Australian Government
Bureau of Meteorology

AWRA Landscape Community Modelling System User Guide

A guide to setup and implement the AWRA Landscape Community Modelling System

Alison Oke, Mohsin Hafeez, Stuart Baron-Hay, Avijeet Ramchurn

Contact details

Dr. Andrew Frost

Manager (acting) - Water Resources Modelling Unit

Bureau of Meteorology
PO Box 413, Darlinghurst NSW 1300

Tel: +61 2 9296 1517
Email: Andrew.Frost@bom.gov.au

Dr. Amgad Elmahdi
Section Head, Water Resources Assessment (WRA)

Bureau of Meteorology
GPO Box 1289 Melbourne VIC 3001

Tel: +61 3 8638 8274
Email: Amgad.Elmahdi@bom.gov.au

AWRA Community Modelling System mailbox

Bureau of Meteorology
Email: awracms@bom.gov.au

CONTENTS

1. Introduction

This Chapter provides background to and introduces the Australian Water Resource Assessment Community Modelling Systems (AWRA CMS) and User Guide.

1.1. Background

Water availability across the Australian continent is limited by varying climate zones and high year-to-year variability. The Australian Government through the Commonwealth Water Act 2007, has given the Bureau of Meteorology (the Bureau), responsibility for compiling and delivering comprehensive water information for Australia. Specifically, to collate information and assess and report on the availability, condition and use of water resources in Australia. To fulfil its legislative responsibilities, the Bureau requires a water balance modelling system that quantifies water flux and storage terms and their respective uncertainties using a combination of data sets (on-ground metering, remotely sensed data) and modelled outputs. The system needs to be applicable across the continent and flexible enough to be able to use all available data sources (when modelling data rich and data limited regions) to provide nationally consistent and robust estimates (Hafeez et al., 2015).

The Australian Water Resources Assessment modelling system (AWRAMS) has been developed through the Water Information R&D Alliance (WIRADA) between the Bureau of Meteorology (the Bureau) and CSIRO since 2008 – towards fulfilling the Bureau's water reporting requirements as specified in the Water Act (2007). AWRAMS has been developed and tested to provide continental-to-regional scale water balance estimates for water resources assessment and water accounting purposes. The science underpinning AWRAMS has now reached maturity and translated into an operational system within the Bureau. AWRAMS ongoing development is being supported into the future by the Bureau and CSIRO. To further enable development and other uses of the system, the AWRAMS is being released as a community model.

The AWRAMS has two modelling components;

- AWRA-L to estimate landscape water balance fluxes, and
- AWRA-R to estimate river water balance fluxes

see Hafeez et al., 2015 for further details. This document describes the AWRA community modelling system containing AWRA-L only.

AWRA-L is a one dimensional, 0.05° grid based water balance model over the continent that has semi-distributed representation of the soil, groundwater and surface water stores. AWRA-L is a three soil layer (top: 0-10cm, shallow: 10cm-100cm, deep: 100cm-600cm), two hydrological response unit (shallow rooted versus deep rooted) model: see conceptual diagram in Figure 1 (with model description provided in Frost, Ramchurn

and Smith, 2016).Within each grid cell the water balance is calculated independently for deep and shallow rooted vegetation with outputs from the model representing the grid cell average or total (depending on variable). The AWRA-L model uses the operational Australian Water Availability Project (AWAP) climate data and uses the same 0.05° AWAP grid (approximately 5 km resolution) and runs with a daily time step from 1911 to now (a solar radiation climatology is used prior to 1990). The model outputs are spatio-temporally aggregated to provide daily, monthly and annual gridded estimates of landscape runoff, evapotranspiration (ET), soil moisture, and deep drainage at the regional and continental scale seamlessly from the past to the present (100 + years).

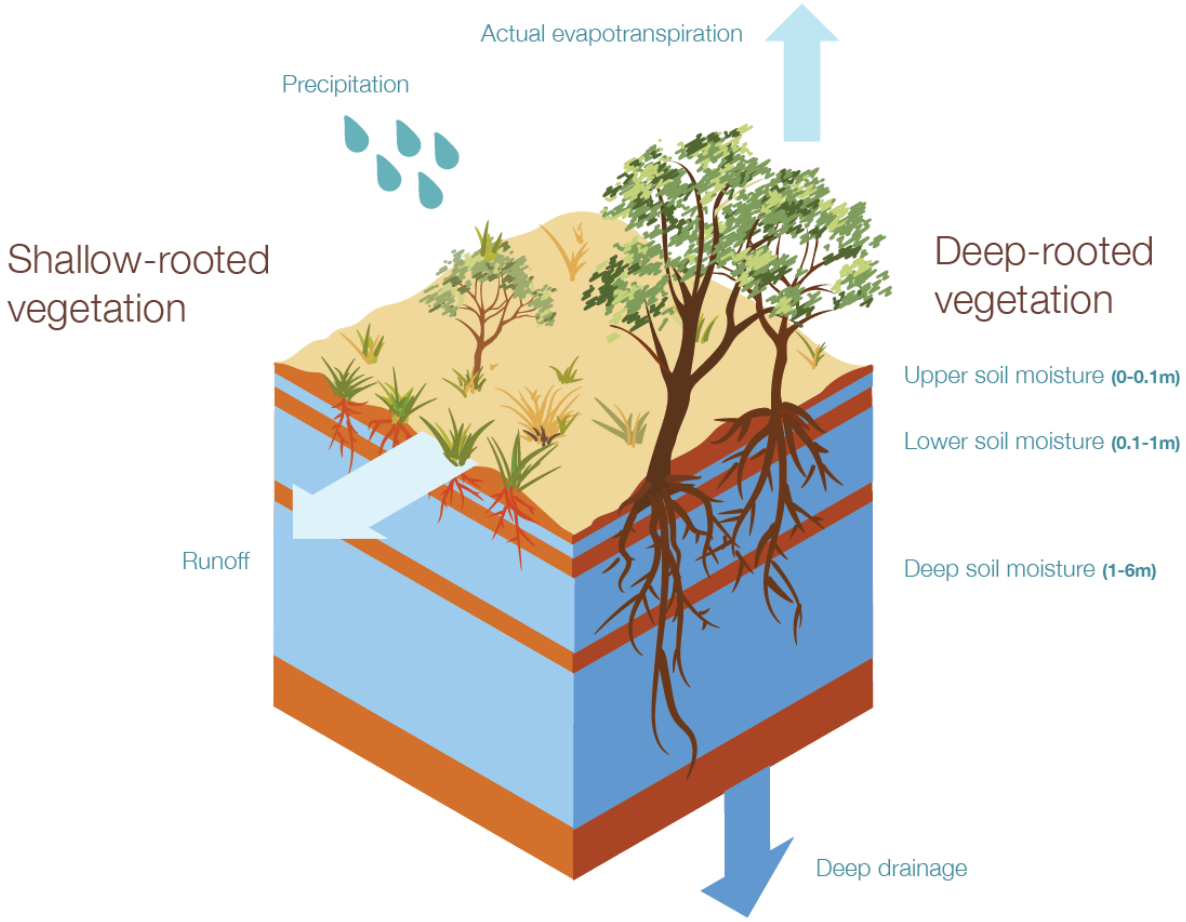


Figure 1. Conceptual AWRA-L grid cell with key water stores and fluxes shown

Frost, Ramchurn and Hafeez (2016) evaluated AWRA-L v5 outputs against a range of data sources including streamflow, probe based estimates of soil moisture, flux tower estimates of evapotranspiration and a national groundwater recharge data set, satellite observations of soil moisture and evapotranspiration. AWRA-L v5 performed relatively well for streamflow and rootzone soil moisture compared to peer models. Benchmark statistics are provided in that report over which future version of AWRA-L and other models can be compared. A key component of the community modelling system is the benchmarking datasets allowing easy comparison to currently AWRA-L performance.

Since July 2015, the AWRA-L model has been operational in the Bureau and is now independent of the research environment in CSIRO. The Bureau's operational AWRA-L model produce daily landscape water balances across Australia. It is a Python (see https://www.python.org/ and Appendix C) programming language based modelling system, with the core model algorithms implemented in high performing native languages (C) and generic functionality provided by robust, open source libraries. The operational AWRA-L model is running on a daily schedule with modelled outputs generated each night to extend the century record to current day, and available at 9am (Smith et al., 2015). Interactive outputs from the AWRA-L model are now available to the public through the Australian Landscape Water Balance website (www.bom.gov.au/water/landscape). Decision-makers in industry, government and the community can now visualise, investigate and download up-to-date water balance information from the operational AWRA-L model by day, month or year, from 1911 onwards and at every scale between point location, catchment and the entire continent, supporting water resources assessment and planning.

AWRA-L model was primarily developed for water resources applications across Australia. Since the availability of the operational AWRA-L modelled outputs from July 2015, the modelled fluxes have been used for various climatological, water and agriculture applications across Australia. The AWRA-L model has been used to estimate the landscape water balance fluxes for the National Water Account and Water Resources Assessment reports since 2010. The model is used as an input to other Bureau products, including our National Climate and Water Briefings, National Water Account, Special Climate Statements, Climate Outlooks and (internal) monthly weather, climate, flood and space weather updates.

Since 2015, the Bureau's AWRA team has been interacting with a wide range of stakeholders about the needs of a daily operational water balance model. These interactions have spanned Commonwealth agencies (Murray-Darling Basin Authority (MDBA), Department of Agriculture and Water Resources (DAWR), and Australian Bureau of Agricultural and Resource Economics and Sciences (ABARES)) and State government water and agriculture agencies (NSW DPI Agriculture and Water, SeqWater, Vic DEDJTR, QLD's DSITI, WaterNSW, Vic DELWP, SA DENWR and WA's DAFWA), catchment management authorities, water utilities (Melbourne Water, SA Water Sydney Water), consultants (HARC, GHD and DHI), Emergency Services (Qld and Tas) water industry professionals (Engineers Australia), research organisations (CRC Sheep and CSIRO), universities (UNSW, Melbourne, Monash, USQ, ANU, Adelaide, Queensland, Murdoch, Edith Cowan, Newcastle and University of Kanstanz, Denmark), conservation organisation (Reef catchments and Royal Botanic Gardens) and farmers. Through these consultations, it is clear there is a high level of interest in accessing the AWRA MS from the research, government and consulting sectors.

Based on the active interactions with wide ranging of stakeholders, the Bureau broadly categorises four specialised areas for potential applications of AWRA-L based modelled fluxes.

Water Resources Assessment and Planning Applications: The Bureau's internal use of modelled outputs for national water account and water resources assessment reports was the initial catalyst for the AWRA modelling system, and this requirement is echoed by many stakeholders operating at different local and regional scales, including MDBA, Melbourne Water, DAWR, Vic DELWP, SA's DENWR and many

postgraduate students. For example, MDBA's Environmental Water Planning section used rainfall, soil moisture and runoff from AWRA-L modelled data as indicators to describe the hydrological antecedent conditions for 2015 and 2016 in defined river catchments (whether the catchment has received relatively much or little water) for their forward planning of delivering environmental water.

Agriculture and Natural Resources Management Applications: The Bureau team is working with state and commonwealth agriculture agencies (NSW DPI Agriculture, Vic DEDJTR, WA's DAFWA, and ABARES) on how to best use the modelled outputs (soil moisture, actual ET and potential ET) for various applications including agriculture crop production and drought monitoring over dryland agriculture areas. For example, the Victorian Department of Primary Industries (DEDJTR) uses the soil moisture model to summarise climatic conditions and help prepare their seasonal climate outlooks. Preliminary analysis shows that the AWRA-L soil moisture model output matches the patterns observed through the Department's soil moisture monitoring network better than previous models. Another success story is AWRA-L based soil moisture model outputs being adopted by the ABARES for regular reporting purposes ranging from the Weekly Australian Climate, Water and Agriculture update as well as for quarterly Australian Crop Report, and Agricultural Commodities.

Flood Applications: Many stakeholders' including the Bureau's National Flood Forecasting team, Seqwater, MDBA and a consulting firm (HARC) are currently using AWRA-L based modelled outputs (soil moisture and runoff) for potential applications in flash flood risk maps, flood forecasting and estimation of initial losses prior to flood events around Australia. For example, AWRA based modelled soil moisture data typically had half as much error in estimating initial losses than the existing operational methods. Results from case study showed that the use of AWRA-L soil moisture provides better understanding of catchment initial loss prior to flood events in NSW, Queensland and Victoria. AWRA-L based soil moisture estimated initial losses will be used in a revision of the loss chapter for Australian Rainfall and Runoff (AR&R). The revised AR&R will be recommending the use of Bureau AWRA-L results for design flood studies.

Groundwater Community: The Australian Groundwater modelling community (MDBA, Bureau's groundwater Unit and several universities) has shown a strong interest in using AWRA-L based modelled deep drainage output for various groundwater applications across Australia. For example, the Bureau is working on a case study for MDBA on evaluating AWRA-L modelled deep drainage fluxes through comparison with field data and peer models over groundwater Sustainable Diversion Limit (SDL) areas in the Murray Darling Basin.

Based on the active engagement with the above mentioned stakeholders, it is apparent that the users have different requirements for applications of AWRA-L modelled outputs ranging from water resources assessment and planning to various other sectors (including agriculture, natural resources management, flood and groundwater). The release of the AWRA MS as a community model provides the best means to engage the most qualified experts in each discipline in the process of refining and applying AWRA-L to these modelling applications.

1.2. The AWRA Community Modelling System (AWRA CMS)

The AWRA CMS is initially limited to Landscape (AWRA-L) model and related functionality. The package includes the following components

- Simulation: generating model outputs over a given spatial extent and time period according to specified input data
- Calibration: automated alteration of model parameters to closely match observed data eg. streamflow,
- Visualisation: eg. viewing of gridded outputs and timeseries, and
- Benchmarking: testing the model against observed data functionality.
- Utilities: common functionality required by the following four components

See the conceptual diagram of the AWRA CMS module components in Figure 2 below. These components are built using open-source software, removing a key a barrier to adoption.

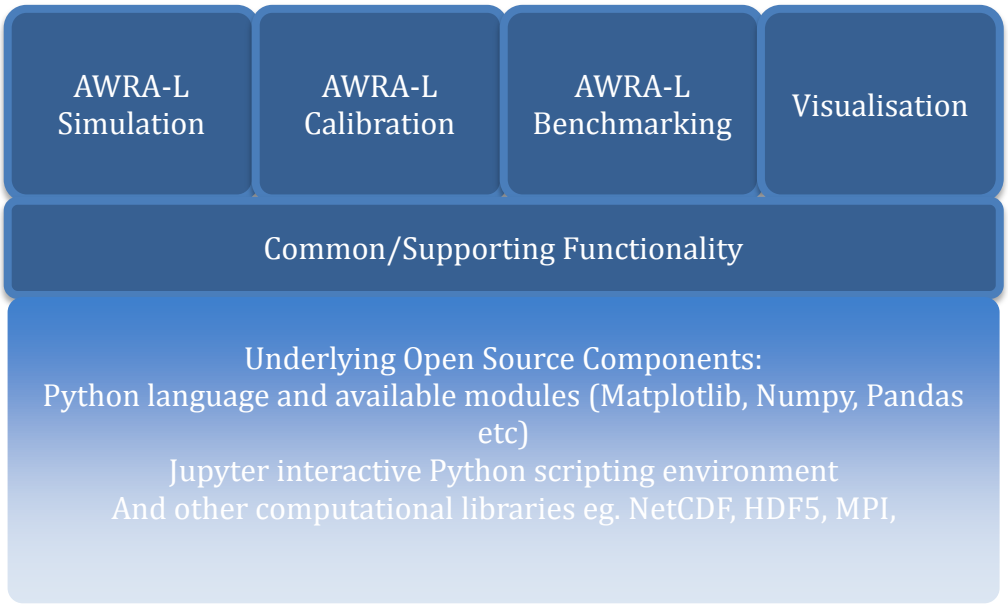


Figure 2 Components of the Community Modelling System

NetCDF – network common data format;
HDF5 – file format and software library;
MPI – message passing interface.

Simulation: The simulation component contains the model code for the current version of the operational AWRA-L (v5.0) model and the required functionality to run the model continentally, over a region, or a point.

The inputs and outputs to the model are user configurable (although current static spatial grids, input climate data and benchmarking data are also available), and advanced users can modify the model code.

Calibration: The calibration component contains the code to calibrate the AWRA-L model with the Shuffled Complex Evolution (SCE-UA) algorithm (Duan et al.,1992). Calibration can be undertaken on a local computer or alternatively on a supercomputer (towards reducing computational time when applying over large spatial and temporal scales). The user can specify the calibration objective functions and calibrate to multiple observations (and observation types) simultaneously. In particular, the simultaneous calibration to streamflow, soil moisture and evapotranspiration (ET) from around 300 unimpaired catchments across Australia is supported as was used in deriving the AWRA-L v5 parameters.

Benchmarking: The benchmarking package facilitates the comparison of AWRA-L model outputs against observations of landscape water stores and fluxes. The package generates key model statistics (for example Nash-Sutcliffe Efficiency, relative bias and correlation) and plots (time-series, scatterplots, box and whiskers, and CDF) of the outputs (catchment aggregated gridded runoff, soil moisture, ET and deep drainage) which can be viewed graphically or in tables. The benchmarking package ships with streamflow from unimpaired catchments across Australia (Zhang et al., 2013), remotely sensed catchment averaged soil moisture (AMSR-E; Owe et al., 2008) and catchment averaged ET derived from MODIS satellite observations (CMRSET; Guerschman et al., 2009), sample observations of in-situ soil moisture from two Australian networks in the Upper Hunter (SASMAS; Rüdiger et al., 2007) and Murrumbidgee (OzNet ; Smith et al., 2012), gap filled site ET from an Australian flux tower network (OzFlux; Beringer, Hutley et al., 2016; infilled according to the processing outlined in Beringer, McHugh et al., 2016). To gain access to the full validation (benchmarking) datasets the user must become an AWRA CMS Registered User (Section 1.5). Users must also abide by individual dataset licences contained with the data.

Visualisation: The visualisation package facilitates the visual inspection of AWRA-L model inputs and outputs both spatially and temporally, and enables comparison of different sets of results from the model. The package allows the plotting of maps or time-series for any specified region such as Australia, a state, a catchment, a bounding box, or a pixel. The package gives the user spatial and temporal aggregation abilities, including the aggregation of gridded model inputs and outputs over ad-hoc polygon areas such as catchments.

Utilities: The support package contains supporting functionality that is required in more than one of the above packages. This package contains support code for file-handling, geospatial operations and data manipulation.

1.3. Potential Users of AWRA CMS

Potential users are anticipated to predominately be interested in the ability to run the system with local data (including scenario modelling) and to modify the system with new capabilities. The potential collaborators have expressed a range of potential goals for their use of the Community Model, including performing comparisons with existing models, tailoring the hydrological performance to specific land uses and cropping types, and applying the model overseas.

Broadly speaking, there are four potential user categories of the AWRA Operational Model outputs and the AWRA Community Modelling System:

- **Data user:** who accessing the model outputs through the Bureau's website
- **Case study user:** who work with the Bureau to evaluate his/her case using 100 years of data
- **Applying users:** who would primarily be interested in applying the current model to a region of interest using localised and/or scenario data where available, and
- **Contributor users**: who will extend the capabilities of the model with new research and coding (modify the system with new capabilities) – see Chapter 8.

It is expected that the majority of early adopters of the AWRA Community Modelling System will be **Applying users** looking to apply the system with local data/scenarios, with more **Contributor users** adopting the system as it becomes well known and established.

Importantly, the community modelling system is not intended to address the needs of users and organisations that simply need to consume the model outputs from the operational system (**Data user** and **Case study users** listed above**)**. For these situations, and in general, the existing Bureau's web site and registered data services will continue to be the point of truth for AWRA-L operational model outputs. However, the release of the modelling system to the community would serve to highlight the potential use of outputs from the AWRA-L model.

