rme_test/runs/run1464_1670/ou*
location.

1.2.1 Automated Water Supply Model

Automated Water Supply Model (AWSM) was developed at the USDA Agricultural Research Service (ARS) in Boise, ID. AWSM was designed to streamline the workflow used by the ARS to forecast the water supply of multiple water basins. AWSM standardizes the steps needed to distribute met. data with SMRF, run an energy and mass balance with iSnobal, and process the results, while maintaining the flexibility of each program.



Quick Start

The fastest way to get up and running with AWSM is to use the docker images that are prebuilt and can deployed cross platform.

To build AWSM natively from source checkout the install instructions here.

Docker

To mount a data volume, so that you can share data between the local filesystem and the docker, the *-v* option must be used. For a more in depth dicussion and tutorial, read about docker volumes. The container has a shared data volume at */data* where the container can access the local filesystem.

When the image is run, it will go into the Python terminal within the image. Within this terminal, AWSM can be imported. The command */bin/bash* can be appended to the end of docker run to enter into the docker terminal for full control. It will start in the */data* location with AWSM code in */code/awsm*.

For Linux:

```
docker run -v <path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

For MacOSX:

```
docker run -v /Users/<path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

For Windows:

```
docker run -v /c/Users/<path>:/data -it usdaarsnwrc/awsm [/bin/bash]
```

Running the test

```
docker run -it usdaarsnwrc/awsm /bin/bash
cd /code/smrf
gen_maxus --out_maxus test_data/topo/maxus.nc test_data/topo/dem.ipw
cd /code/awsm
awsm test_data/RME_run/config_pysnobal.ini
```

The output netCDF files will be placed in the */code/awsm/test_data/RME_run/output/rme/devel/wy1998/rme_test/runs/run1464_1670/ou* location.

1.2.2 Installation

Installing Dependencies

AWSM utilizes many of the utilities within SMRF. The first step is to read and follow the install instructions for SMRF, found here. Make sure to follow all instructions, including installing IPW.

The source code for SMRF is stored on on GitHub.

If you would like to use the PySnobal within AWSM, you can download and install the package following the guidelines on the PySnobal repo . **This is optional**.

Installing AWSM

Once the dependencies have been installed for your respective system, the following will install AWSM. It is preferable to use a Python virtual environment to reduce the possibility of a dependency issue. You should use the same virtual environment in which you installed SMRF. You can just source your smrfenv instead of step number 1.

1. Create a virtualenv and activate it.

   ```
   virtualenv awsmenv
   source awsmenv/bin/activate
   ```

Tip: The developers recommend using an alias to quickly turn on and off your virtual environment.

2. Clone AWSM source code from the ARS-NWRC github.

   ```
   git clone https://github.com/USDA-ARS-NWRC/AWSM.git
   ```

3. Change directories into the AWSM directory. Install the python requirements. After the requirements are done, install AWSM.

   ```
   cd AWSM
   pip install -r requirements_dev.txt
   python setup.py install
   ```

4. (Optional) Generate a local copy of the documentation.

   ```
   cd docs
   make html
   ```

   To view the documentation use the preferred browser to open up the files. This can be done from the browser by opening the index.rst file directly or by the commandline like the following:

```
google-chrome _build/html/index.html
```

Testing AWSM

Once everything is installed, you can run a quick test case over a small catchment in Idaho called Reynolds Mountain East (RME).

1. Move to config file and run case. Start in your AWSM directory

```
cd test_data/RME_run/
awsm config.ini
```

2. Wait for the test run to finish and then view the results.

```
cd output/rme/devel/wy1998/rme_test/
```

The iSnobal model outputs will be in the "runs" folder and the distributed SMRF data will be in the "data" folder. Navigate around and see what the outputs look like. You can visualize the .nc (netCDF) files with the ncview utility.

1.2.3 Usage

To run AWSM, a configuration is require and it's simply passed as the first argument to the awsm command. If the configuration file was named config.ini it could be used like the following.

```
awsm config.ini
```

For configuring AWSM simulations refer to *Using Configuration Files*. If you are interested in using AWSM in a project, getting started looks like this:

```
import awsm

with awsm.framework.framework.AWSM(configFile) as a:
  # Specify functions to run
```

Review the script for running AWSM in "./scripts/awsm" to get a better sense of the methods used to run AWSM and use the *API Documentation*

1.2.4 Using Configuration Files

AWSM simulation details are managed using configuration files. The python package inicheck is used to manage and interpret the configuration files. Each configuration file is broken down into sections containing items and each item is assigned a value.

A brief description of the syntax is:

- Sections are noted by being in a line by themselves and are bracketed.
- Items are denoted by colon (**:**).
- Values are simply written in, and values that are lists are comma separated.
- Comments are preceeded by a **#**

For more information regarding inicheck syntax and utilities refer to the inicheck documentation.

Understanding Configuration Files

The easiest way to get started is to look at one of the config files in the repo already. A simple case to use is our reynolds mountain east test which can be view easily here.

Take a look at the "topo" section from the config file show below

```
##############################################################################
# Files for DEM and vegetation
##############################################################################

[topo]
basin_lon:                      -116.7547
basin_lat:                      43.067
filename:                       ./topo/topo.nc
type:                           netcdf
```

This section describes all the topographic information required for AWSM to run. At the top of the section there is comment that describes the section. The section name "topo" is bracketed to show it is a section and the items underneath are assigned values by using the colon.

Editing/Checking Configuration Files

Use any text editor to make changes to a config file. We like to use atom with the .ini syntax package installed.

If you are unsure of what to use various entries in your config file refer to the config-file-reference or use the inicheck command for command line help. Below is an example of how to use the inicheck details option to figure out what options are available for the topo section type item.

```
inicheck --details topo type -m smrf awsm
```

The output is:

```
Providing details for section topo and item type...

Section      Item    Default    Options                   Description
================================================================================
topo         type    netcdf     ['netcdf', 'ipw']         Specifies the input file␣
↪type
```

Creating Configuration Files

Not all items and options need to be assigned, if an item is left blank it will be assigned a default. If it is a required filename or something it will be assigned a none value and AWSM will throw an error until it is assigned.

To make an up to date config file use the following command to generate a fully populated list of options.

```
inicheck -f config.ini -m smrf awsm -w
```

This will create a config file using the same name but call "config_full.ini" at the end.

Core Configuration File

Each configuration file is checked against the core configuration file stored ./awsm/framework/core_config.ini and various scenarios are guided by the a recipes file that is stored in ./awsm/framework/recipes.ini. These files work together to guide the outcomes of the configuration file.

To learn more about syntax and how to contribute to a Core or Master configuration file see Master Configuration Files in inicheck.

1.2.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/USDA-ARS-NWRC/AWSM/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

awsm could always use more documentation, whether as part of the official awsm docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/USDA-ARS-NWRC/AWSM/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *awsm* for local development.

