RHEAS Documentation

Release 0.2

Kostas Andreadis

February 02, 2017

Contents

The Regional Hydrologic Extremes Assessment System (RHEAS) is a hydrologic nowcast and forecast framework developed at the NASA Jet Propulsion Laboratory.

Contents:

CHAPTER 1

Installation

1.1 Requirements

The requirements for installing RHEAS include

- Python and a number of python packages
  - Numpy
  - NetCDF4
  - GDAL
  - Scipy
- GCC compiler
- Automake
- PostgreSQL server with the PostGIS extension
- VIC model executable
- DSSAT model executable

Thankfully most of these requirements are automatically installed using Buildout, a Python-based build system for creating, assembling and deploying applications from multiple parts.

In terms of hardware, the recommended requirements include any modern computer system with hard drive storage of at least 250 GB, and memory of at least 4 GB.

1.2 Installing on Linux and MacOS

In order to compile RHEAS, we need a C compiler and the Make utilities, as well as some other software packages. Depending on the Linux distribution these can be installed

```
sudo dnf groupinstall "Development Tools"
sudo dnf install scipy numpy gdal-devel python-dateutil libxslt-devel python-lxml libxslt-python rea
```

for RPM-based distros (such as RedHat, Fedora, Centos)

```
sudo aptitude update && sudo aptitude -y upgrade
sudo aptitude -y install git build-essential python-numpy python-scipy python-gdal python-argparse py
sudo dpkg --add-architecture i386
```

```
sudo aptitude update
sudo aptitude -y install wine winetricks
```

for DEB-based distros (such as Ubuntu, Debian, Mint)

If you're on a MacOS, the easiest way to install the dependencies is by using the Homebrew package manager. Follow the instructions to install Homebrew and then run

```
brew update
brew install python readline gdal netcdf wine winetricks sfcgal
pip install numpy scipy gdal argparse py-dateutil lxml requests
```

We then clone and uncompress the software archive

```
git clone https://github.com/nasa/RHEAS.git
cd RHEAS
```

and then run the `buildout` script. Before that, we boostrap the build by running

```
python bootstrap.py
```

After that we can install everything by running

```
./bin/buildout
```

The `buildout` script will install the `PostgresQL` database along with the `PostGIS` extension, and create a database for the user. The script will also install some additional Python modules, and build the `VIC` hydrology model.

After the script finishes you should have a `rheas` executable in your `bin` directory!

1.3 Installing on Windows

It is currently possible to run RHEAS in a bash-shell environment such as Cygwin. However, due to backwards compatibility issues with PostGIS dependencies, this method is not currently recommended.

1.4 Testing the installation

A number of unit tests have been created to validate the installation of RHEAS. The tests create a temporary database and perform the following operations:

- Download and ingest the suite of datasets into the database
- Run nowcasts for VIC and DSSAT
- Run forecasts for VIC and DSSAT

The tests can be run with

```
./bin/test
```

CHAPTER 2

The RHEAS database

The back-end of RHEAS is a PostGIS database (a spatially-enabled PostgreSQL database) that stores the necessary data to run RHEAS including model parameters, meteorological data (both forecasts and nowcasts), and remote sensing as well as in-situ observations.

The VIC and DSSAT model parameters are contained under the `vic` and `dssat` schemas. Depending on the requested spatial resolution for the models, appropriate model parameters and corresponding files are chosen. Tables in the `vic` schema include `soils` which contains the soil parameters, and `input` which contains the necessary files to run the VIC model. Each record in the `input` table should contain the following columns:

- **resolution**: model spatial resolution
- **snowbandfile**: VIC elevation band file
- **vegparam**: VIC vegetation parameter file
- **veglib**: VIC vegetation library file
- **soilfile**: VIC soil parameter file
- **rootzones**: number of root zones
- **basefile**: DSSAT input file template

The requirements for running DSSAT also include the type of cultivar(s), planting start dates, a global crop mask and soil properties which have been ingested in the database from various sources. Soil properties and cultivar information are stored as vector tables (`soils` and `cultivar` respectively), while planting start dates and the crop mask are stored as raster maps with the type of crop as an additional column in the `crops` and `cropland` tables respectively.

Multiple remote sensing datasets can be ingested into the database and are available either to be used as input (i.e. precipitation, temperature, wind speed) or assimilated (i.e. soil moisture, water storage, LAI, evapotranspiration). The following table summarizes the available datasets, their characteristics and the database schema/table they will be installed in.