1.4. Licensing

The Bureau has chosen the Berkeley Software Distribution (BSD) license modified according to the Bureau's legal advice and requirements (see Appendix D), including restricting commercial use of the modelling system. The adoption of a modified BSD-based license is expected to simplify the adoption decision for many potential users as it is simple, well known and open source. Similarly, the Bureau has opted for the Contributor License Agreement (CLA; see Appendix E) for users who wish to submit enhancements to the modelling system they have authored to the Bureau. The CLA will either a) require the contributor to assign copyright in the contribution to the lead organisation, or b) require the contributor to license the lead organisation to use the contribution, including the right to on-license the contribution.

1.5. User Registration

There are two types of Users with the AWRA CMS: Users and Registered Users. Any ***User*** can access the AWRA CMS package and documentation via GitHub, and are bound by the modified BSD license agreement. However, to gain access to the evaluation datasets a user must become an AWRA CMS ***Registered User***.

Reasons to become an AWRA CMS Registered User

- Access to complete calibration datasets (catchment-average time-series)

- Access to complete benchmarking datasets (catchment-average time-series)
- Solar radiation "climatology" gridded dataset for input prior to 1990
- Allowing the user to create a fork of github repository further allowing the opportunity to contribute to the ongoing development of the AWRA CMS code (see Chapter 8)
- User manual (this document)
- Technical Model Description Report (Frost, Ramchurn and Smith, 2016)
- Benchmarking Report (Frost, Ramchurn and Hafeez, 2016)
- Ongoing support

To become a Registered User send an email to awracms@bom.gov.au and the Bureau will send you the licence agreement and instructions on how to access the datasets. At this point information on using the GitHub repository and how Bureau will manage updating the master copy of the AWRA CMS repository will be provided.

1.6. This user guide

1.6.1. Purpose

The remainder of this User Guide describes how you interact with the AWRA CMS, what you can do with the system, and how you do it.

While the AWRA CMS can be applied to smaller catchments, this User Guide uses the Australian continent to provide examples for setup, navigation, simulation, interrogation, calibration and analyses of model results.

This User Guide provides a technical description of the model in the context of the code rather than a scientific description of the model which is provided in The Bureau's Operational AWRA-L Model technical description document (Frost, Ramchurn and Smith, 2016); provided to Registered Users.

1.6.2. Structure

This User Guide has 9 Chapters which are ordered according to how a typical user would work through the system. Chapters cover:

1. Introduction
2. Installation
3. Using the AWRA CMS
4. Visualisation
5. Model Calibration

2. Installing AWRA CMS

Using an open-source Python-based Jupiter notebook solution offers the flexibility that the system can be installed from source code on various **Linux** operating systems, **OS X** (Apple Macintosh Unix based operating systems) and **Windows Subsystem for Linux**. Build steps for the most common operating systems are provided in 2.2.

This section provides an overview of installation processes. The user is advised that system installation requires basic skills in Linux or OS X operating system (navigation, shell interaction, library and programme installation).

2.1. System overview

The Python programming language (Python 3.4 and Jupyter notebook 3.2) underpins the AWRA CMS. Python is a widely used, platform independent programming language. It is recommended that the user complete the tutorials and familiarise themselves with the basics of Python before setting up the AWRA CMS (see Appendix B).

The AWRA-L model kernel is itself coded in C. C is used in the kernel to ensure the fastest computation time for elements of the code (i.e. the models) that get run repeatedly at each time step. The user is not expected to know C programming in routine model interaction. Knowledge of these languages is however necessary if the user wishes to alter the model internal processes.

The user interface to the modelling system departs from a traditional rigid GUI template, and instead makes use of the Jupyter Notebooks (Learn more about notebooks at: http://jupyter.org/) to facilitate experimentation with code, user-driven data analysis and visualisation of outputs. This is web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text (SEE APPENDIX C).

2.2. System Setup

The modelling system has been developed, tested and used on **Linux OS**. Installation guidance is provided for several flavours of **Linux** (latest versions of Ubuntu, Fedora, Centos, Scientific Linux, Debian, Mint and “Bash on Ubuntu on Windows”). Guidance is also provided for installing the system on **OS X** although this platform is not supported.

The modelling system is designed to be run with Python 3 (specifically has been developed and tested with v3.4.2 and tested with 3.5.2) and relies on the following libraries and packages:

- C compiler to build the core model code
- NetCDF library (version 4.3, NOTE: 4.4 creates files that are unreadable with *h5py*) and Python bindings (*netCDF4*)
- HDF5 library (Tested on 1.8 and up) and Python bindings (*h5py*)
- Python *numpy* and *pandas* packages
- IPython/Jupyter notebook for the recommended interactive use of the modelling system
- And Python packages via *pip install*
    - *cffi* – for building the model Python bindings
    - *pyzmq* – for inter-process communication
    - *matplotlib* – for image and graph display
    - *nose* – for running tests

All installations use a virtual environment which allows the convenience of isolating different versions of Python and packages. It is however advised that the installation of all the pre-requisite libraries may require administrator privileges.

Certain spatial analysis functions also rely on the osgeo/GDAL libraries and corresponding Python bindings. To use shape files GDAL needs to be installed. However, this is optional, as the simulation package will work without it. GDAL installation as follows should be performed after building the python virtual environment.

For Red Hat flavours of Linux:

- to install GDAL type: `sudo dnf install gdal gdal-devel`
- to install GDAL Python package type: `pip install gdal`

For Debian flavours of Linux:

- to install GDAL type: `sudo apt-get install gdal-bin libgdal-dev`
- to install GDAL Python package type: `pip install gdal`

There is always a number of ways to install these pre-requisites on any given operating system and the situation will be different on different OS versions. The following sections give guidance for setting up the pre-requisites on different operating systems.

2.2.1. Linux: Fedora

The following installation procedure was successful on 22/09/2016 with Fedora 24 64-bit.

```
export PYENV="/home/[user]/.venv"
export BUILD="$PYENV/build"

mkdir -p $BUILD
cd $BUILD

### building Python requires several libraries - this will install these libraries system-wide
sudo dnf install openssl-devel bzip2-devel freetype-devel libsqlite3-devel
sudo dnf install gcc-c++

### download and install python
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar zxf Python-3.5.2.tgz
cd Python-3.5.2
./configure --prefix=$PYENV
make
make install
cd ..

### create and activate virtual environment
$PYENV/bin/pyvenv $PYENV/virtualenv
. $PYENV/virtualenv/bin/activate

export LD_LIBRARY_PATH="$PYENV/lib:$LD_LIBRARY_PATH"
export CPPFLAGS="$CPPFLAGS -I$PYENV/include -I$PYENV/lib"
export LDFLAGS="$LDFLAGS -L$PYENV/lib"

### download and install latest HDF5(1.8.17)
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.17.tar.gz
tar xzf hdf5-1.8.17.tar.gz
cd hdf5-1.8.17
./configure --prefix=$PYENV
make check
make install
cd ..

### download and install NETCDF (do not use latest(4.4.*) since files created cannot be read by h5py)
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/old/netcdf-4.3.2.tar.gz
tar xzf netcdf-4.3.2.tar.gz
cd netcdf-4.3.2
./configure --prefix=$PYENV --enable-dap
make check
make install
```

```
cd ..

### install python package netCDF4
export USE_SETUPCFG=0
export HDF5_INCDIR=$PYENV/include
export HDF5_LIBDIR=$PYENV/lib
pip install netCDF4==1.2.4

### required python packages
pip install h5py
pip install pyzmq
pip install pandas==0.16.1
pip install matplotlib==1.4.3
pip install ipython[notebook]

### python package cffi and required libraries
sudo dnf install python3-cffi
sudo dnf install libffi-devel
pip install cffi==1.1.2
```

2.2.2. Linux: Ubuntu

The following installation procedure was successful on 19/09/2016 using Ubuntu 16.04 LTS 64-bit.

```
export PYENV="/home/[user]/.venv"
export BUILD="$PYENV/build"

mkdir -p $BUILD
cd $BUILD

### building Python requires several libraries - this will install these libraries system-wide
sudo apt-get install libssl-dev zlib1g-dev libbz2-dev libsqlite3-dev
sudo apt-get install libfreetype6-dev libpng-dev

### download and install python
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar zxf Python-3.5.2.tgz
cd Python-3.5.2
./configure --prefix=$PYENV
make
make install
```

```
cd ..

### create and activate virtual environment
$PYENV/bin/pyvenv $PYENV/virtualenv
. $PYENV/virtualenv/bin/activate

export LD_LIBRARY_PATH="$PYENV/lib:$LD_LIBRARY_PATH"
export CPPFLAGS="$CPPFLAGS -I$PYENV/include -I$PYENV/lib"
export LDFLAGS="$LDFLAGS -L$PYENV/lib"

### download and install latest HDF5(1.8.17)
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.17.tar.gz
tar xzf hdf5-1.8.17.tar.gz
cd hdf5-1.8.17
./configure --prefix=$PYENV
make check
make install
cd ..

### download and install NETCDF (do not use latest(4.4.*) since files created cannot be read by h5py)
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/old/netcdf-4.3.2.tar.gz
tar xzf netcdf-4.3.2.tar.gz
cd netcdf-4.3.2
./configure --prefix=$PYENV --enable-dap
make check
make install
cd ..

### install python packages
pip install numpy==1.9.3

### download and build python package netCDF4
wget https://pypi.python.org/packages/source/n/netCDF4/netCDF4-1.2.4.tar.gz
tar zxf netCDF4-1.2.4.tar.gz
cd netCDF4-1.2.4
cp setup.cfg.template setup.cfg
### if build cannot find hdf5 then need to export these 2 environment variables
export HDF5_DIR=$PYENV
export NETCDF4_DIR=$PYENV
python setup.py build
python setup.py install
cd ..
```

```
### required python packages
pip install h5py
pip install pyzmq
pip install pandas==0.16.1
pip install matplotlib==1.4.3
pip install ipython[notebook]

### python package cffi and required libraries
sudo apt-get install python3-cffi
sudo apt-get install libffi-dev
pip install cffi==1.1.2
```

2.2.3. Linux: Mint

The following installation procedure was successful on 28/09/2016 using Mint 18 ‘Sarah’ 64-bit with cinnamon desktop. The OS image file was downloaded from https://www.linuxmint.com/download.php.

```
export PYENV="/home/[user]/.venv"
mkdir -p "$PYENV/build"
cd $PYENV/build

### building Python requires several libraries - this will install these libraries system-wide
sudo apt-get install build-essential
sudo apt-get install libssl-dev zlib1g-dev libbz2-dev libsqlite3-dev
sudo apt-get install libfreetype6-dev libpng-dev

# download and install Python
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar zxf Python-3.5.3.tgz
cd Python-3.5.2
./configure --prefix=$PYENV
make
make install
cd ..

### create and activate virtual environment
$PYENV/bin/pyvenv $PYENV/virtualenv
. $PYENV/virtualenv/bin/activate

export LD_LIBRARY_PATH="$PYENV/lib:$LD_LIBRARY_PATH"
export CPPFLAGS="$CPPFLAGS -I$PYENV/include"
```

```
export LDFLAGS="$LDFLAGS -L$PYENV/lib"

### download and install latest HDF5(1.8.17)
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.17.tar.gz
tar xzf hdf5-1.8.17.tar.gz
cd hdf5-1.8.17
./configure --prefix=$PYENV
make check
make install
cd ..

### download and install NETCDF (do not use latest(4.4.*) since files created cannot be read by h5py)
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/old/netcdf-4.3.2.tar.gz
tar xzf netcdf-4.3.2.tar.gz
cd netcdf-4.3.2
./configure --prefix=$PYENV --enable-dap
make check
make install
cd ..

### install python packages
pip install numpy==1.9.3

### install netCDF4 python package
export HDF5_DIR=$PYENV
export NETCDF4_DIR=$PYENV
pip install netCDF4==1.2.4

### python package cffi and required libraries
sudo apt-get install python3-cffi
sudo apt-get install libffi-dev
pip install cffi==1.1.2

### required python packages
pip install h5py
pip install pyzmq
pip install pandas==0.16.1
pip install matplotlib
pip install ipython[notebook]
```

2.2.1. Linux: Centos and Scientific Linux

The following installation procedure was successful on 5/10/2016 with Centos 7 64-bit (with EPEL repository enabled) and Scientific Linux 7.2 64bit installations packaged with development tools.

```
export PYENV="/home/[user]/.venv"
export BUILD="$PYENV/build"

mkdir -p $BUILD
cd $BUILD

### building Python requires several libraries - this will install these libraries system-wide
sudo yum install openssl-devel bzip2-devel freetype-devel sqlite3-devel libpng libpng-devel
sudo yum install gcc-c++

### download and install python
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar zxf Python-3.5.2.tgz
cd Python-3.5.2
./configure --prefix=$PYENV
make
make install
cd ..

### create and activate virtual environment
$PYENV/bin/pyvenv $PYENV/virtualenv
. $PYENV/virtualenv/bin/activate

export LD_LIBRARY_PATH="$PYENV/lib:$LD_LIBRARY_PATH"
export CPPFLAGS="$CPPFLAGS -I$PYENV/include -I$PYENV/lib"
export LDFLAGS="$LDFLAGS -L$PYENV/lib"

### download and install latest HDF5(1.8.17)
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.17.tar.gz
tar xzf hdf5-1.8.17.tar.gz
cd hdf5-1.8.17
./configure --prefix=$PYENV
make check
make install
cd ..

### download and install NETCDF (do not use latest(4.4.*) since files created cannot be read by h5py)
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/old/netcdf-4.3.2.tar.gz
```

```
tar xzf netcdf-4.3.2.tar.gz
cd netcdf-4.3.2
./configure --prefix=$PYENV --enable-dap
make check
make install
cd ..

### install python package netCDF4
export USE_SETUPCFG=0
export HDF5_INCDIR=$PYENV/include
export HDF5_LIBDIR=$PYENV/lib
pip install netCDF4==1.2.4

### required python packages
pip install h5py
pip install pyzmq
pip install pandas==0.16.1
pip install matplotlib==1.4.3
pip install ipython[notebook]

### python package cffi and required libraries
sudo yum install python3-cffi
sudo yum install libffi-devel
pip install cffi==1.1.2

## to install gdal
## install latest supported version of geos
wget -N "http://download.osgeo.org/geos/geos-3.5.0.tar.bz2"
tar xjf "geos-3.5.0.tar.bz2"
cd "geos-3.5.0"
./configure --prefix=$PYENV
make && make install
cd ..

## install proj4
wget -N "http://download.osgeo.org/proj/proj-4.9.2.tar.gz"
tar xzf "proj-4.9.2.tar.gz"
cd "proj-4.9.2"
./configure --prefix=$PYENV
make && make install
cd ..

## install swig
```

```
wget -N "http://downloads.sourceforge.net/swig/swig-3.0.10.tar.gz"
tar xvf swig-3.0.10.tar.gz
cd "swig-3.0.10" || fail
./configure --prefix=$PYENV
make && make install
cd ..

## install gdal as a system wide library
wget -N "http://download.osgeo.org/gdal/2.1.0/gdal-2.1.0.tar.gz"
tar xzf "gdal-2.1.0.tar.gz"
cd "gdal-2.1.0"
./configure --with-netcdf=$PYENV  --with-static-proj4=$PYENV --with-geos=$PYENV/bin/geos-config
sudo make && sudo make install
cd ..

export PYTHONPATH="$PYENV/lib/python3.5/site-packages/"
export CPLUS_INCLUDE_PATH=$PYENV/include
export C_INCLUDE_PATH=$PYENV/include
pip install gdal==2.1.0
```

2.2.1. Linux: Bash on Ubuntu on Windows

Microsoft has provided a new feature enabling a Linux environment on Windows called Windows Subsystem for Linux (WSL) also known as Bash on Ubuntu on Windows. This feature is available on a 64 bit version of Windows 10 Anniversary update build 14393 or later. To enable WSL follow the instructions at

https://msdn.microsoft.com/en-au/commandline/wsl/install_guide

It is preferable to setup the virtual environment on the bash file system as this gives control over file permissions with *chmod*.

```
# choose a path to setup virtual env
export PYENV="/home/[user]/.venv"
mkdir -p "$PYENV/build"
cd $PYENV/build

### build requires several libraries - this will install these libraries system-wide
sudo apt-get install build-essential
sudo apt-get install m4
sudo apt-get install libssl-dev zlib1g-dev libbz2-dev libsqlite3-dev
sudo apt-get install libfreetype6-dev libpng-dev

# download and install Python
```

```
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar zxf Python-3.5.3.tgz
cd Python-3.5.2
./configure --prefix=$PYENV
make
make install
cd ..

### create and activate virtual environment
$PYENV/bin/pyvenv $PYENV/virtualenv
. $PYENV/virtualenv/bin/activate

export LD_LIBRARY_PATH="$PYENV/lib:$LD_LIBRARY_PATH"
export CPPFLAGS="$CPPFLAGS -I$PYENV/include"
export LDFLAGS="$LDFLAGS -L$PYENV/lib"

### download and install latest HDF5(1.8.17)
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.17.tar.gz
tar xzf hdf5-1.8.17.tar.gz
cd hdf5-1.8.17
./configure --prefix=$PYENV
make check
make install
cd ..

### download and install NETCDF (do not use latest(4.4.*) since files created cannot be read by h5py)
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/old/netcdf-4.3.2.tar.gz
tar xzf netcdf-4.3.2.tar.gz
cd netcdf-4.3.2
./configure --prefix=$PYENV --enable-dap
make check
make install
cd ..

### install python packages
pip install numpy==1.9.3

### install netCDF4 python package
export HDF5_DIR=$PYENV
export NETCDF4_DIR=$PYENV
pip install netCDF4==1.2.4

### python package cffi and required libraries
```

```
sudo apt-get install python3-cffi
sudo apt-get install libffi-dev
pip install cffi==1.1.2

### required python packages
pip install h5py
pip install pandas==0.16.1
pip install matplotlib
pip install ipython[notebook]

# method to enable notebooks in WSL from http://blog.lanzani.nl/2016/jupyter-wsl/
pip uninstall pyzmq
sudo add-apt-repository ppa:aseering/wsl
sudo apt-get update
sudo apt-get install libzmq3 libzmq3-dev
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/lib/x86_64-linux-gnu"
pip install --no-use-wheel -v pyzmq
pip install jupyter
```

It is recommended to install AWRA CMS code bundle to bash file system to enable file permission changes with chmod. Nosetests will only run python test files with all execution permissions turned off *chmod -x test_file.py*

To start a notebook type *jupyter notebook* at the bash prompt. This will start a notebook server accessible in the browser at *http://localhost:8888/notebooks*.

2.2.2. Mac OS X

Recent versions of OS X come pre-installed with Python 2.7, so you will need to separately install a more recent version (Python 3.5 or later). You will also need to install the HDF5 and NetCDF4 libraries and possibly the GDAL libraries. There are various options for installing each of these packages, but one of the easier options is to use the Homebrew package manager.

- install Homebrew from http://brew.sh
- install Python 3 using *brew install python3*
- install NetCDF (which will install HDF5 as a dependency):

```
brew tap homebrew/science
brew install netcdf
```

- optionally install GDAL using brew install homebrew/versions/gdal111

- install the Python NetCDF4 library, the h5py library, pandas and the ipython notebook

```
pip3 install netcdf4 h5py pandas ipython[notebook]
```

- optionally install GDAL Python package with *pip3 install gdal==1.11.1* (The version number is required because, at the time of writing, homebrew installs a version 1.X release of GDAL. If this changes and Homebrew installs a version 2.X library, you should be able to install the latest python bindings with *pip3 install gdal*).

