|  |
| --- |
| AWRA Landscape Community Modelling System User Guide |
| A guide to setting up and running the AWRA Landscape Community Modelling System (AWRA-CMS) |

|  |
| --- |
|  |

Wendy Sharples, Avijeet Ramchurn, Alison Oke, Bianca Gibson, David Shipman, Stuart Baron-Hay, Sean Loh, Andrew Frost, Chantal Donnelly and David Wright

|  |
| --- |
| Published by the Bureau of Meteorology |

© Commonwealth of Australia 2019

This work is copyright. Apart from any use as permitted under the Copyright Act 1968, no part may be reproduced without prior written permission from the Bureau of Meteorology. Requests and inquiries concerning reproduction and rights should be addressed to the Production Manager, Communication Section, Bureau of Meteorology, GPO Box 1289, Melbourne 3001. Information regarding requests for reproduction of material from the Bureau website can be found at [www.bom.gov.au/other/copyright.shtml](http://www.bom.gov.au/other/copyright.shtml)

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 22/9/2016 | V1 | Final User Guide | Alison Oke |
| 29/08/2017 | Draft V2.0 | Working draft | Avijeet Ramchurn |
| 22/11/2017 | Draft V2.1 | Draft for MODISM Training | Alison Oke |
| 21/12/2017 | V2.6 | Full draft of Guide | Alison Oke |
| 17/1/2018 | V2.7 | Review of draft | Chantal Donnelly |
| 14/02/2018 | V2.8 | Final review and QA | Wendy Sharples |
| 08/03/2018 | V2.9 | Final QA | David Wright |
| 04/03/2019 | V3.0 | Rework for 1.2 Release | Wendy Sharples |
| 12/03/2019 | V3.1 | Incorporate changes after review | Wendy Sharples |
| 13/03/2019 | V3.2 | Incorporate changes after review | Wendy Sharples |

**Review Status**

|  |  |  |
| --- | --- | --- |
| **Reviewer** | **Date Reviewed** | **Version Reviewed** |
| Chantal Donnelly | 11/03/2019 | V3.0 |
| Rob Argent | 13/03/2019 | V3.1 |
|  |  |  |
|  |  |  |
|  |  |  |

**Release Signatories**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval** | **Name** | **Signature** | **Date** |
|  |  |  |  |
|  |  |  |  |

**Contact details**

**Dr. Chantal Donnelly**

Manager - Water Resources Modelling Unit

Bureau of Meteorology

GPO Box 1289 MELBOURNE VIC 3001

Phone: 0732398767

Email: [awracms@bom.gov.au](mailto:awracms@bom.gov.au)

**Anthony Brinkley**

Section Head, Water Resources Assessment (WRA)

Bureau of Meteorology

GPO Box 1289 MELBOURNE VIC 3001

Phone: +61 3 8638 8274

Email: [Anthony.Brinkley@bom.gov.au](mailto:Anthony.Brinkley@bom.gov.au)

CONTENTS

[1. Introduction 6](#_Toc3469688)

[1.1. Background 6](#_Toc3469689)

[1.2. The AWRA Community Modelling System (AWRA-CMS) 6](#_Toc3469690)

[1.3. The AWRA-CMS package 7](#_Toc3469691)

[1.4. Input and test datasets 9](#_Toc3469692)

[1.5. User registration 9](#_Toc3469693)

[1.6. Licensing 10](#_Toc3469694)

[1.7. Understanding this User Guide 10](#_Toc3469695)

[2. Installing AWRA-CMS 11](#_Toc3469696)

[2.1. System overview 11](#_Toc3469697)

[2.2. Installing AWRA-CMS 11](#_Toc3469698)

[1. Linux – Automated 11](#_Toc3469699)

[2. Linux – Manual 13](#_Toc3469700)

[3. Windows – Automated 13](#_Toc3469701)

[4. Windows – Manual 13](#_Toc3469702)

[2.3. Getting started with AWRA-CMS 13](#_Toc3469703)

[1. Directories 13](#_Toc3469704)

[2. Training Notebooks 14](#_Toc3469705)

[3. Training Input Data 14](#_Toc3469706)

[3. AWRA-CMS concepts and classes 15](#_Toc3469707)

[3.1. Setting up AWRA-L simulations and calibrations 15](#_Toc3469708)

[3.2. Model inputs 15](#_Toc3469709)

[3.3. Model configuration and the nodegraph 15](#_Toc3469710)

[1. Configuration 16](#_Toc3469711)

[2. Nodegraphs 16](#_Toc3469712)

[4. Training exercises 17](#_Toc3469713)

[4.1. Overview 17](#_Toc3469714)

[4.2. Simulation 17](#_Toc3469715)

[4.2.1. Using the AWRA MS to run a Simulation of the AWRA-L model 17](#_Toc3469716)

[4.3. Calibration 42](#_Toc3469717)

[4.3.1 Using the AWRA-CMS to run a Calibration of the AWRA-L model 42](#_Toc3469718)

[4.4. Visualisation 63](#_Toc3469719)

[4.4.1 Spatial and Time-series visualisation of AWRA-L inputs and outputs 63](#_Toc3469720)

[4.5. Benchmarking 80](#_Toc3469721)

[4.5.1 Demonstrates AWRA-CMS benchmarking functionality using CMRS ET data 80](#_Toc3469722)

[5. References 93](#_Toc3469723)

[Appendix A. AWRA-L model overview 94](#_Toc3469724)

[Appendix B. AWRA-L parameters 96](#_Toc3469725)

[Appendix C. Python version of AWRA-L 100](#_Toc3469726)

[Appendix D. Contributing to AWRA-CMS 101](#_Toc3469727)

[Appendix E. AWRAMS Licence 102](#_Toc3469728)

[Appendix F. AWRAMS Contributor Licence 105](#_Toc3469729)

## Introduction

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

### 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 been implemented as an operational system within the Bureau. AWRAMS ongoing development is being supported into the future by the Bureau and CSIRO.

The AWRAMS (Hafeez et al., 2015) has two modelling components:

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

To enable further development and other uses of the system, the AWRA-L component has been bundled into a framework for pre and post-processing, calibration and benchmarking to the Australian context and has been released to the community as the AWRA-CMS (Community Modelling System) package. This document describes the AWRA-CMS.

### The AWRA Community Modelling System (AWRA-CMS)

Based on active engagement with a wide range of stakeholders, it is apparent that users have different requirements for applications of AWRA-L modelled ranging from water resources assessment and planning to various other sectors (including agriculture, natural resources management, flood and groundwater) at continental to regional and catchment scales. 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.

### The AWRA-CMS package

The AWRA-CMS is presented as a publicly available python package called **awrams**. The package is designed to provide easy functionality to use, analyse and modify the AWRA-L model code and its inputs and outputs. While the model kernel code is kept in C language, everything else is in Python. The AWRA-CMS is currently limited to Landscape (AWRA-L) model and related functionality.

The package is purpose built using open-source software such as Python, C to facilitate adoption. Model content and functionality are divided into a number of linked components, shown conceptually in Figure 2. These components are expanded on below:

##### 1. awrams.models - The AWRA-L model code

The current version of the operational model (AWRA-L v6) is the default hydrological model (code and parameters) provided as part of the package. For speed and efficiency, the model's spatio-temporal loops are coded in the C-language with Python used to provide the supporting architecture to feed in user-specified input and run settings. Advanced users can modify the model code or add other models if desired.

##### 2. awrams.simulation - Simulation tools to run the model

The simulation component contains the required functionality to interact with the run settings of the model to specify its extent, period, and outputs. The inputs and outputs to the model are user configurable. Notably, the model can be run in two modes:

1. **On-demand**, i.e. keeping the inputs and outputs stored in memory for fast interactive use,
2. **Server**, i.e. writes output directly to disk and is used for spatio-temporally large runs (producing data larger than available memory).

##### 3. awrams.calibration – Tools to calibrate the parameters of the model

This component contains tools for calibration and evaluation of sensitivity to model parameters. It 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 on a supercomputer (if computational time becomes critical when applying over large spatial and temporal scales, using multiple processors). The user can specify the calibration objective functions and calibrate to multiple observations (and observation types) simultaneously.

The Bureau has made use of this capability to fit the AWRA-L v6 model simultaneously to streamflow (specific runoff), soil moisture and evapotranspiration (ET) observations for approximately 300 unimpaired catchments from across Australia with a further 300 catchments reserved for validation.

##### 4. awrams.benchmarking – Benchmarking model outputs against observations

The benchmarking functionality facilitates the repeatable comparison of AWRA-L model outputs as well as other model outputs against observations of landscape water stores and fluxes (e.g. discharge, soil moisture, evapotranspiration, recharge estimates and more). The idea is that the benchmarking can be used to compare models and model versions. The package generates key model-fit statistics (for example Nash-Sutcliffe Efficiency, relative bias and correlation) and plots (e.g. time-series, scatterplots, box-whisker plots, and CDF) of model outputs (e.g. catchment aggregated gridded runoff, soil moisture, ET and deep drainage). The outputs themselves can be viewed either graphically or in tables.

##### 5. awrams.visualisation – Visualisation of inputs and outputs

The visualisation tools enable the visual inspection of AWRA-L model inputs and outputs both spatially and temporally, and 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.

##### 6. awrams.utils – Utilities to support the system

The utils component contains functionality that is required in more than one of the above packages. It contains support code for file-handling, geospatial operations and data manipulation such as fast extraction from gridded outputs; and grid time-series conversion.

Simulation

Calibration

Benchmarking

Visualisation

Common/Supporting Functionality

Underlying Open Source Components:

Python language and libraries (Matplotlib, Numpy, Pandas etc)

Jupyter interactive Python scripting environment

And other computational libraries e.g. NetCDF41, HDF52, MPI3

Hydrological model (AWRA-L)

Datasets

Figure 1. Components of the Community Modelling System

### Input and test datasets

A set of sample datasets for training and testing purposes is provided along (see Section 2) with the CMS package. The datasets contain:

1. Forcing inputs,
2. Spatial parameters,
3. Observed data for calibration and benchmarking.

These datasets were used during the AWRA-L development and are set up within the package for model calibration and benchmarking purposes. Users must also abide by individual dataset licences contained with the data. Details of the data sources are as follows:

* + - * + streamflow from unimpaired catchments across Australia (Zhang et al., 2013),
        + remotely sensed catchment averaged soil moisture ([AMSR-E](http://gcmd.nasa.gov/KeywordSearch/Metadata.do?Portal=GCMD&KeywordPath=Parameters%7CLAND+SURFACE%7CSOILS%7CSOIL+MOISTURE%2FWATER+CONTENT&OrigMetadataNode=GCMD&EntryId=GES_DISC_LPRM_AMSRE_D_SOILM3_V002&MetadataView=Full&MetadataType=0&lbnode=mdlb3); Owe et al., 2008),
        + remotely sensed, catchment averaged ET derived from MODIS satellite observations ([CMRSET](http://www.sciencedirect.com/science/article/pii/S002216940900105X); Guerschman et al., 2009),
        + sample observations of in-situ soil moisture from two Australian networks in the Upper Hunter ([SASMAS](http://www.eng.newcastle.edu.au/sasmas/SASMAS/sasdata.html); Rüdiger et al., 2007) and Murrumbidgee ([OzNet](http://www.oznet.org.au/) ; Smith et al., 2012), and
        + gap filled site ET from an Australian flux tower network ([OzFlux](http://ozflux.org.au/); Beringer, Hutley et al., 2016; infilled according to the processing outlined in Beringer, McHugh et al., 2016). [[1]](#footnote-1)[[2]](#footnote-2)[[3]](#footnote-3)

### 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](https://github.com/awracms/awra_cms), and are bound by the modified BSD license agreement (See Section 1.6). The downloaded data consists of:

* + - * + User Guide (this document),
        + Technical Model Description Report (Frost, Ramchurn and Smith, 2018),
        + Benchmarking Report (Frost, and Wright, 2018),
        + Limited observation and forcing data for calibration and benchmarking.

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*** are:

* + - * + 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 supported [fork](https://help.github.com/articles/fork-a-repo/) of GiThub repository further allowing the opportunity to contribute to the ongoing development of the AWRA-CMS code (Appendix E),
        + Ongoing support and updates to package datasets.

To become a Registered User send an email to [awracms@bom.gov.au](mailto: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 the Bureau will manage updating the master copy of the AWRA-CMS repository will be provided.

### Licensing

The Bureau has chosen the Berkeley Software Distribution (BSD) license modified according to the Bureau’s legal advice and requirements (see Appendix C), including restricting commercial use of the modelling system. The adoption of a modified BSD-based license was done 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 D) 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.

### Understanding this User Guide

The remainder of this User Guide describes how you set yourself up to use the AWRA-CMS, and provides detailed guidance on how to undertake common hydrological modelling tasks (i.e. simulation, calibration, benchmarking, visualisation, and data extraction) using the tools provided. This user guide accompanies AWRA-CMS v1.2 released March 2019. Previous releases include v0.1, and v0.0.

This User Guide takes the approach of prioritising the description of common hydrological modelling workflows for each of the main capabilities of the package rather than explaining all the commands available in each of the packages. The material is thus aimed at educating the user about the essential function calls needed to achieve typical hydrological modelling tasks, and the user is encouraged to build their understanding of the package capabilities through both accessing the help available and exploring the package code. Documentation of the package (using docstrings) is ongoing so users are encouraged to update the package regularly.

Note that the scientific description of the model is not given here and is instead provided in the Bureau’s Operational AWRA-L Model technical description document (Frost, Ramchurn and Smith, 2018). This document is accessible from the following URL: <http://www.bom.gov.au/water/landscape/assets/static/publications/AWRALv6_Model_Description_Report.pdf>.

## Installing AWRA-CMS

AWRA-CMS can be installed on Red Hat Enterprise Linux and CentOS 6 & 7 and Windows 10. Support for debian like Linux distributions is coming soon.

### System overview

The Python programming language (Python 3 and Jupyter notebook) underpins the AWRA-CMS. Python is a widely used, platform independent programming language. A notebook is an interactive document containing code, text, and other elements. Jupyter notebooks provide an interface with which you can easily interact with the python code. It is recommended that the user complete some tutorials and familiarise themselves with the basics of Python before setting up the AWRA-CMS (e.g. see Appendix F).

The AWRA-L model kernel is coded in C. C was chosen 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 for routine model interaction. Knowledge of these languages is however necessary if the user wishes to alter internal model 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.

### Installing AWRA-CMS

#### Linux – Automated

##### With Sudo (administrator privileges)

Run the following command in terminal:

curl -s <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | sudo -E bash

With training data download:

export CLONE\_DATA=True && curl -s <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | sudo -E bash

Alternative Method (if curl is unavailable):

wget -O - <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | sudo -E bash

##### Without Sudo

Run the following command in terminal:

curl -s <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | bash

With Training data download:

export CLONE\_DATA=True && curl -s <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | bash

Alternative Method (if curl is unavailable):

wget -O - <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.sh> | bash

* 1. **Optional Environment Variables:**

# By default the dependencies are installed. If you do not choose to install the dependencies set

# INSTALL\_DEPENDENCIES to FALSE

export INSTALL\_DEPENDENCIES=FALSE # (default: True)

# By default the version downloaded is master branch.

export AWRAMS\_VERSION=dev # (default: master)

# By default the training data set is not cloned. Please set CLONE\_DATA to TRUE to obtain the training data set before

installation

export CLONE\_DATA=TRUE # (default: False)

**NOTE: It is highly recommended that new users clone the data with the AWRA-CMS package install**

# By default AWRAMS\_INSTALL\_PATH is the current directory.

export AWRAMS\_INSTALL\_PATH="/installation/path/" # (default: Current directory)

# By default CLONE\_DATA\_PATH is set to "/install/path/awrams/data". If this variable is set, the data will be installed

to that path.

export CLONE\_DATA\_PATH="/clone/data/to" # (default: AWRAMS\_INSTALL\_PATH/awrams/data)

# By default CONDA\_PATH is set to none. If there is a conda environment already installed, point CONDA\_PATH

to it.

export CONDA\_PATH="/home/miniconda3/" # (default: none)

# By default OPENMPI\_VERSION is set to 3.1.3.

export OPENMPI\_VERSION="1.8.0" # (default: 3.1.3)

# By default OPENMPI\_PATH is /home/user/. If you choose to install it elsewhere specify the path or if you

already have openmpi installed, then point the variable to where it has been installed.

export OPENMPI\_PATH="/usr/bin/" # (default: /home/user)

# By default OPENMPI\_INSTALL is set to install by default. If you do not choose to install OPENMPI and wish

to module load MPI set this variable to false.

export OPENMPI\_INSTALL="false" # (default: true)

# By default MPI4PY\_VERSION is set to 3.0.0. This variable changes the python package version.

export MPI4PY\_VERSION=3.0.0 # (default: 3.0.0)

# By default FROM\_SCRATCH is set to false. This variable deletes the folders created from a previous install and

re-installs everything.

export FROM\_SCRATCH=true # (default: false)

##### Initiate Jupyter notebooks

Run the following command in terminal:

jupyter notebook

#### Linux – Manual

See full documentation: <https://github.com/awracms/awra_cms/blob/AWRA_CMS_v1.2/INSTALL-GUIDE.md>

#### Windows – Automated

##### Install AWRA-CMS

Go to: <https://raw.githubusercontent.com/awracms/awra_cms/AWRA_CMS_v1.2/scripts/install.bat> to download the install.bat script. Double click to run the script and follow the prompts.

##### Initiate Jupyter notebooks

To initiate the Jupyter notebooks, run the following command in the command prompt:

jupyter notebook

#### Windows – Manual

See full documentation: <https://github.com/awracms/awra_cms/blob/AWRA_CMS_v1.2/INSTALL-GUIDE.md>

### Getting started with AWRA-CMS

#### Directories

The directories of the package are structured around the five 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.

#### Training Notebooks

The Training notebooks demonstrate the functionality of the CMS and are located in the Training folder. They are the recommended place for new users to start, as they guide the user through the five major components of AWRA-CMS and serve as living documentation. These notebooks are kept up to date and it is advisable to always refer to them over this document in terms of relevance.

#### Training Input Data

Training data can be cloned along with the AWRA-CMS. This dataset is to be used in conjunction with the Training notebooks (see Section 2 for installation instructions). To gain access to the full datasets become a AWRA-CMS Registered User (see Section 1.5) or an AWAP Registered User[[4]](#footnote-4) for the full forcing datasets.

## AWRA-CMS concepts and classes

This section introduces the essential CMS concepts for running a simulation or calibration.

### Setting up AWRA-L simulations and calibrations

Setting up a simulation or calibration run starts by defining:

1. The **model** (in this case AWRA-L),
2. The inputs and outputs via the nodegraph (the nodegraph is a dictionary used to configure a model run):
   1. Inputs:
      1. the forcing inputs (rain, temperature, radiation),
      2. the spatial inputs (grids of various landscape properties),
      3. model parameters (calibrated or fixed).
   2. Outputs:
      1. the model outputs to be written out (e.g. flow, evaporation, soil moisture),
      2. the grid resolution (default is 0.05 degree).
3. The spatial **extent,**
4. The modelling time **period.**

### Model inputs

The model inputs can be divided into three types

* + - * + Forcing inputs: These are the daily varying climate inputs, e.g. rain, maximum and minimum temperature and solar radiation are provided as netcdf (\*.nc) files.
        + Spatial grids: static soil, vegetation and topography related spatial datasets are stored in a single HDF5 file **spatial\_parameters.h5.** <https://en.wikipedia.org/wiki/Hierarchical_Data_Format>. Currently these are static spatial grids.
        + Calibratable model parameters: These are stored in a \*.json format file: **DefaultParameters.json** <https://en.wikipedia.org/wiki/JSON>

Typically, constants, non-gridded variables and non-forcing arguments to functions are variables defined in the parameter set. The user can modify parameter values at source (i.e. in the file) or on the fly within a notebook prior to setting off the model run.

### Model configuration and the nodegraph

In the AWRA-CMS, loading up the default model (AWRA-L) brings up the default configuration of how all the model input/parameters are combined and what outputs are produced. How these come together for the model run (with any transformation or scaling) is dependent on the model and system configurations located in awrams/config. The configuration is organised by way of a mapping system, which links a variable used by the system to the mechanism by which it is generated. This mapping system, which ultimately defines the data flow from raw input to model parameter, is presented to the model in a python dictionary (each parameter is a key and the transforms that apply is the value) used to describe the data-flow.

#### Configuration

By default, everything is set up at installation (see Section 2) to point at the default model and the Training data set so that the Training notebooks run as designed. In the instance that a user wants to use different machines, different input data and change the model parameters, the configuration scripts in config/model and config/system can be altered as required. These can be saved as objects to be imported as profiles for repeated use and provenance traceability. For more details please see the 'Changing the Model' notebook: Training/Advanced/ChangingtheModel.ipynb, in the AWRA-CMS repository.

#### Nodegraphs

In AWRA-CMS, this mapping dictionary can be called up and edited through '**nodegraph**' functionality, essentially linking (*graph*ing) each piece of data processing (*node*) needed by the system and transforming the configuration information into runnable code, telling the model how to configure the various components required for/by the run. The nodegraph is devised as a python class, and hence comes with certain attributes and applicable methods that facilitate input and output manipulation (awrams.utils.nodegraph).

At runtime, the **input nodegraph** controls the loading of files and any infilling and unit conversion at run time (in memory), hence it provides flexibility in the sense that inputs or outputs can be defined through functions and arguments within a user notebook just prior to a run. It 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.

For example, AWRA-L model uses a single temperature input, but AWAP supplies two (minimum and maximum temperature). The default AWRA-L input mapping loads the AWAP inputs then rectifies and computes a weighted average of these, before passing the single value on to the core model code. Before setting off a model run, the user could modify the calculation of the weighted average by modifying the nodegraph.

Similarly, the **output nodegraph** defines which variables are to be output where, and any transformations to be completed upon output.

## Training exercises

### Overview

The purpose of the Training notebooks is to guide the user through the functionality of the AWRA-CMS with examples and exercises. The Training notebooks reside in the Training folder with sub folders containing notebooks describing the functionality of each module of the AWRA-CMS. It is highly recommended that the user run through all the Training notebooks before setting up his/her own model. In addition, the user should consider the Training notebooks as the most up to date version of documentation and refer to these notebooks over this manual. The following sections contain a selection of example notebooks from the Training folder detailing how to run simulation, calibration, visualisation and benchmarking.

A quick refresher of the AWRA-CMS components:

* **Simulation** *Module:* **awrams.simulation**   
  Produces a specified run of the AWRA-L model over a time period and spatial extent
* **Calibration** *Module:* **awrams.calibration**   
  Obtains the optimum set of AWRA-L model parameters using an optimisation algorithm and an objective function, for a given time period, extent, etc.
* **Visualisation** *Module:* **awrams.visualisation**  
  Produces figures showing AWRA-L inputs and outputs as maps and time-series
* **Benchmarking** *Module:* **awrams.benchmarking**  
  Validates model outputs compared to observed data or other model simulations
* **Utilities** *Module:* **awrams.utils**   
  Contains functionality that is required in more than one of the AWRA-CMS components

### Simulation

In [1]:

### uncomment to display figures

%matplotlib inline

#### 4.2.1. Using the AWRA MS to run a Simulation of the AWRA-L model

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

* Using the On-Demand Simulator
* Using the Server Simulator

This notebook outlines how to run a basic on-demand AWRA-L simulation and get into more detail about how to change the model configuration (nodegraph) to define the inputs to the simulation.

**On-Demand Simulation**

The AWRA MS On-Demand Simulator is designed to run the model for a few years over a small extent. It allows the user to quickly and efficiently assess the impact of changes made to the model or inputs without the need to write 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.

This notebook outlines a basic On-Demand Simulation of the AWRA-L model and goes through some of the options and functionality available from the package to modify a model run and inspect the outputs.

