R package LakeEnsemblR: Basic Use and Sample Applications

johannes.feldbauer@tu-dresden.de tadhg.moore@dkit.ie jorrit.mesman@unige.ch
ladwigjena@gmail.com

2020-04-12

Contents

1 Setting up LakeEnsemblR

1.1 Introduction

A key part of developing LakeEnsemblR was to develop a standardised format for model input data. This involved standard variable naming which includes units.

1.2 DateTime formatting

This package uses international standard format for date and time (ISO 8601), which is `YYYY-mm-dd HH:MM:SS`. For example: `2020-04-03 09:00:00`.

1.3 Time Zones

Currently this is not accounted for so the timezone used in input is the timezone of output data. It is on the list of things to do.

1.4 Hypsograph data

The data needs to be a comma separated values (.csv) file where 0m is the surface and all depths are reported as positive. Area needs to be in meters squared. The column names *must* be `Depth_meter` and `Area_meterSquared`

Example of data:

```
Depth_meter,Area_meterSquared
0,3931000
1,3688025
2,3445050
3,3336093.492
4,3225992.455
5,3133491.11
6,3029720
...
```

1.5 Temperature Profile data

The data needs to be a comma separated values (.csv) file where the datetime column is in the format `YYYY-mm-dd HH:MM:SS`. Depths are positive and relative to the water surface. Water temperature is in degrees Celsius. The column names *must* be `datetime`, `Depth_meter` and `Water_Temperature_celsius`

Example of data:

```
datetime,Depth_meter,Water_Temperature_celsius
2004-01-05 00:00:00,0.9,6.97
2004-01-05 00:00:00,2.5,6.71
2004-01-05 00:00:00,5,6.73
2004-01-05 00:00:00,8,6.76
...
```

1.6 Meteorological data

The data needs to be a comma separated values (.csv) file where the datetime column is in the format `YYYY-mm-dd HH:MM:SS`. See table 1 for the list of variables, units and column names.

Table 1. Description of meteorological variables used within LakeEnsemblR with units and required column names.

Description	Units	Column Name	Status
Downwelling longwave radaiation	W/m2	Longwave_Radiation_Downwelling_wattPerMeterSquared	If not provided,it is calculated internally from air temperature, cloud cover and relative humidity/dewpoint temperature
Downwelling shortwave radaiation	W/m2	Shortwave_Radiation_Downwelling_wattPerMeterSquared	Required
Cloud cover	-	Cloud_Cover_decimalFraction	If not provided,it is calculated internally from air temperature, short-wave radiation, latitude, longitude, elevation and relative humidity/dewpoint temperature
Air temperature	°C	Air_Temperature_celsius	Required
Relative humidity	%	Relative_Humidity_percent	If not provided,it is calculated internally from air temperature and dewpoint temperature

Description	Units	Column Name	Status
Dewpoint temperature	°C	Dewpoint_Temperature_celsius	If not provided,it is calculated internally from air temperature and relative humidity
Wind speed at 10m	m/s	Ten_Meter_Elevation_Wind_Speed_meterPerSecond	Either wind speed or u and v vectors is required
Wind direction at 10m	°C	Ten_Meter_Elevation_Wind_Direction_degree	Not required, but if provided u and v vectors are calculated internally
Wind u-vector at 10m	m/s	Ten_Meter_Uwind_vector_meterPerSecond	Either wind speed or u and v vectors is required
Wind v-vector at 10m	m/s	Ten_Meter_Vwind_vector_meterPerSecond	Either wind speed or u and v vectors is required
Precipitation	m/s	Precipitation_meterPerSecond	Not strictly required but is important for mass budgets in some models
Rainfall	m/s	Rainfall_meterPerSecond	Required
Snowfall	m/day	Snowfall_meterPerDay	If not provided,it is calculated internally from rain when air temperature < 0 degC
Sea level pressure	Pa	Sea_Level_Barometric_Pressure_pascal	Not required
Surface level pressure	Pa	Surface_Level_Barometric_Pressure_pascal	Required
Vapour pressure	mbar	Vapor_Pressure_milliBar	If not provided,it is calculated internally from air temperature and relative humidity/dewpoint temperature

1.7 LakeEnsemblR YAML configuration file

There is an example yaml configuration provided in the example dataset in the package or you can download a copy from GitHub here.