2.3. Installing AWRA CMS packages from source

Download source from Github by either cloning or downloading. Downloading a repository just downloads the files from the most recent commit of the default branch. It doesn't download any of the files in the .git folder and therefore the history. When you clone you get a copy of the history and it is a functional git repository.

To download and install the AWRA CMS repository:

```
### download and install AWRA CMS
### activate virtual environment
. $PYENV/virtualenv/bin/activate

tar zxf awrams_cm.tar.gz
cd awrams_cm

cd utils
python setup.py install
cd ..

cd benchmarking
python setup.py install
python setup.py nosetests
cd ..

cd models
python setup.py install
python setup.py nosetests
cd ..

cd simulation
python setup.py install
python setup.py nosetests
cd ..
```

```
cd visualisation
python setup.py install
python setup.py nosetests
cd ..

cd calibration
python setup.py install
python setup.py nosetests
cd ..

cd utils
python setup.py nosetests
cd ..
```

2.4. Getting started

2.4.1. Interface

Following successful installation, Jupyter Notebooks are initiated from the python command line using the command:

```
user@localhost~: jupyter notebook
```

2.4.2. Directories

The directories of the package are structured around the four major components of the AWRA CMS Package

- Simulation
- Visualisation
- Calibration
- Benchmarking
- Utilities

Each of these components consists of multiple modules that define and run these sections of the AWRA CMS.

2.4.3. User Manual Notebooks

The five AWRA CMS components each have at least one Jupyter Notebook that demonstrates the functionality of the package and aligns with the example code presented in this User Guide. These notebooks are located in the ‘userman’ folder under the root directory of the package. It is recommended that the user access these userman Notebooks to get them started.

2.4.4. Test Input Data

Each of the five AWRA CMS components has a set of test data associated with it that provide an example of the input data format and feeds into the notebooks to demonstrate package functionality. The data is located under each of the package directories eg simulation/tests/data/.

To gain access to the full validation datasets become a AWRA CMS Registered User (Section 1.5). The full 105 years of climate input data can be accessed through the AusCover website.

The AusCover AWAP gridded climate data is produced by the Bureau and distributed by AusCover which is funded by the Terrestrial Ecosystem Research Network (TERN).

3. Using the AWRA CMS

The AWRA CMS environment allows an AWRA-L model simulation to be run according to user specifications including:

- model version
- model parameter set, possibly from user calibration or for sensitivity analysis
- initial soil moisture/groundwater and leaf matter mass state
- spatial extent
- time period
- desired model outputs

The simplest use case is to run AWRA-L v5 with the default parameters and initial states. This case will be demonstrated in this section to get users started and familiar with the system and subsequent sections will introduce the flexibility of the system.

There are two ways to run a simulation of the AWRA-L model in the AWRA CMS,

1. Using the On-Demand Simulator and
2. Using the Server Simulator.

This section will outline how to run a basic AWRA-L simulation with both methods and provide details on how to edit the input and output nodegraphs to customise the simulation.

The Simulation.ipynb userman notebook provides a live interface with these examples.

3.1. Basic On-Demand Simulation

The AWRA CMS On-Demand Simulator is designed to run the model for a few years over a small spatial extent. It allows the user to quickly and efficiently assess the impact of changes made to the model or inputs before running a full simulation and writing all of the outputs to file. The user can write the results out but it is generally designed to hold the results in memory for visualisation and checking.

The following 7 steps outline a basic On-Demand Simulation of the AWRA-L model using default inputs and retaining the results in memory rather than writing them out. There is additional functionality outlined in section 3.3 such as changing the inputs, model and outputs.

1. Import all the necessary packages (e.g. numpy, pandas etc)
2. Read in the default or custom nodegraph input mapping

3. Set climate forcing data paths as part of the nodegraph input
4. Setup the system (runner) using the nodegraph
5. Define the required temporal period and spatial extent
6. Run the model for the defined ‘extent’ and ‘period’
7. Basic gridded image display of results

1. Import the necessary packages

```
# General
import numpy as np
import pandas as pd
import os
from os.path import join,dirname
from os import getcwd

# Utilities
from awrams.utils import extents
from awrams.utils import datetools as dt
from awrams.utils.precision import quantize

# Plotting
from matplotlib import pyplot as plt

# Nodegraph
from awrams.utils.nodegraph import nodes

# Simulation
from awrams.simulation.ondemand import OnDemandSimulator
from awrams.simulation.server import Server

# AWRA model
from awrams.models import awral
awral.CLIMATE_DATA = '../tests/data/'

# Create Results directory in same path as Notebook
os.makedirs('./_results',exist_ok=True)
```

2. Read in default input nodegraph

```
imap = awral.get_default_mapping()
```

```
# View input map
for k in imap.mapping:
    print(k,imap.mapping[k])
```

3. Set forcing data paths

```
def change_path_to_forcing(imap):
    path = getcwd()
    path = join(path,'..','tests','data')

    FORCING = {
        'tmin'  : ('temp_min*','temp_min_day'),
        'tmax'  : ('temp_max*','temp_max_day'),
        'precip': ('rain_day*','rain_day'),
        'solar' : ('solar*'   ,'solar_exposure_day')
    }
    for k,v in FORCING.items():
        imap.mapping[k+'_f'] = nodes.forcing_from_ncfiles(path,v[0],v[1])

change_path_to_forcing(imap)
```

4. Setup system (runner) using the nodegraph

```
runner = OnDemandSimulator(awral,imap.mapping)
```

5. Define the required period and spatial extent

```
period = dt.dates('Dec 2010- Jan 2011')
extent = extents.from_boundary_offset(400,50,450,100)
```

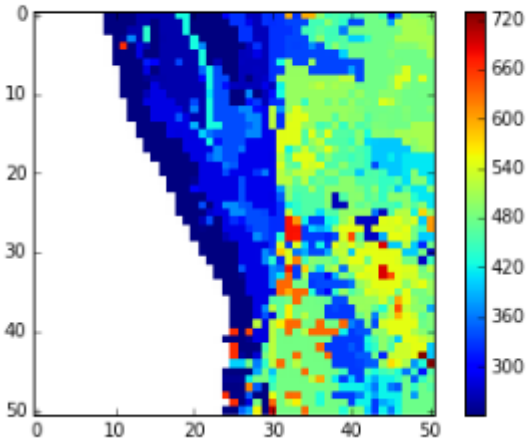
6. Run the model for the defined extent and period

```
r = runner.run(period,extent)
```

7. Basic display of results

```
im = plt.imshow(nodes.get_expanded(r['sd'][0],extent.mask),interpolation='None')
plt.colorbar(im)
```

```
<matplotlib.colorbar.Colorbar at 0x7f561fcd1f28>
```



3.2. Basic Server Simulation

To run the AWRA model for more than a couple of years or for a significant extent the server version of the simulator should be used. The Server Simulation is very similar to the On Demand Simulator in the way that it is run with the main difference being it requires an output map with results paths defined as it does not hold the results in memory. This allows the Server Simulator to run much quicker for a longer period over a larger extent. The same output and input maps can be used for either simulator

1. Define and input and output maps the same as the On-Demand Simulator

```
imap = awral.get_default_mapping()
change_path_to_forcing(imap)
```

```
def build_output_mapping():
    import os
    from awrams.utils.nodegraph import nodes,graph
    from awrams.models.awral.template import DEFAULT_TEMPLATE


    output_map = awral.get_output_nodes(DEFAULT_TEMPLATE)


    outpath = './_results/'
    output_map.mapping['s0_ncsave'] = nodes.write_to_annual_ncfile(outpath,'s0')
    output_map.mapping['ss_ncsave'] = nodes.write_to_annual_ncfile(outpath,'ss')
    output_map.mapping['sd_ncsave'] = nodes.write_to_annual_ncfile(outpath,'sd')
    output_map.mapping['qtot_ncsave'] = nodes.write_to_annual_ncfile(outpath,'qtot')
    output_map.mapping['etot_ncsave'] = nodes.write_to_annual_ncfile(outpath,'etot')
```

```
    return graph.OutputGraph(output_map.mapping)

# Run the function to crate the new output map
omap = build_output_mapping()
```

2. Define the required temporal period and spatial extent

Because it is the Server Simulator the extent can be defined as the full continent which is the default

```
period = dt.dates('Dec 2010- Jan 2011')
extent = extents.default()
```

3. Run the model for the defined extent and period

```
sim = Server(awral)
sim.run(imap,omap,period,extent)
```

3.3. Customise the simulation

3.3.1. Setting the Geographical Extent (extent)

There are several different ways the geographical extent of the model can be setup from single cell, bounding box to region id. Below are some options for setting the extent over which the model is run

```
# The whole continent
extent = extents.default()

# Bounding box grid cell index (x, y, x, y)
extent = extents.from_boundary_offset(400,50,450,100)

# Bounding box (lat, long)
extent = extents.from_boundary_coords(-39.5,143.5,-44,149)

# Location of single cell by index (x, y)
extent = extents.from_cell_offset(430,75)

# Location of single cell by location ('lat', 'long')
extent = extents.from_cell_coords(-31.95,115.85)

# Defined Region via a shapefile (shapefile must have an 'id' column )
```

```
from awrams.utils import catchments
path = getcwd()
shapefilename = join(path, '../..','utils','awrams','utils','data', 'Final_list_all_attributes.shp')
try:
    db = catchments.CatchmentDB(shapefilename)
    extent = db.get_by_id('146095')

except ImportError as e:
    print(e)
```

3.3.2. Setting the period of run (period)

The way the period is set is quite flexible in the AWRA CMS. It can appear in any one of the following formats

```
period = dt.dates('2010 - 2012')
period = dt.dates('1 jan 2010 - 31 dec 2012')
period = dt.dates('1 jan 2010') # a single day
period = dt.dates('dec 2010 - jan 2011')
```

3.3.3. Output Nodegraph (variables)

The AWRA CMS uses nodegraphs to define the input and output mapping for the AWRA-L model. The nodegraphs are a functional-style datagraph that builds the input and output mapping that is then passed to the simulator (On-Demand or Server) to run the model. Generally the default maps are defined and then edited to customise a simulation.

The Output nodegraph defines which variables are to be output where, and any transformations to be completed upon output.

```
def build_output_mapping():
    import os
    from awrams.utils.nodegraph import nodes,graph
    from awrams.models.awral.template import DEFAULT_TEMPLATE

    output_map = awral.get_output_nodes(DEFAULT_TEMPLATE)

    outpath = './_results/'

    output_map.mapping['s0_ncsave'] = nodes.write_to_annual_ncfile(outpath,'s0')
    output_map.mapping['ss_ncsave'] = nodes.write_to_annual_ncfile(outpath,'ss')
```

```
    output_map.mapping['sd_ncsave'] = nodes.write_to_annual_ncfile(outpath,'sd')

    return graph.OutputGraph(output_map.mapping)

# Run the function to crate the new output map
omap = build_output_mapping()
```

```
# To save the results include the output mapping
runner = OnDemandSimulator(awral,imap.mapping,omap.mapping)
```

3.3.4. Input Nodegraph

The input nodegraph can be customised in many ways to reflect changes the user wants to make to the inputs of the AWRA-L model. These changes can include anything from changing the path to the forcing data to defining the initial states to be used in the simulation. A few examples are presented here. All of these examples update the input mapping which is then just run through one of the simulators.

The first step is always to read in the default input map.

```
# Get the default nodegraph and edit to build custom nodegraph
imap = awral.get_default_mapping()
```

1. Changing the path to forcing data

The below will set the naming convention and relative path according to the test data provided with the community model package but any path and naming convention can be set and run through this function. 'tmin' is the node, 'temp_min*' is the filename pattern and 'temp_min_day' is the variable name within the .nc file.

```
def change_path_to_forcing(imap):
    path = getcwd()
    path = join(path,'..','tests','data')

    FORCING = {
        'tmin'  : ('temp_min*','temp_min_day'),
        'tmax'  : ('temp_max*','temp_max_day'),
        'precip': ('rain_day*','rain_day'),
        'solar' : ('solar*'   ,'solar_exposure_day')
    }
    for k,v in FORCING.items():
        imap.mapping[k+'_f'] = nodes.forcing_from_ncfiles(path,v[0],v[1])
```

```
change_path_to_forcing(imap)
```

2. Fill gaps in forcing data with climatology

```
def insert_climatology(imap):
    from awrams.utils.nodegraph import nodes
    from os.path import join,dirname
    from os import getcwd

    path = getcwd()
    path = join(path,'..','tests','data')

    imap.mapping['precip_f'] = nodes.forcing_gap_filler(path,'rain_day*','rain_day',join(path,'climatology_daily_rain_day.nc'))
    imap.mapping['tmax_f'] = nodes.forcing_gap_filler(path,'temp_max*','temp_max_day',join(path,'climatology_daily_temp_max_day.nc'))
    imap.mapping['tmin_f'] = nodes.forcing_gap_filler(path,'temp_min*','temp_min_day',join(path,'climatology_daily_temp_min_day.nc'))
    imap.mapping['solar_f'] = nodes.forcing_gap_filler(path,'solar*','solar_exposure_day',join(path,'climatology_daily_solar_exposure_day.nc'))
```

3. Using the TERN AusCover data as input forcing data

The AusCover AWAP gridded climate data is produced by the Bureau and distributed by AusCover which is funded by the Terrestrial Ecosystem Research Network (TERN).

AusCover data can be downloaded from thredds server with catalog located at http://rs-data1-mel.csiro.au/thredds/catalog/bawap/catalog.html

The AusCover data is in a slightly different file format to the default AWRA forcing data but the following remapping will bring the data into the AWRA CMS format.

```
def use_AusCover_for_forcing(imap):
    from awrams.utils.nodegraph.nodes import forcing_from_ncfiles
    from os.path import join
    from os import getcwd

    path = getcwd()
    path = join(path,'..','tests','data')

    imap.mapping['precip_f'] = forcing_from_ncfiles(path,'bom-rain_day-*','rain_day')
    imap.mapping['tmax_f']   = forcing_from_ncfiles(path,'bom-tmax_day-*','tmax_day')
    imap.mapping['tmin_f']   = forcing_from_ncfiles(path,'bom-tmin_day-*','tmin_day')
    imap.mapping['solar_f']  = forcing_from_ncfiles(path,'bom-rad_day-*','rad_day')
```

4. Getting initial states for the model

In [23]:

```
# Initialise states from dict mapping of numpy arrays

def get_initial_states_dict(imap,period,extent):
    from awrams.utils.io.data_mapping import SplitFileManager
    from awrams.utils.nodegraph import nodes

    from os.path import join,dirname
    from os import getcwd

    path = getcwd()
    path = join(path,'..','tests','data')

    node_names = {'mleaf_dr': 'init_mleaf_hrudr',
                  'mleaf_sr': 'init_mleaf_hrusr',
                  's0_dr': 'init_s0_hrudr',
                  's0_sr': 'init_s0_hrusr',
                  'ss_dr': 'init_ss_hrudr',
                  'ss_sr': 'init_ss_hrusr',
                  'sd_dr': 'init_sd_hrudr',
                  'sd_sr': 'init_sd_hrusr',
                  'sg_bal': 'init_sg',
                  'sr_bal': 'init_sr'}

    data_map = {}
    period = [period[0] - 1]

    for k in node_names:
        sfm = SplitFileManager.open_existing(path,k+'*nc',k)
        data_map[node_names[k]] = sfm.get_data(period,extent)

    nodes.init_states_from_dict(imap,data_map,extent)

get_initial_states_dict(imap,period,extent)
```

```
# Initialise states from netcdf files (from a previous run)

def get_initial_states(imap):
    from awrams.utils.nodegraph import nodes
```

```
    from os.path import join
    from os import getcwd

    path = getcwd()
    path = join(path,'..','tests','data')

    mapping = imap.mapping

    mapping['init_sr'] = nodes.init_state_from_ncfile(path,'sr_bal*','sr_bal')
    mapping['init_sg'] = nodes.init_state_from_ncfile(path,'sg_bal*','sg_bal')

    HRU = {'_hrusr':'_sr','_hrudr':'_dr'}
    for hru in ('_hrusr','_hrudr'):
        for state in ["s0","ss","sd",'mleaf']:
            mapping['init_'+state+hru] = nodes.init_state_from_ncfile(path,state+HRU[hru]+'*',state+HRU[hru])

get_initial_states(imap)
```

5. Defining the static gridded Spatial parameters

Spatial parameters for the model are stored in HDF format and supplied with the awrams.models package. The path to the spatial parameter file is set with awrams.models.awral.settings.SPATIAL_FILE:

default value = 'awrams/models/awral/data/spatial_parameters.h5'

Parameter names in the HDF file correlate to required nodes in the model input node mapping. For example: fraction available water capacity in the surface soil layer (s0) is referenced as 's0fracawc' in the spatial parameter file and 's0fracawc_grid' in the input node mapping:

s0fracawc_grid spatial_from_hdf5([]):{'preload': False,
'filename':'./awrams/models/awral/data/spatial_parameters.h5', 'variable': 'parameters/s0fracAWC'}

The model expects spatial parameter grids to have the same dimensions as the extent geospatial reference (ie extent.parent_ref).

```
import h5py
path = getcwd()
spatial_params_file = join(path, '../..','models','awrams','models','awral','data', 'spatial_parameters.h5')
h = h5py.File(spatial_params_file,'r')

list(h['parameters'].keys())
```

['f_tree',

```
'height',
'hveg_dr',
'k0sat',
'k0sat_v5',
'k_gw',
'kdsat_v5',
'kssat',
'kssat_v5',
'lai_max',
'meanP',
'meanPET',
'ne',
'pref',
's0fracAWC',
'slope',
'ssfracAWC',
'windspeed']
```

3.4. Editing the AWRA-L code

The AWRAL C code can be edited in the awral_t.c template using the Jupyter Notebooks or a basic text editor. Changes to the code are detected at run-time. The model will be compiled if changes are detected and a new version of awral.c is created.

Example: changing the albedo calculation in the model

The user wants to reduce the complexity of the albedo calculations to test the energy balance terms in the model. In awral_t.c, replace the existing equation for albedo (alb) with a constant.

```
double alb_veg = 0.452 * vc;
double alb_soil = alb_wet + (alb_dry - alb_wet) * exp(-w0 / w0ref_alb);
//double alb = fveg * alb_veg + fsoil * alb_soil;
double alb = 0.08;
double rsn = (1.0 - alb) * rgeff;
```

3.5. Results

3.5.1. Location

The location of the output results of an AWRAL simulation is defined in the output_map by:

```
output_map.mapping['s0_ncsave'] = nodes.write_to_annual_ncfile(outpath,'s0')
```

(see section 3.3.3)..

3.5.2. File format and size

The AWRAL results are output in netcdf format with a single .nc file containing all of the daily grids in a year for a single variable. eg qtot_avg_2016.nc which contains the 365 daily runoff grids of 2016. A continental

simulation output is in the order of 250 MB per year per variable. The size of the files decreases with the spatial extent of the simulation.

The best way to handle, extract and manipulate these output files is to use the AWRA CMS tools. These tools are explored in the following Visualisation, Calibration, Benchmarking and Utility sections of this guide.

4. Visualisation

The results of an AWRA-L simulation can be viewed both spatially and temporally using the tools of the AWRA CMS. Some examples of these tools are presented in this section but the Visualisation.ipynb userman notebook provides a live interface with these tools and further examples.