This notebook goes through the following steps:

1. Import required libraries
2. Quick example: the default model run
3. Modifying the model configuration   
   3.1 Change forcing data  
   3.2 Change spatial grids  
   3.3 Change parameter values
4. Put model run specification together  
   4.1 Instantiate the simulator  
   4.2 Specify period and extents  
   4.3 Run the model
5. Configuring model outputs  
   5.1 Check what outputs are available  
   5.2 Add new output  
   5.3 Save outputs to files
6. More examples  
   6.1 Run over entire continent  
   6.2 Run with uniform rain
7. Exercise

##### 1. Import required libraries

In [2]:

## External Python packages

import os

import numpy as np

from matplotlib import pyplot as plt

import pandas as pd

In [3]:

## AWRAMS utilities

from awrams.utils import extents

from awrams.utils import datetools as dt

In [4]:

## AWRAMS input nodegraph. The nodegraph is created when building the input mapping

from awrams.utils.nodegraph import nodes

In [5]:

## AWRAMS parameter utilities

from awrams.utils import parameters

In [6]:

## Select simulation option

from awrams.simulation import ondemand

In [7]:

## Import the config manager - we'll use this a lot!

from awrams.utils import config\_manager

##### 2. Quick example: run the default model and settings

Quick example to show you where we are heading

In [8]:

# The model profile is our entry point into obtaining and configuring information needed for a model run

model\_profile = config\_manager.get\_model\_profile('awral','v6\_default')

# The system profile contains non-model-specific information related to the system we are running on, like file paths

# The following is equivalent to

# sys\_profile = config\_manager.get\_system\_profile('default')

sys\_profile = config\_manager.get\_system\_profile()

# Settings are dictionaries of configurable options - you use profiles to access these

sys\_settings = sys\_profile.get\_settings()

# Model settings uses the system settings dictionary to fill out its paths etc...

# You can use defaults, or pass in a sys\_settings dict

model\_settings = model\_profile.get\_settings(sys\_settings)

In [9]:

# The input mapping contains all the data related input configuration for a run (files, parameters, transforms...)

input\_map = model\_profile.get\_input\_mapping()

# If you want to use custom model settings, you can pass this in manually - more on this soon...

input\_map = model\_profile.get\_input\_mapping(model\_settings)

In [10]:

# The model object represents the actual runnable AWRA-L model, rather than abstract configuration information

# Again, this uses the model\_settings dictionary; if you leave this argument out it will use the defaults

model = model\_profile.get\_model(model\_settings)

In [11]:

sim = ondemand.OnDemandSimulator(model, input\_map) # Define a simulator with available params

results = sim.run(dt.dates('1 Jan 2010', '2 Jan 2010'), extents.get\_default\_extent()) # Runs over the whole country for one day

##### 3. Modifying the model configuration

###### 3.1 Examine the default input nodegraph

In [12]:

# Get our default input map again

input\_map = model\_profile.get\_input\_mapping()

In [13]:

# What's in the map?

list(input\_map)

Out[13]:

['alb\_dry\_hrudr',

'alb\_dry\_hrusr',

'alb\_wet\_hrudr',

'alb\_wet\_hrusr',

'cgsmax\_hrudr',

'cgsmax\_hrusr',

'er\_frac\_ref\_hrudr',

'fsoilemax\_hrudr',

'fsoilemax\_hrusr',

'fvegref\_g\_hrudr',

'fvegref\_g\_hrusr',

'gfrac\_max\_hrudr',

'gfrac\_max\_hrusr',

'hveg\_hrusr',

'k\_gw\_scale',

'k\_rout\_int',

'k\_rout\_scale',

'k0sat\_scale',

'kdsat\_scale',

'kr\_coeff',

'kssat\_scale',

'lairef\_hrudr',

'lairef\_hrusr',

'ne\_scale',

'pref\_gridscale',

'rd\_hrudr',

'rd\_hrusr',

's\_sls\_hrudr',

's\_sls\_hrusr',

's0max\_scale',

'sdmax\_scale',

'sla\_hrudr',

'sla\_hrusr',

'slope\_coeff',

'ssmax\_scale',

'tgrow\_hrudr',

'tgrow\_hrusr',

'tsenc\_hrudr',

'tsenc\_hrusr',

'ud0\_hrudr',

'ud0\_hrusr',

'us0\_hrudr',

'us0\_hrusr',

'vc\_hrudr',

'vc\_hrusr',

'w0lime\_hrudr',

'w0lime\_hrusr',

'w0ref\_alb\_hrudr',

'w0ref\_alb\_hrusr',

'wdlimu\_hrudr',

'wdlimu\_hrusr',

'wslimu\_hrudr',

'wslimu\_hrusr',

'f\_tree\_grid',

'height\_grid',

'hveg\_dr\_grid',

'k0sat\_grid',

'k\_gw\_grid',

'kdsat\_grid',

'kssat\_grid',

'lai\_max\_grid',

'meanpet\_grid',

'ne\_grid',

'pref\_grid',

's0fracawc\_grid',

'slope\_grid',

'ssfracawc\_grid',

'sdfracawc\_grid',

'tmin\_f',

'tmax\_f',

'precip\_f',

'solar\_f',

'wind\_f',

'tmin',

'tmax',

'hypsperc\_f',

'hypsperc',

'u2t',

'height',

'er\_frac\_ref\_hrusr',

'k\_rout',

'k\_gw',

's0max',

'ssmax',

'sdmax',

'k0sat',

'kssat',

'kdsat',

'kr\_0s',

'kr\_sd',

'prefr',

'fhru\_hrusr',

'fhru\_hrudr',

'ne',

'slope',

'hveg\_hrudr',

'laimax\_hrusr',

'laimax\_hrudr',

'pair',

'pt',

'rgt',

'tat',

'avpt',

'radcskyt',

'init\_sr',

'init\_sg',

'init\_mleaf\_hrusr',

'init\_s0\_hrusr',

'init\_ss\_hrusr',

'init\_sd\_hrusr',

'init\_mleaf\_hrudr',

'init\_s0\_hrudr',

'init\_ss\_hrudr',

'init\_sd\_hrudr']

In [14]:

input\_map['hveg\_hrusr']

Out[14]:

parameter([]):{'value': 0.5, 'min\_val': 0.1, 'max\_val': 50.0, 'fixed': True, 'description': 'HRU:SR Height of Vegetation Canopy'}

Changing the inputs to the model can be done by operating on the input map.   
This can be done by changing properties of the existing items, inserting new values, or by using functions that transform the map

###### 3.2 Change forcing inputs

In [15]:

# The system settings dictionary contains several preconfigured climate datasets

# Have a look at the default config file for examples of how these are laid out; this is a useful template if you

# want to add your own later on

print(list(sys\_settings.CLIMATE\_DATASETS))

['TRAINING', 'TESTING']

In [16]:

# Let's have a look at the training dataset

sys\_settings.CLIMATE\_DATASETS['TRAINING']

Out[16]:

CLIMATOLOGY: {

solar: ('<path\_to\_repository>/awrams\_cm/awrams/data/climatology/climatology\_daily\_solar\_exposure\_day.nc', 'solar\_exposure\_day')

}

FORCING: {

MAPPING: {

precip: ('rain\_day/rain\_day\*.nc', 'rain\_day')

solar: ('solar\_exposure\_day/solar\_exposure\_day\*.nc', 'solar\_exposure\_day')

tmax: ('temp\_max\_day/temp\_max\_day\*.nc', 'temp\_max\_day')

tmin: ('temp\_min\_day/temp\_min\_day\*.nc', 'temp\_min\_day')

wind: ('wind/wind\*.nc', 'wind')

}

PATH: '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap'

}

In [17]:

# As you can see, this is identical to the model defaults

# This is because we used sys\_settings.CLIMATE\_DATASETS['TRAINING'] in the default model config; ie there is no

# need to manually copy these paths, just refer to the existing dataset

model\_settings.CLIMATE\_DATASET

Out[17]:

CLIMATOLOGY: {

solar: ('<path\_to\_repository>/awrams\_cm/awrams/data/climatology/climatology\_daily\_solar\_exposure\_day.nc', 'solar\_exposure\_day')

}

FORCING: {

MAPPING: {

precip: ('rain\_day/rain\_day\*.nc', 'rain\_day')

solar: ('solar\_exposure\_day/solar\_exposure\_day\*.nc', 'solar\_exposure\_day')

tmax: ('temp\_max\_day/temp\_max\_day\*.nc', 'temp\_max\_day')

tmin: ('temp\_min\_day/temp\_min\_day\*.nc', 'temp\_min\_day')

wind: ('wind/wind\*.nc', 'wind')

}

PATH: '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap'

}

In [18]:

# You can see these reflected in the forcing items of the input map; these are appended with '\_f'

# to indicate that the forcing data is sourced from a file (this is just a convention, but worth remembering)

input\_map['tmax\_f']

Out[18]:

forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'temp\_max\_day/temp\_max\_day\*.nc', 'nc\_var': 'temp\_max\_day', 'cache': False, 'map\_func': None}

In [19]:

# It's easy to change datasets to any of the defaults, or you can construct your own as a dictionary

model\_settings.CLIMATE\_DATASET = sys\_settings.CLIMATE\_DATASETS['TESTING']

# Remember to regenerate the input map using your updated settings...

input\_map = model\_profile.get\_input\_mapping(model\_settings)

# The forcing nodes reflect the updated paths

input\_map['tmax\_f']

Out[19]:

forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/test\_data/simulation/climate', 'pattern': 'temp\_max\_day/temp\_max\_day\*.nc', 'nc\_var': 'temp\_max\_day', 'cache': False, 'map\_func': None}

In [20]:

# Alternatively, you can operate directly on the input map itself. This is convenient for quickly pointing to new data

# without editing config files

input\_map['tmax\_f'] = nodes.forcing\_from\_ncfiles(model\_settings.CLIMATE\_DATASET.FORCING.PATH,\

model\_settings.CLIMATE\_DATASET.FORCING.MAPPING['tmax'][0],

model\_settings.CLIMATE\_DATASET.FORCING.MAPPING['tmax'][1])

###### 3.3 Changing the spatial input grids

## Example. Not run

In [21]:

model\_settings.SPATIAL\_FILE

Out[21]:

'<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5'

In [22]:

import h5py

h = h5py.File(model\_settings.SPATIAL\_FILE,'r')

In [23]:

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

Out[23]:

['f\_tree',

'height',

'hveg\_dr',

'k0sat',

'k\_gw',

'kdsat',

'kssat',

'lai\_max',

'meanPET',

'ne',

'pref',

's0fracAWC',

'sdfracAWC',

'slope',

'ssfracAWC']

In [24]:

kdsat\_grid = h['parameters']['kdsat'][:]

In [25]:

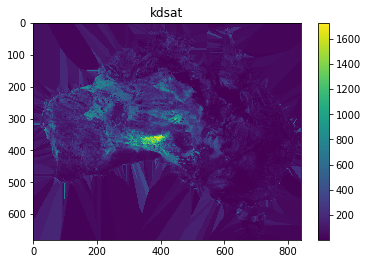
im = plt.imshow(kdsat\_grid)

plt.title("kdsat")

plt.colorbar(im)

#Don't panic at the disco effect. Grids have been infilled to cater for potential edge effects

Out[25]:



In [26]:

# Couple of ways you can look at the mapping for a particular grid

input\_map['kdsat\_grid']

Out[26]:

spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/kdsat', 'preload': False, 'force\_grid': None}

In [27]:

# If you want to modify any of the arguments

input\_map.kdsat\_grid.args

Out[27]:

{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/kdsat', 'preload': False, 'force\_grid': None}

In [28]:

# If you want to modify any of the arguments

input\_map.kdsat\_grid.args['filename']

Out[28]:

'<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5'

In [29]:

# The nodes library allows you to point to different grid file (.nc, .flt, anything recognised by gdal) to load up the data

# input\_map.f\_tree\_grid = nodes.spatial\_from\_file(PATH\_TO\_NEW\_FTREE\_FILE)

###### 3.4 How to change a parameter

In [30]:

# In most cases there are convenience functions for dealing with common tasks

# Use the parameters module we imported from awrams.utils earlier;

# The following Dataframe uses a standard layout than many parts of AWRAMS understands

param\_df = parameters.input\_map\_to\_param\_df(input\_map)

param\_df

Out[30]:

|  | **min\_val** | **max\_val** | **value** | **fixed** | **description** |
| --- | --- | --- | --- | --- | --- |
| **alb\_dry\_hrudr** | 0.100 | 0.50 | 0.260000 | True | HRU:DR Dry Soil Albedo |
| **alb\_dry\_hrusr** | 0.100 | 0.50 | 0.260000 | True | HRU:SR Dry Soil Albedo |
| **alb\_wet\_hrudr** | 0.100 | 0.50 | 0.160000 | True | HRU:DR Wet Soil Albedo |
| **alb\_wet\_hrusr** | 0.100 | 0.50 | 0.160000 | True | HRU:SR Wet Soil Albedo |
| **cgsmax\_hrudr** | 0.020 | 0.05 | 0.021248 | False | HRU:DR Conversion Coefficient From Vegetation ... |
| **cgsmax\_hrusr** | 0.020 | 0.05 | 0.032094 | False | HRU:SR Conversion Coefficient From Vegetation ... |
| **er\_frac\_ref\_hrudr** | 0.040 | 0.25 | 0.064258 | False | HRU:DR Ratio of Average Evaporation Rate Over ... |
| **fsoilemax\_hrudr** | 0.200 | 1.00 | 0.999608 | False | HRU:DR Soil Evaporation Scaling Factor When So... |
| **fsoilemax\_hrusr** | 0.200 | 1.00 | 0.585005 | False | HRU:SR Soil Evaporation Scaling Factor When So... |
| **fvegref\_g\_hrudr** | 0.100 | 0.25 | 0.150000 | True | HRU:DR Reference Soil Cover Fraction That Dete... |
| **fvegref\_g\_hrusr** | 0.100 | 0.25 | 0.150000 | True | HRU:SR Reference Soil Cover Fraction That Dete... |
| **gfrac\_max\_hrudr** | 0.250 | 0.50 | 0.300000 | True | HRU:DR Fraction of Daytime Net Radiation Lost ... |
| **gfrac\_max\_hrusr** | 0.250 | 0.50 | 0.300000 | True | HRU:SR Fraction of Daytime Net Radiation Lost ... |
| **hveg\_hrusr** | 0.100 | 50.00 | 0.500000 | True | HRU:SR Height of Vegetation Canopy |
| **k\_gw\_scale** | 0.100 | 10.00 | 0.939081 | False | K\_gw\_scale |
| **k\_rout\_int** | 0.050 | 3.00 | 0.165420 | False | K\_rout\_int |
| **k\_rout\_scale** | 0.050 | 3.00 | 0.048550 | False | K\_rout\_scale |
| **k0sat\_scale** | 0.100 | 10.00 | 3.892007 | False | Scale for saturated hydraulic conductivity sur... |
| **kdsat\_scale** | 0.010 | 1.00 | 0.012202 | False | Scale for saturated hydraulic conductivity dee... |
| **kr\_coeff** | 0.010 | 1.00 | 0.661590 | False | Kr\_coeff |
| **kssat\_scale** | 0.010 | 10.00 | 0.052349 | False | Scale for saturated hydraulic conductivity sha... |
| **lairef\_hrudr** | 1.300 | 2.50 | 2.500000 | True | HRU:DR Reference Leaf Area Index (at which fve... |
| **lairef\_hrusr** | 1.300 | 2.50 | 1.400000 | True | HRU:SR Reference Leaf Area Index (at which fve... |
| **ne\_scale** | 0.010 | 1.00 | 0.042534 | False | Scale for effective porosity |
| **pref\_gridscale** | 0.100 | 3.00 | 2.564201 | False | Pref\_gridscale |
| **rd\_hrudr** | 3.000 | 20.00 | 6.000000 | True | HRU:DR Rooting Depth |
| **rd\_hrusr** | 0.500 | 2.00 | 1.000000 | True | HRU:SR Rooting Depth |
| **s\_sls\_hrudr** | 0.030 | 0.80 | 0.067438 | False | HRU:DR Specific Canopy Rainfall Storage Capaci... |
| **s\_sls\_hrusr** | 0.030 | 0.80 | 0.292770 | False | HRU:SR Specific Canopy Rainfall Storage Capaci... |
| **s0max\_scale** | 0.500 | 3.00 | 2.804879 | False | Scale for Maximum water storage surface layer ... |
| **sdmax\_scale** | 0.500 | 1.00 | 0.884361 | False | Scale for Maximum water storage deep layer (Deep) |
| **sla\_hrudr** | 0.700 | 70.00 | 3.000000 | True | HRU:DR Specific Leaf Area |
| **sla\_hrusr** | 0.700 | 70.00 | 10.000000 | True | HRU:SR Specific Leaf Area |
| **slope\_coeff** | 0.010 | 1.00 | 0.438796 | False | slope\_coeff |
| **ssmax\_scale** | 0.500 | 3.00 | 1.993221 | False | Scale for Maximum water storage shallow layer ... |
| **tgrow\_hrudr** | 20.000 | 1000.00 | 1000.000000 | True | HRU:DR Characteristic Time Scale for Vegetatio... |
| **tgrow\_hrusr** | 20.000 | 1000.00 | 150.000000 | True | HRU:SR Characteristic Time Scale for Vegetatio... |
| **tsenc\_hrudr** | 10.000 | 200.00 | 60.000000 | True | HRU:DR Characteristic Time Scale for Vegetatio... |
| **tsenc\_hrusr** | 10.000 | 200.00 | 10.000000 | True | HRU:SR Characteristic Time Scale for Vegetatio... |
| **ud0\_hrudr** | 0.001 | 10.00 | 11.569894 | False | HRU:DR Maximum Root Water Uptake Rates From De... |
| **ud0\_hrusr** | 0.000 | 10.00 | 0.000000 | True | HRU:SR Maximum Root Water Uptake Rates From De... |
| **us0\_hrudr** | 1.000 | 7.00 | 6.000000 | True | HRU:DR Maximum Root Water Uptake Rates From Sh... |
| **us0\_hrusr** | 1.000 | 7.00 | 6.000000 | True | HRU:SR Maximum Root Water Uptake Rates From Sh... |
| **vc\_hrudr** | 0.050 | 1.00 | 0.350000 | True | HRU:DR Vegetation Photosynthetic Capacity Inde... |
| **vc\_hrusr** | 0.050 | 1.00 | 0.650000 | True | HRU:SR Vegetation Photosynthetic Capacity Inde... |
| **w0lime\_hrudr** | 0.600 | 0.90 | 0.850000 | True | HRU:DR Relative Top Soil Water Content at Whic... |
| **w0lime\_hrusr** | 0.600 | 0.90 | 0.850000 | True | HRU:SR Relative Top Soil Water Content at Whic... |
| **w0ref\_alb\_hrudr** | 0.200 | 0.50 | 0.300000 | True | HRU:DR Reference Value of w0 Determining the R... |
| **w0ref\_alb\_hrusr** | 0.200 | 0.50 | 0.300000 | True | HRU:SR Reference Value of w0 Determining the R... |
| **wdlimu\_hrudr** | 0.150 | 0.50 | 0.300000 | True | HRU:DR Deep Water-Limiting Relative Water Content |
| **wdlimu\_hrusr** | 0.150 | 0.50 | 0.300000 | True | HRU:SR Deep Water-Limiting Relative Water Content |
| **wslimu\_hrudr** | 0.150 | 0.50 | 0.300000 | True | HRU:DR Shallow Water-Limiting Relative Water C... |
| **wslimu\_hrusr** | 0.150 | 0.50 | 0.300000 | True | HRU:SR Shallow Water-Limiting Relative Water C... |

In [31]:

# Change some values in the DataFrame

# Set the value of a single parameter

param\_df.set\_value('ssmax\_scale','value',2.1)

# Change the 'fixed' property to True - this means the calibration system will now use

# this value directly ie. it is fixed rather than calibrated

param\_df.set\_value('ssmax\_scale','fixed',True)

param\_df.loc['ssmax\_scale']

Out[31]:

min\_val 0.5

max\_val 3

value 2.1

fixed True

description Scale for Maximum water storage shallow layer ...

Name: ssmax\_scale, dtype: object

In [32]:

# So far we've just been modifying a DataFrame; update the input map to tell AWRAMS to use the new values

input\_map = parameters.param\_df\_to\_mapping(param\_df,input\_map)

input\_map.ssmax\_scale

Out[32]:

parameter([]):{'value': 2.1, 'min\_val': 0.5, 'max\_val': 3.0, 'fixed': True, 'description': 'Scale for Maximum water storage shallow layer (Shallow)'}

In [33]:

# For simple changes, it may be easier to operate directly on the mapping..

input\_map.ssmax\_scale.args

Out[33]:

{'value': 2.1, 'min\_val': 0.5, 'max\_val': 3.0, 'fixed': True, 'description': 'Scale for Maximum water storage shallow layer (Shallow)'}

In [34]:

input\_map.ssmax\_scale.args.value = 1.95

##### 4. Put model run specification together

Like we did at the start for the default version

###### 4.1 Instantiate the simulator

In [35]:

sim = ondemand.OnDemandSimulator(model,input\_map)

###### 4.2 Define the required period and spatial extent

In [36]:

period = dt.dates('dec 2010 - jan 2011')

In [37]:

## Set the starting extent as the extent of the AWAP grid

## in the background this picks up the geospatial references associated with the grid set as default in the configuration file

## Config.py ?

extent\_default = extents.get\_default\_extent()

In [38]:

## Select a sub-area of that grid, say the Perth region [450 grid cells south, 20 grid cells across]

extent = extent\_default.ioffset[400:450,50:100]

In [39]:

extent

Out[39]:

origin: -30.0,114.5, shape: (50, 50), cell\_size: 0.05

###### 4.3 Run the model for the defined extent and period

In [40]:

results = sim.run(period, extent)

A switch is available in the model run command that allows the process to capture the inputs, i,e. forcing data and grid values that apply specifically to the extent run.