Variable	Dataset	Time Coverage	Temporal Resolution	Spatial Resolution	Spatial Coverage	Table	Mode
Precipitation	CHIRPS	1981-	Daily	5km	Africa	precip.chirps	IN
Precipitation	TRMM	1998-	Daily	0.25 º	Global	precip.trmm	IN
Precipitation	RFE2	2001-	Daily	0.10 º	Africa	precip.rfe2	IN
Precipitation	CMORPH	1998-	Daily	0.25 º	Global	precip.cmorph	IN
Precipitation	GPM	2014-	Daily	0.10 º	Global	precip.gpm	IN
Meteorology	NCEP	1981-	Daily	1.875 º	Global	*.ncep	IN
Meteorology	PRISM	1981-	Daily	4km	CONUS	*.prism	IN
Soil moisture	AMSR-E	2002-2011	Daily	0.25 º	Global	soilm.amsre	AS
Soil moisture	SMOS	2009-	Daily	~40km	Global	soilm.smos	AS
Soil moisture	SMAP	2015-	Daily	3/9km	Global	soilm.smap	AS
Evapotranspiration	MOD16	2000-	8 days	1km	Global	evap.modis	AS
Water storage	GRACE	2002-	Monthly	1.0 º	Global	tws.grace	AS
Snow cover	MOD10	2001-	Daily	1km	Global	snow.mod10	AS
Snow cover	MOD-SCAG	2001-	Daily	1km	Global	snow.modscag	AS
Leaf Area Index	MCD15	2002-	8 days	1km	Global	lai.modis	AS
Meteorology	IRI	2000-	Monthly	2.5 º	Global	*.iri	FC
Meteorology	NMME	2000-	Daily	0.5 º	Global	*.nmme	FC

There are three modes for the datasets: `IN` corresponds to datasets being used as inputs to the model, `AS` refers to datasets being assimilated, and `FC` are datasets that are used to provide the meteorological (i.e. precipitation, temperature, and in some cases wind speed) forecasts.

CHAPTER 3

Populating the configuration file

The configuration file follows the INI format to allow for simplicity. When running the RHEAS executable, the configuration file is the only position argument while the database name needs to be provided with the `-d` switch.

Running RHEAS with the help switch

```
./rheas -h
```

produces the proper usage command

```
usage: rheas.py [-h] [-d DB] [-u] config

Runs RHEAS simulation.

positional arguments:
config      configuration file

optional arguments:
-h, --help  show this help message and exit
-d DB       name of database to connect
-u          update database
```

There are four possible sections for the configuration file:

- **nowcast**: nowcast simulation options
- **forecast**: forecast simulation options
- **vic**: VIC model options
- **dssat**: DSSAT model options

Each section needs to be given inside braces, e.g. `[nowcast]`.

3.1 Nowcast options

The available options for a nowcast simulation include:

- `model`: the model(s) to be used for this simulation. Valid options include `vic` and `dssat`; if both models are requested they need to be separated by a comma (*required*)
- `startdate`: the start date of the simulation in the "year-month-day" format (*required*)
- `enddate`: the start date of the simulation in the "year-month-day" format (*required*)
- `name`: name of the simulation (*required*)

- `basin`: path to a shapefile of the model domain. If not provided, the `name` option should correspond to a previously performed simulation
- `resolution`: spatial resolution of the simulation (*required*)

3.2 Forecast options

The available options for a forecast simulation include:

- `model`: the model(s) to be used for this simulation. Valid options include `vic` and `dssat`; if both models are requested they need to be separated by a comma (*required*)
- `startdate`: the start date of the simulation in the "year-month-day" format (*required*)
- `enddate`: the start date of the simulation in the "year-month-day" format (*required*)
- `name`: name of the simulation (*required*)
- `basin`: path to a shapefile of the model domain. If not provided, the `name` option should correspond to a previously performed simulation
- `resolution`: spatial resolution of the simulation (*required*)
- `ensemble size`: the size of the forecast ensemble (*required*)
- `method`: method to use to generate the meteorological forcings for VIC (*required*). The options that have been implemented include:
  - `esp`: use the Ensemble Streamflow Prediction approach that randomly resamples the climatology
  - `iri`: resample climatology based on the probabilities in the IRI meteorological forecasts

3.3 VIC options

The options for the VIC model include:

- `precip`: dataset to use for precipitation forcing (*required*)
- `temperature`: dataset to use for maximum and minimum temperature forcing (*required*)
- `wind`: dataset to use for wind speed forcing (*required*)
- `lai`: dataset to use for leaf area index forcing
- `save state`: directory where VIC model state file is saved in
- `save to`: option for saving output variables. Can be one of
  - `db`: save output to database
  - path to copy raw VIC output files to
- `initialize`: whether to initialize the model from a previously saved state file (can be given as `on/off`, `true/false` or `yes/no`)
- `save`: a comma-separated list of variables to be saved from VIC. The variable names can be:
  - `net_long`: net downward longwave flux [W/m2]
  - `net_short`: net downward shortwave flux [W/m2
  - `snow_cover`: fractional area of snow cover [fraction]

- `salbedo`: snow pack albedo [fraction]
- `snow_depth`: depth of snow pack [cm]
- `tdepth`: depth of thawing fronts [cm] for each thawing front
- `fdepth`: depth of freezing fronts [cm] for each freezing front
- `rootmoist`: root zone soil moisture [mm]
- `smfrozfrac`: fraction of soil moisture (by mass) that is ice, for each soil layer
- `smliqfrac`: fraction of soil moisture (by mass) that is liquid, for each soil layer
- `snow_canopy`: snow interception storage in canopy [mm]
- `soil_moist`: soil total moisture content [mm] for each soil layer
- `soil_wet`: vertical average of (soil moisture - wilting point)/(maximum soil moisture - wilting point) [mm/mm]
- `surfstor`: storage of liquid water and ice (not snow) on surface (ponding) [mm]
- `swe`: snow water equivalent in snow pack (including vegetation-intercepted snow) [mm]
- `wdew`: total moisture interception storage in canopy [mm]
- `baseflow`: baseflow out of the bottom layer [mm]
- `evap`: total net evaporation [mm]
- `evap_bare`: net evaporation from bare soil [mm]
- `evap_canop`: net evaporation from canopy interception [mm]
- `prec`: incoming precipitation [mm]
- `rainf`: rainfall [mm]
- `refreeze`: refreezing of water in the snow [mm]
- `runoff`: surface runoff [mm]
- `snow_melt`: snow melt [mm]
- `snowf`: snowfall [mm]
- `transp_veg`: net transpiration from vegetation [mm]
- `albedo`: average surface albedo [fraction]
- `baresoilt`: bare soil surface temperature [C]
- `snow_surf_temp`: snow surface temperature [C]
- `soil_temp`: soil temperature [C] for each soil layer
- `surf_temp`: average surface temperature [C]
- `vegt`: average vegetation canopy temperature [C]
- `advection`: advected energy [W/m2]
- `grnd_flux`: net heat flux into ground [W/m2]
- `latent`: net upward latent heat flux [W/m2]
- `melt_energy`: energy of fusion (melting) in snowpack [W/m2]
- `rfrz_energy`: net energy used to refreeze liquid water in snowpack [W/m2]

  - `sensible`: net upward sensible heat flux [W/m2]
  - `aero_cond`: "scene" aerodynamic conductance [m/s]
  - `air_temp`: air temperature [C]
  - `longwave`: incoming longwave [W/m2]
  - `shortwave`: incoming shortwave [W/m2]

- `observations`: a comma-separated list of the observations to be assimilated into VIC. Any of the datasets with `AS` mode outlined in the *database table* can be used with their table name (without the schema, e.g. `grace`)
- `update`: the date or frequency when assimilation should be performed. Valid options for the assimilation frequency are: `daily`, `weekly`, and `monthly`. If this option is not set, assimilation is performed whenever the observation is available during the simulation period. When performing a forecast simulation, this option is not taken into account and assimilation is performed at the forecast initialization date

3.4 DSSAT options

The options for the DSSAT model include:

- `shapefile`: a shapefile contains the areas (e.g. administrative boundaries) for which DSSAT will be run (*required*)
- `ensemble size`: the size of the ensemble to be used (*optional*)
- `assimilate`: flag indicating whether to assimilate soil moisture (`sm`), LAI (`lai`) observations or both (*optional*)

CHAPTER 4

Running a nowcast

We begin by creating the necessary RHEAS configuration file. In your favorite text editor create a new file, let's assume that its name is `nowcast.conf`. The first section contains the simulation options and has a `nowcast` header:

```
[nowcast]
```

Let's assume that we are requesting both a hydrologic and agricultural simulation. In that case, we need to set both models (VIC and DSSAT) as an option

```
[nowcast]
model: vic, dssat
```

We then set the starting and ending dates of the nowcast

```
[nowcast]
model: vic, dssat
startdate: 2003-1-1
enddate: 2003-3-31
```

Let's also assume that we have a shapefile of the study domain (the sample data contain such a file in the `data/tests` directory). Moreover, let's set the name of the simulation (that name will be useful to retrieve the output from the database).

```
[nowcast]
model: vic, dssat
startdate: 2003-1-1
enddate: 2003-3-31
basin: data/tests/basin.shp
name: basin
```

The model spatial resolution is next, here set as 0.25 degrees (~25 km).

```
[nowcast]
model: vic, dssat
startdate: 2003-1-1
enddate: 2003-3-31
basin: data/tests/basin.shp
name: basin
resolution: 0.25
```

These are all the necessary options to define our forecast simulation. Then we need to define the parameters for the two models (VIC and DSSAT).