4.1. Setup

1. Import the necessary packages

```
# General
import os
from os.path import join,dirname
from os import getcwd

# Visualisation
import awrams.visualisation.vis as vis
import awrams.visualisation.results as res

# Catchment definition tools
from awrams.utils.catchments import CatchmentDB,CATCHMENT_SHAPEFILE
catchments = CatchmentDB()
CATCHMENT_SHAPEFILE

# Get working directory
wd = os.getcwd()
```

2. Load results from simulation of AWRA-L

These results will be netcdf files

```
# The relative path that the Simulation notebook puts results when run
path = join(wd, '../..','simulation','notebooks','_results')
results = res.load_results(path)
```

3. Inspect the variables present in results

Tab completable access to variable list

```
results.variables
OrderedDict([('s0', s0), ('etot', etot), ('ss', ss), ('qtot', qtot), ('sd', sd)])
```

4.2. Display a spatial slice

Define a slice of interest for viewing = [variables, period, extent]

Variables slice

- display a single variable - results.variables.qtot_avg
- display multiple variables with a tuple - results.variables.qtot_avg,results.variables.ss_avg
- display all variables using standard slicing syntax - ":"

Period slice

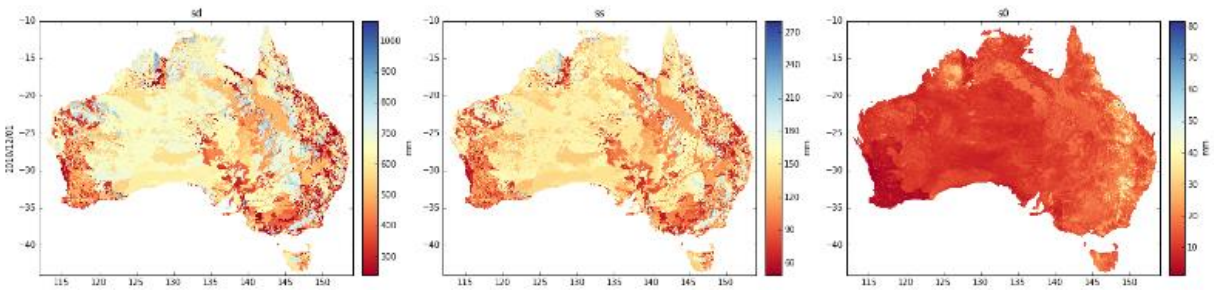
- a single day - "1 dec 2010"
- a period - "dec 2010" or "dec 2010 - jan 2011"
- data will be aggregated over the period using the specified method (pass aggregate_method='average' or aggregate_method='sum')
  or the default method for a variable

Extent slice

- entire spatial extents - ":" or vis.extents.ExtentAll()
- a bounding box - vis.extents.GeoBounds(lat0,lon0,lat1,lon1) eg vis.extents.GeoBounds(-40,145,-44,149)
- a catchment - catchments.by_name.Murrumbidgee_MittagangCrossing()

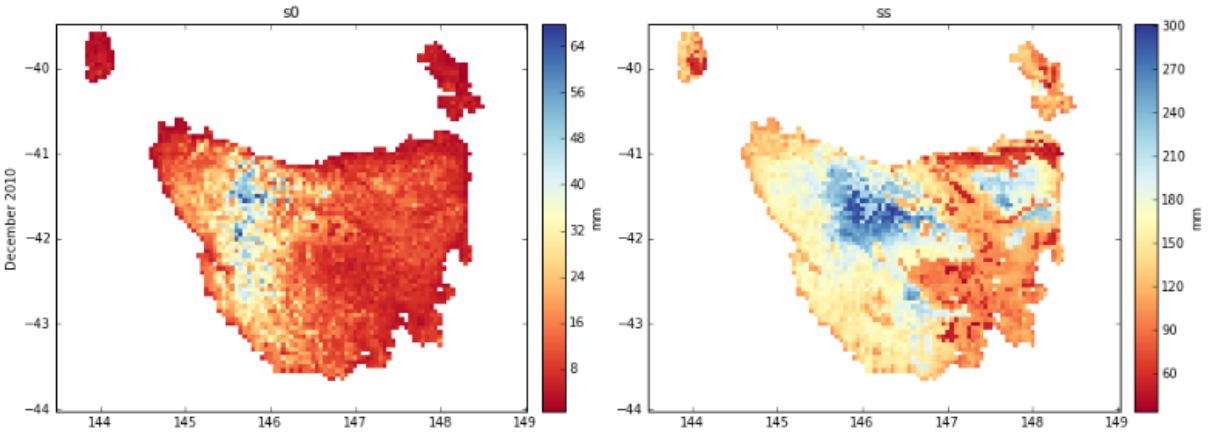
4.2.1. Display all variables, for a single day, at the default continental extent

```
results[:,'1 Dec 2010',:].spatial()
```



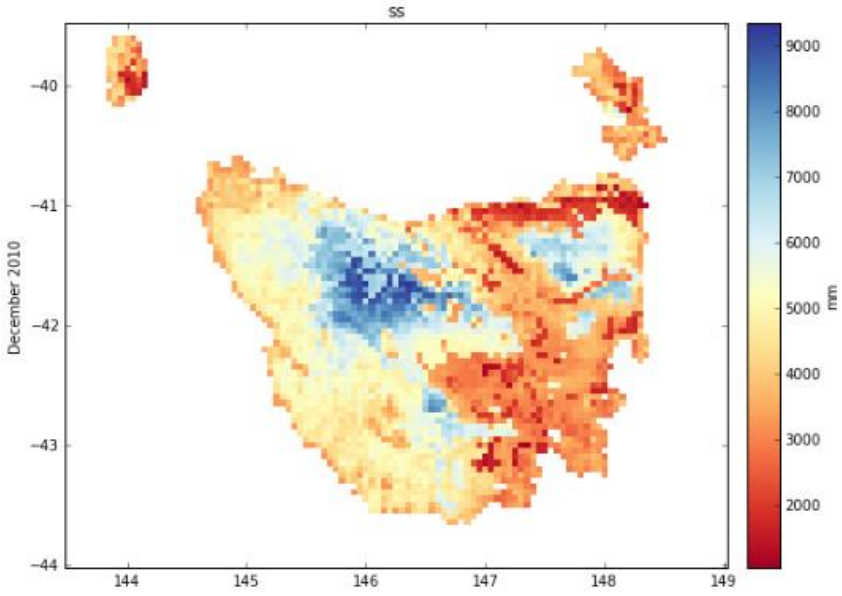
4.2.2. Aggregate over a month for specified region

```
v = results.variables.s0,results.variables.ss
results[v,'dec 2010',vis.extents.from_boundary_coords(-39.5,143.5,-44,149)].spatial()
vis.plt.savefig('map_of_tasmania.png', format='png', dpi=120)
```



Change aggregation method from default of mean to sum

```
v = results.variables.ss
v.agg_method = 'sum'
results[v,'dec 2010',vis.extents.from_boundary_coords(-39.5,143.5,-44,149)].spatial()
```



4.2.3. Accessing the underlying data

Raw data cube

```
results.variables.s0.data.shape
```

Temporally aggregated data

```
results.variables.s0.agg_data.shape
```

4.2.4. Aggregate over a catchment

By default catchments are read from shapefile *"awrams/utils/data/Final_list_all_attributes.shp"*
with features in the shapefile referenced by *key_field: 'StationID'*
and features named with *name_fields: ['GaugeName', 'RiverName']*
to supply a different shapefile call:

catchments = CatchmentDB(shp_file="shape_file_name")

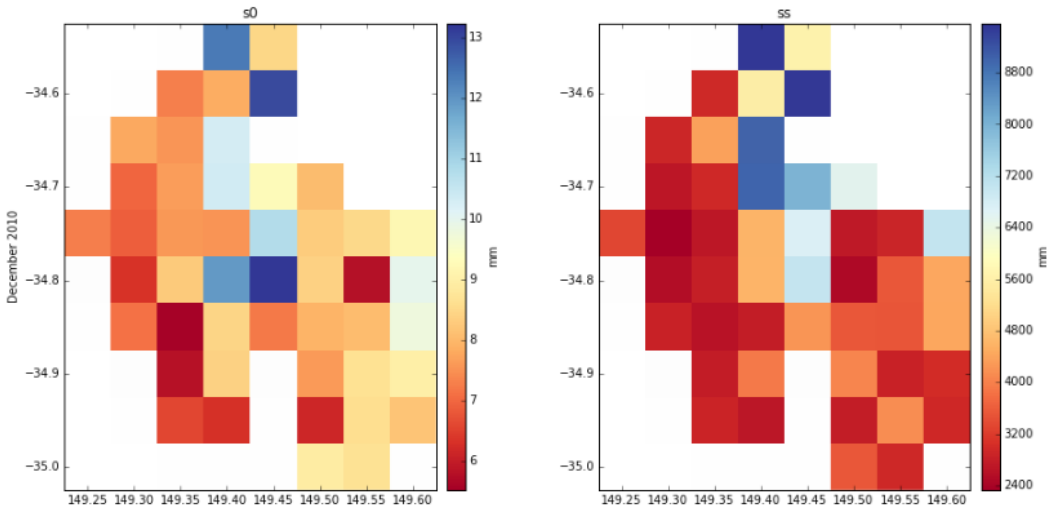
to specify different feature field names for key_field and name_fields, before calling CatchmentDB set:

CatchmentDB.key_field = "new_id_field"

CatchmentDB.name_fields =["new_name_field_1","new_name_field_2,...]

CatchmentDB.name_format = "%s (%s)" ### number of format specifiers = number of name_fields

```
v = results.variables.s0,results.variables.ss
results[v,'dec 2010',catchments.by_name.Lachlan_Gunning()].spatial(interpolation=None) #interpolation="bilinear")
```



Show location of Catchment

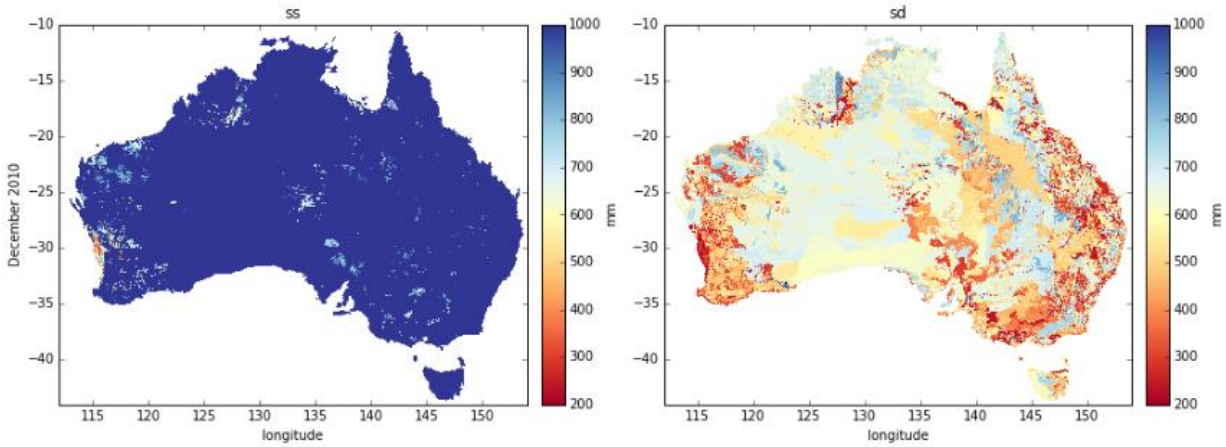
```
vis.show_extent(catchments.by_name.Lachlan_Gunning())
vis.show_extent(catchments.by_name.Lachlan_Gunning(),vis.extents.from_boundary_coords(-40,142,-30,154))
```

4.2.5. List of catchments in default shapefile "*awrams/utils/data/Final_list_all_attributes.shp"*

```
catchments.list()
```

4.2.6. Manipulating matplotlib settings

```
v = results.variables.ss,results.variables.sd
results[v,'dec 2010',:].spatial(clim=(200,1000),xlabel="longitude")
```



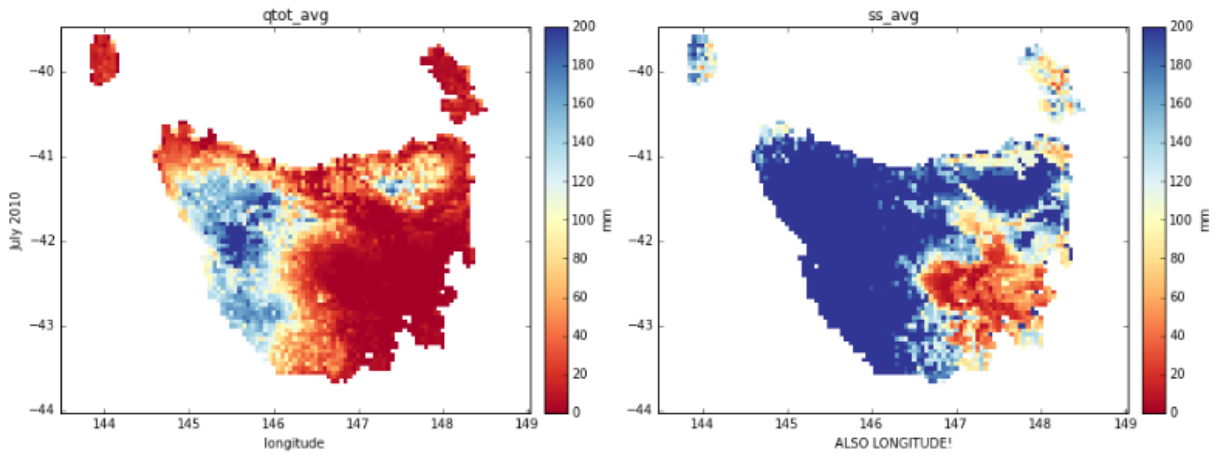
4.2.7. Accessing the underlying axes

Get the range of data for the selection

```
q = results[v,'dec 2010',vis.extents.from_boundary_coords(-39.5,143.5,-44,149)]
q.get_data_limits()
```

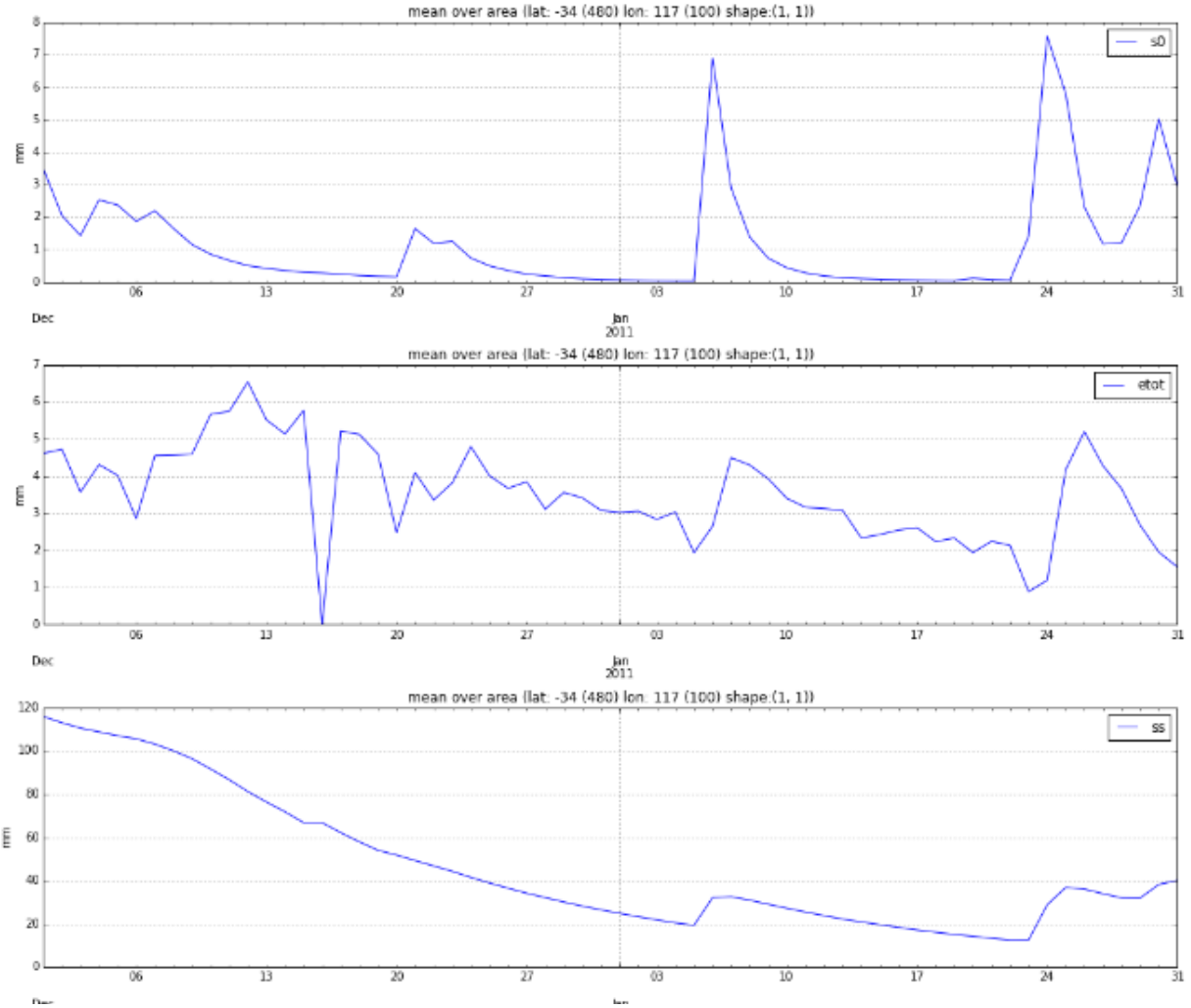
Set colour range limits and horizontal axis labels

```
q.spatial(clim=(0,200),xlabel="longitude")


gridview = q.mpl
view = gridview.children[0,1]


view.ax.set_xlabel("ALSO LONGITUDE!")
vis.plt.show()
```



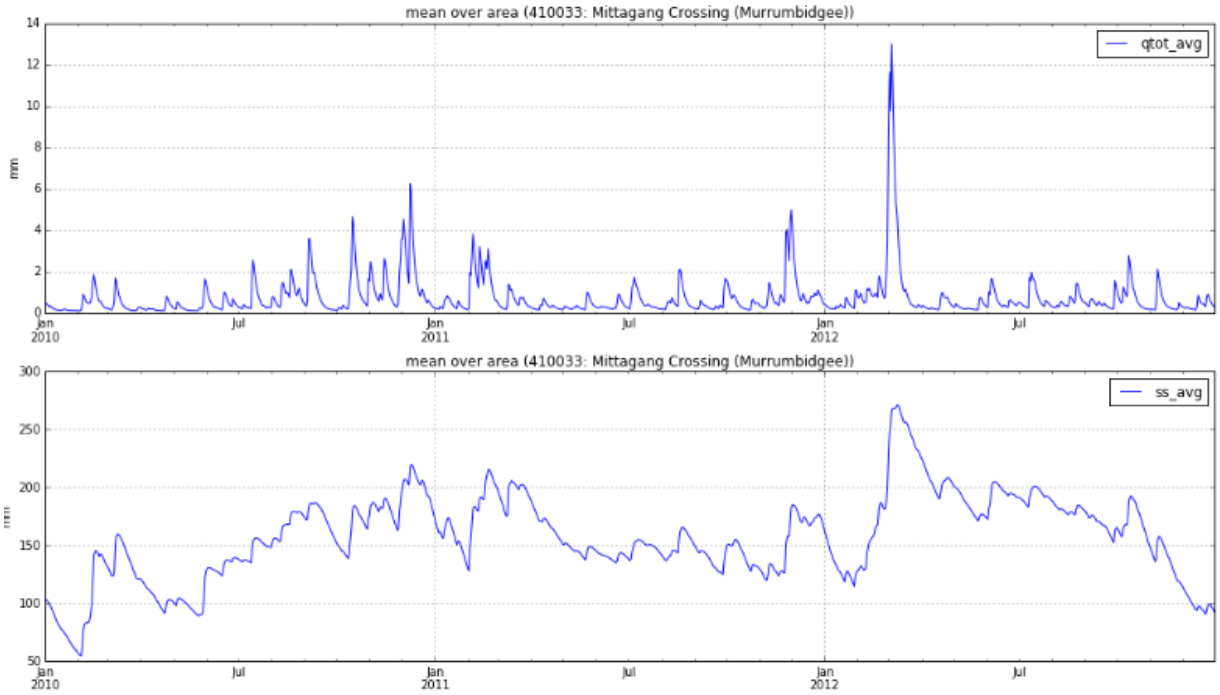
4.3. Display results as time-series

```
p = 'dec 2010 - jan 2011'
e = vis.extents.from_cell_coords(-34,117)
results[:,p,e].timeseries()
```