In [41]:

## 4.3.1 Option to capture model inputs

results, inputs = sim.run(period, extent, return\_inputs = True) # results holds model outputs, inputs holds model gridded inputs/parameters

In [42]:

## Have a look at some forcing input data

forcing\_keys = ['tmin\_f','tmax\_f','precip\_f','wind\_f']

climate\_inputs = {k:inputs[k] for k in forcing\_keys}

list(climate\_inputs)

Out[42]:

['tmin\_f', 'tmax\_f', 'precip\_f', 'wind\_f']

In [43]:

climate\_inputs['precip\_f'].shape

Out[43]:

(62, 50, 50)

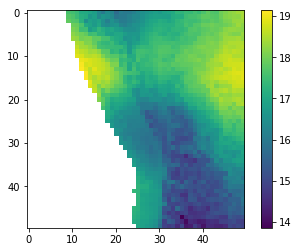
This indicates a 50x50 grid with 62 timesteps

In [44]:

im = plt.imshow(climate\_inputs['tmin\_f'][0])

plt.colorbar(im)

Out[44]:



##### 5. Configuring model outputs

###### 5.1 Check what outputs are available

This should be as per what's in the default output mapping + an extra one for the model states captured to be able to hotstart

In [45]:

list(results) # this is to see what's in the model run outputs

Out[45]:

['e0',

'etot',

'dd',

's0',

'ss',

'sd',

'qtot',

'sr',

'sg',

's0\_hrusr',

'ss\_hrusr',

'sd\_hrusr',

'mleaf\_hrusr',

's0\_hrudr',

'ss\_hrudr',

'sd\_hrudr',

'mleaf\_hrudr',

'final\_states']

In [46]:

results['ss\_hrusr']

Out[46]:

masked\_array(data =

[[[-- -- -- ..., 114.4369349693651 112.54745210954304 111.775712041619]

[-- -- -- ..., 114.42449845497929 114.13237492020525 114.11574120226166]

[-- -- -- ..., 114.35751429223369 114.242736331367 114.19190709930814]

...,

[-- -- -- ..., 114.53840442745907 114.2881122612743 114.59117339956673]

[-- -- -- ..., 114.47333630533458 114.6435959920111 114.68054942893707]

[-- -- -- ..., 114.92963310351558 114.77736785574145 114.81602667757917]]

[[-- -- -- ..., 108.71680387800343 108.78498849310108 107.9700396605628]

[-- -- -- ..., 108.44121602140434 108.30038174998543 108.84686064843429]

[-- -- -- ..., 108.90198881397079 108.72304732835093 108.59523119302705]

...,

[-- -- -- ..., 109.26848914523906 109.2510203441351 109.323393416688]

[-- -- -- ..., 109.52982327422332 109.1744085229064 108.63829604331859]

[-- -- -- ..., 108.73268531580406 108.51410387641258 108.90627215065312]]

[[-- -- -- ..., 102.34771709529687 103.67297217935072 102.84144746076126]

[-- -- -- ..., 101.92052893569694 101.88164950837113 102.91105904648275]

[-- -- -- ..., 102.89276987326302 102.64456604797134 102.48935639223501]

...,

[-- -- -- ..., 104.37786964860001 104.591230793131 104.38038141431225]

[-- -- -- ..., 104.8744187261157 104.20399148332599 103.26089818280333]

[-- -- -- ..., 103.40116593728997 103.11503061270881 103.69073720480004]]

...,

[[-- -- -- ..., 4.059723881300041 3.6819367973932655 4.269386147444381]

[-- -- -- ..., 4.411404732872124 5.40158178458388 4.755222563119076]

[-- -- -- ..., 4.8761204952989585 5.052338375631086 6.058234086266255]

...,

[-- -- -- ..., 4.304508080840092 4.755028538618111 4.800852452699001]

[-- -- -- ..., 4.10845158452996 4.482972936927684 5.15624552969872]

[-- -- -- ..., 5.379934941468914 5.06631115311433 5.48448098998151]]

[[-- -- -- ..., 5.1873796182431064 4.859976909303218 5.227370179876082]

[-- -- -- ..., 5.828319583609162 6.624773919261219 5.638762979079206]

[-- -- -- ..., 6.452649357539555 6.360530667274828 7.304481331215306]

...,

[-- -- -- ..., 12.300173156871733 12.60415109920609 11.905664563960633]

[-- -- -- ..., 12.494395154712592 12.628123335700204 12.947671927503885]

[-- -- -- ..., 14.245176767699537 13.262983259797968 13.374269295461781]]

[[-- -- -- ..., 5.039190856076261 4.839890650721992 5.1415393001527905]

[-- -- -- ..., 5.649203863581855 6.3669006749237536 5.451900374718312]

[-- -- -- ..., 6.274543863876956 6.164572243935609 7.01567546125222]

...,

[-- -- -- ..., 12.984137798328158 13.364136784071329 12.549689557990819]

[-- -- -- ..., 13.303388937503842 13.224477584840164 13.422344216308556]

[-- -- -- ..., 14.648093307973253 13.672907450647289 13.868605772137816]]],

mask =

[[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]

[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]

[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]

...,

[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]

[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]

[[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]

...,

[ True True True ..., False False False]

[ True True True ..., False False False]

[ True True True ..., False False False]]],

fill\_value = nan)

In [47]:

# Examine one of the cells (30,30) over the whole modelling period

# We'll use Pandas to make this easier...

df = pd.DataFrame(index=period)

# Examine the shallow soil layers for both HRUs

df['ss\_hrusr'] = results['ss\_hrusr'][:,30,30]

df['ss\_hrudr'] = results['ss\_hrudr'][:,30,30]

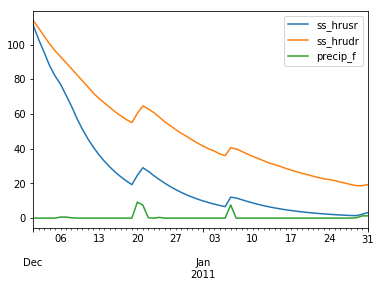
# Include some input data

df['precip\_f'] = inputs['precip\_f'][:,30,30]

df.plot()

Out[47]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3ec010fc50>



###### 5.2 Add extra outputs to the model

In [48]:

# See the currently selected set of model outputs

model\_settings.OUTPUTS

Out[48]:

OUTPUTS\_AVG: ['e0', 'etot', 'dd', 's0', 'ss', 'sd']

OUTPUTS\_CELL: ['qtot', 'sr', 'sg']

OUTPUTS\_HRU: ['s0', 'ss', 'sd', 'mleaf']

In [49]:

# Turn on the individual HRU outputs for interflow for the shallow soil layer, "ifs".

model\_settings.OUTPUTS['OUTPUTS\_HRU'].append('ifs')

In [50]:

# Now add ifs to OUTPUTS\_AVG; the area-weighted average of ifs across both HRUs

model\_settings.OUTPUTS['OUTPUTS\_AVG'].append('ifs')

In [51]:

# Obtain a new model object with the updated settings

model = model\_profile.get\_model(model\_settings)

In [52]:

## Run a simulation with the new outputs

sim = ondemand.OnDemandSimulator(model, input\_map)

results = sim.run(period, extent)

2019-03-07 01:04:57,664 INFO Rebuilding model

2019-03-07 01:04:58,029 INFO

2019-03-07 01:04:58,032 INFO

2019-03-07 01:04:58,038 INFO Build completed

In [53]:

results['ifs'].shape

Out[53]:

(62, 50, 50)

In [54]:

df = pd.DataFrame(index=period)

df['ifs\_hrusr'] = results['ifs\_hrusr'][:,30,30]

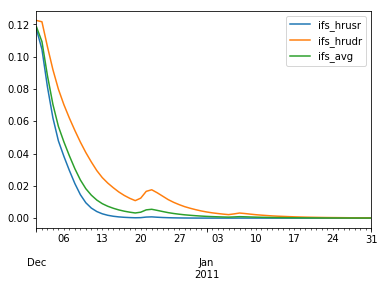
df['ifs\_hrudr'] = results['ifs\_hrudr'][:,30,30]

df['ifs\_avg'] = results['ifs'][:,30,30]

df.plot()

Out[54]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3ecc416da0>



###### 5.3 Save outputs to file

By default, data is generated only in memory, not written out to files.   
Typically you would use the SimulationServer for this purpose, however it is still possible to do so with the OnDemandSimulator

In [55]:

from awrams.simulation.support import build\_output\_mapping

In [56]:

## Re-run

In [57]:

# See what data the model is outputting (in memory)

model.get\_output\_variables()

Out[57]:

['e0',

'etot',

'dd',

's0',

'ss',

'sd',

'ifs',

'qtot',

'sr',

'sg',

's0\_hrusr',

's0\_hrudr',

'ss\_hrusr',

'ss\_hrudr',

'sd\_hrusr',

'sd\_hrudr',

'mleaf\_hrusr',

'mleaf\_hrudr',

'ifs\_hrusr',

'ifs\_hrudr']

In [58]:

# We probably don't want \_all\_ those written out

save\_vars = ['qtot','s0\_hrusr','s0\_hrudr']

# Set a path to write to

outpath = './\_results/'

output\_map = build\_output\_mapping(model, outpath, save\_vars = save\_vars)

In [59]:

sim\_with\_outputs = ondemand.OnDemandSimulator(model, input\_map, output\_map)

In [60]:

period = dt.dates('dec 2010 - jan 2011')

results, iresults = sim\_with\_outputs.run(period, extent, True)

In [61]:

# Files generated by this run...

os.listdir('./\_results')

Out[61]:

['qtot\_2011.nc',

'qtot\_2010.nc',

's0\_hrusr\_2010.nc',

's0\_hrusr\_2011.nc',

's0\_hrudr\_2011.nc',

's0\_hrudr\_2010.nc']

Additional things you can do with the model configuration, such as changing initial states and infilling gaps in the forcing inputs are presented in the <https://github.com/awracms/awra_cms/blob/AWRA_CMS_v1.2/Training/Simulation/SimulationServer.ipynb> notebook

##### 6. More examples

###### 6.1 Run over the entire continent.

This will run the model over all of Australia just for one day. Nb You are likely to have memory issues if want to run for longer periods. The size of the output data is 2M B per output variable per day.

In [62]:

period = dt.dates('jan 1 2011')

results, inputs = sim.run(period,extent\_default,True)

In [63]:

from matplotlib import rcParams

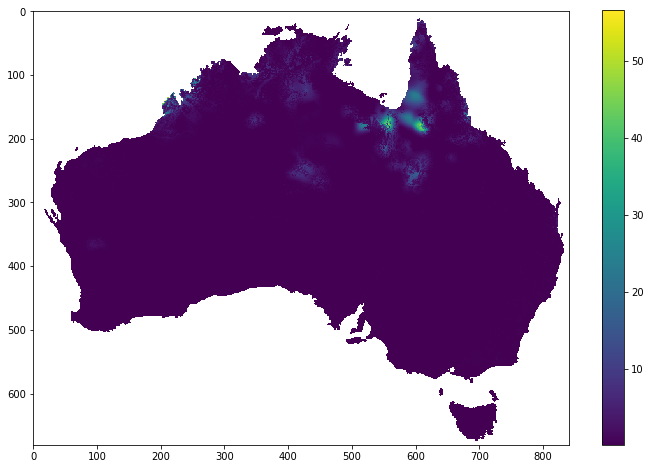
rcParams['figure.figsize'] = [12.,8.]

In [64]:

im = plt.imshow(results['qtot'][0],interpolation='None')

plt.colorbar(im)

Out[64]:



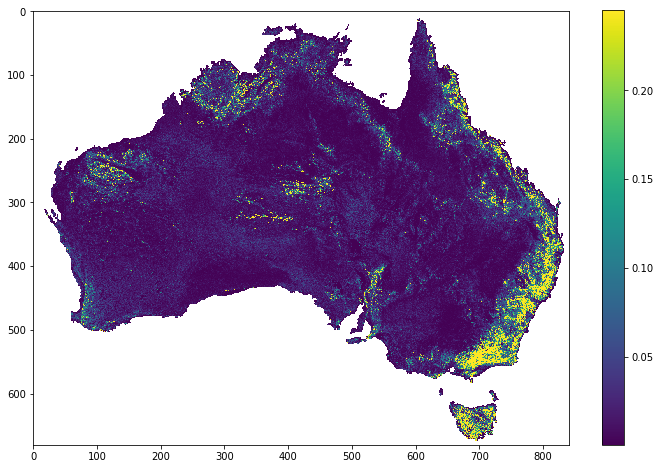
In [65]:

# Also view slope gridded input

im = plt.imshow(inputs['slope'],interpolation='None')

plt.colorbar(im)

Out[65]:



In [66]:

# Grid cell elevation range (highest point of hypsometric curve - lowest point)

im = plt.imshow(inputs['height'][-1]-inputs['height'][0],interpolation='None')

plt.colorbar(im)

Out[66]:



In [67]:

# Rainfall input on the first day of simulation

im = plt.imshow(inputs['pt'][0],interpolation='None')

plt.colorbar(im)

Out[67]:



###### 6.2 Run with a uniform rain input across the country

In [68]:

input\_map.pt = nodes.const(1000)

In [69]:

runner = ondemand.OnDemandSimulator(model, input\_map)

2019-03-07 01:05:11,808 INFO Rebuilding model

2019-03-07 01:05:12,230 INFO

2019-03-07 01:05:12,233 INFO

2019-03-07 01:05:12,238 INFO Build completed

In [70]:

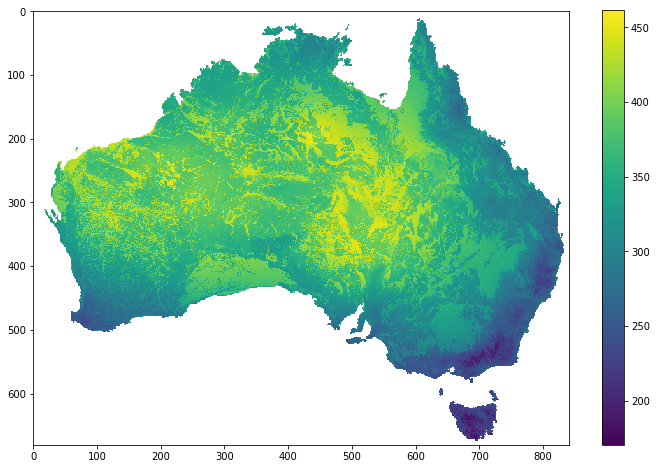
results\_fixedpt = runner.run(period, extent\_default)

In [71]:

im = plt.imshow(results\_fixedpt['qtot'][0],interpolation='None')

plt.colorbar(im)

Out[71]:



##### 7. **Exercise**

This section provides an example exercise to help increase your understanding:

Run the model over the same catchment multiple times, each time with a different parameter value for a parameter of your choice. Then plot the parameter values vs the average flow over the modelled period.

### Calibration

In [1]:

%matplotlib inline

#### Using the AWRA-CMS to run a Calibration of the AWRA-L model

This notebook demonstrates how the user might calibrate the model to multiple gauging stations.   
The default approach in the AWRACMS is to calibrate multiple catchments to a single set of parameters.

This notebook goes through the following steps:

1. Import required libraries
2. Set up calibration configuration   
   2.1 Catchments to be calibrated  
   2.2 Period to calibrate over  
   2.3 Define forcing and observation inputs  
   2.4 Select Optimiser, e,g SCE  
   2.5 Define objective functions  
   2.6 Combine into a specification dictionary
3. Run model calibration
4. Extract best parameter set
5. Visualise calibration process outputs
6. Exercises

##### 1. Import required libraries

In [2]:

from os.path import join

import h5py

from matplotlib import pyplot as plt

import pandas as pd

from awrams.utils import config\_manager, extents, gis

from awrams.utils import datetools as dt

from awrams.utils.gis import ShapefileDB

from awrams.utils.nodegraph import nodes, graph

from awrams.calibration.optimizers import sce

from awrams.calibration.objectives import test\_objectives as tobj

from awrams.calibration import support

from awrams.calibration import cluster

from awrams.calibration.launch\_calibration import run\_from\_pickle

import awrams.visualisation.vis as vis

##### 2. Define calibration configuration

###### 2.1 Pick a catchment from the default catchments dataset

In [3]:

# Use sys\_settings for paths

sys\_settings = config\_manager.get\_system\_profile().get\_settings()

base\_data\_path = sys\_settings['DATA\_PATHS']['BASE\_DATA']

catchments\_shapefile = join(base\_data\_path, 'spatial/shapefiles/Final\_list\_all\_attributes.shp')

calvalshapefile = ShapefileDB(catchments\_shapefile)

# Define the extents of the calibration

def\_extent = extents.get\_default\_extent()

In [4]:

# Single catchment

my\_extent = calvalshapefile.get\_extent\_by\_field('StationID', '003303', parent\_extent=def\_extent)

my\_extent

Out[4]:

origin: -24.25,145.45, shape: (17, 27), cell\_size: 0.05

In [5]:

## Create a dict with multiple extents

cal\_dict = {}

cal\_catchments = ['105001', '145003']

for catchment in cal\_catchments:

cal\_dict[catchment] = calvalshapefile.get\_extent\_by\_field('StationID', catchment.zfill(6), parent\_extent=def\_extent)

cal\_dict

Out[5]:

{'105001': origin: -15.2,143.6, shape: (9, 8), cell\_size: 0.05,

'145003': origin: -28.2,152.6, shape: (4, 5), cell\_size: 0.05}

In [6]:

## Check out where the catchments are

vis.show\_extent(cal\_dict['105001'], def\_extent)

vis.show\_extent(cal\_dict['145003'], def\_extent)





###### 2.2 Specify running and calibration period

In [7]:

run\_period = dt.dates('2009 - 2010')

eval\_period = dt.dates('2009 - 2010')

###### 2.3 Specify forcing inputs and observations

In [8]:

# 2.3.1 Specify observation location

observations = dict(qtot=join(base\_data\_path, 'observations/runoff/awrams\_v5\_cal\_qobs.csv'))

In [9]:

model\_profile = config\_manager.get\_model\_profile('awral', 'v6\_default')

input\_map = model\_profile.get\_input\_mapping()

# Suggested exercise: Fix any or change the ranges of the parameters to be calibrated

# Tip: Use the method shown in the simulation notebook to access the parameters and modify them

###### 2.4 Select optimiser

The **Shuffled-Complex evolution (SCE)** algorithm of Duan et al (1993) is used for finding and optimum parameter set

* + Duan, Q.Y., Gupta, V.K. & Sorooshian, S. J Optim Theory Appl (1993) 76: 501. <https://doi.org/10.1007/BF00939380>

In [10]:

evolver\_spec = support.EvolverSpec(sce.CCEvolver,

evolver\_run\_args=dict(n\_offspring=1, n\_evolutions=5, elitism=2.0))

optimizer\_spec = support.OptimizerSpec(sce.ShuffledOptimizer,

evolver\_spec=evolver\_spec,

n\_complexes=5,

max\_nsni=500,

min\_complexes=1,

max\_eval=2000) #n\_complex 14

###### 2.5 Import objective function

In [11]:

from awrams.calibration.objectives import test\_objectives as tobj

In [12]:

# %load ../../calibration/awrams/calibration/objectives/test\_objectives.py

from awrams.calibration import objectives

from awrams.calibration.support import input\_group

import numpy as np

obs = np.linspace(0.,1.,365)

def test\_nse():

nse = NSE(obs)

assert( nse(obs) == 1.0 )

assert( nse(np.repeat(obs.mean(),365)) == 0.0 )

def test\_bias():

bias = Bias(obs)

assert( bias(obs) == 0.0 )

assert( bias(obs\*2.0) == 1.0 )

class LocalQTotal:

'''

Simple sum of run

'''

#input\_schema = ['qtot','etot','dd']

input\_schema = input\_group(['qtot','etot','dd'],'volume')

output\_schema = ['qtot\_vol','etot\_vol','dd\_vol']

def \_\_init\_\_(self,obs,eval\_period):

pass

def evaluate(self,modelled):

return np.array((np.sum(modelled['qtot']),np.sum(modelled['etot']),np.sum(modelled['dd'])))

class GlobalQTotal:

output\_schema = ['qtot\_vol','etot\_vol','dd\_vol']

objective\_key = 'qtot\_vol'

def evaluate(self,l\_results):

out\_d = dict( [(k, np.sum(l\_results[k])) for k in self.output\_schema] )

return out\_d

class TestLocalSingle:

input\_schema = input\_group(['qtot'])

output\_schema = ['qtot\_nse']

def \_\_init\_\_(self,obs,eval\_period,min\_valid=15):

self.valid\_idx = {}

self.nse = {}

self.flow\_variable = 'qtot'

for k in [self.flow\_variable]:

data = obs[k]

if np.isnan(data).any():

nan\_mask = np.isnan(data)

self.valid\_idx[k] = np.where(nan\_mask == False)

else:

self.valid\_idx[k] = slice(0,len(eval\_period))

self.nse[k] = NSE(data[self.valid\_idx[k]])

def evaluate(self,modelled):

qtot\_nse = self.nse[self.flow\_variable](modelled[self.flow\_variable][self.valid\_idx[self.flow\_variable]])

return np.array(qtot\_nse)

class TestGlobalSingle:

output\_schema = ['objf\_val']

objective\_key = 'objf\_val'

def evaluate(self,l\_results):

objf\_val = 1.0 - np.mean(l\_results['qtot\_nse'])

return dict(objf\_val = objf\_val)

class TestLocalMulti:

input\_schema = input\_group(['qtot','etot'])

output\_schema = ['qtot\_nse','etot\_nse']

def \_\_init\_\_(self,obs,eval\_period,min\_valid=15,flow\_variable='qtot\_avg',et\_variable='etot\_avg'):

self.valid\_idx = {}

self.nse = {}

self.flow\_variable = flow\_variable

self.et\_variable = et\_variable

for k in [flow\_variable,et\_variable]:

data = obs[k]

if np.isnan(data).any():

nan\_mask = np.isnan(data)

self.valid\_idx[k] = np.where(nan\_mask == False)

else:

self.valid\_idx[k] = slice(0,len(eval\_period))

self.nse[k] = NSE(data[self.valid\_idx[k]])

def evaluate(self,modelled):

qtot\_nse = self.nse[self.flow\_variable](modelled[self.flow\_variable][self.valid\_idx[self.flow\_variable]])

etot\_nse = self.nse[self.et\_variable](modelled[self.et\_variable][self.valid\_idx[self.et\_variable]])

return dict(qtot\_nse=qtot\_nse,etot\_nse=etot\_nse)

class TestGlobalMultiEval:

output\_schema = ['objf\_val','qtot\_nse','etot\_nse']

objective\_key = 'objf\_val'

def evaluate(self,l\_results):

qtot\_nse = np.mean(l\_results['qtot\_nse'])

etot\_nse = np.mean(l\_results['etot\_nse'])

objf\_val = 1.0 - (qtot\_nse+etot\_nse) \* 0.5

#return 1.0 - np.mean((l\_results['qtot\_nse'] + l\_results['etot\_nse']) \* 0.5)

return dict(objf\_val = objf\_val, qtot\_nse=qtot\_nse, etot\_nse=etot\_nse)

In [13]:

local\_objfspec = support.ObjectiveFunctionSpec(tobj.TestLocalSingle) # this function to load up the ObjFunc comes from awrams.calibration.support

global\_objfspec = tobj.TestGlobalSingle

In [14]:

objective\_spec = support.ObjectiveSpec(global\_objfspec,

local\_objfspec,

observations,

eval\_period)

###### 2.6. Build spec dict

Assemble above settings into specification dictionary

In [15]:

# Get an instance of the model

model = model\_profile.get\_model()

In [16]:

# Create the calibration specification dictionary

'''

User specifiable calibration description

'''

cal\_spec = {}

cal\_spec['optimizer\_spec'] = optimizer\_spec

cal\_spec['objective\_spec'] = objective\_spec

cal\_spec['extent\_map'] = cal\_dict

cal\_spec['run\_period'] = run\_period

cal\_spec['model'] = model

cal\_spec['node\_mapping'] = input\_map

cal\_spec['logfile'] = './calres.h5'

In [17]:

# Save a file containing all the calibration specifications

nnodes = 1

ncores = 4

cluster.build\_pickle\_from\_spec(cal\_spec, ncores, nnodes, 'test\_cal.pkl')

Out[17]:

{'optimizer\_spec': <awrams.calibration.support.OptimizerSpec at 0x7fc07d9a88d0>,

'objective\_spec': <awrams.calibration.support.ObjectiveSpec at 0x7fc07d934278>,

'extent\_map': {'105001': origin: -15.2,143.6, shape: (9, 8), cell\_size: 0.05,

'145003': origin: -28.2,152.6, shape: (4, 5), cell\_size: 0.05},

'run\_period': DatetimeIndex(['2009-01-01', '2009-01-02', '2009-01-03', '2009-01-04',

'2009-01-05', '2009-01-06', '2009-01-07', '2009-01-08',

'2009-01-09', '2009-01-10',

...