You will need to update each of the required variables in the location block to reflect your own site.

```
location:
   name: Feeagh                                # station name used in output [default=GOTM site]
   latitude: 53.9                              # latitude [degrees North; min=-90.0; max=90.0; default=
   longitude: -9.5                             # longitude [degrees East; min=-360.0; max=360.0; defaul
   elevation: 15                               # elevation of lake surface above sea level [m]
   depth: 46.8                                 # maximum water depth [m; min=0.0; default=100.0]
   hypsograph: LakeEnsemblR_bathymetry_standard.csv                    # hypsograph [default=]
   init_depth: 46.8                            # initial height of lake surface relative to the bottom
```

Then input the filepaths to the bathymetry, meteorlogical and water temperature profile observations file. For first time users we would recommend to set up a folder with just these three files plus the LakeEnsemblR yaml configuration file.

Now you should be ready to run LakeEnsemblR on your site.

2 Running LakeEnsemblR

Once you have your hypsograph, water temperature observations and meteorological files prepared tunning LakeEnsemblR is relatively straightforward.

2.1 Example model run

```
# Install packages - Ensure all packages are up to date - parallel development ongoing
#install.packages('devtools')
devtools::install_github('GLEON/GLM3r')
devtools::install_github('USGS-R/glmtools', ref = 'ggplot_overhaul')
devtools::install_github('aemon-j/FLakeR')
devtools::install_github('aemon-j/GOTMr')
devtools::install_github('aemon-j/gotmtools')
devtools::install_github('aemon-j/SimstratR')
devtools::install_github('aemon-j/LakeEnsemblR')
devtools::install_github('aemon-j/MyLakeR')

# Load libraries
library(gotmtools)
library(LakeEnsemblR)

# Set working directory
setwd('example') # Change working directory to example folder

# Set models & config file
model <- c('GLM',  'FLake', 'GOTM', 'Simstrat', 'MyLake')
config_file <- 'Feeagh_master_config.yaml'

# 1. Example - creates directories with all model setup
export_config(config_file = config_file, model = model, folder = '.')

# 2. Create meteo driver files
export_meteo(config_file = config_file, model = model)

# 3. Create initial conditions
start_date <- get_yaml_value(file = config_file, label =  "time", key = "start")

export_init_cond(config_file = config_file,
                 model = model,
                 date = start_date,
                 print = TRUE)

# 4. Run ensemble lake models
wtemp_list <- run_ensemble(config_file = config_file,
                           model = c('FLake', 'GLM', 'GOTM', 'Simstrat', 'MyLake'),
                           return_list = TRUE)
```

2.2 Post-processing

```
# Load libraries for post-processing
library(ggpubr)
library(ggplot2)

## Plot model output using gotmtools/ggplot2

# Extract names of all the variables in netCDF
```

```r
ens_out <- 'output/ensemble_output.nc'
vars <- gotmtools::list_vars(ens_out)
vars # Print variables

plist <- list() # Initialize empty list for storing plots of each variable
for(i in 1:5){
  p1 <- gotmtools::plot_vari(ncdf = ens_out,
                             var = vars[i],
                             incl_time = FALSE,
                             limits = c(0,25),
                             zlab = 'degC')
  p1 <- p1 + scale_y_reverse() + #Reverse y-axis
    coord_cartesian(ylim = c(45,0))+ # ggplot2 v3.3 is sensitive to order of ylim
    ggtitle(vars[i]) + # Add title using variable name
    xlab('')+ # Remove x-label
    theme_bw(base_size = 18) # Increase font size of plots
  plist[[i]] <- p1
}

# Plot all model simulations
# install.packages('ggpubr')
g1 <- ggpubr::ggarrange(plotlist = plist, ncol = 1, common.legend = TRUE, legend = 'right')
g1
ggsave('output/model_ensemble_watertemp.png', g1,  dpi = 300,width = 384,height = 300, units = 'mm')
```

3 References