4.3.1. Aggregate over catchment and display as timeseries

```
v = results.variables.qtot,results.variables.ss
p = 'dec 2010 - jan 2011'
e = catchments.by_name.Murrumbidgee_MittagangCrossing()
results[v,p,e].timeseries()
```

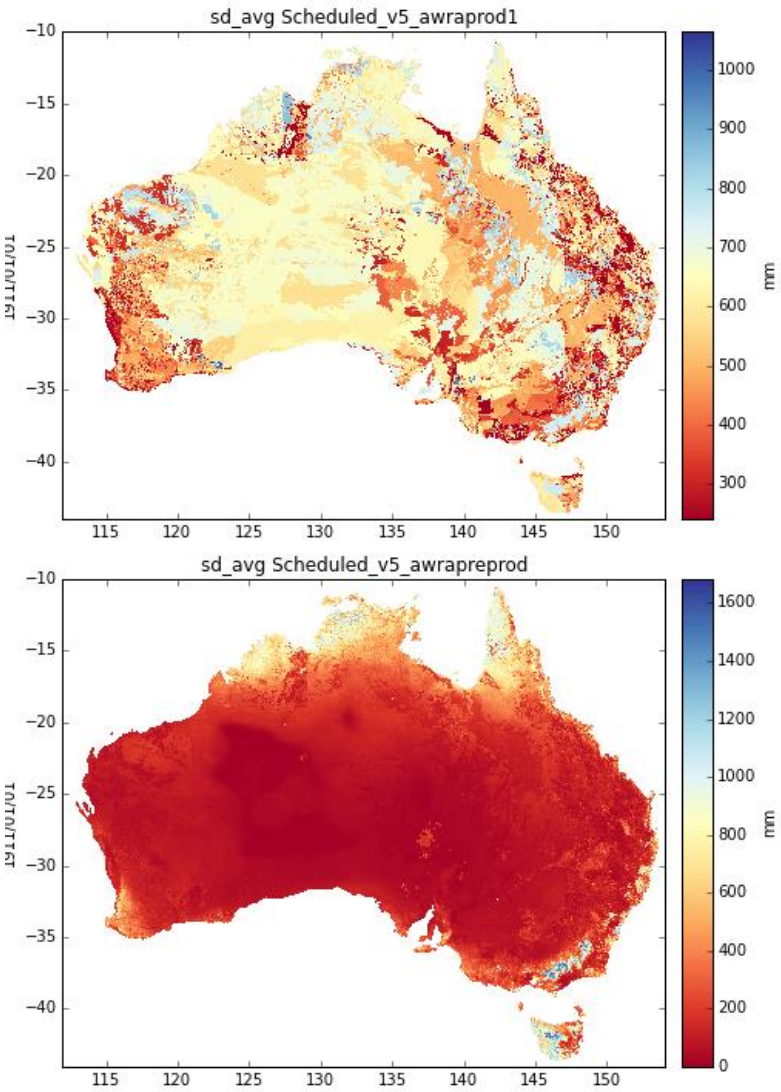



```
results.variables.qtot_avg.data.shape,results.variables.qtot_avg.agg_data.shape
```

4.3.2. Comparing results from different simulations

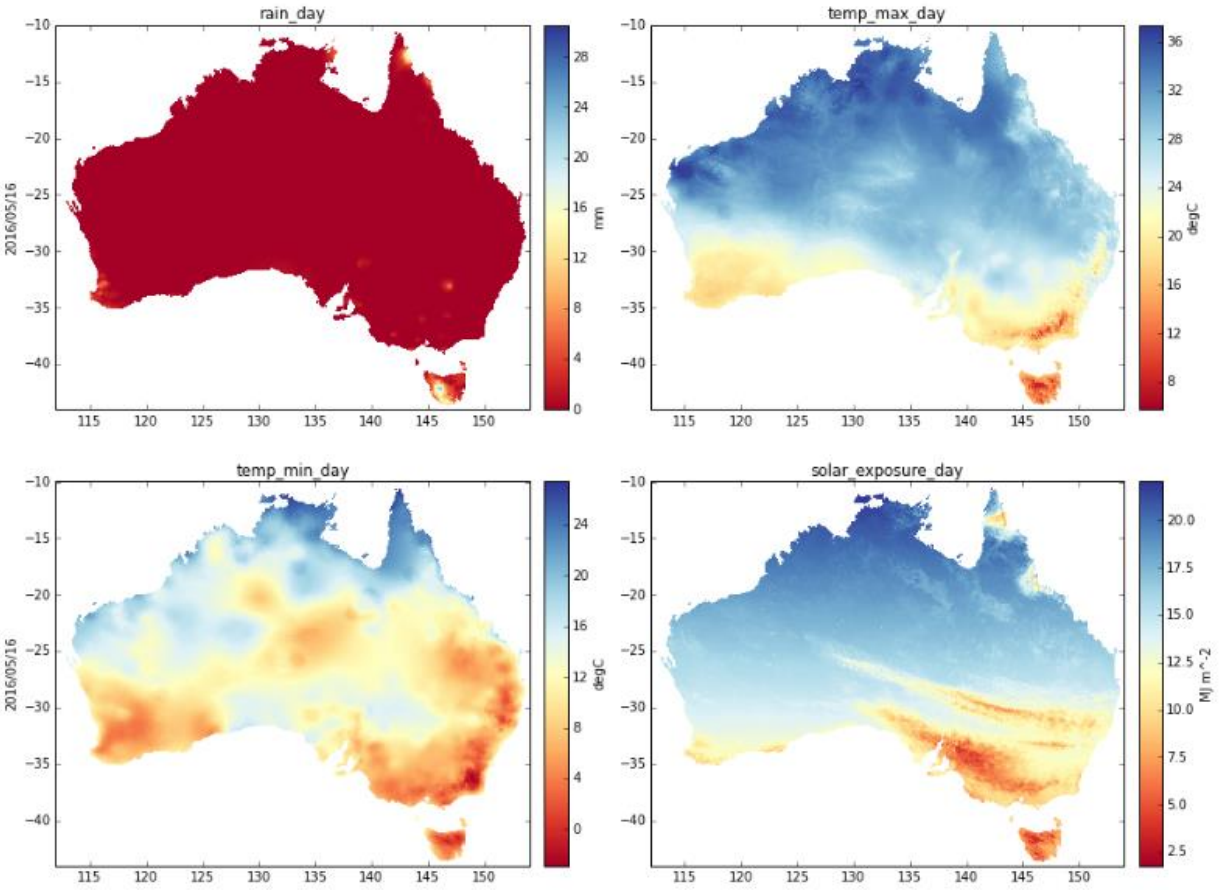
This Notebook is example only at is relies on two different simulations having been run which is not currently available from the test data provided. Once the User has two simulations the paths can be simply replaced and the notebook will run.

```
r1 = res.load_results('/data/cwd_awra_data/awra_test_outputs/Scheduled_v5_awraprod1_bu/')
r2 = res.load_results('/data/cwd_awra_data/awra_test_outputs/Scheduled_v5_awrapreprod/')


v = r1.variables.sd_avg,r2.variables.sd_avg
p = '1 jan 1911'
e = vis.extents.from_cell_coords(-34,117)
r1.name = 'Scheduled_v5_awraprod1'
r2.name = 'Scheduled_v5_awrapreprod'
vis.spatial(v, period='1 jan 1911') #,clim=(0,1000))
```



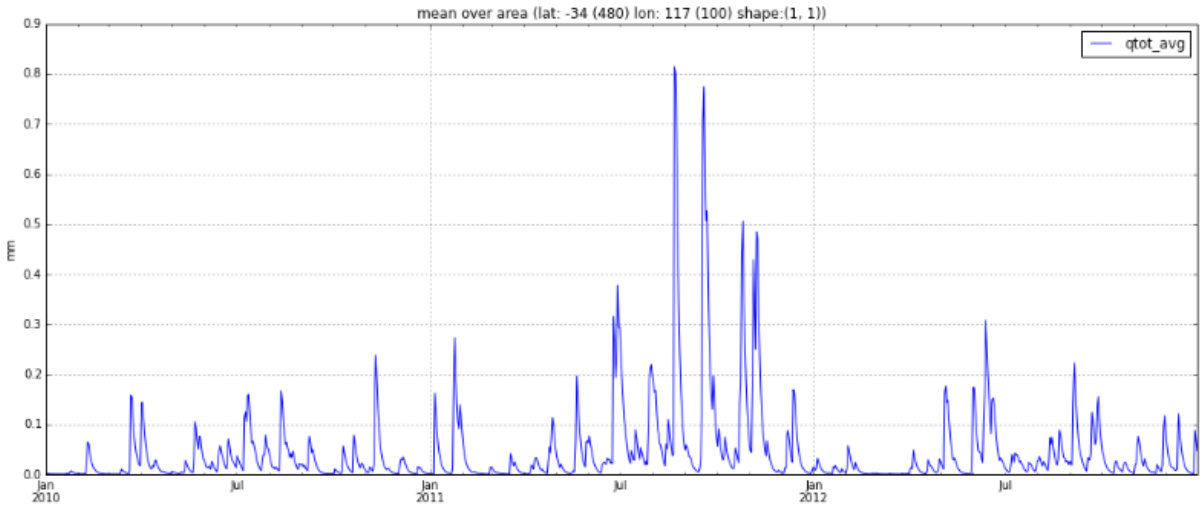
4.4. Visualising climatic inputs

```
precip = res.load_results('/data/cwd_awra_data/awra_inputs/climate_generated_awrapreprod/rr/')
tmax = res.load_results('/data/cwd_awra_data/awra_inputs/climate_generated_awrapreprod/tmax/')
tmin = res.load_results('/data/cwd_awra_data/awra_inputs/climate_generated_awrapreprod/tmin/')
solar = res.load_results('/data/cwd_awra_data/awra_inputs/climate_generated_awrapreprod/solar/')


v = precip.variables.rain_day,tmax.variables.temp_max_day,tmin.variables.temp_min_day,solar.variables.solar_exposure_day
vis.ROW_FIELD = None
vis._layout.MAX_COLUMNS = 2
vis.spatial(v, period='16 may 2016')
```



4.4.1. Time-series matplotlib settings

```
v = r1.variables.qtot_avg
e = vis.extents.from_cell_coords(-34,117)
p = '2010-2012'

q = r1[v,p,e]
q.timeseries()
```




```
q.timeseries() #ylim=(0,20))
ax = q.mpl.children[0,0].ax
lines = ax.get_lines()
line = lines[0]
line.set_color('m')
leg = ax.get_legend()
legline = leg.get_lines()[0]
legline.set_color('m')
```

5. Model Calibration

As a conceptual model, AWRA-L includes a large number of parameters whose values are not easily prescribed and which may vary between catchments and modelling periods. The values of these parameters need to be calibrated. This is done in an automatic procedure by comparing modelled and observed responses and seeking to minimise the differences between them. In the AWRA CMS, AWRA-L can be calibrated to multiple response series including observed streamflow together with remotely sensed surface soil moisture and actual evapotranspiration. Only test soil moisture and ET datasets are provided with the AWRA CMS package and full version 5 calibration and benchmarking data is available to registered users.

5.1. The calibration datasets

5.1.1. Streamflow

Zhang et al. (2013) collated streamflow for a set of Australian catchments that are amenable to catchment modelling. 295 of these catchments, with areas less than 5000km^2, have been randomly nominated as calibration catchments, with 291 nominated as validation catchments (Viney et al., 2015).

5.1.2. Evaporation

CMRSET : CSIRO MODIS reflectance-based Scaling ET (Guerschman et al., 2009), which uses monthly Enhanced Vegetation Index and Global Vegetation Moisture Index derived from MODIS to scale Priestley-Taylor potential evapotranspiration derived from meteorological surfaces. The monthly values were disaggregated to daily for the calibration catchments according to PET estimates across the month.

5.1.3. Soil Moisture

AMSR-E: Vrije Universiteit Amsterdam (VUA)-NASA product (Owe, et al. 2008) derived from passive Advanced Microwave Scanning Radiometer for the Earth Observing System aboard the Aqua polar orbiting satellite, covering 20/06/2002-01/10/2011. The data were resampled (using nearest-neighbour interpolation) from their original 0.25° resolution to the AWRA-L modelling grid of 0.05° resolution for Australia. The units are volumetric soil moisture.

5.2. Model Parameters

AWRA-L v5.0 contains 49 notionally optimisable parameters. However, not all these parameters are typically calibrated. Sensitivity studies have shown that streamflow and other hydrological and vegetation responses are relatively insensitive to some model parameters (Viney et al., 2015). The values of these parameters can therefore be fixed to reasonable prescribed values. The time and computing resources required for calibration are substantially reduced when fewer parameters are calibrated. In AWRA-L v5.0, only 21 of the optimisable

parameters are typically calibrated, while the remaining 28 are fixed – see Frost, Ramchurn and Hafeez (2016) for further details.

5.3. Optimisation Methods/Algorithms and Parameters

The AWRA-L calibration implementation in AWRA CMS is focussed on a framework enabling computational efficiency via parallel processing of large datasets (e.g. on NCI), by simulating points/catchments independently of one another (on differing nodes). This approach is open to implementation of any particular optimisation algorithm or objective function preferred by the user;

The system is by default initially able to optimise parameters for runoff, against observations from a series of catchments used in the WIRADA research. However the system is amenable to optimisation against other observed data such as surface soil moisture or evapotranspiration. The code is modular and allows expert users to implement other optimization strategies within the existing parallelization framework. AWRA-L uses a message brokering system that allows a user specifiable number of computational nodes on high performance computing architectures.

AWRA CMS uses the Shuffled Complex Evolution (SCE) algorithm. Each complex in a shuffled population (specified by n_complexes in the ShuffledOptimizer constructor), is launched in its own process. Simple implementations of AWRA CMS evaluators will therefore by parallelized by the number of complexes.

5.4. Objective function

Default national calibration is applies a single set of parameters simultaneously on all 295 calibration catchments and for each catchment we calculate the objective function

$$F = (E_d + E_m)/2 - 5 \left| \ln(1 + B) \right|^{2.5}$$

where E_d and E_m are respectively the Nash-Sutcliffe efficiencies of daily and monthly streamflow, and B is the bias (total prediction error divided by total observed streamflow) – see Viney et al (2009). The final grand objective function (collating all catchment/point based results) is then taken as the mean of the 25th, 50th, 75th and 100th percentiles of the F values of the 295 calibration catchments. This objective function value is maximised in calibration.

For further details of the model, parameters and calibration approach see Viney et al (2015) and Frost, Ramchurn and Smith (2016).

5.5. Basic Calibration Procedure

An example of how to calibrate AWRA-L is outlined in the Calibration_SCE.ipynb userman notebook.

5.5.1. Setup

A small test set of data is provided with the AWRA CMS package to demonstrate the calibration procedure. The data to complete the calibration of a single catchment in the AWRA CMS is provided in /calibration/tests/data. The following steps are defined for the setup of a model calibration.

1. Import necessary packages and modules

```
# AWRA MS calibration modules
import awrams.calibration.calibrate as cal
from awrams.calibration.sce import SCEOptimizer,ProxyOptimizer

# The AWRA-L model
from awrams.models import awral

# AWRA MS Utility modules
from awrams.utils import datetools as dt
from awrams.utils import catchments
from awrams.utils.nodegraph import nodes

# General
import pandas as pd
import os
from matplotlib import pyplot
```

2. Set period for calibration

```
period = dt.dates('1990 - 1995')
period
```

3 Set spatial extent

```
# Define calibration catchment
cal_catchment= '421103'

# Get the catchment as a spatial extent either from the inbuilt catchment file or load it from a pkl file
from awrams.utils import catchments

try:
    db = catchments.CatchmentDB()
    extent = db.get_by_id(cal_catchment)

except ImportError as e:
```

```
    print(e)
    # read catchment extent from a pickle
    import pickle
    path = os.getcwd()
    pkl = os.path.join(path,'..','tests','data','extent_421103.pkl')
    extent = pickle.load(open(pkl,'rb'))

extent.cell_count
```

4. Setup input map for climate inputs

```
# View default calibration input map cal.input_map

for k in cal.input_map.mapping:
    print(k,cal.input_map.mapping[k])
```

```
def change_path_to_forcing(imap):
    from awrams.utils.nodegraph import nodes

    path = os.getcwd()
    path = os.path.join(path,'..','tests','data')

    FORCING = {
        'tmin'  : ('temp_min*','temp_min_day'),
        'tmax'  : ('temp_max*','temp_max_day'),
        'precip': ('rain_day*','rain_day'),
        'solar' : ('solar*'   ,'solar_exposure_day')
    }
    for k,v in FORCING.items():
        imap.mapping[k+'_f'] = nodes.forcing_from_ncfiles(path,v[0],v[1],cache=True)

change_path_to_forcing(cal.input_map)
```

5. Load the observed streamflow data

```
path = os.getcwd()
csv = os.path.join(path,'..','tests','data','q_obs.csv')
qobs = pd.read_csv(csv,parse_dates=[0])
qobs = qobs.set_index(qobs.columns[0])
```

```
obs = qobs[cal_catchment]
```

6. Find all parameters in the input map that are able to be calibrated map

```
# This calibrates all of them, but you could equally create a subset...
parameters = cal.get_parameter_df(cal.input_map.mapping)
parameters
```

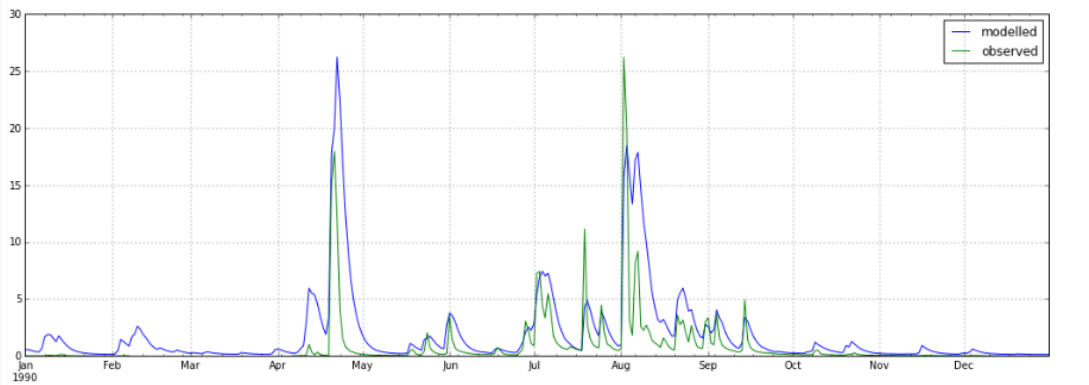
5.5.2. Model evaluator

1 Create the model evaluator

```
evaluator = cal.RunoffEvaluator(period,extent,obs)
```

2 Plot evaluator initial results

```
evaluator.plot(evaluator.initial_results,period=dt.dates('1990'))
```



5.5.3. Optimization

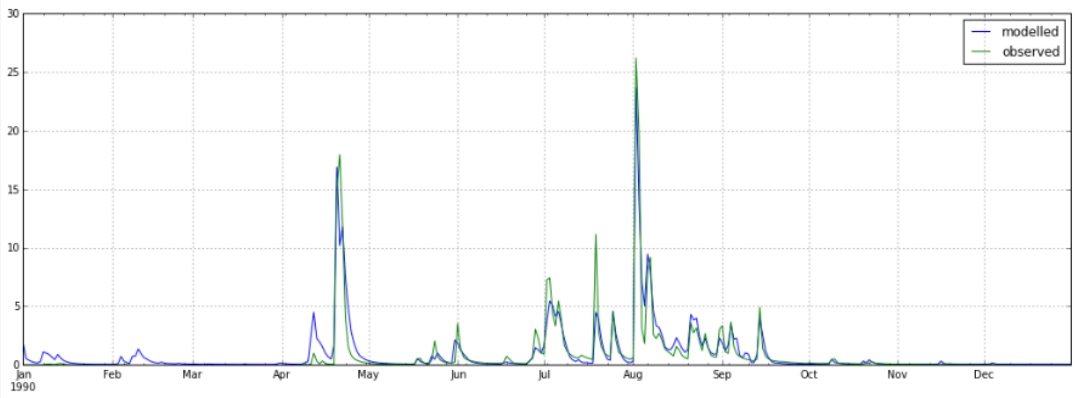
1 Create the SCE instance

```
sce = ProxyOptimizer(13,5,4,3,3,parameters,evaluator)
```

2 Run basic optimizer

```
sce.run_optimizer()
```

```
# View sce population
sce.population.iloc[0]
evaluator.plot(evaluator.run_sim(sce.population.iloc[0]),period=dt.dates('1990'))
```



3 Run optimizer with seed population

```
sce.run_optimizer(seed=sce.population.iloc[0])
```

5.5.4. Cleanup the workspace

It is important to terminate the SCE worker processors at the end of a calibration session. The SCE worker processors don't get destroyed automatically when the sce finishes, that way if the calibration is restarted from a seed population the processors still exist.