'2010-12-22', '2010-12-23', '2010-12-24', '2010-12-25',

'2010-12-26', '2010-12-27', '2010-12-28', '2010-12-29',

'2010-12-30', '2010-12-31'],

dtype='datetime64[ns]', length=730, freq='D'),

'model': <awrams.models.awral.model.AWRALModel at 0x7fc07d9346a0>,

'node\_mapping': alb\_dry\_hrudr: parameter([]):{'value': 0.26, 'min\_val': 0.1, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Dry Soil Albedo'}

alb\_dry\_hrusr: parameter([]):{'value': 0.26, 'min\_val': 0.1, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Dry Soil Albedo'}

alb\_wet\_hrudr: parameter([]):{'value': 0.16, 'min\_val': 0.1, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Wet Soil Albedo'}

alb\_wet\_hrusr: parameter([]):{'value': 0.16, 'min\_val': 0.1, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Wet Soil Albedo'}

avpt: transform(['tmin']):{'tfunc': {'name': 'pe', 'objtype': 'class', 'module': 'awrams.models.awral.transforms'}, 'func\_args': {}}

cgsmax\_hrudr: parameter([]):{'value': 0.02124781, 'min\_val': 0.02, 'max\_val': 0.05, 'fixed': False, 'description': 'HRU:DR Conversion Coefficient From Vegetation Photosynthetic Capacity Index to Maximum Stomatal Conductance'}

cgsmax\_hrusr: parameter([]):{'value': 0.03209419, 'min\_val': 0.02, 'max\_val': 0.05, 'fixed': False, 'description': 'HRU:SR Conversion Coefficient From Vegetation Photosynthetic Capacity Index to Maximum Stomatal Conductance'}

er\_frac\_ref\_hrudr: parameter([]):{'value': 0.06425805, 'min\_val': 0.04, 'max\_val': 0.25, 'fixed': False, 'description': 'HRU:DR Ratio of Average Evaporation Rate Over Average Rainfall Intensity During Storms Per Unit Canopy Cover'}

er\_frac\_ref\_hrusr: transform(('er\_frac\_ref\_hrudr', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

f\_tree\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/f\_tree', 'preload': False, 'force\_grid': None}

fhru\_hrudr: assign(['f\_tree\_grid']):{}

fhru\_hrusr: transform([1.0, 'f\_tree\_grid']):{'tfunc': {'name': 'subtract', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

fsoilemax\_hrudr: parameter([]):{'value': 0.99960822, 'min\_val': 0.2, 'max\_val': 1.0, 'fixed': False, 'description': 'HRU:DR Soil Evaporation Scaling Factor When Soil Water Supply is Not Limiting Evaporation'}

fsoilemax\_hrusr: parameter([]):{'value': 0.58500527, 'min\_val': 0.2, 'max\_val': 1.0, 'fixed': False, 'description': 'HRU:SR Soil Evaporation Scaling Factor When Soil Water Supply is Not Limiting Evaporation'}

fvegref\_g\_hrudr: parameter([]):{'value': 0.15, 'min\_val': 0.1, 'max\_val': 0.25, 'fixed': True, 'description': 'HRU:DR Reference Soil Cover Fraction That Determines The Rate of Decline in Soil Heat Flux With Increasing Canopy Cover'}

fvegref\_g\_hrusr: parameter([]):{'value': 0.15, 'min\_val': 0.1, 'max\_val': 0.25, 'fixed': True, 'description': 'HRU:SR Reference Soil Cover Fraction That Determines The Rate of Decline in Soil Heat Flux With Increasing Canopy Cover'}

gfrac\_max\_hrudr: parameter([]):{'value': 0.3, 'min\_val': 0.25, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Fraction of Daytime Net Radiation Lost To Soil Heat Storage for an Unvegetated Surface'}

gfrac\_max\_hrusr: parameter([]):{'value': 0.3, 'min\_val': 0.25, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Fraction of Daytime Net Radiation Lost To Soil Heat Storage for an Unvegetated Surface'}

height: assign(['height\_grid']):{}

height\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/height', 'preload': False, 'force\_grid': None}

hveg\_dr\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/hveg\_dr', 'preload': False, 'force\_grid': None}

hveg\_hrudr: assign(['hveg\_dr\_grid']):{}

hveg\_hrusr: parameter([]):{'value': 0.5, 'min\_val': 0.1, 'max\_val': 50.0, 'fixed': True, 'description': 'HRU:SR Height of Vegetation Canopy'}

hypsperc: transform(('hypsperc\_f', 0.01)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

hypsperc\_f: const\_from\_hdf5([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'dimensions/hypsometric\_percentile', 'dims': ['hypsometric\_percentile']}

init\_mleaf\_hrudr: transform([2.0, 'sla\_hrudr']):{'tfunc': {'name': 'true\_divide', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_mleaf\_hrusr: transform([2.0, 'sla\_hrusr']):{'tfunc': {'name': 'true\_divide', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_s0\_hrudr: transform(('s0max', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_s0\_hrusr: transform(('s0max', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_sd\_hrudr: transform(('sdmax', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_sd\_hrusr: transform(('sdmax', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_sg: constant([]):{'value': 100.0}

init\_sr: constant([]):{'value': 0.0}

init\_ss\_hrudr: transform(('ssmax', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

init\_ss\_hrusr: transform(('ssmax', 0.5)):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

k0sat: transform(('k0sat\_scale', 'k0sat\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

k0sat\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/k0sat', 'preload': False, 'force\_grid': None}

k0sat\_scale: parameter([]):{'value': 3.89200702, 'min\_val': 0.1, 'max\_val': 10.0, 'fixed': False, 'description': 'Scale for saturated hydraulic conductivity surface layer'}

k\_gw: transform(('k\_gw\_scale', 'k\_gw\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

k\_gw\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/k\_gw', 'preload': False, 'force\_grid': None}

k\_gw\_scale: parameter([]):{'value': 0.93908135, 'min\_val': 0.1, 'max\_val': 10.0, 'fixed': False, 'description': 'K\_gw\_scale'}

k\_rout: transform(('k\_rout\_scale', 'k\_rout\_int', 'meanpet\_grid')):{'tfunc': {'name': 'k\_rout', 'objtype': 'class', 'module': 'awrams.models.awral.transforms'}, 'func\_args': {}}

k\_rout\_int: parameter([]):{'value': 0.16542047, 'min\_val': 0.05, 'max\_val': 3.0, 'fixed': False, 'description': 'K\_rout\_int'}

k\_rout\_scale: parameter([]):{'value': 0.04855041, 'min\_val': 0.05, 'max\_val': 3.0, 'fixed': False, 'description': 'K\_rout\_scale'}

kdsat: transform(('kdsat\_scale', 'kdsat\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

kdsat\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/kdsat', 'preload': False, 'force\_grid': None}

kdsat\_scale: parameter([]):{'value': 0.0122021, 'min\_val': 0.01, 'max\_val': 1.0, 'fixed': False, 'description': 'Scale for saturated hydraulic conductivity deep layer'}

kr\_0s: transform(('k0sat', 'kssat')):{'tfunc': {'name': 'interlayer\_k', 'objtype': 'class', 'module': 'awrams.models.awral.transforms'}, 'func\_args': {}}

kr\_coeff: parameter([]):{'value': 0.66159026, 'min\_val': 0.01, 'max\_val': 1.0, 'fixed': False, 'description': 'Kr\_coeff'}

kr\_sd: transform(('kssat', 'kdsat')):{'tfunc': {'name': 'interlayer\_k', 'objtype': 'class', 'module': 'awrams.models.awral.transforms'}, 'func\_args': {}}

kssat: transform(('kssat\_scale', 'kssat\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

kssat\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/kssat', 'preload': False, 'force\_grid': None}

kssat\_scale: parameter([]):{'value': 0.05234894, 'min\_val': 0.01, 'max\_val': 10.0, 'fixed': False, 'description': 'Scale for saturated hydraulic conductivity shallow layer'}

lai\_max\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/lai\_max', 'preload': False, 'force\_grid': None}

laimax\_hrudr: assign(['lai\_max\_grid']):{}

laimax\_hrusr: assign(['lai\_max\_grid']):{}

lairef\_hrudr: parameter([]):{'value': 2.5, 'min\_val': 1.3, 'max\_val': 2.5, 'fixed': True, 'description': 'HRU:DR Reference Leaf Area Index (at which fveg = 0.63)'}

lairef\_hrusr: parameter([]):{'value': 1.4, 'min\_val': 1.3, 'max\_val': 2.5, 'fixed': True, 'description': 'HRU:SR Reference Leaf Area Index (at which fveg = 0.63)'}

meanpet\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/meanPET', 'preload': False, 'force\_grid': None}

ne: transform(('ne\_scale', 'ne\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

ne\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/ne', 'preload': False, 'force\_grid': None}

ne\_scale: parameter([]):{'value': 0.04253378, 'min\_val': 0.01, 'max\_val': 1.0, 'fixed': False, 'description': 'Scale for effective porosity'}

pair: constant([]):{'value': 97500.0}

precip\_f: forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'rain\_day/rain\_day\*.nc', 'nc\_var': 'rain\_day', 'cache': False, 'map\_func': None}

pref\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/pref', 'preload': False, 'force\_grid': None}

pref\_gridscale: parameter([]):{'value': 2.56420087, 'min\_val': 0.1, 'max\_val': 3.0, 'fixed': False, 'description': 'Pref\_gridscale'}

prefr: transform(('pref\_gridscale', 'pref\_grid')):{'tfunc': {'name': 'multiply', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

pt: assign(['precip\_f']):{}

radcskyt: radcskyt([]):{}

rd\_hrudr: parameter([]):{'value': 6.0, 'min\_val': 3.0, 'max\_val': 20.0, 'fixed': True, 'description': 'HRU:DR Rooting Depth'}

rd\_hrusr: parameter([]):{'value': 1.0, 'min\_val': 0.5, 'max\_val': 2.0, 'fixed': True, 'description': 'HRU:SR Rooting Depth'}

rgt: transform(['solar\_f', 0.1]):{'tfunc': {'name': 'maximum', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

s0fracawc\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/s0fracAWC', 'preload': False, 'force\_grid': None}

s0max: transform(('s0max\_scale', 's0fracawc\_grid', 100.0)):{'tfunc': {'name': '\_mul3', 'objtype': 'class', 'module': 'awrams.utils.nodegraph.nodes'}, 'func\_args': {}}

s0max\_scale: parameter([]):{'value': 2.80487914, 'min\_val': 0.5, 'max\_val': 3.0, 'fixed': False, 'description': 'Scale for Maximum water storage surface layer (Top)'}

s\_sls\_hrudr: parameter([]):{'value': 0.06743782, 'min\_val': 0.03, 'max\_val': 0.8, 'fixed': False, 'description': 'HRU:DR Specific Canopy Rainfall Storage Capacity Per Unit Leaf Area'}

s\_sls\_hrusr: parameter([]):{'value': 0.29276959, 'min\_val': 0.03, 'max\_val': 0.8, 'fixed': False, 'description': 'HRU:SR Specific Canopy Rainfall Storage Capacity Per Unit Leaf Area'}

sdfracawc\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/sdfracAWC', 'preload': False, 'force\_grid': None}

sdmax: transform(('sdmax\_scale', 'sdfracawc\_grid', 5000.0)):{'tfunc': {'name': '\_mul3', 'objtype': 'class', 'module': 'awrams.utils.nodegraph.nodes'}, 'func\_args': {}}

sdmax\_scale: parameter([]):{'value': 0.88436116, 'min\_val': 0.5, 'max\_val': 1.0, 'fixed': False, 'description': 'Scale for Maximum water storage deep layer (Deep)'}

sla\_hrudr: parameter([]):{'value': 3.0, 'min\_val': 0.7, 'max\_val': 70.0, 'fixed': True, 'description': 'HRU:DR Specific Leaf Area'}

sla\_hrusr: parameter([]):{'value': 10.0, 'min\_val': 0.7, 'max\_val': 70.0, 'fixed': True, 'description': 'HRU:SR Specific Leaf Area'}

slope: assign(['slope\_grid']):{}

slope\_coeff: parameter([]):{'value': 0.43879647, 'min\_val': 0.01, 'max\_val': 1.0, 'fixed': False, 'description': 'slope\_coeff'}

slope\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/slope', 'preload': False, 'force\_grid': None}

solar\_f: forcing\_with\_climatology([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'solar\_exposure\_day/solar\_exposure\_day\*.nc', 'nc\_var': 'solar\_exposure\_day', 'filler\_fn': '<path\_to\_repository>/awrams\_cm/awrams/data/climatology/climatology\_daily\_solar\_exposure\_day.nc'}

ssfracawc\_grid: spatial\_from\_file([]):{'filename': '<path\_to\_repository>/awrams\_cm/awrams/data/model\_data/awral/spatial\_parameters\_v6.h5', 'variable': 'parameters/ssfracAWC', 'preload': False, 'force\_grid': None}

ssmax: transform(('ssmax\_scale', 'ssfracawc\_grid', 900.0)):{'tfunc': {'name': '\_mul3', 'objtype': 'class', 'module': 'awrams.utils.nodegraph.nodes'}, 'func\_args': {}}

ssmax\_scale: parameter([]):{'value': 1.99322065, 'min\_val': 0.5, 'max\_val': 3.0, 'fixed': False, 'description': 'Scale for Maximum water storage shallow layer (Shallow)'}

tat: transform(['tmin', 'tmax']):{'tfunc': {'name': '\_mix', 'objtype': 'class', 'module': 'awrams.utils.nodegraph.nodes'}, 'func\_args': {'ratio': 0.75}}

tgrow\_hrudr: parameter([]):{'value': 1000.0, 'min\_val': 20.0, 'max\_val': 1000.0, 'fixed': True, 'description': 'HRU:DR Characteristic Time Scale for Vegetation Growth Towards Equilibrium'}

tgrow\_hrusr: parameter([]):{'value': 150.0, 'min\_val': 20.0, 'max\_val': 1000.0, 'fixed': True, 'description': 'HRU:SR Characteristic Time Scale for Vegetation Growth Towards Equilibrium'}

tmax: transform(['tmin\_f', 'tmax\_f']):{'tfunc': {'name': 'maximum', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

tmax\_f: forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'temp\_max\_day/temp\_max\_day\*.nc', 'nc\_var': 'temp\_max\_day', 'cache': False, 'map\_func': None}

tmin: transform(['tmin\_f', 'tmax\_f']):{'tfunc': {'name': 'minimum', 'objtype': 'class', 'module': 'numpy'}, 'func\_args': {}}

tmin\_f: forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'temp\_min\_day/temp\_min\_day\*.nc', 'nc\_var': 'temp\_min\_day', 'cache': False, 'map\_func': None}

tsenc\_hrudr: parameter([]):{'value': 60.0, 'min\_val': 10.0, 'max\_val': 200.0, 'fixed': True, 'description': 'HRU:DR Characteristic Time Scale for Vegetation Senescence Towards Equilibrium'}

tsenc\_hrusr: parameter([]):{'value': 10.0, 'min\_val': 10.0, 'max\_val': 200.0, 'fixed': True, 'description': 'HRU:SR Characteristic Time Scale for Vegetation Senescence Towards Equilibrium'}

u2t: assign(['wind\_f']):{}

ud0\_hrudr: parameter([]):{'value': 11.569894099999999, 'min\_val': 0.001, 'max\_val': 10.0, 'fixed': False, 'description': 'HRU:DR Maximum Root Water Uptake Rates From Deep Soil'}

ud0\_hrusr: parameter([]):{'value': 0.0, 'min\_val': 0.0, 'max\_val': 10.0, 'fixed': True, 'description': 'HRU:SR Maximum Root Water Uptake Rates From Deep Soil'}

us0\_hrudr: parameter([]):{'value': 6.0, 'min\_val': 1.0, 'max\_val': 7.0, 'fixed': True, 'description': 'HRU:DR Maximum Root Water Uptake Rates From Shallow Soil'}

us0\_hrusr: parameter([]):{'value': 6.0, 'min\_val': 1.0, 'max\_val': 7.0, 'fixed': True, 'description': 'HRU:SR Maximum Root Water Uptake Rates From Shallow Soil'}

vc\_hrudr: parameter([]):{'value': 0.35, 'min\_val': 0.05, 'max\_val': 1.0, 'fixed': True, 'description': 'HRU:DR Vegetation Photosynthetic Capacity Index Per Unit Canopy Cover'}

vc\_hrusr: parameter([]):{'value': 0.65, 'min\_val': 0.05, 'max\_val': 1.0, 'fixed': True, 'description': 'HRU:SR Vegetation Photosynthetic Capacity Index Per Unit Canopy Cover'}

w0lime\_hrudr: parameter([]):{'value': 0.85, 'min\_val': 0.6, 'max\_val': 0.9, 'fixed': True, 'description': 'HRU:DR Relative Top Soil Water Content at Which Evaporation is Reduced'}

w0lime\_hrusr: parameter([]):{'value': 0.85, 'min\_val': 0.6, 'max\_val': 0.9, 'fixed': True, 'description': 'HRU:SR Relative Top Soil Water Content at Which Evaporation is Reduced'}

w0ref\_alb\_hrudr: parameter([]):{'value': 0.3, 'min\_val': 0.2, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Reference Value of w0 Determining the Rate of Albedo Decrease With Wetness'}

w0ref\_alb\_hrusr: parameter([]):{'value': 0.3, 'min\_val': 0.2, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Reference Value of w0 Determining the Rate of Albedo Decrease With Wetness'}

wdlimu\_hrudr: parameter([]):{'value': 0.3, 'min\_val': 0.15, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Deep Water-Limiting Relative Water Content'}

wdlimu\_hrusr: parameter([]):{'value': 0.3, 'min\_val': 0.15, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Deep Water-Limiting Relative Water Content'}

wind\_f: forcing\_from\_ncfiles([]):{'path': '<path\_to\_repository>/awrams\_cm/awrams/data/training/climate/bom\_awap', 'pattern': 'wind/wind\*.nc', 'nc\_var': 'wind', 'cache': False, 'map\_func': None}

wslimu\_hrudr: parameter([]):{'value': 0.3, 'min\_val': 0.15, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:DR Shallow Water-Limiting Relative Water Content'}

wslimu\_hrusr: parameter([]):{'value': 0.3, 'min\_val': 0.15, 'max\_val': 0.5, 'fixed': True, 'description': 'HRU:SR Shallow Water-Limiting Relative Water Content'},

'logfile': './calres.h5',

'node\_alloc': OrderedDict([(0,

{'catchments': [('105001',

{'start\_cell': 0, 'ncells': 49},

{'owns': True, 'remote': False}),

('145003',

{'start\_cell': 0, 'ncells': 16},

{'owns': True, 'remote': False})],

'cell\_counts': [49, 16]})]),

'catch\_node\_map': {'145003': [0], '105001': [0]},

'n\_workers': 1,

'n\_sub\_workers': 4}

##### 3. Run the calibration

In [18]:

cal = run\_from\_pickle('./test\_cal.pkl')

s: 23568

Node starting on awraprod2.bom.gov.au

Logger setup complete

evaluating initial population

{}

.........

running

{'n\_eval': 215, 'n\_shuffle': 0, 'best\_params': Score: 0.736880821118, Params: [ 0.0458999 0.04326281 0.2074315 0.82519241 0.65576483 7.29582519

4.62578128 2.41653654 2.2434171 0.86762866 0.9282149 0.1462411

0.48561721 2.75974071 0.92483746 0.06020117 0.36121213 0.71562666

0.53588317 1.67661838 6.9802402 ], Meta: None}

...

running

{'n\_eval': 1993, 'n\_shuffle': 259, 'best\_params': Score: 0.448452564874, Params: [ 0.02427486 0.04953662 0.04004621 0.99903329 0.99701031 9.5164319

0.72253581 0.45342914 0.05005853 0.9737941 0.67728464 0.05942686

0.31669019 2.99686114 1.35706566 0.03120805 0.03249839 0.99973579

0.13308893 1.37225413 9.95132623], Meta: None}

complete

{'condition': 'Maximum evaluations completed', 'n\_eval': 2000, 'best\_params': Score: 0.448439719471, Params: [ 0.02498835 0.0494128 0.04001745 0.99970235 0.99723899 9.49629781

0.71502023 0.43395212 0.05008656 0.99112018 0.68805701 0.05833789

0.31433659 2.99930748 1.32969116 0.03077901 0.03210068 0.99910553

0.2205591 1.38994553 9.93292314], Meta: None}

Completed

Server exiting, attempting cleanup

Logger terminate received

Logger finally

Logger file close

Termination request received from server

Node 23569 terminating

Logger closed

Optimizer closed

Server final exit

Node 23569 complete

426.73 seconds

##### 4. Extract best parameter set

The calibration process creates a file containing all the parameters and objective scores of each iteration of the calibration.

In [19]:

cr = support.CalibrationResults('./calres.h5')

In [20]:

# These are the parameters we calibrated against

cr.get\_best\_paramset()

Out[20]:

|  | **min\_val** | **max\_val** | **value** | **fixed** | **description** |
| --- | --- | --- | --- | --- | --- |
| **alb\_dry\_hrudr** | 0.100 | 0.50 | 0.260000 | True | HRU:DR Dry Soil Albedo |
| **alb\_dry\_hrusr** | 0.100 | 0.50 | 0.260000 | True | HRU:SR Dry Soil Albedo |
| **alb\_wet\_hrudr** | 0.100 | 0.50 | 0.160000 | True | HRU:DR Wet Soil Albedo |
| **alb\_wet\_hrusr** | 0.100 | 0.50 | 0.160000 | True | HRU:SR Wet Soil Albedo |
| **cgsmax\_hrudr** | 0.020 | 0.05 | 0.024988 | False | HRU:DR Conversion Coefficient From Vegetation ... |
| **cgsmax\_hrusr** | 0.020 | 0.05 | 0.049413 | False | HRU:SR Conversion Coefficient From Vegetation ... |
| **er\_frac\_ref\_hrudr** | 0.040 | 0.25 | 0.040017 | False | HRU:DR Ratio of Average Evaporation Rate Over ... |
| **fsoilemax\_hrudr** | 0.200 | 1.00 | 0.999702 | False | HRU:DR Soil Evaporation Scaling Factor When So... |
| **fsoilemax\_hrusr** | 0.200 | 1.00 | 0.997239 | False | HRU:SR Soil Evaporation Scaling Factor When So... |
| **fvegref\_g\_hrudr** | 0.100 | 0.25 | 0.150000 | True | HRU:DR Reference Soil Cover Fraction That Dete... |
| **fvegref\_g\_hrusr** | 0.100 | 0.25 | 0.150000 | True | HRU:SR Reference Soil Cover Fraction That Dete... |
| **gfrac\_max\_hrudr** | 0.250 | 0.50 | 0.300000 | True | HRU:DR Fraction of Daytime Net Radiation Lost ... |
| **gfrac\_max\_hrusr** | 0.250 | 0.50 | 0.300000 | True | HRU:SR Fraction of Daytime Net Radiation Lost ... |
| **hveg\_hrusr** | 0.100 | 50.00 | 0.500000 | True | HRU:SR Height of Vegetation Canopy |
| **k\_gw\_scale** | 0.100 | 10.00 | 0.715020 | False | K\_gw\_scale |
| **k\_rout\_int** | 0.050 | 3.00 | 0.433952 | False | K\_rout\_int |
| **k\_rout\_scale** | 0.050 | 3.00 | 0.050087 | False | K\_rout\_scale |
| **k0sat\_scale** | 0.100 | 10.00 | 9.496298 | False | Scale for saturated hydraulic conductivity sur... |
| **kdsat\_scale** | 0.010 | 1.00 | 0.991120 | False | Scale for saturated hydraulic conductivity dee... |
| **kr\_coeff** | 0.010 | 1.00 | 0.688057 | False | Kr\_coeff |
| **kssat\_scale** | 0.010 | 10.00 | 0.058338 | False | Scale for saturated hydraulic conductivity sha... |
| **lairef\_hrudr** | 1.300 | 2.50 | 2.500000 | True | HRU:DR Reference Leaf Area Index (at which fve... |
| **lairef\_hrusr** | 1.300 | 2.50 | 1.400000 | True | HRU:SR Reference Leaf Area Index (at which fve... |
| **ne\_scale** | 0.010 | 1.00 | 0.314337 | False | Scale for effective porosity |
| **pref\_gridscale** | 0.100 | 3.00 | 2.999307 | False | Pref\_gridscale |
| **rd\_hrudr** | 3.000 | 20.00 | 6.000000 | True | HRU:DR Rooting Depth |
| **rd\_hrusr** | 0.500 | 2.00 | 1.000000 | True | HRU:SR Rooting Depth |
| **s\_sls\_hrudr** | 0.030 | 0.80 | 0.030779 | False | HRU:DR Specific Canopy Rainfall Storage Capaci... |
| **s\_sls\_hrusr** | 0.030 | 0.80 | 0.032101 | False | HRU:SR Specific Canopy Rainfall Storage Capaci... |
| **s0max\_scale** | 0.500 | 3.00 | 1.329691 | False | Scale for Maximum water storage surface layer ... |
| **sdmax\_scale** | 0.500 | 1.00 | 0.999106 | False | Scale for Maximum water storage deep layer (Deep) |
| **sla\_hrudr** | 0.700 | 70.00 | 3.000000 | True | HRU:DR Specific Leaf Area |
| **sla\_hrusr** | 0.700 | 70.00 | 10.000000 | True | HRU:SR Specific Leaf Area |
| **slope\_coeff** | 0.010 | 1.00 | 0.220559 | False | slope\_coeff |
| **ssmax\_scale** | 0.500 | 3.00 | 1.389946 | False | Scale for Maximum water storage shallow layer ... |
| **tgrow\_hrudr** | 20.000 | 1000.00 | 1000.000000 | True | HRU:DR Characteristic Time Scale for Vegetatio... |
| **tgrow\_hrusr** | 20.000 | 1000.00 | 150.000000 | True | HRU:SR Characteristic Time Scale for Vegetatio... |
| **tsenc\_hrudr** | 10.000 | 200.00 | 60.000000 | True | HRU:DR Characteristic Time Scale for Vegetatio... |
| **tsenc\_hrusr** | 10.000 | 200.00 | 10.000000 | True | HRU:SR Characteristic Time Scale for Vegetatio... |
| **ud0\_hrudr** | 0.001 | 10.00 | 9.932923 | False | HRU:DR Maximum Root Water Uptake Rates From De... |
| **ud0\_hrusr** | 0.000 | 10.00 | 0.000000 | True | HRU:SR Maximum Root Water Uptake Rates From De... |
| **us0\_hrudr** | 1.000 | 7.00 | 6.000000 | True | HRU:DR Maximum Root Water Uptake Rates From Sh... |
| **us0\_hrusr** | 1.000 | 7.00 | 6.000000 | True | HRU:SR Maximum Root Water Uptake Rates From Sh... |
| **vc\_hrudr** | 0.050 | 1.00 | 0.350000 | True | HRU:DR Vegetation Photosynthetic Capacity Inde... |
| **vc\_hrusr** | 0.050 | 1.00 | 0.650000 | True | HRU:SR Vegetation Photosynthetic Capacity Inde... |
| **w0lime\_hrudr** | 0.600 | 0.90 | 0.850000 | True | HRU:DR Relative Top Soil Water Content at Whic... |
| **w0lime\_hrusr** | 0.600 | 0.90 | 0.850000 | True | HRU:SR Relative Top Soil Water Content at Whic... |
| **w0ref\_alb\_hrudr** | 0.200 | 0.50 | 0.300000 | True | HRU:DR Reference Value of w0 Determining the R... |
| **w0ref\_alb\_hrusr** | 0.200 | 0.50 | 0.300000 | True | HRU:SR Reference Value of w0 Determining the R... |
| **wdlimu\_hrudr** | 0.150 | 0.50 | 0.300000 | True | HRU:DR Deep Water-Limiting Relative Water Content |
| **wdlimu\_hrusr** | 0.150 | 0.50 | 0.300000 | True | HRU:SR Deep Water-Limiting Relative Water Content |
| **wslimu\_hrudr** | 0.150 | 0.50 | 0.300000 | True | HRU:DR Shallow Water-Limiting Relative Water C... |
| **wslimu\_hrusr** | 0.150 | 0.50 | 0.300000 | True | HRU:SR Shallow Water-Limiting Relative Water C... |