1. Fork the *awsm* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/awsm.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv awsm
$ cd awsm/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 awsm tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/micahsandusky5/awsm/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest discover -v
```

If any code edits you add require new configuration file items, then they must be added to the core configuration file to be registered with AWSM. Please see *Core Configuration File* to learn more.

1.2.6 AWSM Configuration File Reference

The AWSM config file is a combination of SMRF and AWSM's own configuration files. Pleas refer to the Reference for SMRF Configuration Files for any questions regarding the configuration file sections from SMRF which are as follows:

- topo
- time
- stations
- csv
- gridded
- mysql
- air_temperature
- vapor pressure
- wind
- precip
- albedo
- thermal
- soil_temp
- output
- logging
- system

All other sections and details for AWSM configuration files can be seen in the below.

For configuration file syntax information please visit http://inicheck.readthedocs.io/en/latest/

awsm master

make_in
: Convert SMRF outputs to iSnobal inputs
  Default: True
  Type: bool

make_nc

Convert iSnobal outputs to netCDF
Default: True
Type: bool

mask_isnobal
Mask iSnobal outputs
Default: False
Type: bool

prompt_dirs
ask yes or no when making new directories
Default: False
Type: bool

run_ipysnobal
Run iPySnobal like iSnobal
Default: False
Type: bool

run_isnobal
Run iSnobal for specified simulation
Default: True
Type: bool

run_smrf
Specifies whether or not to run SMRF
Default: True
Type: bool

run_smrf_ipysnobal
Run SMRF and iPySnobal together
Default: False
Type: bool

paths

basin
name of basin to run
Default: None
Type: string

desc
description for set of runs
Default: None
Type: string

folder_date_style
style of date that gets appended to generated folders
Default: wyhr
Type: string
Options: wyhr day start_end

isops
if running operational or development
Default: False
Type: bool

path_dr
path to starting drive MUST exist
Default: None
Type: directory

proj
name of project if running devel
Default: None
Type: string

grid

active_layer
height of iSnobal active layer
Default: 0.25

Type: float

csys
coordinate type
Default: UTM
Type: string
Options: UTM

nbits
number of bits for IPW images
Default: 16
Type: int

thresh_medium
medium mass threshold for timestep refinement
Default: 10
Type: int

thresh_normal
normal mass threshold for timestep refinement
Default: 60
Type: int

thresh_small
small mass threshold for timestep refinement
Default: 1
Type: int

files

prev_mod_file
last snow output file from iSnobal for restart
Default: None
Type: filename

roughness_init

standard init file used only for roughness band
Default: None
Type: filename

awsm system

daily_folders
seperate daily output folders mainly for HRRR
Default: False
Type: bool

em_name
name of energetics ouput file without extension
Default: em
Type: string

ithreads
numbers threads for running iSnobal
Default: 1
Type: int

log_level
level of information to be logged
Default: debug
Type: string
Options: debug info error

log_to_file
log to file or print to screen
Default: True
Type: bool

output_frequency
frequency of iSnobal outputs in hours
Default: 24
Type: int

run_for_nsteps
number of timesteps to run iSnobal This is optional
Default: None
Type: int

snow_name
name of snow ouput file without extension
Default: snow
Type: string

variables
Variables for PySnobal to output after being calculated
Default: thickness snow_density specific_mass liquid_water temp_surf temp_lower temp_snowcover thickness_lower water_saturation net_rad sensible_heat latent_heat snow_soil precip_advected sum_eb evaporation snowmelt SWI cold_content
Type: string
Options: thickness snow_density specific_mass liquid_water temp_surf temp_lower temp_snowcover thickness_lower water_saturation net_rad sensible_heat latent_heat snow_soil precip_advected sum_eb evaporation snowmelt SWI cold_content

isnobal restart

depth_thresh
threshold in meters for low snow depths for restart
Default: 0.05
Type: float

restart_crash
whether or not to restart from crashed run
Default: False
Type: bool

wyh_restart_output
last iSnobal output hour to restart from
Default: None
Type: int

ipysnobal

forcing_data_type
file type from which to get input data
Default: ipw
Type: string
Options: ipw netcdf

ipysnobal initial conditions

init_file
full path to init file or last output file
Default: None
Type: filename

input_type
type of file for initializing ipysnobal run
Default: ipw
Type: string
Options: netcdf ipw ipw_out netcdf_out

ipysnobal constants

z_g
depth of soil temperature in meters
Default: 0.5
Type: float

z_t
height of temperature in meters
Default: 5.0
Type: float

z_u
height of wind speed in meters
Default: 5.0
Type: float

1.2.7 API Documentation

The API here describes all the classes and functions used in AWSM.

1.2.8 Credits

Development Lead

- Micah Sandusky <micah.sandusky@ars.usda.gov>

Contributors

- Micah Johnson <micah.johnson150@gmail.com>

1.2.9 History

0.1.0 (2017-08-18)

- Create package

0.2.0 (2018-01-04)

- Incorporation scripts used to run SMRF and iSnobal
- Creation of rigid directory structure
- Creation of entire framework
- Incorporation of PySnobal package
- Automated run procedure

0.3.0 (2018-01-10)

- General cleanup
- Documentation

0.4.0 (2018-05-03)

- Put into docker package, continuous integration
- Conforming to Pep8 standards
- Improved restart procedure for iSnobal
- Improved gridded forecast ability
- Improved user configuration
- Fast user test cases and unit test capability
- Repeatable runs from station data with git version tracking

CHAPTER 2

Indices and tables

- genindex
- modindex
- search