5.5.5. Optimising the model with PEST

The procedure described so far is focused on using the AWRA CMS SCE optimizer but the AWRA CMS is designed to enable the use of model-independent parameters estimation systems such as PEST. The details of this capability are described in the Calibration_PEST.ipynb provided with the AWRA CMS package.

6. Benchmarking

A suite of Benchmarking tools are available within the AWRA CMS. Benchmarking a model run enables the user to confirm that the model is performing to expectations relative to observations and other models. A small set of example datasets are provided with this package but the benchmarking tools are designed to work with comparing any observation dataset with the relevant AWRA-L variable. The benchmarking data and methods used to evaluate the AWRA-L model are detailed in Frost, Ramchurn and Hafeez (2016). In that document, the following national datasets have been compared with the relevant AWRA-L variables for the latest AWRA-L model version. Subsets of these datasets, with data for only ten catchments, have been provided with the AWRA CMS package to demonstrate the tools. The benchmarking of the model over these ten catchments is presented in a series of notebooks relevant to each dataset, with just the runoff presented here.

AWRA-L Variable	**Description**	**AWRA CMS Dataset**	**Demo Notebook**
Runoff (Qtot)	Observed streamflow: 786 catchment (Zhang et al., 2013)	runoff	BenchmarkingDemo_QObs
Actual Evapo-transpiration (Etot)	Satellite derived data: CMRSET(Guerschman et al., 2009) Flux tower data: OzFlux (Beringer et al., 2016)	cmrset ozflux	BenchmarkingDemo_CMRS_ET BenchmarkingDemo_OzFlux_ET
Upper Soil Moisture (S0)	Satellite derived soil moisture data: AMSRE (Owe et al., 2008) ASCAT (Bartalis et al., 2007)	amsre ascat	BenchmarkingDemo_wunsat_AMSRE BenchmarkingDemo_wunsat_ASCAT
Surface (S0) and Shallow (SS) Soil Moisture	Soil moisture monitoring networks: SASMAS (Rüdiger et al., 2007) OzNET (Smith et al., 2012)	sasmas oznet	BenchmarkingDemo_SM_SASMAS BenchmarkingDemo_SM_OZNET

The User is alerted to the fact that the soil moisture benchmarking components (SASMAS and OzNET may further necessitate some configuration settings and metadata adjustment, if for example the file naming convention is different or the observed data is to be modified. The relevant files are located at the following locations: awrams_cm\benchmarking\awrams\benchmarking\config.py, and the folder awrams_cm\benchmarking\awrams\benchmarking\meta

6.1. Input data

6.1.1. Observation data format

The benchmarking system is designed with the flexibility to compare multiple datasets in the simple format of a csv format. The observation data just needs to be arranged according to an id and date and there can be gaps in the data

	432157	42434	462374	426258	167210	439704
31/10/2001	76.7064	107.8738			31.7719	
30/11/2001	94.98	114.063			38.598	85.284
31/12/2001	110.4778	138.5948			72.6547	

6.1.2. AWRA-L data

AWRA-L data needs to be extracted at the desired location for comparison with the observed datasets. The User can run the extraction commands detailed in Section 7.2 to extract the AWRA-L data from the netcdf grids output generated by a Simulation.

6.2. Benchmarking Example: Qtot and Streamflow Observations

6.2.1. Setup

1. Import the necessary packages

```
from awrams.benchmarking import Benchmark
from awrams.benchmarking.utils import read_id_csv
from awrams.utils import datetools as dt
import awrams.benchmarking
awrams.benchmarking.__file__
```

'/data/cwd_awra_data/AWRAMSI/IWRM_0042_WP3/GIT/aoke/community_model/git_lab/awrams_cm/benchmarking/awrams/benchmarking/__init__.py'

2. Define observed data and parameters

Compare AWRA-L runoff (Qtot) against observed streamflow

```
obs_csv = '../tests/data/runoff/q_obs.csv'
catchment_csv = '../tests/data/catchment_ids.csv'
```

```
id_list=read_id_csv(catchment_csv)
```

The command below creates and initialises a "q" benchmark object which will contain all the data and the statistics pertaining to the datasets to be included in the benchmarking.

```
q = Benchmark("QObs", "runoff")
```

The main settings (period, catchments) for the benchmarking can then be modified as per the examples below.

```
q.period = dt.dates("1981", "30/12/2011")
q.load(obs_csv, id_list=id_list)
```

The list of sites being benchmarked can also be checked.

```
q.sites
```

```
['116008',
 '105001',
 '107002',
 '108003',
 '113004',
 '112102',
 '109001',
 '4508',
 '111101',
 '5115']
```

3. Add models to the comparison

Time series of modelled data in csv format, with the column names being the catchment gauge number (which have to be elements of the full list of calibration/validation catchments) can then be added to the benchmarking. In this case, the data originates from pre-processed AWRA-L data extracted from the simulation grids at the observation locations using the AWRA CMS extract tool.

```
csv_data='../tests/data/runoff/awral_qtot_avg.csv'
q.add_model("AWRAMSI_v4_0_AWRAL", data_csv=csv_data)
```

4. Show list of loaded or selected models

List of loaded models is available with activated dropdown by typing "q.models."
Can "select" or "unselect" models for displaying

```
q.benchmark.selection
```

```
['AWRAMSI_v4_0_AWRAL']
```

```
q.benchmark.selection.AWRAMSI_v4_0_AWRAL.unselect()
q.benchmark.selection.AWRAMSI_v4_0_AWRAL.select()
```

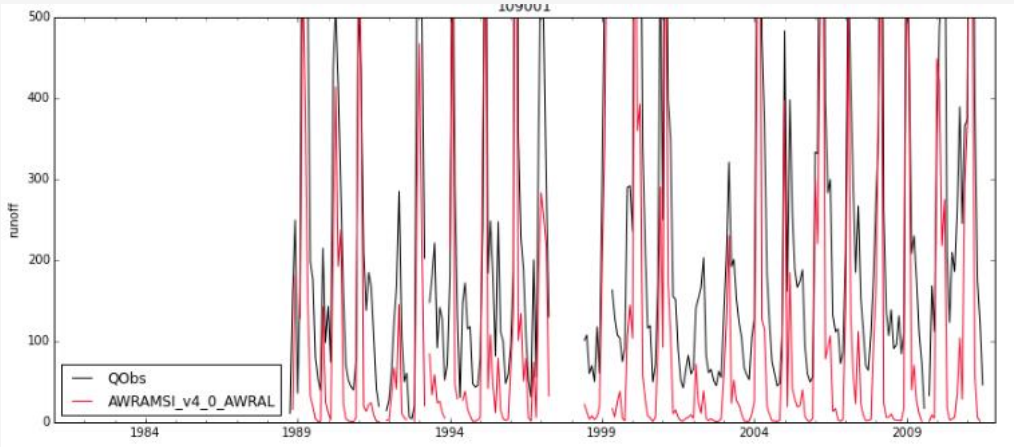
6.2.2. Plotting

1. Time-series

Display the modelled and observed flow for benchmarked locations as a time-series. The frequency can be specified by "freq=d" for daily, "freq=m" for monthly, "freq=y" for yearly

The titles, labels, scales etc can all be customised using standard matplotlib keyword arguments similar to the visualisation tools.

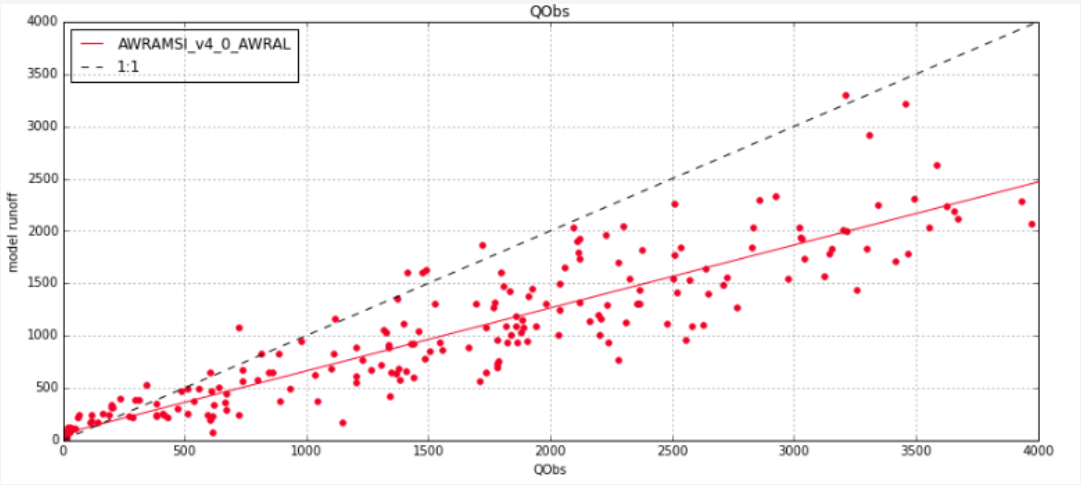
```
p = q.benchmark.plot_timeseries('109001',freq='m',ylim=[0,500])
```

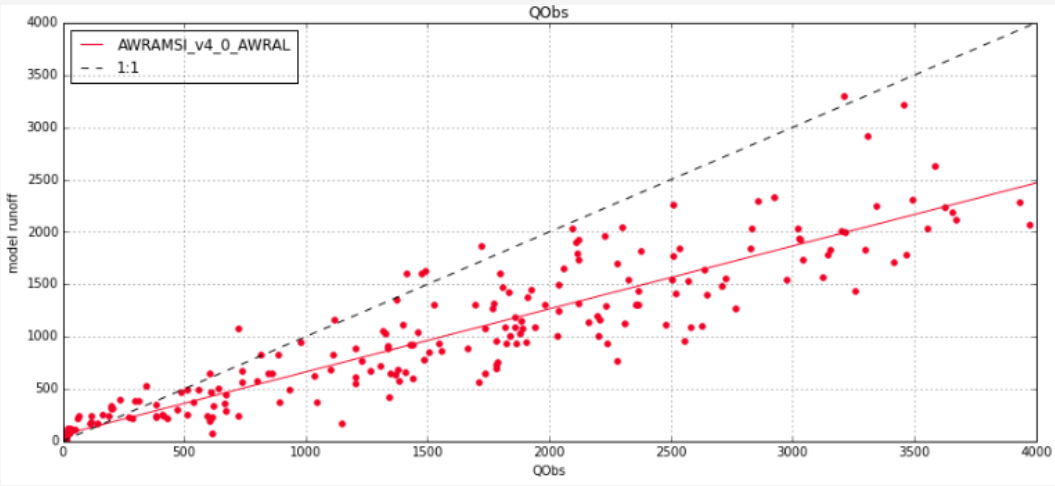


2. Regression

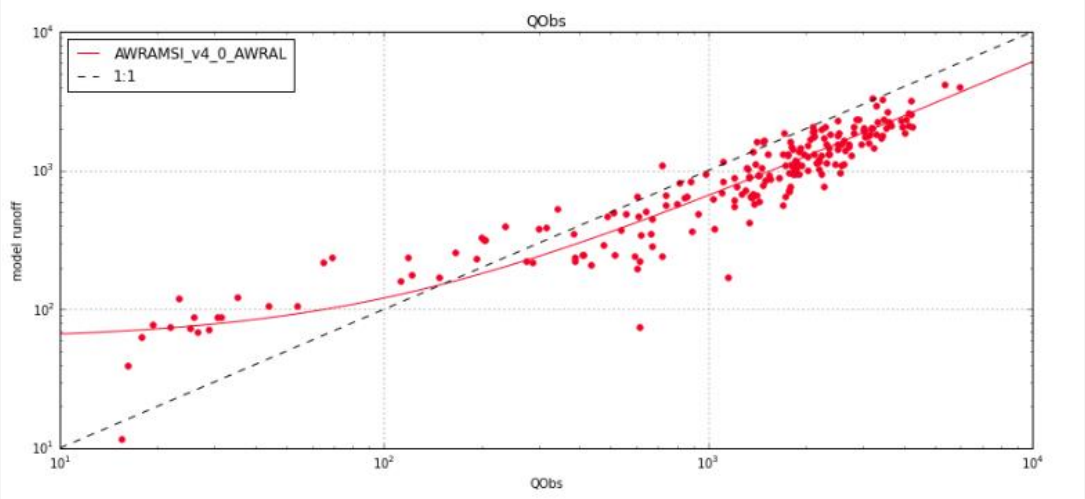
Regress the each of the models daily, monthly or annually aggregated data against the observed values. If more than one model present it will plot in a different colour on the same graph.

```
p = q.benchmark.plot_regression(title="QObs", freq='y', xlim=[0,4000],ylim=[0,4000])
```





p = q.benchmark.plot_regression(title="QObs", freq='y', yscale='log', xscale='log', ylim=[10,10000], xlim=[10,10000])



6.2.3. Statistics

A full explanation of the benchmarking statistics used to assess model performance is provided in Frost, Ramchurn and Hafeez (2016). In summary the following statistics are built into the AWRA CMS benchmarking statistics and are used to evaluate AWRA-L.

Relative bias (B)

$$B_i = \sum_{t=1}^{T_i} \frac{Q_o^{ti} - Q_m^{ti}}{\overline{Q}_o^i}$$

Monthly and daily Nash-Sutcliffe Efficiency (NSE) and

$$NSE_i = 1 - \frac{\sum_{t=1}^{T_i} \left(Q_o^{ti} - Q_m^{ti} \right)^2}{\sum_{t=1}^{T_i} \left(Q_o^{ti} - \bar{Q}_o^{i} \right)^2}$$

Pearson's correlation coefficient (r)

$$r = \frac{\sum_{t=1}^{T_i} (Q_o^{ti} - \overline{Q_o^{i}})(Q_m^{ti} - \overline{Q_m^{i}})}{\sqrt{\sum_{t=1}^{T_i} (Q_o^{ti} - \overline{Q_o^{i}})^2} \sqrt{\sum_{t=1}^{T_i} (Q_m^{ti} - \overline{Q_m^{i}})^2}}$$

where Q_o^{ti} and Q_m^{ti} are the observed and modelled value for site i and time step t, for a total of T_i available observations, and $\overline{Q_o^{i}}$ is the mean observed and $\overline{Q_m^{i}}$ the modelled data for site *i*.

The bias and monthly NSE statistics in particular are seen as good metrics for judging the models performance for AWRA-L purposes. Pearson's correlation coefficient is a good indicator for variables where the bias (and absolute value) of the variable is not as important as matching the variability (e.g. soil moisture and actual ET).

The objective function can also be assessed through the benchmarking package by calling ‘grand_f’ (Section 5.4).

Summary percentiles can be printed out by specifying a statistic to the 'stat_percentiles' function.

By default the timeframe used for statistics is monthly, but can be specified

Objective function (grand_f)

```
q.benchmark.stat_percentiles('grand_f', freq='daily')
```

	grand_f
AWRAMSI_v4_0_AWRAL	0.015785

```
q.benchmark.stat_percentiles('grand_f', freq='monthly')
```

	grand_f
AWRAMSI_v4_0_AWRAL	0.079414

Nash-Sutcliffe Efficiency (nse)

```
q.benchmark.stat_percentiles('nse')
```

	0%	5%	25%	50%	75%	95%	100%
AWRAMSI_v4_0_AWRAL	0.271805	0.382997	0.628121	0.677263	0.834034	0.898544	0.914612

Relative bias (bias_relative)

```
q.benchmark.stat_percentiles('bias_relative')
```

	0%	5%	25%	50%	75%	95%	100%
AWRAMSI_v4_0_AWRAL	-0.534929	-0.507981	-0.45654	-0.322629	-0.096231	1.389183	1.944231

Pearson's correlation coefficient (pearsons_r)

```
q.benchmark.stat_percentiles('pearsons_r',freq='d')
```

	0%	5%	25%	50%	75%	95%	100%
AWRAMSI_v4_0_AWRAL	0.605472	0.676093	0.82781	0.857553	0.908226	0.934009	0.934931

Compare raw observations and modelled data

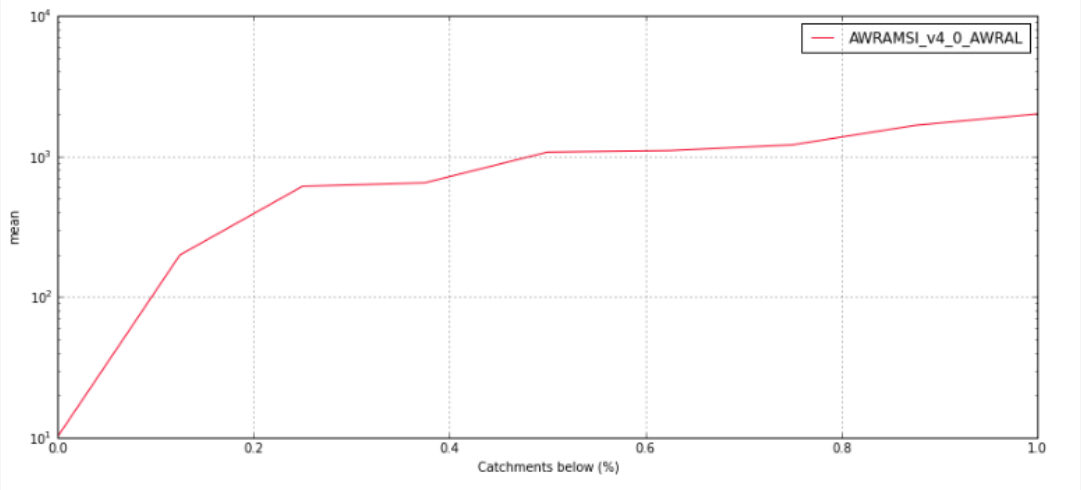
```
q.benchmark.data_percentiles(freq='d')
```

	0%	5%	25%	50%	75%	95%	100%
QObs	0.007468	0.033215	0.817031	3.959605	5.534223	8.695649	9.399966
AWRAMSI_v4_0_AWRAL	0.027104	0.253661	1.790567	3.053136	3.642913	5.440798	5.887065

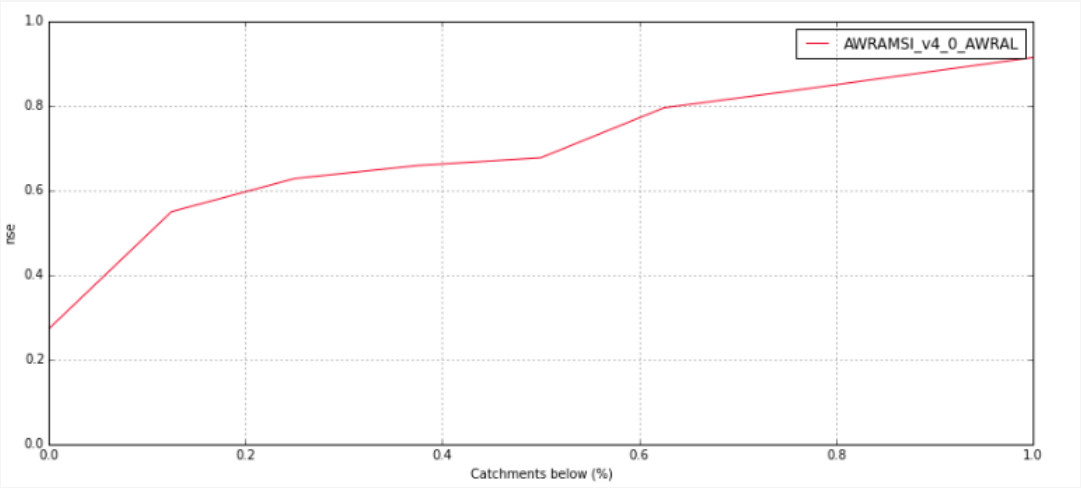
6.2.4. Statistics plotting

A particularly useful feature of the AWRA CMS benchmarking package is the inbuilt statistics plotting functions where the user just has to specify the statistic and frequency but all of the normal plot elements can still be customised.