In [21]:

# Get a DataFrame of the parameter values over time (iterations)

params\_all = cr.get\_parameter\_values()

In [22]:

# Obtain the best (minimum score) parameter set

# Note that CalibrationResults.get\_best\_paramset() performs the same function

best\_params = params\_all.iloc[cr.best\_param\_index()]

best\_params

#best\_params.to\_csv('cal\_params\_test.csv')

Out[22]:

cgsmax\_hrudr 0.024988

cgsmax\_hrusr 0.049413

er\_frac\_ref\_hrudr 0.040017

fsoilemax\_hrudr 0.999702

fsoilemax\_hrusr 0.997239

k0sat\_scale 9.496298

k\_gw\_scale 0.715020

k\_rout\_int 0.433952

k\_rout\_scale 0.050087

kdsat\_scale 0.991120

kr\_coeff 0.688057

kssat\_scale 0.058338

ne\_scale 0.314337

pref\_gridscale 2.999307

s0max\_scale 1.329691

s\_sls\_hrudr 0.030779

s\_sls\_hrusr 0.032101

sdmax\_scale 0.999106

slope\_coeff 0.220559

ssmax\_scale 1.389946

ud0\_hrudr 9.932923

Name: 1994, dtype: float64

##### 5. Take a look at the calibrated model outputs

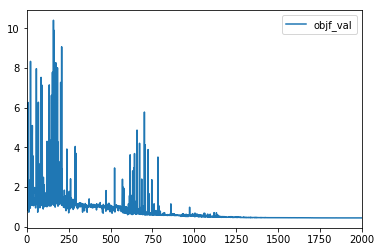
In [23]:

# Examine the change in global objective function values over time

gscores = cr.get\_global\_scores()

gscores.plot()

Out[23]:



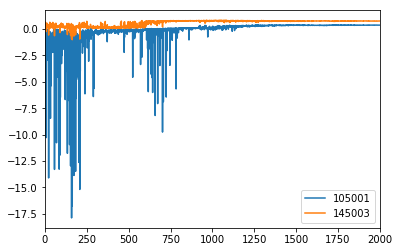
In [24]:

#Get some local (catchment level) scores

qnse\_l = cr.get\_local\_scores('qtot\_nse')

qnse\_l.plot()

Out[24]:



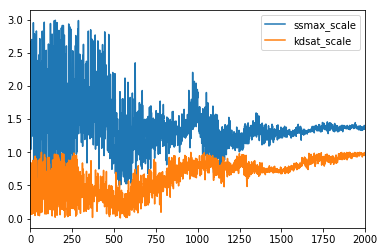
In [25]:

#Observe the evolution in parameter values over iterations

#Great for examining behaviour of parameter space for different optimizers!

params\_all[['ssmax\_scale', 'kdsat\_scale']].plot()

Out[25]:



##### 6. Compare modelled and observed outputs

In [26]:

#Load up an OnDemand simulator to run the model

from awrams.simulation import ondemand

sim = ondemand.OnDemandSimulator(model, input\_map)

In [27]:

# Load observations for comparison

obs\_df = pd.DataFrame.from\_csv(observations['qtot'])

In [28]:

# Set the new (calibrated) parameters

sim.input\_runner.set\_parameters(best\_params)

In [29]:

sim\_res = sim.run(run\_period, cal\_dict['145003'])

sim\_res['dd'].shape

Out[29]:

(730, 4, 5)

In [30]:

pd.DataFrame.from\_records(sim\_res['dd'])

Out[30]:

|  | **0** | **1** | **2** | **3** |
| --- | --- | --- | --- | --- |
| **0** | [--, 9.08104628905, 13.2426937464, 10.98628124... | [10.7947291418, 9.97809832922, 16.9347795001, ... | [7.87273833135, 7.10127097323, 14.4202605666, ... | [--, 7.93619359574, 10.8562891348, 12.98757251... |
| **1** | [--, 8.58885467729, 12.2641331787, 10.27926792... | [10.1320001256, 9.42034801627, 15.3780628579, ... | [7.52214009654, 6.81174650618, 13.2768698573, ... | [--, 7.55826042278, 10.1659624968, 12.01899162... |
| **2** | [--, 8.13811558074, 11.3907819692, 9.637905578... | [9.53300247557, 8.91392348442, 14.0320740913, ... | [7.19927595955, 6.54561078184, 12.2787473205, ... | [--, 7.21736630237, 9.55641645676, 11.17080577... |
| **3** | [--, 7.70780436772, 10.5881698892, 9.040667998... | [8.9697282461, 8.4366543941, 12.83517792, 9.78... | [6.88826703996, 6.28765225842, 11.3718072719, ... | [--, 6.88612979705, 8.9814237528, 10.388471053... |
| **4** | [--, 7.28434277204, 9.83152599319, 8.464976865... | [8.41249074428, 7.96205365225, 11.7254297869, ... | [6.56751612922, 6.01987530622, 10.509608229, 1... | [--, 6.53616652618, 8.40586682378, 9.633360938... |
| **...** | ... | ... | ... | ... |
| **725** | [--, 0.659399364014, 1.43048721434, 0.93772741... | [1.47072951802, 1.60527305147, 2.61724433641, ... | [1.32257459908, 1.25163217981, 2.38658183423, ... | [--, 1.10152124309, 1.61621044171, 1.870417565... |
| **726** | [--, 0.68255539011, 1.49333306323, 0.974149218... | [1.50672364174, 1.63412648302, 2.71830581267, ... | [1.34466446756, 1.26701648707, 2.45868467409, ... | [--, 1.12774409051, 1.6553227731, 1.9213440057... |
| **727** | [--, 0.699882339488, 1.53949519057, 1.00124765... | [1.52935947453, 1.65131506739, 2.78412010003, ... | [1.36155716165, 1.27888777607, 2.51355949639, ... | [--, 1.14758217366, 1.68408601622, 1.958736782... |
| **728** | [--, 0.706243119458, 1.55993148965, 1.01258053... | [1.52620305199, 1.64417423232, 2.7955818197, 1... | [1.35617858303, 1.27135791818, 2.51519582902, ... | [--, 1.14294852684, 1.67520191832, 1.955450040... |
| **729** | [--, 0.7103256077, 1.57179303023, 1.0200887478... | [1.518692328, 1.63322543733, 2.79005864248, 1.... | [1.34901351685, 1.26361542301, 2.50677052568, ... | [--, 1.13700990539, 1.66302935336, 1.947605945... |

730 rows × 4 columns

In [31]:

#sim\_res.to\_csv('/data/cwd\_awra\_data/AWRACMS/Training/test\_data/105001sim.csv')

In [32]:

#Simulations output gridded data; get some convenience aggregation functions

from awrams.calibration.support import aggregate, flattened\_areas

def mean\_by\_extent(in\_data, extent):

weights = flattened\_areas(extent)

return aggregate(in\_data, weights, weights.sum())

In [33]:

comp\_df = pd.DataFrame()

sim\_res['qtot'].shape

#cal\_dict['145003'].shape

#comp\_df['modelled'] = pd.Series(mean\_by\_extent(sim\_res['qtot'],cal\_dict['145003']),index=run\_period)

#comp\_df['obs'] = obs\_df['145003'][run\_period]

#comp\_df.plot()

Out[33]:

(730, 4, 5)

In [34]:

# Deliberately get the worst parameter set, just to see how bad it is...

worst\_params = cr.get\_best\_paramset(reverse=True)

In [35]:

cr.get\_best\_paramset?

In [36]:

sim.input\_runner.set\_parameters(worst\_params)

bad\_res = sim.run(run\_period, cal\_dict['105001'], False, False)

bad\_res['qtot'].shape

Out[36]:

(730, 49)

In [37]:

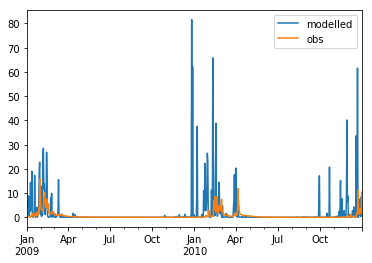
comp\_df = pd.DataFrame()

comp\_df['modelled'] = pd.Series(mean\_by\_extent(bad\_res['qtot'], cal\_dict['105001']), index=run\_period)

comp\_df['obs'] = obs\_df['105001'][run\_period]

comp\_df.plot()

Out[37]:



##### Exercise A

1. Re-calibrate to only one catchment

2. Use the parameters from the two calibrations in 2 ondemand simulations for the set of catchments

3. Compare flows from the models to the observations

##### Exercise B

Try the <https://github.com/awracms/awra_cms/blob\AWRA_CMS_v1.2/Training/Advanced/RemoteJobChaining.ipynb> workbook in the Training/Workspace folder where the steps are outlined in a bit more detail, and:

1. Use the provided set of Tasmanian catchments and repeat the calibration process.

2. Follow it up by doing some benchmarking, visualisation in the following session

### Visualisation

#### Spatial and Time-series visualisation of AWRA-L inputs and outputs

In [1]:

## Change to code if plots do not show up.

%matplotlib inline

The inputs to and results of an AWRA-L simulation can be viewed both spatially and temporally using the tools of the AWRA MS.

This notebook will demonstrate the available functionality to visualise outputs or inputs that are in the format used by the model,   
i.e. in CF compliant netcdf files that contain a year's worth of daily data for a single variable.

In addition, it will be shown how other tools can be used for extracting subsets of data from the overall outputs.

The following steps are covered:

1. Import required libraries

2. Load inputs or results

2.1 Inspect which variables are available

3. Spatial plots for different data slices

3.1 A single day, whole extents

3.2 Aggregated over a month for specified region

3.3 Change aggregation method

3.4 Accessing the underlying data

3.5 Specifying a catchment extent

3.6 Show location of catchment

3.7 Manipulating matplotlib settings

4. Timeseries plots for selected locations/regions

4.1 Time series for a single location

4.2 Time series for a catchment

5. Exercise

##### 1. Import required modules

In [2]:

import os

from awrams.utils import config\_manager, extents

from awrams.utils.gis import ShapefileDB

import awrams.visualisation.vis as vis

import awrams.visualisation.results as res

##### 2. Load inputs or results

The code expects to find netcdf files in the target folder

In [3]:

# NOTE: This notebook depends on results generated by ../Simulation/SimulationServer.ipynb.

# If you haven't already done so, run this notebook before running the following.

sim\_output\_dir = '../Simulation/results\_simserver'

results = res.load\_results(sim\_output\_dir)

sys\_settings = config\_manager.get\_system\_profile().get\_settings()

climate\_training\_path = sys\_settings['CLIMATE\_DATASETS']['TRAINING']['FORCING']['PATH']

inputs = res.load\_results(climate\_training\_path + '/rain\_day/')

In [4]:

results.path

Out[4]:

'/data/cwd\_awra\_data/AWRAMSI/IWRM\_0042\_WP3/GIT/sloh/awracms/awrams\_cm/Training/Simulation/results\_simserver'

In [5]:

inputs.path

Out[5]:

'/mnt/awramsi\_test\_data/awracms\_release/.awrams/data/training/climate/bom\_awap/rain\_day'

###### 2.1 Inspect variables present in results

Tab completable access to variable list

In [6]:

results

Out[6]:

<awrams.visualisation.results.Results at 0x7ffc4cea24a8>

In [7]:

results.period

Out[7]:

DatetimeIndex(['2010-12-01', '2010-12-02', '2010-12-03', '2010-12-04',

'2010-12-05', '2010-12-06', '2010-12-07', '2010-12-08',

'2010-12-09', '2010-12-10', '2010-12-11', '2010-12-12',

'2010-12-13', '2010-12-14', '2010-12-15', '2010-12-16',

'2010-12-17', '2010-12-18', '2010-12-19', '2010-12-20',

'2010-12-21', '2010-12-22', '2010-12-23', '2010-12-24',

'2010-12-25', '2010-12-26', '2010-12-27', '2010-12-28',

'2010-12-29', '2010-12-30', '2010-12-31', '2011-01-01',

'2011-01-02', '2011-01-03', '2011-01-04', '2011-01-05',

'2011-01-06', '2011-01-07', '2011-01-08', '2011-01-09',

'2011-01-10', '2011-01-11', '2011-01-12', '2011-01-13',

'2011-01-14', '2011-01-15', '2011-01-16', '2011-01-17',

'2011-01-18', '2011-01-19', '2011-01-20', '2011-01-21',

'2011-01-22', '2011-01-23', '2011-01-24', '2011-01-25',

'2011-01-26', '2011-01-27', '2011-01-28', '2011-01-29',

'2011-01-30', '2011-01-31'],

dtype='datetime64[ns]', freq='D')

In [8]:

results.variables.qtot

Out[8]:

qtot

In [9]:

inputs.variables

Out[9]:

OrderedDict([('rain\_day', rain\_day)])

##### 3. Spatial plots for different data slices

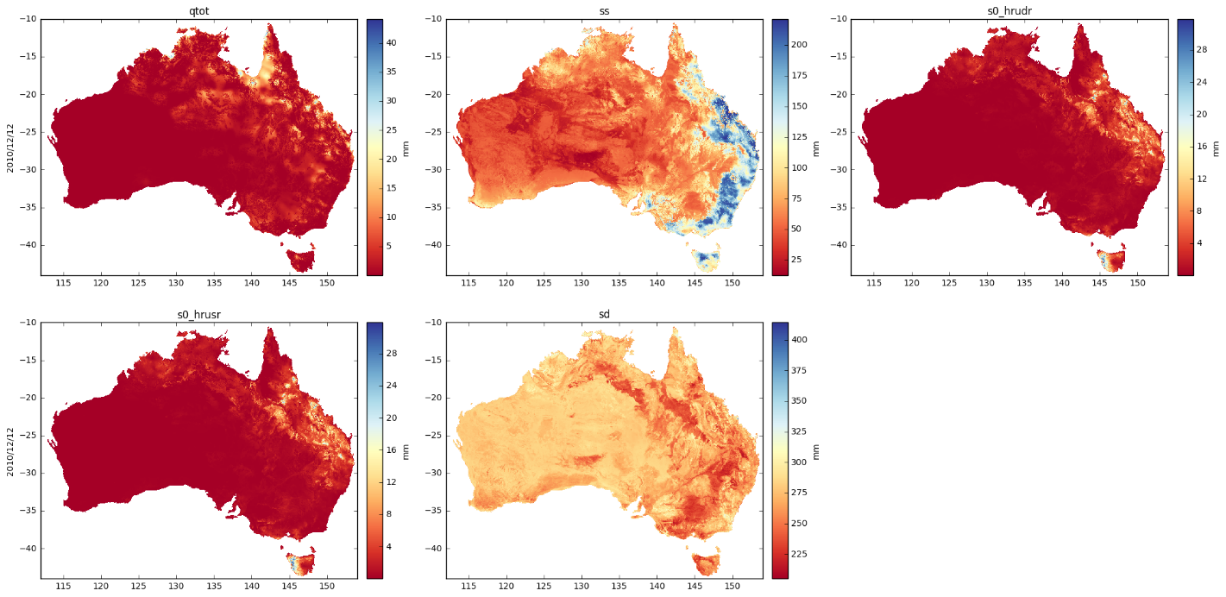
Approach: define slice of interest for viewing = [variables, period, extent]

* variables slice
* display a single variable - results.variables.qtot
* display multiple variables with a tuple - results.variables.qtot,results.variables.ss
* display all variables using standard slicing syntax - ":"
* period slice
* a single day - "1 jul 2010"
* a period - "jul 2010" or "jul 2010 - jun 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 extents.get\_default\_extent()
* a bounding box - results.extent.icoords[-39.5:-44,143.5:149]
* a catchment - catchments.get\_extent\_by\_field('GaugeName','Gunning',results.extent)

###### 3.1 A single day, whole extents

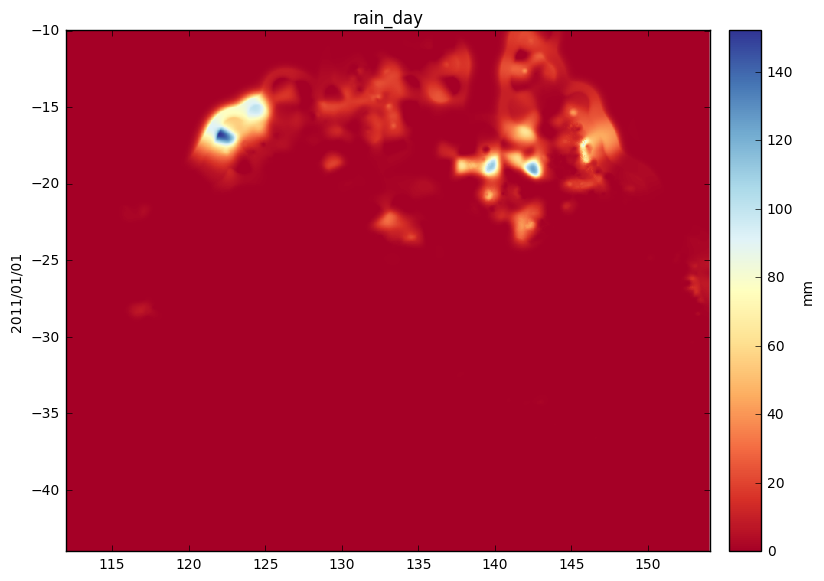
In [10]:

results[:, '12 dec 2010', :].spatial()



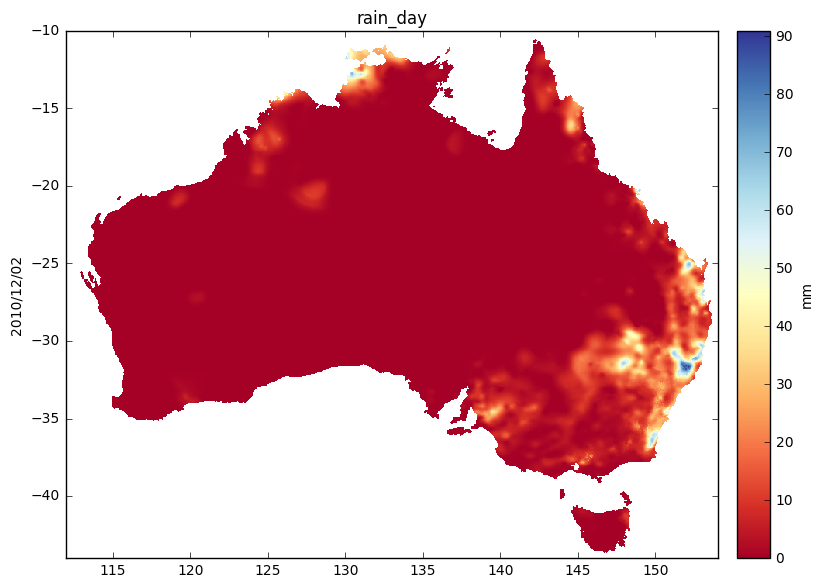
In [11]:

inputs[:, '1 jan 2011', :].spatial() # the entire grid is covered in data because the input file is not masked



In [12]:

inputs[:, '2 dec 2010', extents.get\_default\_extent()].spatial() # one way to mask it is to specify the default extent



###### 3.2 Aggregated over a month for specified region

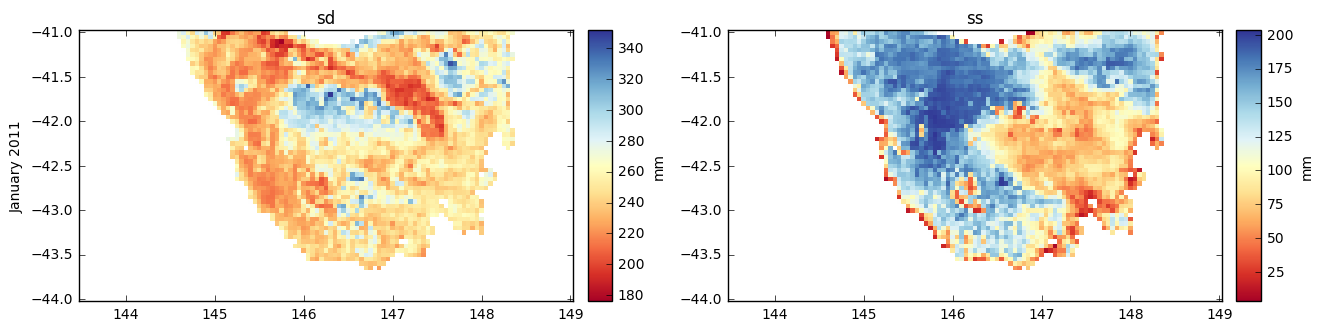
Also saves picture to png

In [13]:

v = results.variables.sd, results.variables.ss

results[v, 'jan 2011', results.extent.icoords[-41:-44, 143.5:149]].spatial()

vis.plt.savefig('map\_of\_tasmania.png', format='png', dpi=120)



###### 3.3. Change aggregation method

Here, from default of mean to sum

In [14]:

v

Out[14]:

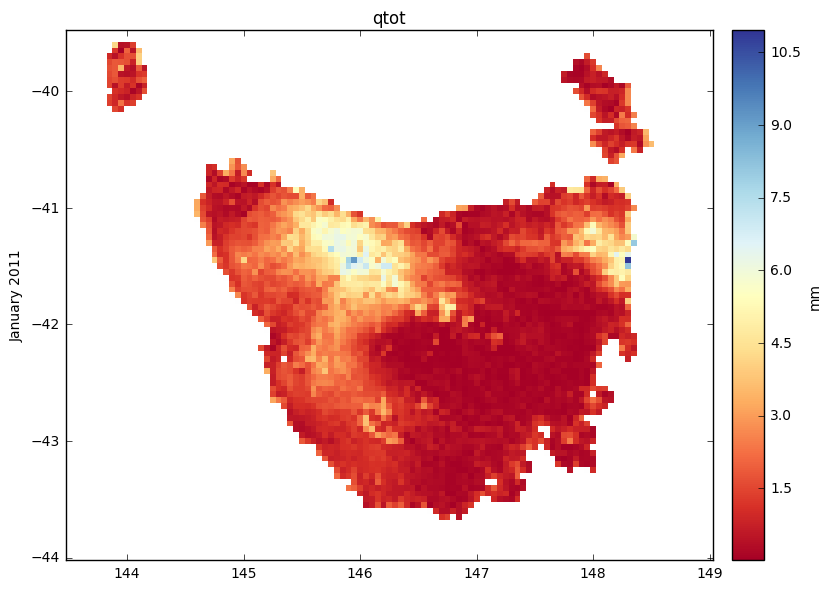
(sd, ss)

In [15]:

v = results.variables.qtot

v.agg\_method = 'mean'

results[v, 'jan 2011', results.extent.icoords[-39.5:-44, 143.5:149]].spatial()



###### 3.4 Accessing the underlying data

Get raw data cube

In [16]:

ss\_data = results.variables.ss.data

Data shape

In [17]:

results.variables.ss.data.shape

Out[17]:

(31, 61, 111)

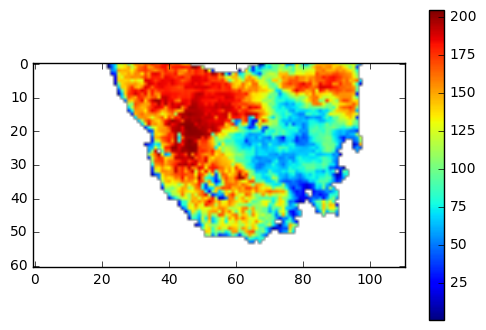
In [18]:

import matplotlib.pyplot as plt

plt.imshow(ss\_data[30])

plt.colorbar()

Out[18]:



Temporally aggregated data

In [19]:

results.variables.ss.agg\_data.shape

Out[19]:

(61, 111)

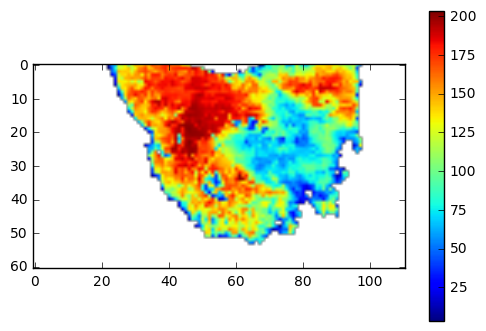
In [20]:

ss\_agg\_data = results.variables.ss.agg\_data

plt.imshow(ss\_agg\_data)

plt.colorbar()

Out[20]:



###### 3.5 Specifying a catchment extent

In [21]:

## A quick way to inspect the contents of the shapefile

base\_data\_path = sys\_settings['DATA\_PATHS']['BASE\_DATA']

catchment\_shapefile = os.path.join(base\_data\_path, 'spatial/shapefiles/Final\_list\_all\_attributes.shp')

default\_catchments = ShapefileDB(catchment\_shapefile)

default\_catchments\_df = default\_catchments.get\_records\_df()

default\_catchments\_df .head()

Out[21]:

|  | **AI** | **AlbersArea** | **Cal\_Val** | **CentrLat** | **CentrLon** | **DateOpen** | **DateQfinish** | **Downstr\_1** | **Downstr\_2** | **Downstr\_3** | **...** | **MeanP** | **QComplete** | **RiverName** | **SlopeMean** | **SlopeRange** | **State** | **StationID** | **Whpk1\_Mean** | **Whpk2\_Mean** | **WrscID** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 3.035136 | 5792.427800 | 0 | -24.659546 | 146.188956 | 1/01/1969 | 10/4/2011 | 0 | 0 | 0 | ... | 556.338623 | 93.998816 | Barcoo | 1.01441 | 11.2756 | QLD | 003303 | 32.0211 | 70.6515 | 003303A |
| **1** | 5.982206 | 320.401464 | -1 | -30.021958 | 139.517736 | 1/01/1972 | 1/10/1991 | 0 | 0 | 0 | ... | 241.771332 | 29.554536 | Hamilton Ck | 3.66823 | 18.2268 | SA | 004502 | 35.0000 | 77.0000 | AW004502 |
| **2** | 5.581191 | 1013.515346 | 1 | -30.692950 | 139.006100 | 1/01/1973 | 13/3/1989 | 0 | 0 | 0 | ... | 265.750458 | 35.540921 | Mt McKinlay Ck | 4.07388 | 22.4773 | SA | 004508 | 34.9490 | 76.5210 | AW004508 |
| **3** | 5.224682 | 3323.972600 | 1 | -23.974046 | 133.340785 | 1/01/1972 | 7/12/2011 | 0 | 0 | 0 | ... | 314.112885 | 99.822406 | Hugh | 2.11346 | 28.3030 | NT | 005115 | 27.2579 | 81.0000 | G0050115 |
| **4** | 4.809843 | 434.946800 | -1 | -23.436722 | 134.315857 | 1/01/1967 | 31/12/2011 | 0 | 0 | 0 | ... | 352.356567 | 82.618026 | Trephina Ck | 2.64208 | 24.1171 | NT | 006005 | 22.8571 | 0.0000 | G0060005 |

5 rows × 35 columns

In [22]:

gunning = default\_catchments.get\_extent\_by\_field('GaugeName', 'Gunning', results.extent)

gunning

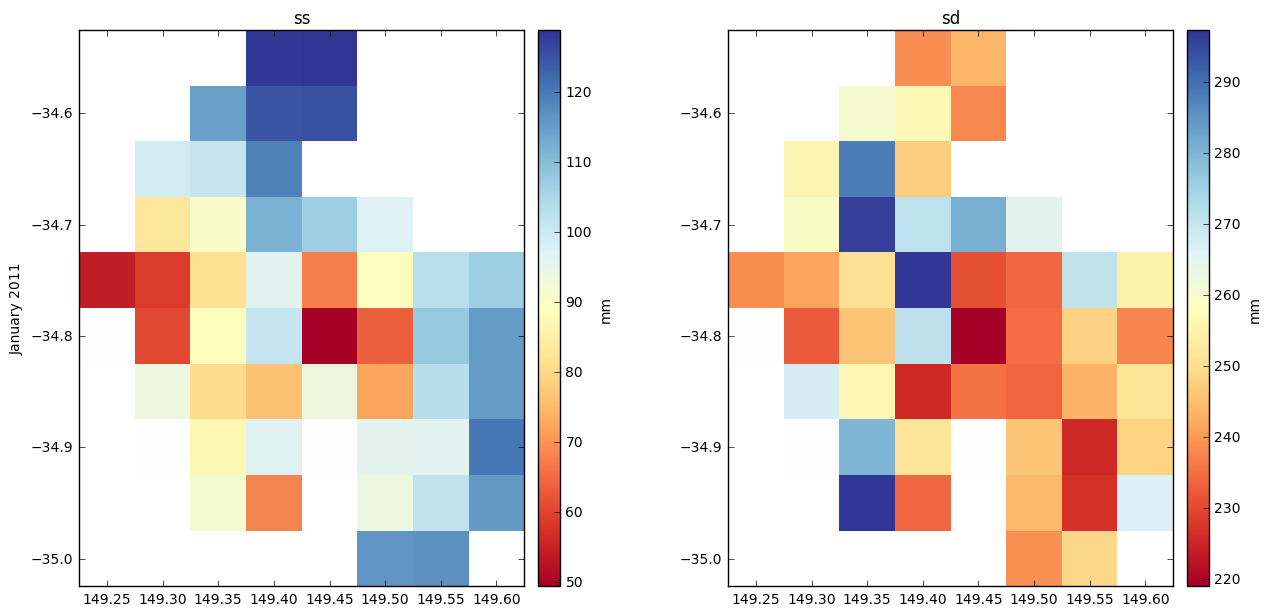
Out[22]:

origin: -34.55,149.25, shape: (10, 8), cell\_size: 0.05

In [23]:

v = results.variables.ss, results.variables.sd

results[v,'jan 2011',gunning].spatial(interpolation=None) #interpolation="bilinear")



## QUICK TIP TO INSPECT ## Create an extent from a shapefile from awrams.utils.gis import ShapefileDB ## Create a shapefile database from the file myShapefile = "INSERT PATH TO YOUR SHAPEFILE" mySDB = ShapefileDB(myShapefile) ## Inspect contents of shapefile shapefile\_df = mySDB.get\_records\_df() shapefile\_df my\_extent = mySDB.get\_extent\_by\_field('key\_field', 'field\_value', parent\_extent=edef)

###### 3.6 Show location of catchment

In [24]:

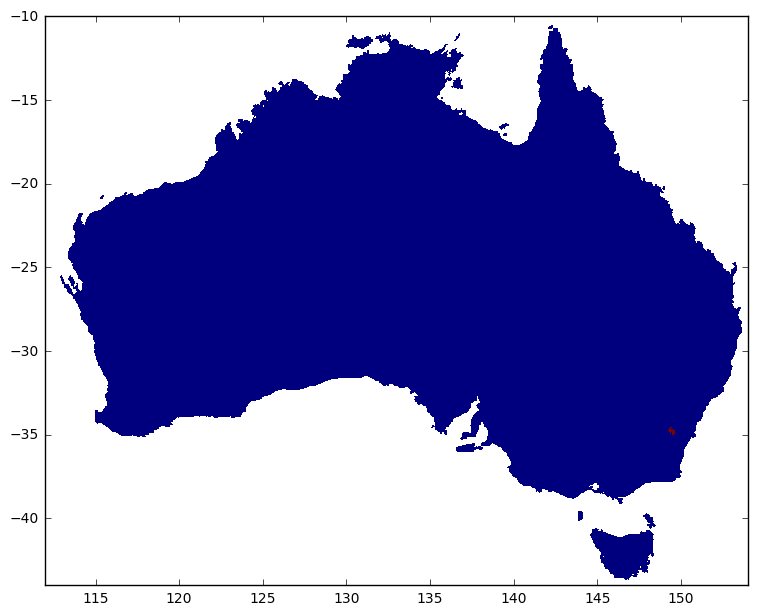
#vis.show\_extent(gunning)

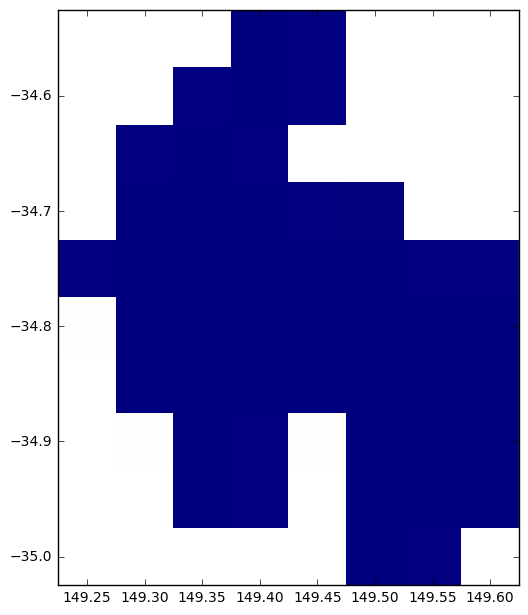
vis.show\_extent(gunning, results.extent.icoords[-30:-40,142:154])

vis.show\_extent(gunning, results.extent)

vis.show\_extent(gunning, gunning)







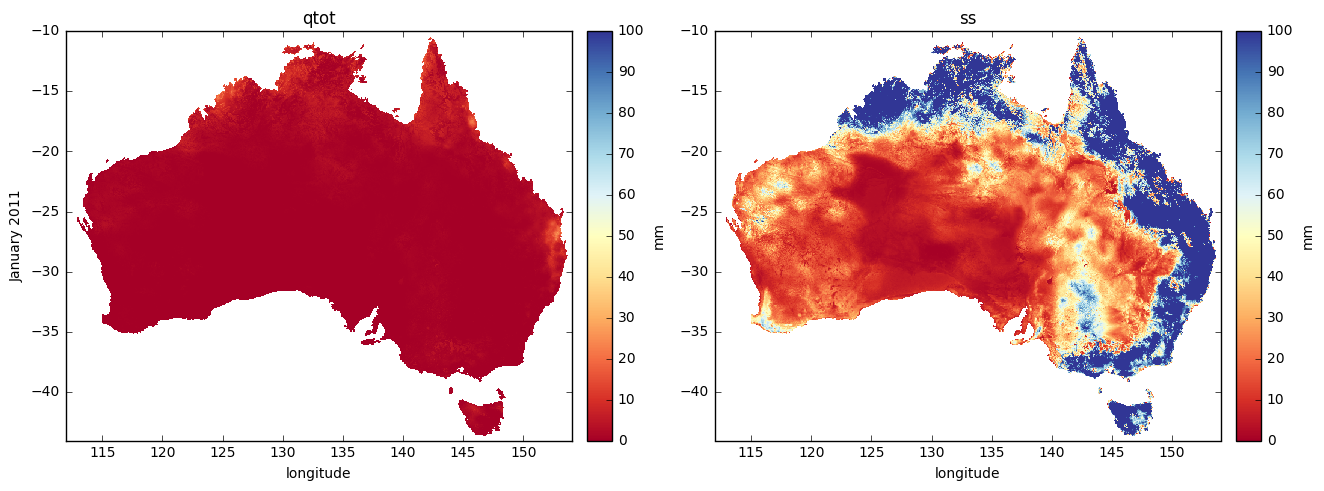
###### 3.7 Manipulating matplotlib settings

##### Specifying plotting ranges and labelling

In [25]:

v = results.variables.qtot, results.variables.ss

results[v,'jan 2011',:].spatial(clim=(0, 100), xlabel="longitude")



accessing the underlying axes and get the range of data for the selection

In [26]:

q = results[v,'jan 2011', results.extent.icoords[-30:-40,142:154]]

q.get\_data\_limits()

Out[26]:

(0.0, 205.4)

set colour range limits and horizontal axis labels

In [27]:

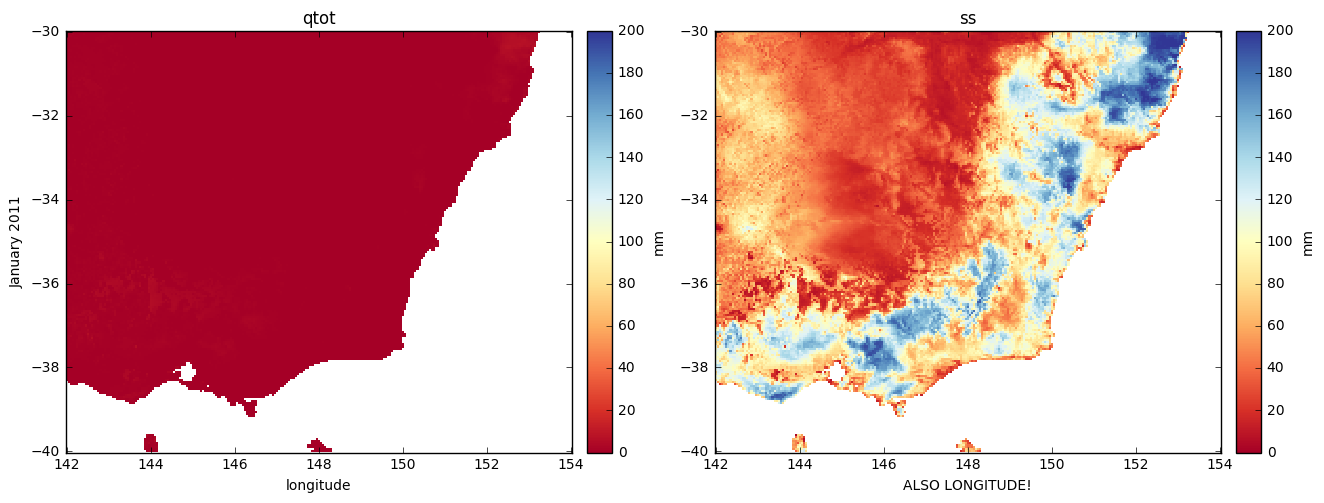
q.spatial(clim=(0, 200), xlabel="longitude")

gridview = q.mpl

view = gridview.children[0, 1]

view.ax.set\_xlabel("ALSO LONGITUDE!")

vis.plt.show()



##### 4. Timeseries plots for selected locations/regions

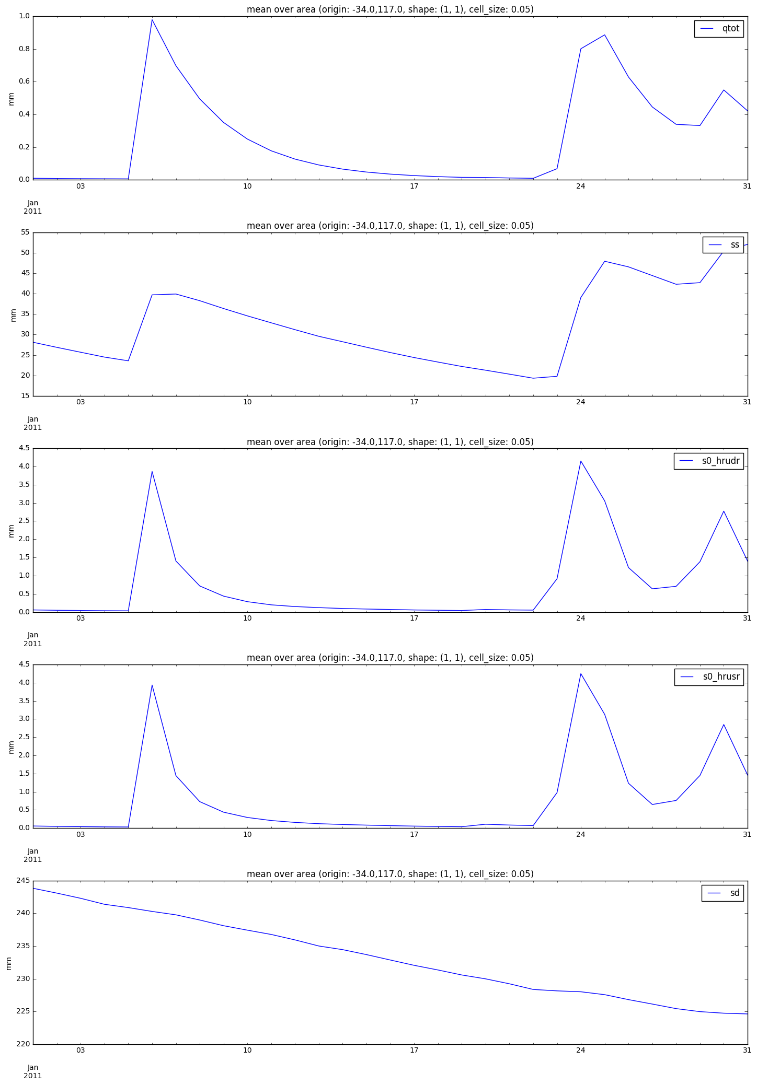
###### 4.1 Time series for a single location

In [28]:

p = 'jan 2011'

e = results.extent.icoords[-34, 117]

results[:,p,e].timeseries()



###### 4.2 Time series of aggregate over catchment

By default, the average over a catchment will be produced if you provide a catchment extent

In [29]:

edef = results.extent

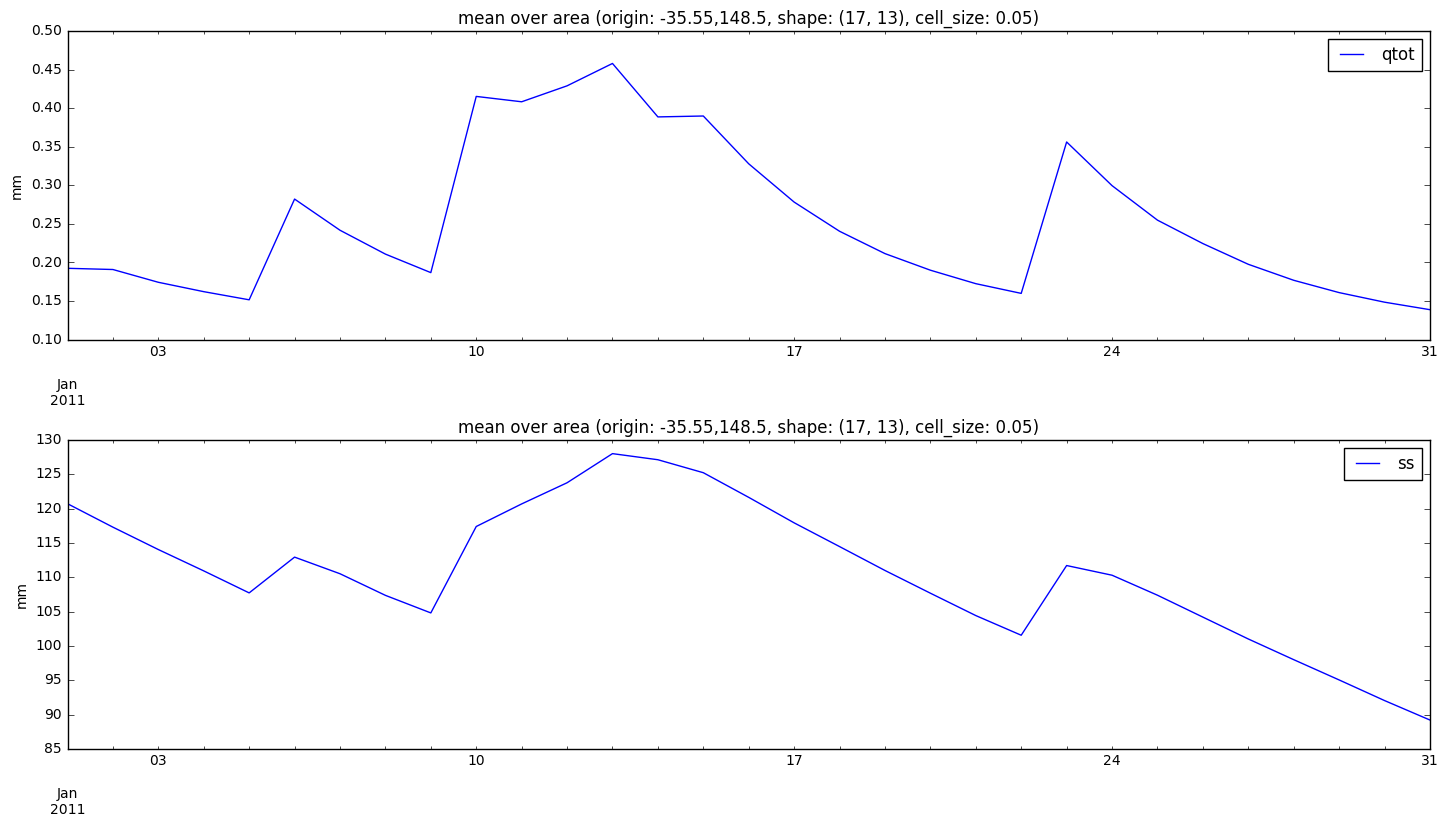
In [30]:

v = results.variables.qtot,results.variables.ss

p = 'jan 2011'

e = default\_catchments.get\_extent\_by\_field('GaugeName', 'Mittagang Crossing', results.extent)

results[v,p,e].timeseries()



In [31]:

results.variables.qtot.data.shape,results.variables.qtot.agg\_data.shape

Out[31]:

((31, 17, 13), (31,))

###### 4.3 More Matplotlib formatting options

In [32]:

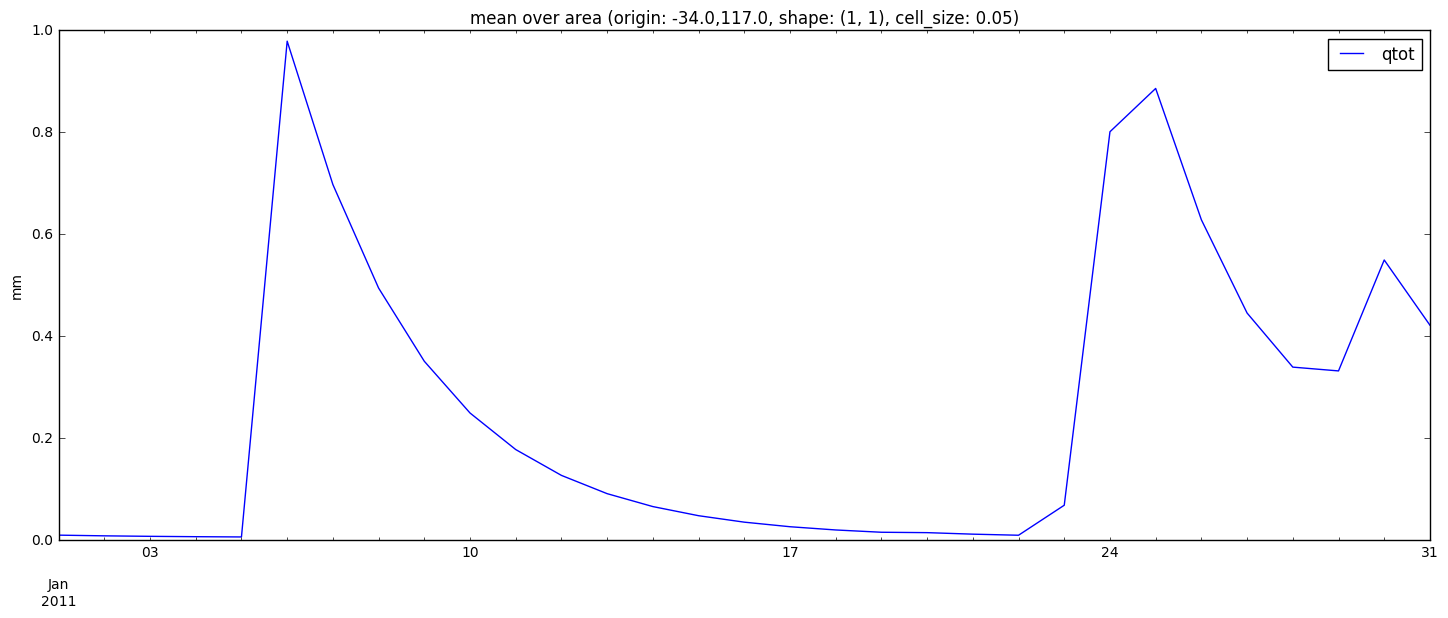
v = results.variables.qtot

e = results.extent.icoords[-34, 117]

p = 'jan 2011'

q = results[v,p,e]

q.timeseries()



In [33]:

q.timeseries() #ylim=(0,20)) # q is an object that carries data as well as matplotlib configurations

ax = q.mpl.children[0,0].ax

lines = ax.get\_lines()

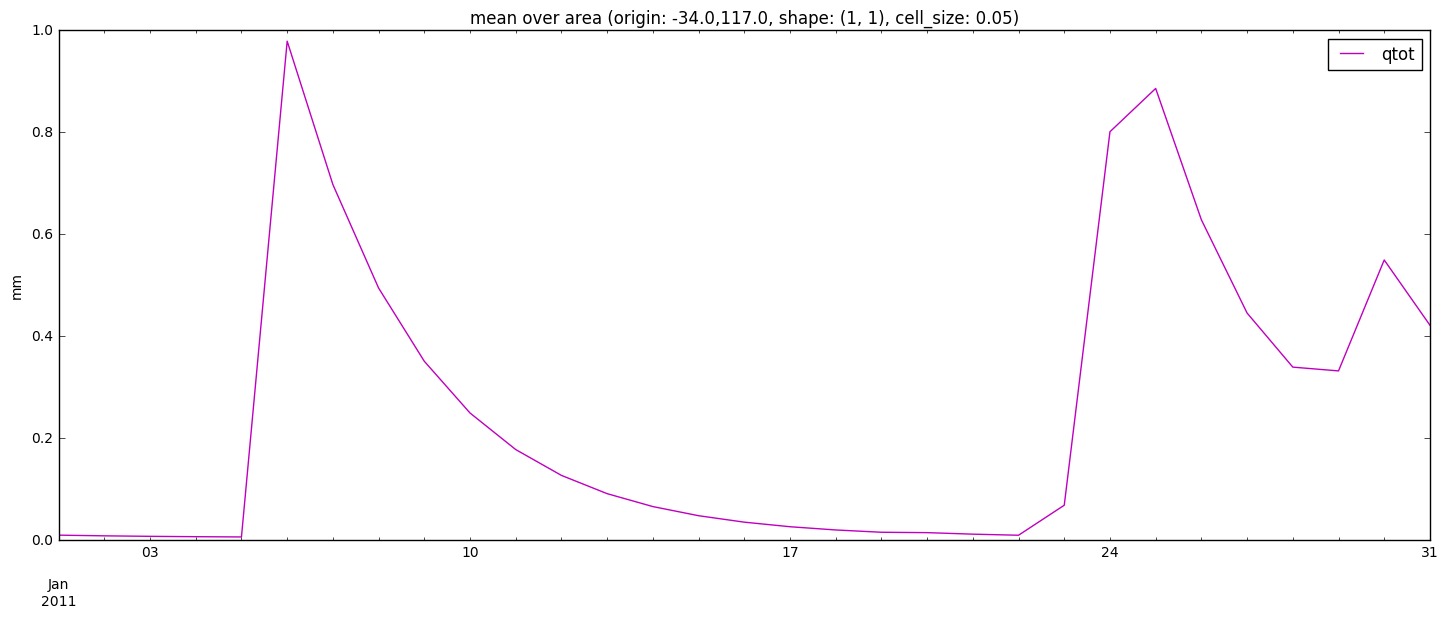
line = lines[0]

line.set\_color('m')

leg = ax.get\_legend()

legline = leg.get\_lines()[0]

legline.set\_color('m')



##### 5. Exercise: Visualising climatic inputs

In [34]:

sys\_settings = config\_manager.get\_system\_profile().get\_settings()

climate\_training\_path = sys\_settings['CLIMATE\_DATASETS']['TRAINING']['FORCING']['PATH']

In [35]:

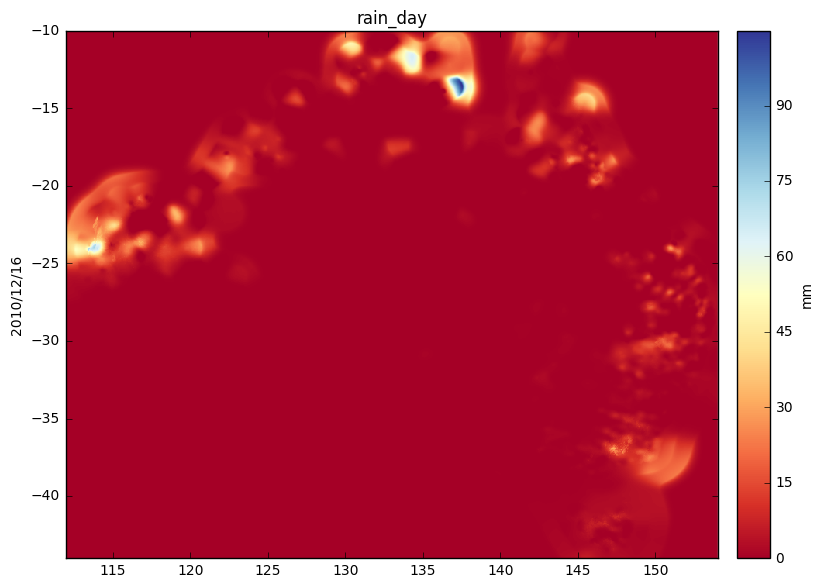
precip = res.load\_results(climate\_training\_path + '/rain\_day/')

vis.spatial(precip.variables.rain\_day, period='16 dec 2010')

## If you want to see the data only over Australia, use the defaul extent definition

vis.spatial(precip.variables.rain\_day, period='16 dec 2010', extent=extents.get\_default\_extent())

Out[35]:



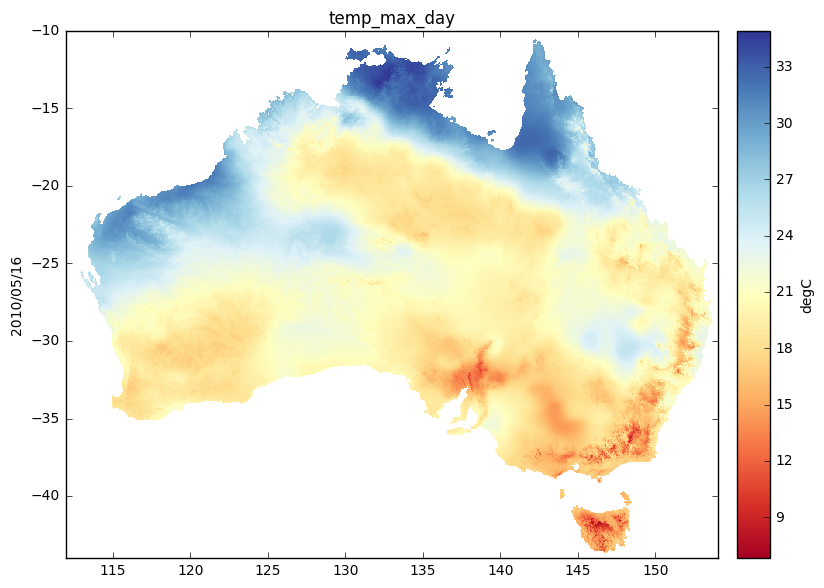


In [36]:

tmax = res.load\_results(climate\_training\_path + '/temp\_max\_day/')

vis.spatial(tmax.variables.temp\_max\_day, period='16 may 2010', extent=extents.get\_default\_extent())

Out[36]:

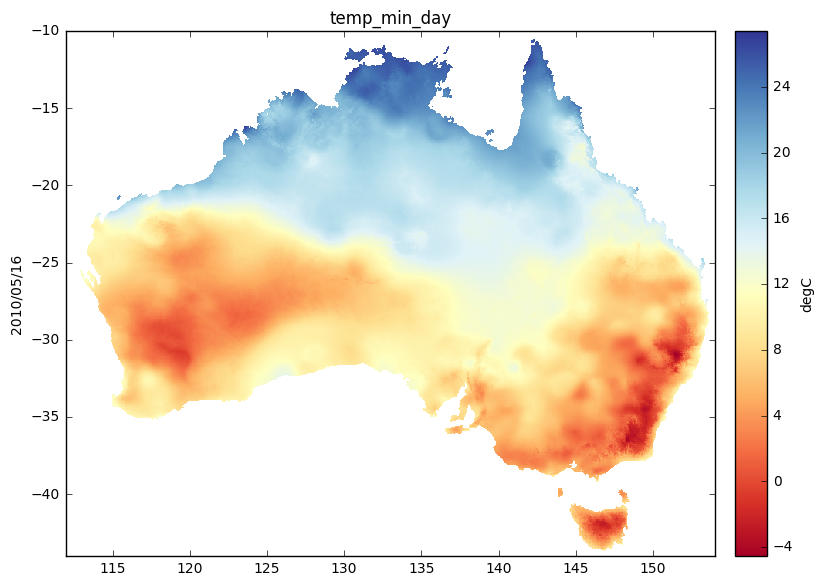


In [37]:

tmin = res.load\_results(climate\_training\_path + '/temp\_min\_day/')

vis.spatial(tmin.variables.temp\_min\_day, period='16 may 2010', extent=extents.get\_default\_extent())

Out[37]:

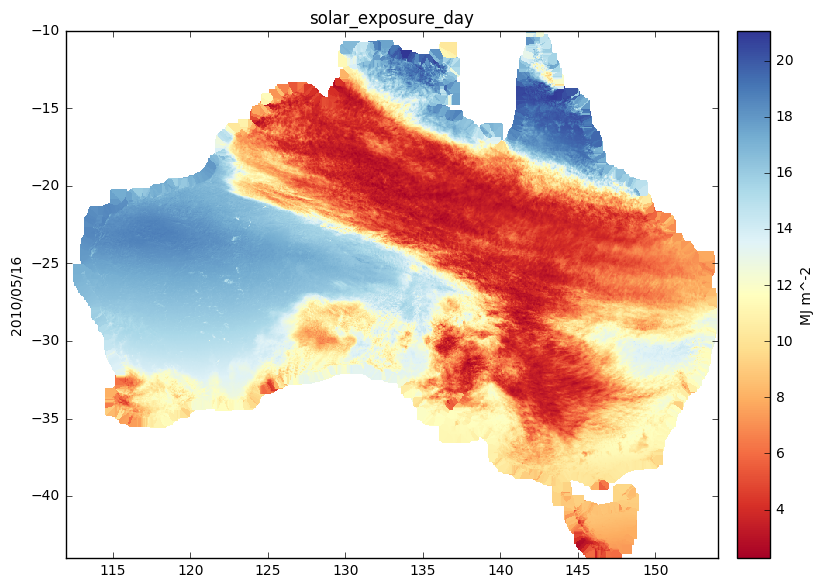


In [38]:

solar = res.load\_results(climate\_training\_path + '/solar\_exposure\_day/')

vis.spatial(solar.variables.solar\_exposure\_day, period='16 may 2010')

Out[38]:

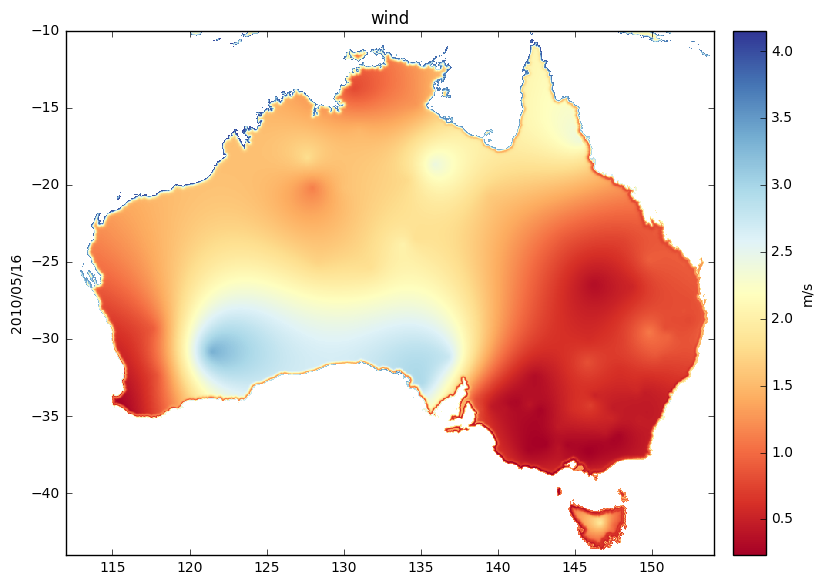


In [39]:

wind = res.load\_results(climate\_training\_path + '/wind/')

vis.spatial(wind.variables.wind, period='16 may 2010')

Out[39]:



### Benchmarking

#### Demonstrates AWRA-CMS benchmarking functionality using CMRS ET data

In [1]:

### uncomment to display figures

%matplotlib inline

Benchmarking dataset information

The actual ET (in mm) dataset consists of catchment-averaged actual evapotranspiration estimates from CMRSET supplied by CSIRO.

Source:

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.

This notebook goes through the following steps:

1. Import required libraries
2. Set up benchmarking configuration
   1. Catchments to be benchmarked
   2. Define observation inputs
3. Create benchmark object
4. Add models to be benchmarked
   1. Select or unselect models
5. View benchmarking statistics
6. View benchmarking plots
7. Statistics plotting

##### 1. Import required libraries

In [2]:

from awrams.benchmarking.benchmark import Benchmark

from awrams.benchmarking.utils import read\_id\_csv

from awrams.utils import datetools as dt

from awrams.utils import config\_manager

##### 2. Set up comparison

Compare CMRS ET aggregated to region of interest  
You can use your own data in csv form similar to the example provided.   
It just needs to have column names matching the names used in extracting AWRA data

###### 2.1 Catchments to be benchmarked

In [3]:

sys\_profile = config\_manager.get\_system\_profile().get\_settings()

TRAINING\_DATA\_PATH = sys\_profile['DATA\_PATHS']['TRAINING\_DATA']

data\_path = TRAINING\_DATA\_PATH + '/benchmarking/'

catchment\_csv = data\_path+'catchment\_ids.csv'

id\_list=read\_id\_csv(catchment\_csv)

###### 2.2. Observations to use

In [4]:

obs\_csv = data\_path+'/cmrset/cmrset.csv'

obs\_csv = data\_path+'/cmrset/Catchments\_ET\_CMRS.csv'

##### 3. Create the benchmark object

An object of "Benchmark" class is created by defining what variable is to be benchmarked. Everything else gets progressively added, and statistics are calculated when the observation and model outputs are added.

In [5]:

et = Benchmark("CMRS\_ET", "evapotranspiration")

# Specify benchmarking period

et.period = dt.dates("1981", "30/12/2011")

# Add observations and catchment subset [the id list needs to be present in the column names of the observation file]

et.load(obs\_csv, id\_list=id\_list)

In [6]:

et.sites # Would be nice to have a function to map the catchments selected

Out[6]:

['4508',

'105001',

'107002',

'108003',

'109001',

'111101',

'112102',

'113004',

'116008']

##### 4. Add models to be benchmarked

This step processes the data and calculates all the statistics [can take a while]

In [7]:

# Reading data from pre-processed csv

csv\_data = data\_path+'/cmrset/awral\_etot\_avg.csv'

et.add\_model("AWRAMSI\_v4\_0\_AWRAL", data\_csv=csv\_data)

In [8]:

## Sample code here to extract data on any catchment in the default catchment dataset

## A similar approach would work with a user defined catchment shapefile.

# import pandas as pd

# from awrams.utils import extents

# from awrams.utils.io.data\_mapping import SplitFileManager

# from awrams.utils.processing.extract import extract\_from\_filemanager

# from awrams.utils.gis import ShapefileDB, CATCHMENT\_SHAPEFILE

# catchments = ShapefileDB(CATCHMENT\_SHAPEFILE)

# var\_name = 'qtot\_avg'

# model\_data\_path = '/data/cwd\_awra\_data/awra\_test\_outputs/Scheduled\_v5\_awraprod1/'

# period = dt.dates('jul 2010 - jun 2011')

# pattern = data\_path + '/%s\*' % var\_name

# sfm = SplitFileManager.open\_existing(model\_data\_path,pattern,var\_name)

# georef = sfm.get\_extent()

# extent\_map = {}

# for site in q.sites:

# extent\_map[site] = catchments.get\_extent\_by\_field('StationID',site.zfill(6),georef)

# df = extract\_from\_filemanager(sfm, extent\_map, period)

# df

# df.to\_csv('./for\_streamflow/'+ var\_name+ '.csv'

In [9]:

csv\_data = data\_path+'/cmrset/AWRAMSI\_v5QES\_AWRAL\_etot\_avg\_CMRS\_ET.csv'

et.add\_model("AWRAMSI\_v5\_0\_AWRAL", data\_csv=csv\_data)

###### 4.1 Show list of loaded or selected models

list of loaded models is available with activated dropdown by typing "et.models."  
can "select" or "unselect" models for displaying

In [10]:

et.benchmark.selection

Out[10]:

['AWRAMSI\_v4\_0\_AWRAL', 'AWRAMSI\_v5\_0\_AWRAL']

In [11]:

#et.benchmark.selection.AWRAMSI\_v4\_0\_AWRAL.unselect()

et.benchmark.selection.AWRAMSI\_v4\_0\_AWRAL.select()

##### 5. View the benchmarking statistics tables

Summary percentiles can be printed out by specifying a statistic from:   
"grand\_f", "nse", "bias\_relative", "pearsons\_r" (default), "mean"   
to the 'stat\_percentiles' function  
The timeframe defaults to monthly, but can be specified

These tables are pandas dataframes, so they can be exported to csv

In [12]:

et.benchmark.stat\_percentiles('grand\_f')

Out[12]:

|  | **grand\_f** |
| --- | --- |
| **AWRAMSI\_v4\_0\_AWRAL** | 0.270952 |
| **AWRAMSI\_v5\_0\_AWRAL** | 0.298980 |

In [13]:

et.benchmark.stat\_percentiles('nse')

Out[13]:

|  | **0%** | **5%** | **25%** | **50%** | **75%** | **95%** | **100%** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AWRAMSI\_v4\_0\_AWRAL** | -4.459714 | -3.574903 | -0.459160 | 0.533537 | 0.541291 | 0.629999 | 0.639714 |
| **AWRAMSI\_v5\_0\_AWRAL** | -5.162047 | -4.319804 | -0.327347 | 0.309657 | 0.589680 | 0.755268 | 0.756441 |

In [14]:

et.benchmark.stat\_percentiles('bias\_relative')

Out[14]:

|  | **0%** | **5%** | **25%** | **50%** | **75%** | **95%** | **100%** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AWRAMSI\_v4\_0\_AWRAL** | -0.240516 | -0.232698 | -0.109853 | -0.089583 | -0.040577 | -0.035093 | -0.034863 |
| **AWRAMSI\_v5\_0\_AWRAL** | -0.260659 | -0.232667 | -0.147821 | -0.101505 | -0.055168 | -0.003524 | -0.000305 |

In [15]:

et.benchmark.stat\_percentiles('pearsons\_r')

Out[15]:

|  | **0%** | **5%** | **25%** | **50%** | **75%** | **95%** | **100%** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AWRAMSI\_v4\_0\_AWRAL** | 0.644339 | 0.650187 | 0.755195 | 0.840095 | 0.856408 | 0.862268 | 0.862610 |
| **AWRAMSI\_v5\_0\_AWRAL** | 0.650217 | 0.659570 | 0.730668 | 0.825253 | 0.858632 | 0.889464 | 0.898616 |

##### 6. View the statistic plots [regression, cumulative exceedance]

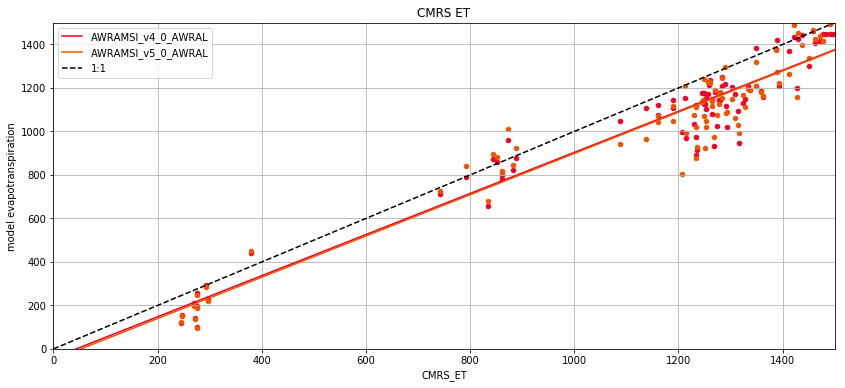
specify frequency by "freq=d" for daily, "freq=m" for monthly, "freq=y" for yearly  
can customise titles, labels, scaling etc using standard matplotlib keyword arguments

In [16]:

## should there be time series plots first

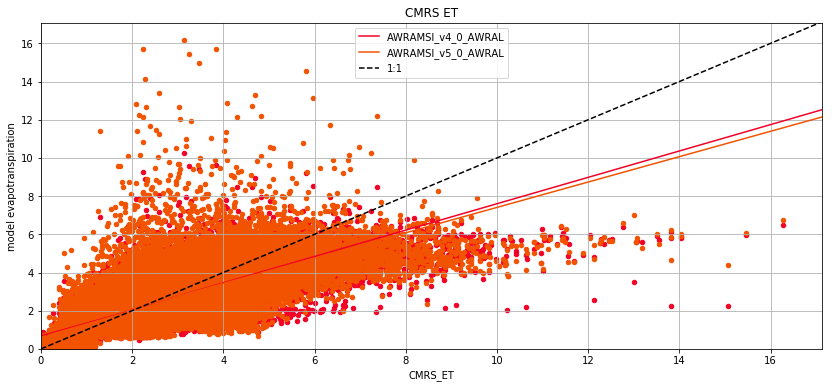
In [17]:

p = et.benchmark.plot\_regression(title="CMRS ET", freq='y', xlim=[0,1500], ylim=[0,1500])



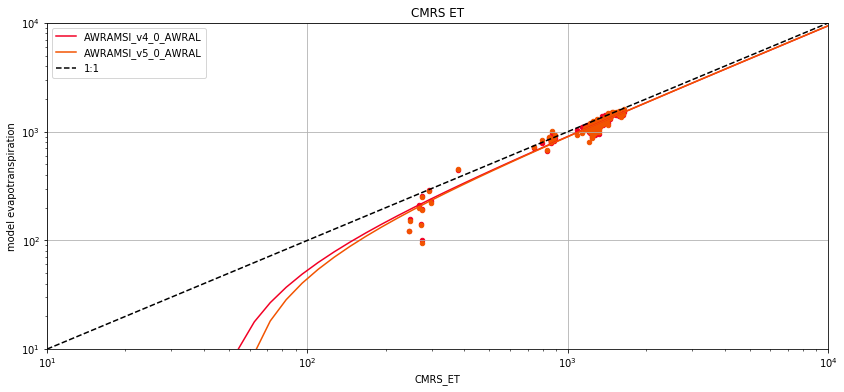
In [18]:

p = et.benchmark.plot\_regression(title="CMRS ET", freq='d', xlim=[0,None], ylim=[0,None])



In [19]:

p = et.benchmark.plot\_regression(title="CMRS ET", freq='y', yscale='log', xscale='log', ylim=[10,10000], xlim=[10,10000])



##### 7. Statistics plotting

specify statistic type from "fobj", "nse", "rmse", "bias\_relative", "pearsons\_r" (default), "mean" and   
frequency from 'd', 'm', 'y'

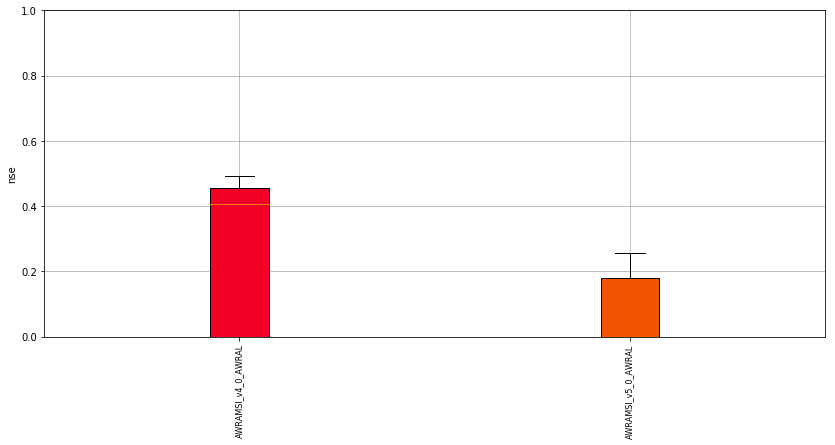
In [20]:

p = et.benchmark.plot\_box('fobj', freq='m', ylim=[0,1]) #fobj is viney function used in calibration



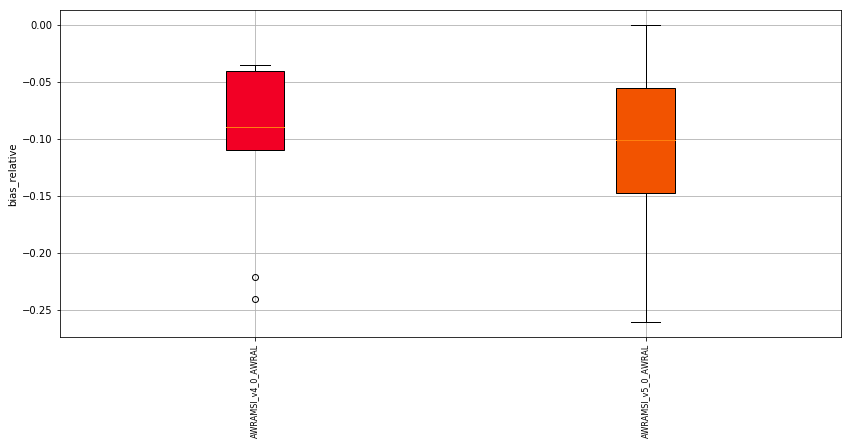
In [21]:

p = et.benchmark.plot\_box('nse', freq='d', ylim=[0,1])



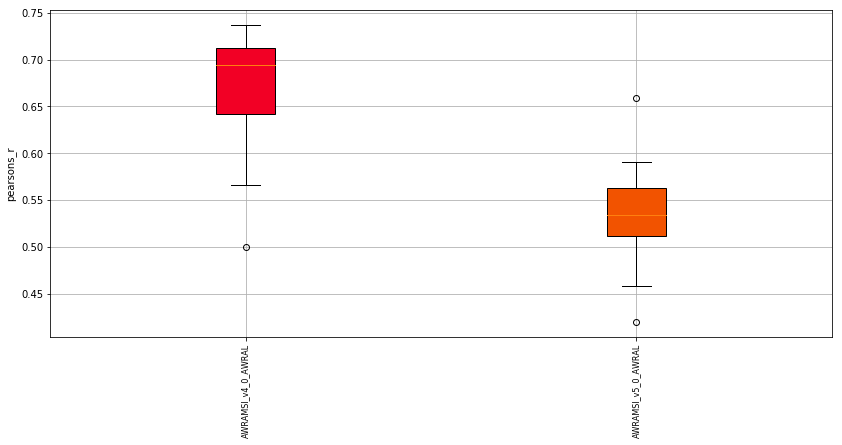
In [22]:

p = et.benchmark.plot\_box('bias\_relative', freq='m')



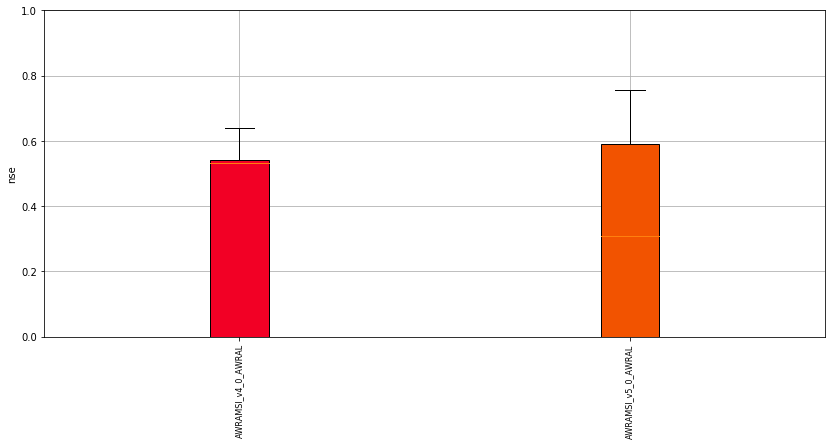
In [23]:

p = et.benchmark.plot\_box('pearsons\_r', freq='d')



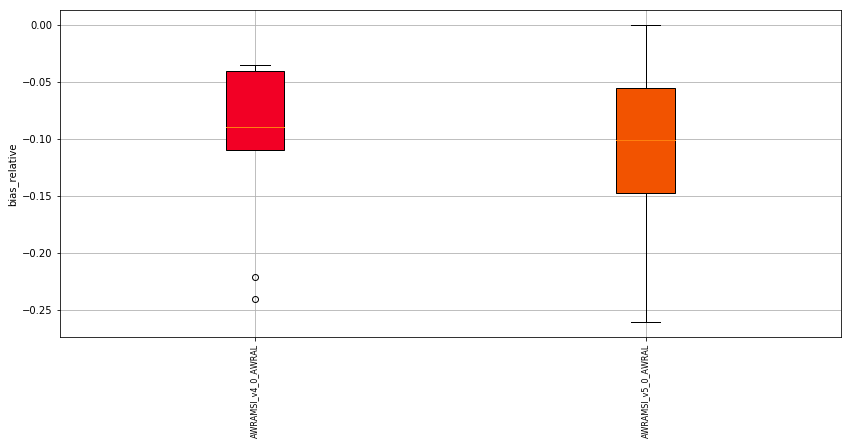
In [24]:

p = et.benchmark.plot\_box('nse', freq='m', ylim=[0,1])



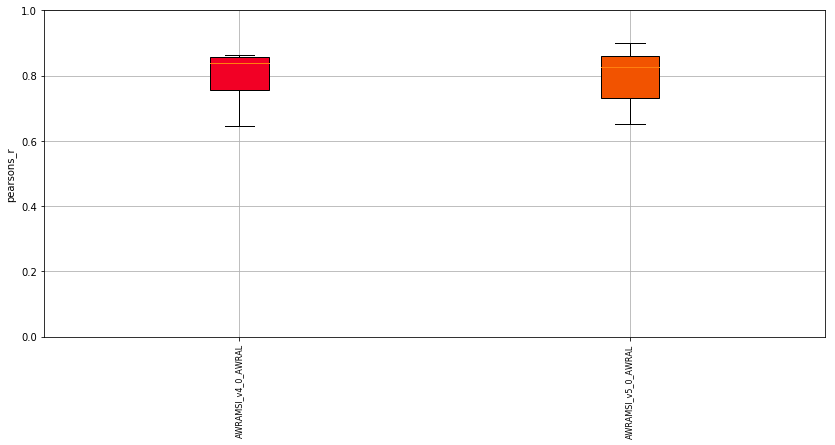
In [25]:

p = et.benchmark.plot\_box('bias\_relative', freq='m')



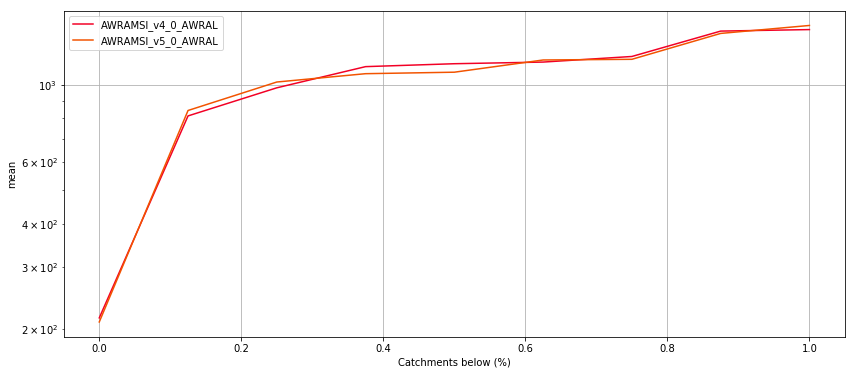
In [26]:

p = et.benchmark.plot\_box('pearsons\_r', freq='m', ylim=[0,1])



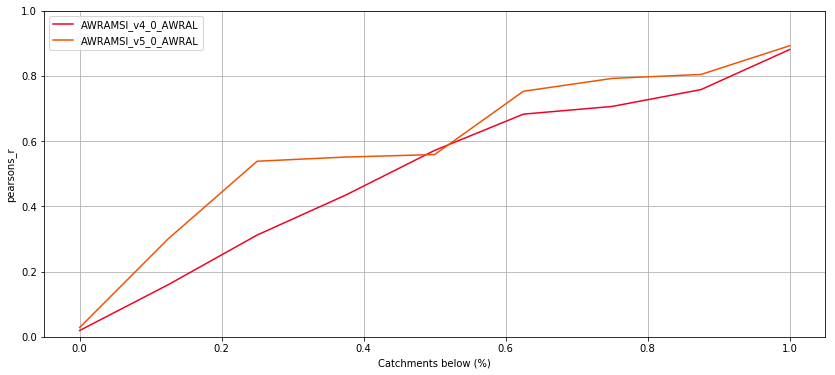
In [27]:

p = et.benchmark.plot\_cdf('mean', freq='y', yscale='log')



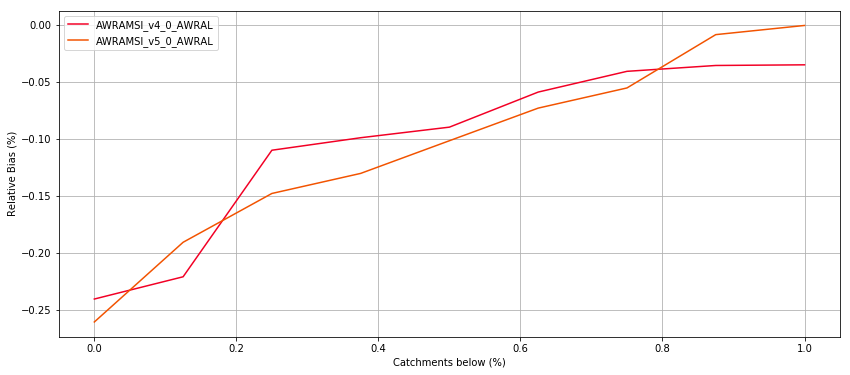
In [28]:

p = et.benchmark.plot\_cdf('pearsons\_r', freq='y', ylim=[0,1])



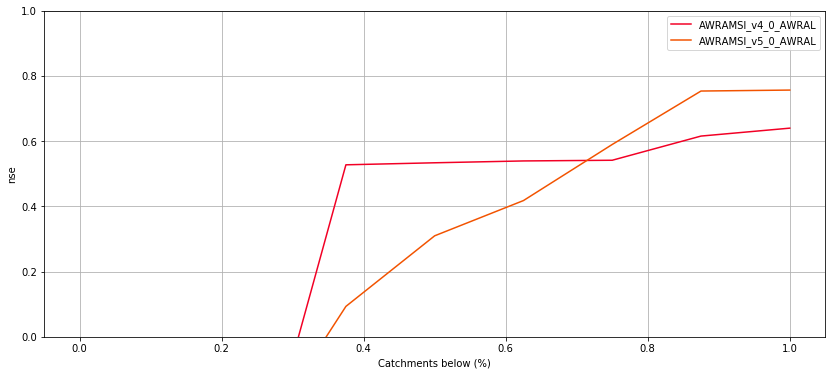
In [29]:

p = et.benchmark.plot\_cdf('bias\_relative',freq='y', ylabel='Relative Bias (%)')



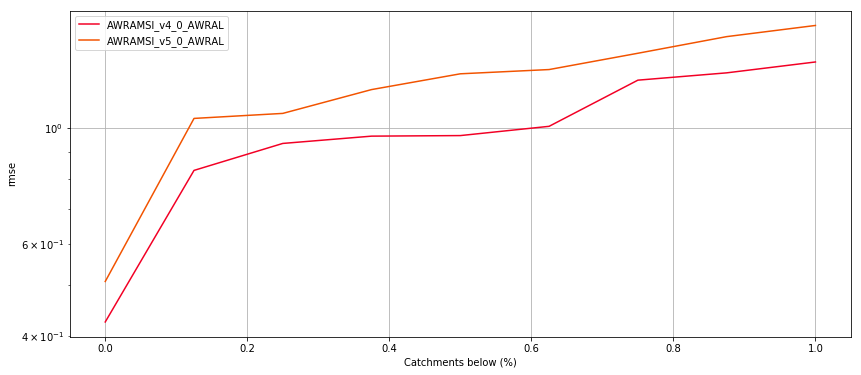
In [30]:

p = et.benchmark.plot\_cdf('nse',freq='m', ylim=[0,1])



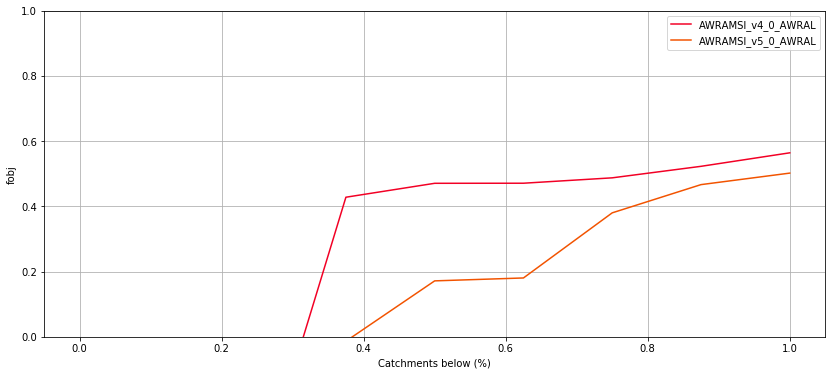
In [31]:

p = et.benchmark.plot\_cdf('rmse',freq='d', yscale='log')



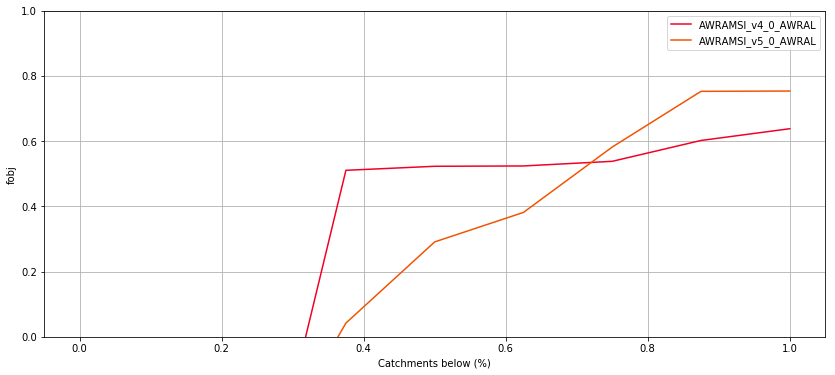
In [32]:

p = et.benchmark.plot\_cdf('fobj',freq='d', ylim=[0,1])



In [33]:

p = et.benchmark.plot\_cdf('fobj',freq='m', ylim=[0,1])



## 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. J., and Wright, D. P., 2018, Evaluation of the Australian Landscape Water Balance Model: AWRA-L v6. Bureau of Meteorology Technical Report.

Frost, A. J., Ramchurn, A. and Smith, A., 2018. The Australian Landscape Water Balance model (AWRA-L v6). Technical Description of the Australian Water Resources Assessment Landscape model version 6. 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.](http://www.bom.gov.au/water/landscape/static/publications/Hafeez_et_al_2015_AWRA_Paper_AWA.pdf)

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.](http://www.bom.gov.au/water/landscape/static/publications/Viney_et_al_2015_AWRA_L_5.0_model_description.pdf)

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.

## AWRA-L model overview

AWRA-L is a one-dimensional, 0.05° grid based water balance model over the Australian continent that has a semi-distributed representation of the soil, groundwater and surface water stores. For soil moisture accounting and runoff generation purposes, each grid cell in AWRA-L is divided into three soil layers (top: 0-10cm, shallow: 10cm-100cm, deep: 100cm-600cm) and two hydrological response units (shallow rooted versus deep-rooted vegetation) (with model description provided in Frost, Ramchurn and Smith, 2016). Within each grid cell the water balance is calculated independently for the 2 HRUs (deep and shallow rooted vegetation) with outputs from the model representing the grid cell average or total (depending on variable). Climatological forcing for AWRA-L model is taken from the operational Australian Water Availability Project ([AWAP](http://www.bom.gov.au/jsp/awap/)) climate data set which is available for the same 0.05° grid as AWRA-L (approximately 5 km resolution) and runs with a daily time step from 1911 to 'today' (a solar radiation climatology is used prior to 1990). The model also requires spatial inputs of soil, geological and terrain properties as well as the distribution of the HRUs in each grid cell.

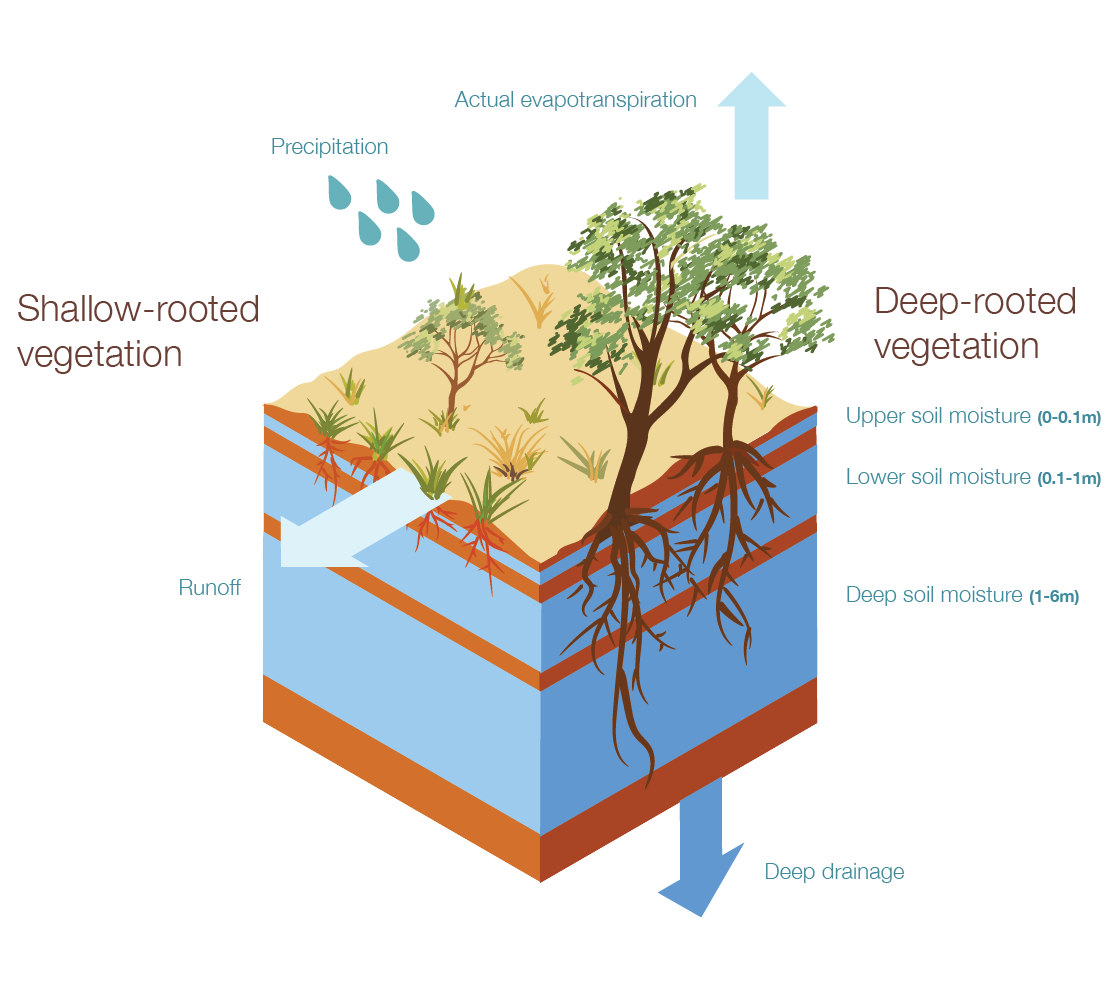


Figure A1. Conceptual AWRA-L grid cell with key water stores and fluxes shown

The model provides daily outputs of runoff from the landscape (surface and subsurface), actual and potential evapotranspiration (ET), soil moisture in each layer, and deep drainage at high-resolution, seamlessly across Australia from the past to the present (100 + years). These outputs can be aggregated spatially or temporally to provide daily, monthly and annual, regional or continental estimates of the landscape water balance.

The model is under constant development, and the current operational version at the Bureau of Meteorology is v5. 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 runoff (compared with streamflow) and rootzone soil moisture compared to peer models. Benchmark statistics are provided in that report over which future versions 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 current AWRA-L performance.

## 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 Kr (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 Kr (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 fv = 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 (m2 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 w0 at which evaporation is reduced (dimensionless) |
| w0ref\_alb | Reference value of w0 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) |

## Python version of AWRA-L

The model code has been translated from C into Python towards understanding the code more easily as compilation is not required. AWRA is run in two ways here:

* Component-by-component
* All components put together

The below code is commented with reference back to the model documentation and the model is run for a year for a chosen point to understand the dynamics.

C.1. The AWRA-L Workflow

AWRA-L broadly follows the following workflow:

* + - * + Pre-calculate and convert constants for use in the model
        + Initialise models states
        + Loop over Time steps
  1. Calculate saturated fraction of cell/catchment for a given groundwater storage (Sg) [a single store - no HRUs
  2. Calculate potential evaporation (E0) according to Energy balance [for each HRU]
  3. Calculate actual ET based on E0, water storage and vegatation state [for each HRU]
  4. Calculate Water balance and flows [for each HRU]
  5. Calculate Vegetation growth and senescence [for each HRU]
  6. Get cell values of variables weighting shallow and deep rooted components depending on fraction of each

These components are coded in Python below (rather than C) to facilitate understanding and testing the AWRA-L model.

C.1.1.AWRA-L model processes and algorithms overview

Run a model simulation to get input data and AWRA-L parameters i

AWRA-L calculations deconstructed - run for time period and parameters supplied

the broad steps listed above are undertaken for demonstration without time step interaction between the energy, water and vegetation components of the model. This is to show th equations used in calculation.

Note there is no dynamic interaction of variables in this case. e.g. fraction saturated, relative top layer soil content kept constant for E0 estimation

The subsequent model run with the components put into the same cell (including time step looping) incorporates this interaction.

## Contributing to AWRA-CMS

One of the main motivations of releasing AWRA as a community model is to gain the benefits of the scientific community in contributing improvements to the system. We hope that other research scientists and agencies will be interested in:

* altering model process descriptions (algorithms) towards better performance,
* increasing functionality of the modelling system
* testing alternative input datasets to the system.
* Improving the calibration and validation procedure
* And more

To make changes to the AWRA-L community model source code the User needs to create their own ['fork'](https://help.github.com/articles/fork-a-repo/) (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, for example by submitting an email outlining the proposed changes to the Bureau
2. User submits their fork (new code) and any documentation and testing that has been conducted
3. The Bureau assesses conceptually the proposed change the submission of fork/code provided in step 2.
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. Note that the bureau is also open to improvements to the benchmarking process and data sets if this can be scientifically justified.

The above list is an outline of the process, once users contact the Bureau we will be able to give instructions and guidance in greater depth.

## AWRAMS Licence

AWRAMS LICENCE

(Variation of BSD 3 licence)

Copyright © 2016, Commonwealth of Australia as represented by the Bureau of Meteorology. All rights reserved.

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

**USER DETAILS** (please complete)

*If signing as an individual and not on behalf of an organisation*

Name (individual signing in their own right):

Email address:

Address:

Phone number:

*If signing on behalf of an organisation*

Name (individual representative signing on behalf of organisation):

Name of Organisation (legal entity):

ABN

Email address:

Address:

Phone number:

Please state the nature and purpose of proposed use of AWRAMS Material:

…………………………………………………………………………………………………………………………………………………………………………………………

Signed:

………………………………………………

Date:……………

## 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**

If signing as an individual and not on behalf of an organisation

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:…

If signing on behalf of an organisation

Name (organisation): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature (duly authorised representative):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:…”

**Us**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Title: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature (duly authorised representative):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:

1. [NetCDF](http://www.unidata.ucar.edu/software/netcdf/)4 – network common data format [↑](#footnote-ref-1)
2. [HDF5](https://support.hdfgroup.org/HDF5/) – file format and software library [↑](#footnote-ref-2)
3. [MPI](http://www.open-mpi.org) – message passing interface [↑](#footnote-ref-3)
4. To become an AWAP Registered User, go to: http://reg.bom.gov.au/reguser/reguser.shtml [↑](#footnote-ref-4)