We begin with the `vic` section

```
[vic]
```

VIC requires precipitation, temperature and wind speed at a minimum, which we define below

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
```

Setting the meteorological forcings this way will perform a deterministic nowcast simulation. Alternatively, a stochastic simulation can be performed by either requesting multiple datasets for precipitation or explicitly requesting an ensemble simulation

```
[vic]
precip: chirps, trmm
temperature: ncep
wind: ncep
ensemble size: 3
```

We can opt to initialize the VIC from a a state file saved in the database or an explicitly file name

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
initialize: yes
initial state: state/vic.state_20030101
```

If the option `initial state` is not set, RHEAS will look into the database for a statefile that is closest to the requested start date.

We choose the model output, specifically net shortwave radiation and soil moisture, to the database

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
initialize: yes
save to: db
save: net_short, soil_moist
```

The DSSAT section `dssat` requires fewer options than VIC, with the shapefile being the first

```
[dssat]
shapefile: data/tests/basin.shp
```

and the ensemble size being an optional parameter. Each of the DSSAT ensemble members is forced by the same VIC output retaining the uncertainty only in the DSSAT model parameters. There's also an option to enable or disable the assimilation of soil moisture and LAI data (enabled by default)

```
[dssat]
shapefile:
ensemble size: 50
assimilate: on
```

Finally, let's run the system (inside the `rheas` directory)

```
./bin/rheas nowcast.conf
```

CHAPTER 5

Running a forecast

As any RHEAS simulation, we begin by creating a configuration file. In your favorite text editor create a new file, let's assume that its name is `forecast.conf`. The first section contains the simulation options and has a `forecast` header:

```
[forecast]
```

Let's assume that we are requesting both a hydrologic and agricultural simulation. In that case, we need to set both models (VIC and DSSAT) as an option

```
[forecast]
model: vic, dssat
```

We then set the starting and ending dates of the forecast

```
[forecast]
model: vic, dssat
startdate: 2001-4-1
enddate: 2001-6-30
```

Let's also assume that we have a shapefile of the study domain (the sample data contain such a file in the `data/tests` directory). Moreover, let's set the name of the simulation (that name will be useful to retrieve the output from the database).

```
[forecast]
model: vic, dssat
startdate: 2001-4-1
enddate: 2001-6-30
basin: data/tests/basin.shp
name: basin
```

The model spatial resolution is next, here set as 0.25 degrees (~25 km).

```
[forecast]
model: vic, dssat
startdate: 2001-4-1
enddate: 2001-6-30
basin: data/tests/basin.shp
name: basin
resolution: 0.25
```

In order to capture the uncertainty in the forecast, we perform an ensemble simulation and set the ensemble size to 10. RHEAS will run the VIC ensemble in parallel (using multiple threads), thus reducing computation time. The ensemble simulation also requires a method to generate the meteorological forcings ensemble (currently, model uncertainty due to parameter errors is not implemented). Available methods are the Ensemble Streamflow Prediction

(ESP, resampling the climatology); resampling from the IRI meteorological forecasts; bias-correcting and downscaling the CFSv2 meteorological forecasts.

```
[forecast]
model: vic, dssat
startdate: 2001-4-1
enddate: 2001-6-30
basin: data/tests/basin.shp
name: basin
resolution: 0.25
ensemble size: 10
method: esp
```

These are all the necessary options to define our forecast simulation. Then we need to define the parameters for the two models (VIC and DSSAT).

We begin with the `vic` section

```
[vic]
```

The generation of the meteorological forecast ensemble requires a base dataset for each of the required variables. Therefore, we need to define the source for precipitation, temperature and wind speed

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
```

We can opt to initialize the ensemble of VIC models from a a state file saved in the database, by randomly selecting a model initial condition from climatology, or by starting a 1-year model run. All these options do not need to be set by the user apart from

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
initialize: yes
```

If RHEAS cannot find previous model states in the database, it will by default run a 1-year simulation for each ensemble member.

We choose the model output, specifically net shortwave radiation and soil moisture, to the database

```
[vic]
precip: chirps
temperature: ncep
wind: ncep
initialize: yes
save to: db
save: net_short, soil_moist
```

The DSSAT section `dssat` requires fewer options than VIC, with the domain shapefile being the first

```
[dssat]
shapefile: data/tests/basin.shp
```

and the ensemble size being another parameter that can optionally be set. The forecasts from VIC are randomly paired with each of the DSSAT ensemble members in order to capture uncertainty both in the DSSAT model parameters and the hydroclimatological forcings. Similarly to a nowcast, assimilation cna be enabled/disabled with an option.

```
[dssat]
shapefile: data/tests/basin.shp
ensemble size: 50
assimilate: no
```

Finally, let's run the system (inside the `rheas` directory)

```
./bin/rheas forecast.conf
```

CHAPTER 6

Customizing RHEAS

6.1 Initializing and updating the database

The RHEAS database needs to be populated with a variety of datasets that can be fetched automatically. A configuration file can be used to control and define which datasets are downloaded and imported in the PostGIS database. The configuration file follows the INI format, with each section corresponding to a dataset.

Unless the entire spatial extent of the datasets needs to be ingested into the database, a `domain` section should be created in the configuration file describing the bounding box for the domain of interest.

- `minlat`: the minimum latitude of the bounding box
- `maxlat`: the maximum latitude of the bounding box
- `minlon`: the minimum longitude of the bounding box
- `maxlon`: the maximum longitude of the bounding box

If the dataset is being fetched for the first time, a starting date needs to be provided within each dataset section while an ending date can be optionally provided.

- `startdate`: start date of data fetched (format is year-month-day)
- `enddate`: the last date of data to be fetched

If the dataset already exists in the database, then RHEAS will only download data after the last date available in the database (unless this is bypassed by the `startdate` keyword).

An example configuration file (named `data.conf`) that will download TRMM, CHIRPS and IRI datasets is given below.

```
[domain]
minlat: -2
maxlat: -2
minlon: 30
maxlon: 34

[trmm]

[iri]
startdate: 2000-2-1
enddate: 2000-3-1

[chirps]
startdate: 2014-1-3
```

Since no option is provided under the `trmm` section, RHEAS will download data from the latest date available in the database to today.

After the configuration file has been created, the database can be initialized/updated by calling RHEAS as

```
./bin/rheas -u data.conf
```

where `data.conf` is the name of the configuration file.

6.2 Using a custom database

Assuming that you have created (using RHEAS or not) a PostGIS database (named `customdb` here as an example) that contains the necessary schemas and tables, it can be used to perform nowcast and forecast simulations by

```
./bin/rheas -d customdb nowcast.conf
```

where `nowcast.conf` is the RHEAS configuration file.

6.3 Writing custom scripts with the RHEAS API

RHEAS exposes most of its functions within each module, allowing for a relatively simple API to be used to further customize it.

In the first example, we will assume that we want to run a deterministic VIC simulation but use custom meteorological data (from Numpy arrays). We first initialize a VIC object

```
import vic
model = vic.VIC(".", "customdb", 0.25, 2015, 1, 1, 2015, 3, 31, "customname")
```

and then write the VIC global and soil parameter files

```
model.writeParamFile()
model.writeSoilFile("basin.shp")
```

where `basin.shp` is a shapefile that describes our basin. Assuming that we have Numpy arrays containing the data for precipitation (`prec`), air temperature (`tmax` and `tmin`) and wind speed (`wind`) we can then write out the forcings for VIC, run the model and save the output into the database

```
model.writeForcings(prec, tmax, tmin, wind)
model.run(vicexe)
model.save("db", ["runoff"])
```

In a second example of using the RHEAS API, we will assume that we have a customized version of the DSSAT model (as an executable) that needs an additional line written in its configuration file. In order to achieve that, we will decorate the corresponding DSSAT class method and replace one of its parameters. We begin by initializing the DSSAT object

```
model = dssat.DSSAT("customdb", "customname", 0.25, 2015, 1, 1, 2015, 3, 31, 40, vicoptions, "basin.s
```

and then decorate the function `writeConfigFile` to change its behavior

```
def addLineToConfig(func):
    def wrapper(*args, **kwargs):
        fname = func(args, kwargs)
        with open(fname, 'a') as fout:
            fout.write("Additional line with parameters")
```

```
    return wrapper

model.writeConfigFile = addLineToConfig(model.writeConfigFile)
```

Finally, we run the customized DSSAT model

```
model.run("dssat_new.exe")
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search