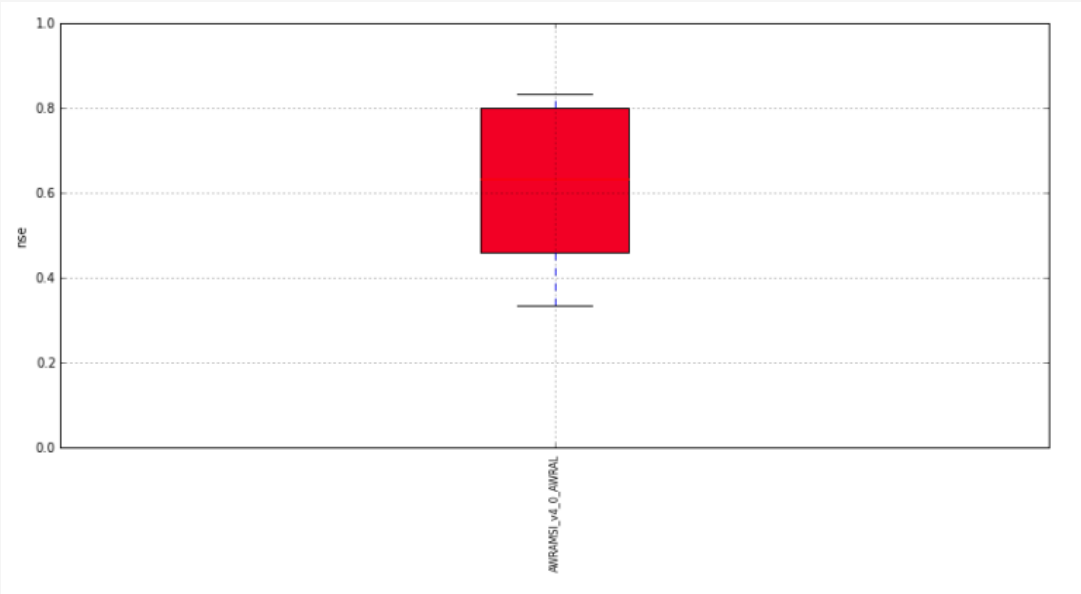
```
p = q.benchmark.plot_cdf('mean', freq='y', yscale='log')
```



p = q.benchmark.plot_cdf('nse',freq='m', ylim=[0,1])



p = q.benchmark.plot_box('nse', freq='d', ylim=[0,1])



7. General Utility Tools of the AWRA CMS

A part of dealing with gridded time series data, some useful functionality for data extraction and aggregation has been in included in the package.

Example applications of those tools have been provided in notebooks located in the folder: awrams_cm\utils\notebooks

- Extraction of time series from grids (over Catchments, at points)
- Temporal aggregation of netcdf outputs

7.2. Extract time-series from grid

This section follows the commands listed in the awrams_cm\utils\notebooks\XXX.ipynb notebook.

7.2.1. Extraction of extents from netcdf dataset

```
from awrams.utils.extract import extract,localise_extent_to_ncfile
from awrams.utils.ts.gridded_time_series import TimeSeriesDataSet
import awrams.utils.ts.processing as processing
from awrams.utils.io.general import h5py_cleanup_nc_mess

from awrams.utils.catchments import CatchmentDB
catchments = CatchmentDB()

import awrams.utils.datetools as dt

import os

# Plotting
from matplotlib import pyplot as plt
%matplotlib inline
```

7.2.2. Extract and spatially aggregate catchments

```
var_name = 'rain_day'

path = os.getcwd()
path = os.path.join(path,'..','..','calibration','tests','data')
```

```
pattern = path + '/%s*' % var_name


period = dt.dates('jul 1990 - jun 1995')


df = extract(var_name,
             pattern,
             {'204007':localise_extent_to_ncfile(catchments.get_by_id('204007')),
              '421103':localise_extent_to_ncfile(catchments.get_by_id('421103'))},
             period)
df
```

	204007	421103
1990-07-01	1.074465	8.188409
1990-07-02	5.240235	16.567407
1990-07-03	0.347165	11.315883
1990-07-04	0.262357	6.869347
..	..	..
1995-06-29	0.975766	8.172905
1995-06-30	0.025161	2.418514

1826 rows × 2 columns

7.2.3. Extract a catchment

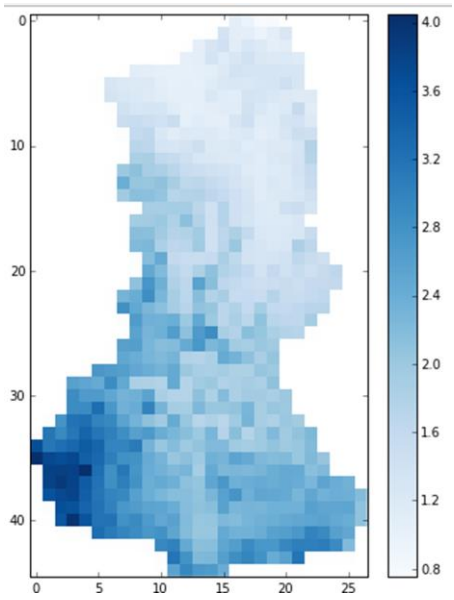
```
period = dt.dates('jul 1990')
### requires osgeo.ogr to process shapefiles
catchment = catchments.get_by_id('204007')
catchment.cell_count
```

729

```
path = os.getcwd()
path = os.path.join(path,'..','..','calibration','tests','data')


pattern = path + '/rain_day*'


ds = TimeSeriesDataSet('rain_day',pattern)
data = ds.get_data(period,catchment)
ds.close_all()
# h5py_cleanup_nc_mess(show_log=True)


data.shape,catchment.cell_count
```

((31, 45, 27), 729)

```
plt.figure(figsize=(6,8))
im = plt.imshow(data.mean(axis=0),interpolation='None',cmap=plt.get_cmap('Blues'))
plt.colorbar(im)
```



7.3. Create aggregated output netcdfs

This section follows the commands listed in the awrams_cm\utils\notebooks\XXX.ipynb notebook

```
import awrams.utils.aggregate_time as agg
import awrams.utils.datetools as dt


import os
```

7.3.1. Daily to monthly

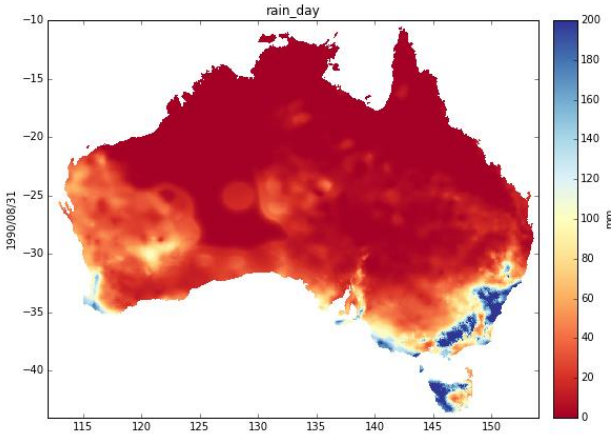
```
var_name = 'rain_day'


# path = os.getcwd()
# path = os.path.join(path,'..','..','calibration','tests','data')
# pattern = path + '/%s*' % var_name
pattern = '/data/cwd_awra_data/awra_inputs/climate/rr/rr_*nc'


var_map = {var_name:pattern}
```

```
out_path = './_results'

period = dt.dates('jan 1990 - dec 1995')

to_freq = 'M'

agg.process(var_map,out_path,period,to_freq,method='sum')
```

```
from awrams.utils.io.data_mapping import SplitFileManager
fm = SplitFileManager.open_existing(out_path,'rain_day.nc','rain_day')
fm.time_file_map
```

```
{'./_results/rain_day.nc': time (days since 1900-01-01 00:00:00) :
 PeriodIndex(['1990-01', '1990-02', '1990-03', '1990-04', '1990-05', '1990-06',
              '1990-07', '1990-08', '1990-09', '1990-10', '1990-11', '1990-12',
              '1991-01', '1991-02', '1991-03', '1991-04', '1991-05', '1991-06',
              '1991-07', '1991-08', '1991-09', '1991-10', '1991-11', '1991-12',
              '1992-01', '1992-02', '1992-03', '1992-04', '1992-05', '1992-06',
              '1992-07', '1992-08', '1992-09', '1992-10', '1992-11', '1992-12',
              '1993-01', '1993-02', '1993-03', '1993-04', '1993-05', '1993-06',
              '1993-07', '1993-08', '1993-09', '1993-10', '1993-11', '1993-12',
              '1994-01', '1994-02', '1994-03', '1994-04', '1994-05', '1994-06',
              '1994-07', '1994-08', '1994-09', '1994-10', '1994-11', '1994-12',
              '1995-01', '1995-02', '1995-03', '1995-04', '1995-05', '1995-06',
              '1995-07', '1995-08', '1995-09', '1995-10', '1995-11', '1995-12'],
             dtype='int64', freq='M')}
```

```
import awrams.visualisation.results as res
results = res.load_results('./_results')
results[:,'aug 1990',:].spatial(clim=(0,200))
```



7.3.2. Daily to annual

```
var_name = 'rain_day'

# path = os.getcwd()
# path = os.path.join(path,'..','..','calibration','tests','data')
# pattern = path + '/%s*' % var_name
pattern = '/data/cwd_awra_data/awra_inputs/climate/rr/rr_*nc'

var_map = {var_name:pattern}

out_path = './_results'

period = dt.dates('jan 1990 - dec 1995')

to_freq = 'A'

agg.process(var_map,out_path,period,to_freq,method='sum')
```

```
from awrams.utils.io.data_mapping import SplitFileManager
fm = SplitFileManager.open_existing(out_path,'rain_day.nc','rain_day')
fm.time_file_map
```

```
{'./_results/rain_day.nc': time (days since 1900-01-01 00:00:00) :
 PeriodIndex(['1990', '1991', '1992', '1993', '1994', '1995'], dtype='int64', freq='A-DEC')}
```

```
import awrams.visualisation.results as res
results = res.load_results('./_results')
results[:,'1990',:].spatial(clim=(0,1000))
```

8. Contributing to AWRA CMS

One of the main motivations of the community model is to gain the benefits of the AWRA CMS community contributing improvements to the system. It is anticipated that research scientists and agencies will be interested in:

- altering model algorithms towards better performance,
- increasing functionality and
- adding new datasets to the system.

To make changes to the AWRA-L community model source code the User needs to create their own 'fork' (altered copy of the community model code) of the code on the GitHub repository.

After ensuring the robustness of the changes locally, developers are able to contribute to the community modelling system by proposing their alterations to the Bureau according to the following steps:

1. The User proposes changes to the Bureau
2. The Bureau assesses scientifically the proposed change. If successful the Bureau grants access for submission of fork/code.
3. User submits their fork (new code) and any documentation and testing that been conducted
4. The Bureau assesses that code against various performance criteria including performance against benchmark data, system performance, code complexity and maintainability. If successful according to that testing the Bureau releases the new version of the CMS.

The above methods will be refined according to the community feedbacks.

9. References

Bartalis, Z., Wagner, W., Naeimi, V., Hasenauer, S., Scipal, K., Bonekamp, H., Figa, J. and Anderson, C., 2007. Initial soil moisture retrievals from the METOP-A Advanced Scatterometer (ASCAT). Geophysical Research Letters, 34(20): L20401.

Beringer, J., Hutley, L. B., McHugh, I., Arndt, S. K., Campbell, D., Cleugh, H. A., Cleverly, J., Resco de Dios, V., Eamus, D., Evans, B., Ewenz, C., Grace, P., Griebel, A., Haverd, V., Hinko-Najera, N., Huete, A., Isaac, P., Kanniah, K., Leuning, R., Liddell, M. J., Macfarlane, C., Meyer, W., Moore, C., Pendall, E., Phillips, A., Phillips, R. L., Prober, S. M., Restrepo-Coupe, N., Rutledge, S., Schroder, I., Silberstein, R., Southall, P., Yee, M. S., Tapper, N. J., van Gorsel, E., Vote, C., Walker, J., and Wardlaw, T, 2016. An introduction to the Australian and New Zealand flux tower network – OzFlux, Biogeosciences, 13, 5895-5916.

Beringer, J., McHugh, I., Hutley, L.B., Isaac, P. and Kljun, N., 2016. Dynamic INtegrated Gap-filling and partitioning for OzFlux (DINGO). Biogeosciences Discuss., 2016: 1-36.

Duan, Q., Sorooshian, S. and Gupta, V. 1992. Effective and efficient global optimization for conceptual rainfall-runoff models. Water Resources Research, 28, 1015-1031.

Frost, A, Ramchurn, A. and Hafeez, M, 2016, Evaluation of the Bureau's Operational AWRA-L Model. Bureau of Meteorology Technical Report.

Frost, A, Ramchurn, A. and Smith, A., 2016. The Bureau's Operational AWRA-L Model: Technical description of AWRA-L version 5. Bureau of Meteorology Technical Report.

Guerschman, J.P., Van Dijk, A.I.J.M., Mattersdorf, G., Beringer, J., Hutley, L.B., Leuning, R., Pipunic, R.C. and Sherman, B.S., 2009. Scaling of potential evapotranspiration with MODIS data reproduces flux observations and catchment water balance observations across Australia. Journal of Hydrology, 369(1-2): 107-119.

Hafeez, M., Frost, A., Vaze, J., Dutta, D., Smith, A. and Elmahdi, A., 2015. A New Integrated Continental Hydrological Simulation System: An overview of the Australian Water Resource Assessment Modelling System (AWRAMS). Water: Journal of the Australian Water Association, 42(3): 75-82.

Owe, M., de Jeu, R. and Holmes, T., 2008. Multisensor historical climatology of satellite-derived global land surface moisture. Journal of Geophysical Research: Earth Surface, 113(F1): F01002.

Rüdiger, C., Hancock, G., Hemakumara, H.M., Jacobs, B., Kalma, J.D., Martinez, C., Thyer, M., Walker, J.P., Wells, T. and Willgoose, G.R., 2007. Goulburn River experimental catchment data set. Water Resources Research, 43(10): W10403.

Smith, A.B., Walker, J.P., Western, A.W., Young, R.I., Ellett, K.M., Pipunic, R.C., Grayson, R.B., Siriwardena, L., Chiew, F.H.S. and Richter, H., 2012. The Murrumbidgee soil moisture monitoring network data set. Water Resources Research, 48(7): W07701.

Viney, N., Vaze, J., Crosbie, R., Wang, B., Dawes, W. and Frost, A., 2015. AWRA-L v5.0: technical description of model algorithms and inputs, CSIRO, Australia.

Zhang, Y.Q., Viney, N., Frost, A., Oke, A., Brooks, M., Chen, Y. and Campbell, N., 2013. Collation of streamflow and catchment attribute dataset for 780 unregulated Australian catchments, CSIRO: Water for a Healthy Country National Research Flagship.

Appendix A. AWRA-L Parameters

Parameter	Description
alb	Surface albedo (dimensionless)
alb_dry	Dry soil albedo (dimensionless)
alb_soil	Albedo of soil surface (dimensionless)
alb_veg	Albedo of vegetated surfaces (dimensionless)
alb_wet	Wet soil albedo (dimensionless)
cGsmax	Coefficient relating vegetation photosynthetic capacity to maximum stomatal conductance (m s^{-1})
D0	Vertical drainage from the bottom of the surface soil layer (mm)
Dd	Vertical drainage from the bottom of the deep soil layer (mm)
delta	Slope of the saturation vapour pressure curve (Pa K^{-1})
Ds	Vertical drainage from the bottom of the shallow soil layer (mm)
E0	Potential evaporation (mm d^{-1})
Eg	Evaporation flux from the groundwater store (mm d^{-1})
Ei	Evaporation flux from canopy interception (mm d^{-1})
ER_frac_ref	Ratio of the mean evaporation rate and the mean rainfall intensity during storms (dimensionless)
Es	Evaporation flux from the surface soil store (mm d^{-1})
Et	Actual total transpiration flux (mm d^{-1})
Etmax	Potential transpiration rate (mm d^{-1})
Etot	Total evapotranspiration (mm d^{-1})
f_tree	Fraction of tree cover within each grid cell (dimensionless)
fEgt	Fraction of the grid cell that is accessible for transpiration from groundwater (dimensionless)
fsat	Fraction of the grid cell that is saturated at the surface (dimensionless)
FsoilEmax	Soil evaporation scaling factor corresponding to unlimited soil water supply (dimensionless)
fveg	Fractional canopy cover (dimensionless)
fvegref_G	Reference Soil Cover Fraction That Determines The Rate of Decline in Soil Heat Flux With Increasing Canopy Cover
fveq	Equilibrium canopy cover (dimensionless)
ga	Aerodynamic conductance (m s^{-1})
gamma	Psychrometric constant (Pa K^{-1})
Gfrac_max	Fraction of Daytime Net Radiation Lost To Soil Heat Storage for an Unvegetated Surface
gs	Canopy conductance (m s^{-1})
hruDR	Hydrological Response Unit: Deep Rooted
hruSR	Hydrological Response Unit: Shallow Rooted
hveg	Height of Vegetation Canopy (m)
I	Infiltration (mm)
IAL	Interaquifer leakage (mm)

IF0	Interflow draining laterally from the surface soil layer (mm)
IFs	Interflow draining laterally from the shallow soil layer (mm)
Init_	Initial state of variable
K_gw	Groundwater drainage coefficient (d^{-1})
K_gw_grid	Groundwater drainage coefficient obtained from continental mapping (d^{-1})
K_gw_scale	Scaling factor for groundwater drainage coefficient (dimensionless)
K_rout	Rate coefficient controlling discharge to stream (dimensionless)
K_rout_scale	Scale coefficient for calculating K_r ($d\ mm^{-1}$)
K0sat	Saturated hydraulic conductivity of surface soil layer ($mm\ d^{-1}$)
K0sat_grid	Saturated hydraulic conductivity of surface soil layer from pedtransfer ($mm\ d^{-1}$)
K0sat_scale	Scaling factor for hydraulic conductivity of surface soil layer (dimensionless)
Kdsat	Saturated hydraulic conductivity of deep soil layer ($mm\ d^{-1}$)
Kdsat_grid	Saturated hydraulic conductivity of deep soil layer from pedotransfer ($mm\ d^{-1}$)
Kdsat_scale	Scaling factor for hydraulic conductivity of deep soil layer (dimensionless)
Kr_coeff	Scaling factor for ratio of saturated hydraulic conductivity (dimensionless)
Krout_int	Intercept coefficient for calculating K_r (dimensionless)
Kssat	Saturated hydraulic conductivity of shallow soil layer ($mm\ d^{-1}$)
Kssat_grid	Saturated hydraulic conductivity of shallow soil layer from pedotransfer ($mm\ d^{-1}$)
Kssat_scale	Scaling factor for hydraulic conductivity of shallow soil layer (dimensionless)
LAI	Leaf area index (LAI) (dimensionless)
LAImax	Maximum achievable LAI value (dimensionless)
LAIref	Reference LAI value corresponding to $f_v = 0.63$ (dimensionless)
lambda	Latent heat of vaporisation ($MJ\ kg^{-1}$)
latitude	Latitude (radians), and is negative in the southern hemisphere
Mleaf	Leaf biomass ($kg\ m^{-2}$)
Mleafnet	Change in leaf biomass at each time step ($kg\ m^{-2}\ d^{-1}$)
ne	Effective porosity (dimensionless)
ne_scale	Scaling factor for effective porosity (dimensionless)
pair	Air pressure (Pa)
pe	Actual vapour pressure (Pa)
pes	Saturation vapour pressure (Pa)
Pg	Precipitation (mm)
PI	Sunset hour angle (radians)
Pn	Net precipitation – precipitation minus interception (mm)
Pref_gridscale	Scaling factor for reference precipitation (dimensionless)
PrefR	Reference value for precipitation (mm)
Pwet	Reference threshold precipitation amount (mm)
Q0	Function of the day of the year (radians)
Qg	Groundwater discharge to the surface water (mm)
QIF	Interflow (mm)
QR	Surface runoff (mm)
Qtot	Total discharge to stream (mm)

RadClearSky	Expected downwelling shortwave radiation on a cloudless day (MJ m^{-2} d^{-1})
RD	Rooting depth (m)
Rgeff	Daily downwelling shortwave (solar) radiation (MJ m^{-2} d^{-1})
Rh_0s	Partitioning factor for vertical and lateral drainage from the surface soil layer (dimensionless)
Rh_sd	Partitioning factor for vertical and lateral drainage from the shallow soil layer (dimensionless)
Rhof	Infiltration-excess runoff component (mm)
RLin	Daily downwelling longwave radiation (MJ m^{-2} d^{-1})
RLout	Daily upwelling longwave radiation (MJ m^{-2} d^{-1})
Rneff	Daily net radiation (MJ m^{-2} d^{-1})
Rsof	Saturation-excess runoff component (mm)
S	Storativity of the confined aquifer (mm)
S_sls	Specific canopy rainfall storage per unit leaf area (mm)
S0	Water storage in the surface soil layer (mm)
S0fracAWC_grid	Available water holding capacity in the surface soil (dimensionless)
S0max	Maximum storage of the surface soil layer (mm)
S0max_scale	Scaling parameter for maximum storage of the surface soil layer (dimensionless)
Sd	Water content of the deep soil store (mm)
Sdmax	Maximum storage of the deep soil layer (mm)
Sdmax_scale	Scale for Maximum water storage deep layer (Deep) (dimensionless)
Sg	Groundwater storage in the unconfined aquifer (mm)
SLA	Specific leaf area (m^2 kg^{-1})
slope	Slope of the land surface (percent)
slope_coeff	Scaling factor for slope (dimensionless)
Sr	Volume of water in the surface water store (mm)
Ss	Water content of the shallow soil store (mm)
SsfracAWC_grid	Available water holding capacity in the shallow soil (dimensionless)
Ssmax	Maximum storage of the shallow soil layer (mm)
Ssmax_scale	Scale for Maximum water storage shallow layer (Shallow) (dimensionless)
StefBolz	Stefan-Boltzmann constant (MJ m^{-2} d^{-1} K^{-4})
Ta	Daily mean temperature (°C)
Tgrow	Characteristic time scale for vegetation growth towards equilibrium (d)
Tmax	Maximum air temperature (°C)
Tmin	Minimum air temperature (°C)
Tsenc	Characteristic time scale for vegetation senescence towards equilibrium (d)
U0	Maximum root water uptake (mm d^{-1})
u2	Wind speed at a height of 2 m (m s^{-1})
Ud	Root water uptake (transpiration) from the deep soil store (mm d^{-1})
Ud0	Maximum possible root water uptake from the deep soil store (mm d^{-1})

Udmax	Maximum root water uptake from the deep soil store at prevailing moisture content (mm d^{-1})
Us	Root water uptake (transpiration) from the shallow soil store (mm d^{-1})
Us0	Maximum possible root water uptake from the shallow soil store (mm d^{-1})
Usmax	Maximum root water uptake from the shallow soil store at prevailing moisture content (mm d^{-1})
Vc	Greenness index per unit canopy cover
w0	Relative soil moisture content of the top soil layer (dimensionless)
w0limE	Limiting the value of w_0 at which evaporation is reduced (dimensionless)
w0ref_alb	Reference value of w_0 that determines the rate of albedo decrease with wetness (dimensionless)
wd	Relative water content of the deep soil store (dimensionless)
wdlimU	Water-limiting relative water content of the deep soil store (dimensionless)
ws	Relative water content of the shallow soil store (dimensionless)
wslimU	Water-limiting relative water content of the shallow soil store (dimensionless)
Y	Root water uptake (transpiration) from the groundwater store via capillary rise (mm d^{-1})

Appendix B. What is Python?

The AWRA CMS system runs on Python 3.4 and above. The following material relates to Python in general, as well as a few key tools and libraries used in the AWRA CMS system.

A very brief overview

- Python is a programming language.
- Many fields of application.
- Works on many platforms.
- Equipped with many modern programming structures and methods that you can 'grow' into.
- Well-supported by many 'packages'
- Especially useful for geospatial applications

Where you are coming from	**Python pathway**
Fortran, C, IDL, etc	Python can be programmed in that style: loops, arrays, if-blocks, etc.
Shell/DOS scripting	Python can be used for text processing, file operations and task control.
MATLAB	Python can call the *PYLAB* and *MATPLOTLIB* packages with similar functionality.
Statistics Packages: S, SyStat, SPSS, R	Python can use *SciPy* and *Pandas* for similar tasks.
ArcGIS, etc	Many libraries with overlapping functionality, but via scripting/programming.
Excel, spreadsheets	Maybe not, but Python can work with spreadsheet files.

Platform independence

Most Python programs will run without change on Linux, Unix, Windows and other platforms (including mobile devices).

A modern language

Python offers modern and convenient data structures and programming idioms:

- Lists
- Dictionaries
- Object-oriented programming
- Functional Programming

Applications for Python

Built in Modules/Packages

Python comes with a standard library of "modules" ready to use.

- Date and time handling
- Text processing
- File services
- Database tools
- Internet handling
- Compression
- Cryptography
- Etc!

If you need a tool, check the Standard Library first!

Third-Party Packages¶

If you need functionality not in the Standard Library, check the Python Package Index.

Packages include: * Web frameworks: (e.g. Flask) * Scientific/Engineering * Image, Video, Audio Processing * Network control * Email handling

The Python community is a rich ecosystem, with many well-supported packages.

The Geospatial Stack¶

There are a number of special heavily duty modules for geospatial work:

- **Numpy** - for fast array processing (e.g. of rasters)
- **Scipy** - for lots of mathematical and statistical functions and operations
- **MatPlotLib** - for flexible 2D, 3D and animated plotting

- **Pandas** - for statistical analysis of data
- **GDAL** - for image processing, especially of geospatial images
- **Basemap** - for manipulating maps
- **OGRtools** - for working with shapefiles
- **netCDF4** and **hdf**

...and utility packages for creating GUI applications (PyQT), databases (e.g. SQLalchemy), packaging big data, shipping output via the web, etc.

Why's it Called Python?

...because the designers liked Monty Python.

Appendix C. Introduction to IPython Notebook

IPython is a set of tools originally developed to make it easier for scientists to work with Python and data. It allows you to combine interactive Python exploration with prewritten programs and even text and equations for documentation.

IPython isn't a different programming language, it's just a set of computer programs for working with the Python language.

The notebook extends the console-based approach to interactive computing, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.

Notebook documents contains the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects.

Documentation on how to set up and use the notebook is available online at:

http://ipython.org/ipython-doc/1/interactive/notebook.html

There are a number of tutorials and demos available on the web, e.g.

http://www.randalolson.com/2012/05/12/a-short-demo-on-how-to-use-ipython-notebook-as-a-research-notebook/

Sneak Preview of IPython

Use matplotlib to display a nice plot

Note: To execute a cell hit "shift-enter". As well as Shift-Enter there are other keyboard shortcuts in Notebook. Look under the Help menu -> Keyboard Shortcuts to see them all.

```
%matplotlib inline
```

```
from matplotlib.pyplot import *
import numpy
x = numpy.linspace(0, 3*numpy.pi, 500)
plot(x, numpy.sin(x**2))
```

```
title("hello birdy")
```

Viewing help (docstrings) associated with functions,methods, classes

```
numpy.linspace?
```

Magic commands

It may also be useful to know what magic commands are. IPython has a set of predefined 'magic functions' that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented. **Line magics** are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. **Cell magics** are prefixed with a double %%, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument. More at:

https://ipython.org/ipython-doc/dev/interactive/magics.html

Loading a python script into a cell, then execute the cell

```
%load misc/hello_AWRA CMSi.py
```

Running a python script¶

```
%run misc/hello_AWRA CMSi.py
```

loading a python script as a module¶

```
import misc.hello_AWRA CMSi as m
```

Notebook Gotchas: Beware when updating modules

Ipython has several ways of reloading modules that you have edited, but be wary that the changes have actually been reloaded

%reload(module_name)
%autoreload

if in doubt do a Kernel->Restart (Ctrl-m 0 0) and Cell->Run all above

Ipython tips

http://pages.physics.cornell.edu/~myers/teaching/ComputationalMethods/python/ipython.html

Debugging tips

A cool feature of iPython notebook is the %debug magic. If you encounter an exception in your code, just type %debug 0 on the next line or cell.

Once in the debugger, you can navigate up and down the stack trace, examine variables, execute code, etc. This is a really powerful way to quickly get to the root cause of an error.

Appendix D. AWRAMS Licence

AWRAMS LICENCE
(Variation of BSD 3 licence)

By accessing or using the AWRAMS software, code, data (input or validation data) or documentation ("AWRAMS Material"), You agree to be bound by these licence terms including any terms or notices incorporated by reference (as updated from time to time) on and from the date You first access or use of the AWRAMS Material. You acknowledge that Your agreement to these licence terms constitutes a legally binding agreement between You and the Commonwealth of Australia as represented by the Bureau of Meteorology ("copyright holder"). If You do not agree to be bound by any of these licence terms, You are not authorised to access and use the AWRAMS Material.

The reference to "You" ("or Your") is a reference to an individual or legal entity exercising permissions granted under these licence terms.

Redistribution and use of AWRAMS Material in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of AWRAMS Material in source code must retain this AWRAMS licence in its entirety.

* Redistributions of AWRAMS Material in binary form must reproduce this AWRAMS licence in its entirety.

* Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from AWRAMS Material without specific prior written permission.

* Redistribution and use of any part of the AWRAMS Material for commercial purposes must not occur without prior written permission from the Bureau of Meteorology (email: awracms@bom.gov.au). For the avoidance of doubt, 'commercial purposes' includes without limitation developing and supplying a product or service to another party for profit or other commercial gain.

* You will make any and all of Your modifications to the AWRAMS Material (including modifications to code, data or documentation) available to the Bureau of Meteorology for possible use by it in any way, including, but not limited to, incorporating modifications into the existing AWRAMS Material if they meet the relevant criteria and making these modifications available to third parties.

* You will complete and sign and return to the Bureau of Meteorology (by email to awracms@bom.gov.au) the Contributor Licence Agreement ("CLA") which accompanies these licence terms in relation to any and all of Your modifications to the AWRAMS Material referred to above.

TO THE FULLEST EXTENT PERMITTED BY LAW, THE AWRAMS MATERIAL IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED OR

EXCLUDED. TO THE FULLEST EXTENT PERMITTED BY LAW, NEITHER THE COPYRIGHT HOLDER NOR ANY CONTRIBUTOR WILL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWSOEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE AWRAMS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH CLAIM, LIABILITY, LOSS OR DAMAGE.

ACCESS TO INPUT/VALIDATION DATA

You will need to complete and sign this AWRAMS licence and the CLA and return signed copies to the Bureau of Meteorology (by email to awracms@bom.gov.au) to enable the Bureau of Meteorology to assess whether to issue You with a link to the input data and validation data for AWRAMS together with a user name and password for access to the data. You agree that the Bureau of Meteorology may allow or deny You access to the input data or validation data for AWRAMS in its absolute discretion and without giving reasons following receipt of the completed and signed AWRAMS licence and CLA. If the Bureau of Meteorology grants you access to the input and validation data you agree that Your access to and use of this data will be governed by the terms of this AWRAMS licence.

Irrespective of whether or not these licence terms and the CLA are signed and submitted to the Bureau as required above, You agree that the licence obligations remain legally binding and enforceable to the extent permitted by law as a consequence of You accessing or using the AWRAMS Material and on and from the date on which You first access or use the AWRAMS Material.

USER DETAILS (please complete)
Name (individual):
Name of Organisation (legal entity):
ABN
Email address:
Address:
Phone number:
Please state the nature and purpose of proposed use of AWRAMS Material:
……
……
Signed (individual):
……
Date:
Signed (legal entity) by its duly authorised representative:
……
Date:

Appendix E. AWRAMS CONTRIBUTOR LICENCE

AWRAMS CONTRIBUTOR LICENCE AGREEMENT
(FOR INDIVIDUALS AND LEGAL ENTITIES)

This contributor agreement ("Agreement") documents the rights granted by contributors to the Commonwealth of Australia as represented by the Bureau of Meteorology ("Us"). To make this document effective, please sign it and send it to Us (by email to awracms@bom.gov.au).

1. Definitions

"You" means as applicable the individual ('natural person') or organisation ('legal entity') who Submits a Contribution to Us.
"Contribution" means any work of authorship or invention that is Submitted by You to Us in which You own or assert ownership of any intellectual property rights (**IP**), including without limitation any copyright, patent rights or related rights. If You do not own IP in the entire work of authorship or invention, please clarify the basis on which the work of authorship or invention was created, contributed or used by You (where indicated in clause 3 below).
"Material" means the work of authorship or invention which is made available by Us to third parties. When this Agreement covers more than one software project, the Material means the work of authorship or invention to which the Contribution was Submitted. After You Submit the Contribution, it may be included in the Material.
"Submit" means any form of electronic, verbal, or written communication sent to Us or our representatives, including but not limited to electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, Us for the purpose of discussing and improving the Material.

"Effective Date" means the date You execute this Agreement or the date You first Submit a Contribution to Us, whichever is earlier.

2. Grant of Rights

2.1 Copyright License

(a) You retain ownership of the copyright in Your Contribution and have the same rights to use or license the Contribution which You would have had without entering into the Agreement.
(b) To the maximum extent permitted by the relevant law, You grant to Us a perpetual, worldwide, non-exclusive, transferable, royalty-free, irrevocable copyright licence covering the Contribution, with the right to sublicense such rights through multiple tiers of sublicensees, to reproduce, modify, display, perform and distribute the Contribution as part of the Material.

2.2 Patent License

For patent claims including, without limitation, method, process, and apparatus claims which You own, control or have the right to grant, now or in the future, You grant to Us a perpetual, worldwide, non-exclusive, transferable, royalty-free, irrevocable patent licence, with the right to sublicense these rights to multiple tiers of sublicensees, to make, have made, use, sell, offer for sale, import and otherwise transfer the Contribution and the Contribution in combination with the Material (and portions of such combination). This licence is granted only to the extent that the exercise of the licensed rights infringes such patent claims.

2.3 Outbound License

Based on the grant of rights in clauses 2.1 and 2.2, if We include Your Contribution in Material, We may license the Contribution under any license, including copyleft, permissive, commercial, or proprietary licenses.

2.4 Moral Rights

If moral rights apply to the Contribution, to the maximum extent permitted by law, You waive and agree not to assert such moral rights against Us or our successors in interest, or any of our licensees, either directly or indirectly. In addition, You consent to Us or our successors in interest or licensees using the Contribution consistently with the terms of this Agreement for moral rights purposes. If You are an organisation or not the moral rights holder, You agree to obtain written consents from the authors of the relevant parts of the Contribution to allow Us or our successors in interest or licensees to use the Contribution consistently with the terms of this Agreement for moral rights purposes.

2.5 Our Rights

You acknowledge that We are not obliged to use Your Contribution as part of the Material and may decide to include any Contribution We consider appropriate.

2.6 Reservation of Rights

Any rights not expressly licensed under this clause are expressly reserved by You.

3. Agreement

You warrant and represent that:

(a) You have the legal authority to enter into this Agreement.

(b) You own or otherwise have rights in respect of the copyright and patent claims covering the Contribution which are required to grant the rights under clause 2.

(c) the grant of rights under clause 2 does not violate any grant of rights which You have made to third parties, including Your employer. If You are an employee, You have had Your employer approve this Agreement or sign another copy of this document. If You are less than eighteen years old, please have Your parents or guardian sign the Agreement.

(d) the Contribution does not contain any malicious code, virus or any intentionally deleterious modification.

(d) If You do not own the IP in the entire work of authorship or invention Submitted, the legal basis on which You have created or used the work of authorship or invention and on which You have legal authority to enter into this Agreement and to grant the licence rights under this Agreement is as follows:

[insert details]

4. Disclaimer

EXCEPT FOR THE EXPRESS WARRANTIES IN CLAUSE 3, THE CONTRIBUTION IS PROVIDED "AS IS". MORE PARTICULARLY, ALL EXPRESS OR IMPLIED WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY DISCLAIMED BY YOU TO US. TO THE EXTENT THAT ANY SUCH WARRANTIES CANNOT BE DISCLAIMED, SUCH WARRANTY IS LIMITED IN DURATION TO THE MINIMUM PERIOD PERMITTED BY LAW.

5. Miscellaneous

5.1 This Agreement will be governed by and construed in accordance with the laws of the Australian Capital Territory excluding its conflicts of law provisions. Under certain circumstances, the governing law in this clause might be superseded by the United Nations Convention on Contracts for the International Sale of Goods ("UN Convention") and the parties intend to avoid the application of the UN Convention to this Agreement and, thus, exclude the application of the UN Convention in its entirety to this Agreement.

5.2 This Agreement sets out the entire agreement between You and Us for Your Contributions to Us and overrides all other agreements or understandings.
5.3 If You or We assign the rights or obligations received through this Agreement to a third party, as a condition of the assignment, that third party must agree in writing to abide by all the rights and obligations in the Agreement.
5.4 The failure of either party to require performance by the other party of any provision of this Agreement in one situation shall not affect the right of a party to require such performance at any time in the future. A waiver of performance under a provision in one situation shall not be considered a waiver of the performance of the provision in the future or a waiver of the provision in its entirety.
5.5 If any provision of this Agreement is found void and unenforceable, such provision will be replaced to the extent possible with a provision that comes closest to the meaning of the original provision and which is enforceable. The terms and conditions set forth in this Agreement shall apply notwithstanding any failure of essential purpose of this Agreement or any limited remedy to the maximum extent possible under law.

You
Name (individual): _____
Name (organisation): _____
Address: _____
_____
Signature (duly authorised representative):
_____
Date:

Us
_____
Name: _____
Title: _____
Address: _____
_____
Signature (duly authorised representative):
_____
Date: