Intro to LakeEnsemblR - CIBR 2021

Tadhg Moore, Jorrit Mesman, Robert Ladwig, Johannes Feldbauer

2021-05-27

1. Introduction

LakeEnsemblR is a new R package created by GLEON-members, that facilitates running ensemble model runs of up to five different models (FLake, GLM, GOTM, Simstrat, and MyLake) in a consistent and standardised manner.

During this workshop, you will learn how to set-up and run LakeEnsemblR for an example set-up, how to calibrate the models, and how to use several post-processing functions that are included in the package. Once you understand how the package works, you can also try to change model parameters, or apply it to your own lake data.

The code is available at https://github.com/aemon-j/LakeEnsemblR, and several example set-ups can be found on https://github.com/aemon-j/LER_examples. There is also a Wiki page that hosts information and some Frequently Asked Questions (FAQ): https://github.com/aemon-j/LakeEnsemblR/wiki

2. Installation and loading of packages

Install LakeEnsemblR from Github and if you haven't done so already, `install.packages("remotes")` first.

We will be running five different models, each of which has a corresponding R package that contains the model executables. We will also need `gotmtools` and `glmtools`, which help in reading and writing model-specific files. The GLEON package `rLakeAnalyzer` is also used for running analysis and formatting of data allows for compatibility with this package. Install these packages first (unless you've already done this), followed by `LakeEnsemblR` itself.

WARNING: Installing all these packages can potentially take a long time so it is **RECOMMENDED** that you do this **PRIOR** to attending the workshop.

```
#install.packages("remotes")

remotes::install_github("GLEON/rLakeAnalyzer")
remotes::install_github("USGS-R/glmtools", ref = "ggplot_overhaul")
remotes::install_github("aemon-j/gotmtools", ref = "yaml")
remotes::install_github("FLARE-forecast/GLM3r")
remotes::install_github("aemon-j/GOTMr")
remotes::install_github("aemon-j/SimstratR")
remotes::install_github("aemon-j/FLakeR", ref = "inflow")
remotes::install_github("aemon-j/MyLakeR")
```

Once you have the packages installed you can install `LakeEnsemblR`. As a side note, if you only wish to use one or two of the lake models with `LakeEnsemblR` then you only need to install those packages and all the `LakeEnsemblR` functions will work with those models.

For this workshop we will use a branch from Tadhg's repo as these are updates that he has worked on for integration with FLARE.

```
remotes::install_github("tadhg-moore/LakeEnsemblR", ref = "flare")
```

3. Load example set-up

An example set-up for Lough Feeagh (Ireland) has been included in the package. Load it with the following lines of code.

Now we copy Lough Feeagh example folder into the working directory and change our working directory to the newly created "feeagh" folder

```
template_folder <- system.file("extdata/feeagh", package= "LakeEnsemblR")
file.copy(from = template_folder, to = ".", recursive = TRUE)
```

Set working directory to the new folder (again, in a regular script by using `setwd()`).

Load libraries

```
library(gotmtools)
library(LakeEnsemblR)
```

Have a look at the feeagh folder. There will be six files

```
list.files()
```

```
##  [1] "cali"
##  [2] "FLake"
##  [3] "GLM"
##  [4] "GOTM"
##  [5] "LakeEnsemblR.yaml"
##  [6] "LakeEnsemblR_bathymetry_standard.csv"
##  [7] "LakeEnsemblR_ice-height_standard.csv"
##  [8] "LakeEnsemblR_inflow_standard.csv"
##  [9] "LakeEnsemblR_initial.yaml"
## [10] "LakeEnsemblR_meteo_standard.csv"
## [11] "LakeEnsemblR_wtemp_profile_standard.csv"
## [12] "MyLake"
## [13] "output"
## [14] "Simstrat"
```

"LakeEnsemblR.yaml" is the configuration file for LakeEnsemblR. It contains all information needed to run the run the models; start and end of simulation, time step, names of input files, output settings, etc. You can open the file directly in RStudio by holding "Ctrl" and left-clicking the filename ("LakeEnsemblR.yaml") in the R editor. Alternatively you can use a text editor (e.g. Notepad++).

Have a look at the LakeEnsemblR.yaml file and see if you understand what each section does. You will see that in this file, there are the names of the other files in the directory, e.g. "LakeEnsemblR_meteo_standard.csv", for meteorological input. Open the meteo file in a text editor. The column names are important; this is how LakeEnsemblR knows what column is what variable. The required headers for any variable can be found in the dictionary which comes with the package.

```
## location:
##    name: Feeagh                                  # name of the lake
##    latitude: 53.9                                # latitude [degrees North; min=-90.0; max=90.0]
##    longitude: -9.5                               # longitude [degrees East; min=-360.0; max=360.0]
##    elevation: 15                                 # elevation of lake surface above sea level [m]
##    depth: 46.8                                   # maximum water depth [m; min=0.0]
##    hypsograph: LakeEnsemblR_bathymetry_standard.csv         # hypsograph [csv file]
##    init_depth: 46.8                              # initial height of lake surface relative to the bot
## time:
##    start: 2010-01-01 00:00:00                    # start date and time [yyyy-mm-dd HH:MM:SS]
##    stop: 2010-12-31 00:00:00                     # stop date and time [yyyy-mm-dd HH:MM:SS]
##    time_step: 3600.0                             # time step for integration [s; min=0.0]
## config_files:
##    GOTM: GOTM/gotm.yaml                          # GOTM config file [yaml file]
##    GLM: GLM/glm3.nml                             # GLM config file [nml file]
##    Simstrat: Simstrat/simstrat.par               # Simstrat config file [json-format file]
##    FLake: FLake/flake.nml                        # FLake config file [nml file]
##    MyLake: MyLake/mylake.Rdata                   # MyLake config file [Rdata file]
## observations:
##    temperature:
##       file: LakeEnsemblR_wtemp_profile_standard.csv         # file with observed water temperature pr
##    ice_height:
##       file: NULL                                 # file with observed ice height, with col
## input:
##    init_temp_profile:
##       file: NULL                                 # initial temperature profile [csv file; if none use
##    meteo:
##       file: LakeEnsemblR_meteo_standard.csv      # file with meteorological forcing data, with column
##    light:
##       Kw: 0.98                                   # light extinction coefficient [m-1, or csv file]
##    ice:
##       use: true                                  # turn on ice models? [true/false]
## inflows:
##    use: true                                     # use in- and outflows? [true/false]
##    file: LakeEnsemblR_inflow_standard.csv        # file with inflow data, with column headers accordi
##    scale_param: 1.0                              # scaling factor for discharge in inflow, for exampl
##    mass-balance: true                            # enforce pseudo mass-balance by adding an artifical
## output:
##    file: ensemble_output                         # name of output file, excluding extension
##    format: netcdf                                # format [text/netcdf]
##    depths: 0.5                                   # depths to extract output [m]
##    compression: 4                                # set to an integer between 1 (least compression) and
##    time_unit: hour                               # time unit [second, hour, day]
##    time_step: 24                                 # number of time units between output [min=1]
##    time_method: mean                             # treatment of time dimension [point=instantaneous, m
##    variables:
##       - temp
##       - ice_height
## scaling_factors:                                 # scaling factors to apply to meteorological input, c
##    all:
##       wind_speed: 1.0
##       swr: 1.0
##    Simstrat:
##       wind_speed: 1.0
```

```
## model_parameters:
##    FLake:                                          # FLake specific parameters
##       LAKE_PARAMS/fetch_lk: 2000.0                           # typical wind fetch [m]
##    GLM:                                            # GLM specific parameters
##       morphometry/bsn_len: 3678                              # length of the lake basin, at crest hei
##       morphometry/bsn_wid: 944                               # width of the lake basin, at crest heig
##    GOTM:                                           # GOTM specific parameters
##       turbulence/turb_param/k_min: 3.6E-6                                 # minimum turbulent kinetic en
##    Simstrat:                                       # Simstrat specific parameters
##      ModelParameters/a_seiche: 0.003
##    MyLake:                                         # MyLake specific parameters
##       Phys.par/C_shelter: 0.15                     # wind sheltering coefficient [min=0; max=1; if not
## calibration:                                       # calibration section
##    met:                                            # meteo scaling parameter
##       wind_speed:                                  # wind speed scaling
##          lower: 0.5                                # lower bound for wind speed scaling
##          upper: 2                                  # upper bound for wind speed scaling
##          initial: 1                                # initial value for wind speed scaling
##          log: false                                # log transform scaling factor
##       swr:                                         # shortwave radiation scaling
##          lower: 0.5                                # lower bound for shortwave radiation scaling
##          upper: 1.5                                # upper bound for shortwave radiation scaling
##          initial: 1                                # initial value for shortwave radiation scaling
##          log: false                                # log transform scaling factor
##    FLake:                                          # FLake specific parameters
##       LAKE_PARAMS/c_relax_C:
##          lower: 0.0001                             # lower bound for parameter
##          upper: 0.01                               # upper bound for parameter
##          initial: 0.0030                           # initial value for parameter
##          log: false                                # log transform scaling factor
##    GLM:                                            # GLM specific parameters
##       mixing/coef_mix_hyp:
##          lower: 0.1                                # lower bound for parameter
##          upper: 2                                  # upper bound for parameter
##          initial: 1                                # initial value for parameter
##          log: false                                # log transform scaling factor
##    GOTM:                                           # GOTM specific parameters
##       turbulence/turb_param/k_min:
##          lower: 5E-6                               # lower bound for parameter
##          upper: 5E-4                               # upper bound for parameter
##          initial: 1E-5                             # initial value for parameter
##          log: true
##    Simstrat:                                       # Simstrat specific parameters
##       ModelParameters/a_seiche:
##          lower: 0.0008                             # lower bound for parameter
##          upper: 0.003                              # upper bound for parameter
##          initial: 0.001                            # initial value for parameter
##          log: false                                # log transform scaling factor
##    MyLake:                                         # MyLake specific parameters
##       Phys.par/C_shelter:
##          lower: 0.14                               # lower bound for parameter
##          upper: 0.16                               # upper bound for parameter
##          initial: 0.15                             # initial value for parameter
##          log: false                                # log transform scaling factor
```

You can view the standardized naming for the meteorological variables here:

```
data("met_var_dic", package = "LakeEnsemblR")
print(met_var_dic)
```

```
##                                        standard_name short_name
## 1                                           datetime       time
## 2   Longwave_Radiation_Downwelling_wattPerMeterSquared        lwr
## 3  Shortwave_Radiation_Downwelling_wattPerMeterSquared        swr
## 4                          Cloud_Cover_decimalFraction         cc
## 5                              Air_Temperature_celsius       airt
## 6                            Relative_Humidity_percent       relh
## 7                         Dewpoint_Temperature_celsius       dewt
## 8        Ten_Meter_Elevation_Wind_Speed_meterPerSecond wind_speed
## 9            Ten_Meter_Elevation_Wind_Direction_degree   wind_dir
## 10               Ten_Meter_Uwind_vector_meterPerSecond        u10
## 11               Ten_Meter_Vwind_vector_meterPerSecond        v10
## 12                     Precipitation_millimeterPerDay     precip
## 13                    Precipitation_millimeterPerHour   precip_h
## 14                         Rainfall_millimeterPerDay       rain
## 15                        Rainfall_millimeterPerHour     rain_h
## 16                         Snowfall_millimeterPerDay       snow
## 17                        Snowfall_millimeterPerHour     snow_h
## 18                Sea_Level_Barometric_Pressure_pascal      p_sea
## 19            Surface_Level_Barometric_Pressure_pascal     p_surf
## 20                           Vapour_Pressure_milliBar      vap_p
## 21                    Extinction_Coefficient_perMeter   ext_coef
```

Or alternatively, this is further described on our Wiki page: https://github.com/aemon-j/LakeEnsemblR/wiki/2.-Setting-up-LakeEnsemblR which gives further details on which variables are required and which can be internally calculated if not present.

(From this point on, we'll refer to LakeEnsemblR as "LER")

4. Export configuration files

Each of the five models requires its own set of configuration and input files, preferably in different file formats and units. But, LER takes care of this for you! Using the LER config file and your input files, the `export_config()` function will create separate folders for each model, with the right set-up for each model.

```
export_config("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                             "Simstrat", "MyLake"))
```

Now have a look at the folder again. Folders for each model have been created. Because our folder was empty, LER used templates stored in the package and the information in the input and config files to create these folders (if the files in the "config_files" section already exist, it will make changes in these files instead).

Each model can now be run in its respective folder.

Something we won't treat further now, but which may be interesting for you when you set up your own simulations; `export_config()` only exports the information included in the LER config file. Other settings in the model-specific config files are kept in their defaults. Should you wish to change model-specific parameters, you can add them to the "model_parameters" section in the LER config file. In our example, this has been

done for several parameters in FLake, GLM, GOTM, and MyLake. Should you wish to change settings that are more difficult to access than by just changing a parameter (e.g. inflow-depth in Simstrat), you can do that in between running `export_config()` and `run_ensemble()`

5. Running the model ensemble

As mentioned at the start of this script, the models are actually run by other packages; FLakeR, GLM3r, GOTMr, SimstratR, and MyLakeR. These packages contain executables for Windows, MacOS, and Linux (except for MyLakeR, which instead contains the MyLake code in R). LER calls these packages to run the models in the folders we've just created.

```
run_ensemble("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                            "Simstrat", "MyLake"))
```

Each model folder will now have an "output" folder which contains the model-specific output. However, LER has also compiled these results in a standardised netcdf format, in an "output" folder in our main directory. NetCDF files can be easily accessed by PyNcView https://sourceforge.net/projects/pyncview/. (If you wish to find out what a NetCDF file is and why we use it, check out our Wiki: https://github.com/aemon-j/LakeEnsemblR/wiki/What-is-a-NetCDF-file%3F)

If you prefer text output, you can also opt for that type of output in the LER config file. The output generated is in the same format that is used in `rLakeAnalyzer` so will allow you to use functions from that package, which you may be familiar with, for loading and plotting the data. Currently the post-processing functions in LER only work with the netCDF format.

Let's generate some text output, and also make this change in the LER config file from within R, using LER's input_yaml_multiple function.

```
ler_yaml <- "LakeEnsemblR.yaml"
yaml <- read_yaml(ler_yaml)
yaml$output$format <- "text" # Change output to .csv files and rerun the ensemble

# Need to write the updated config
write_yaml(yaml, ler_yaml)
run_ensemble("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                            "Simstrat", "MyLake"))
```

Now text output has been generated in the "output" folder. But let's set the output format back to netcdf, as we'll need that later.

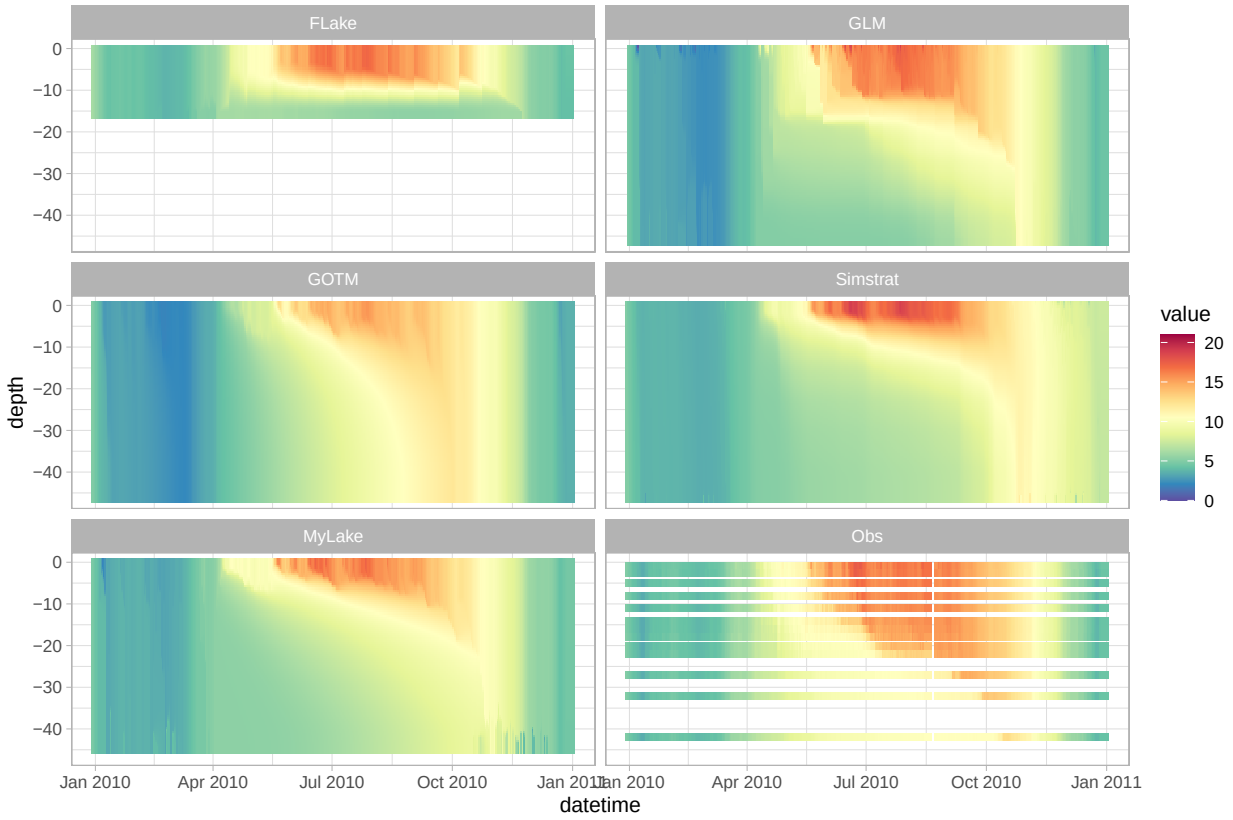
```
yaml <- read_yaml(ler_yaml)
yaml$output$format <- "netcdf"
write_yaml(yaml, ler_yaml)
```

Exploring the netCDF file

Have a look at the netcdf output by opening it in PyNcView. In the left menu, select "z,time,model,member,lat,lon" and click "temp". Then in the right-hand menu, tick the boxes "member" and "model". You can increase the value of "model" by increments. The order of the models is the same as in our function call (i.e. "FLake", "GLM", "GOTM", "Simstrat", "MyLake"), followed by observations. We'll forget about "member" right now, and treat that later. Under "time,model,member,lat,lon", you can also see plots of ice thickness, but that's not too exciting as Lough Feeagh has no ice cover...

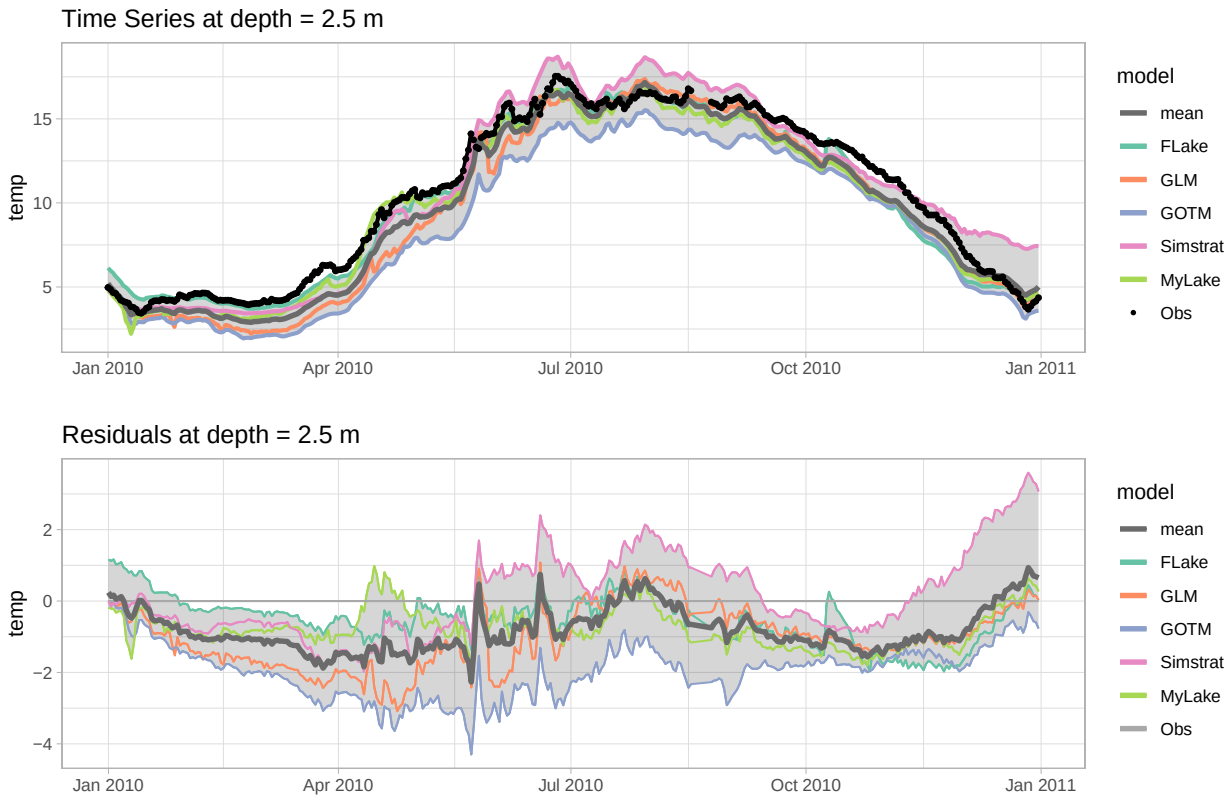
You can also plot the output with the LER `plot_heatmap()` function. It returns a ggplot object, that you can modify further if needed (as done here with `scale_colour_gradientn()` and `theme_light()`).

```
library(ggplot2)
plot_heatmap("output/ensemble_output.nc") +
  scale_colour_gradientn(limits = c(0, 21),
                         colours = rev(RColorBrewer::brewer.pal(11, "Spectral")))+
  theme_light()
```



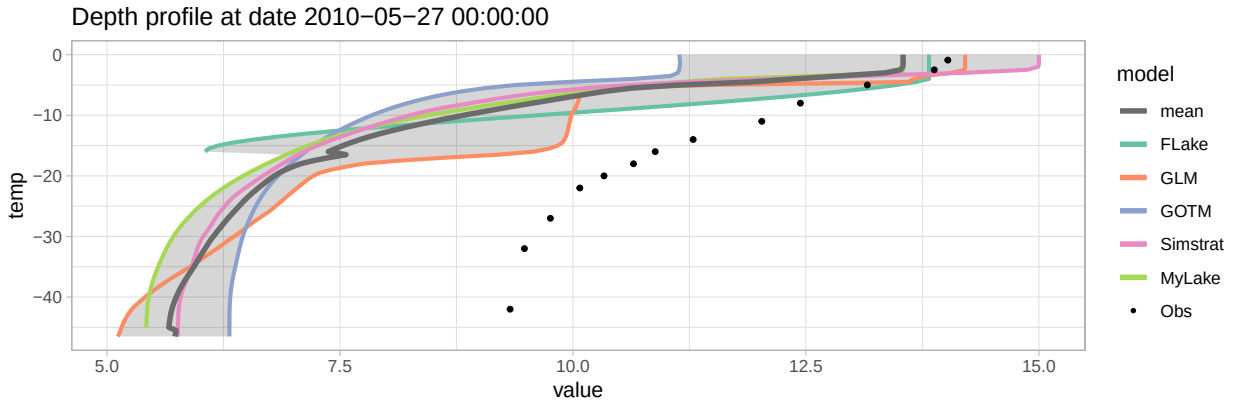
Using the `plot_ensemble()` function, plots of temperature at a certain depth over time can easily be created. If observed data is available, the function also plots the residuals:

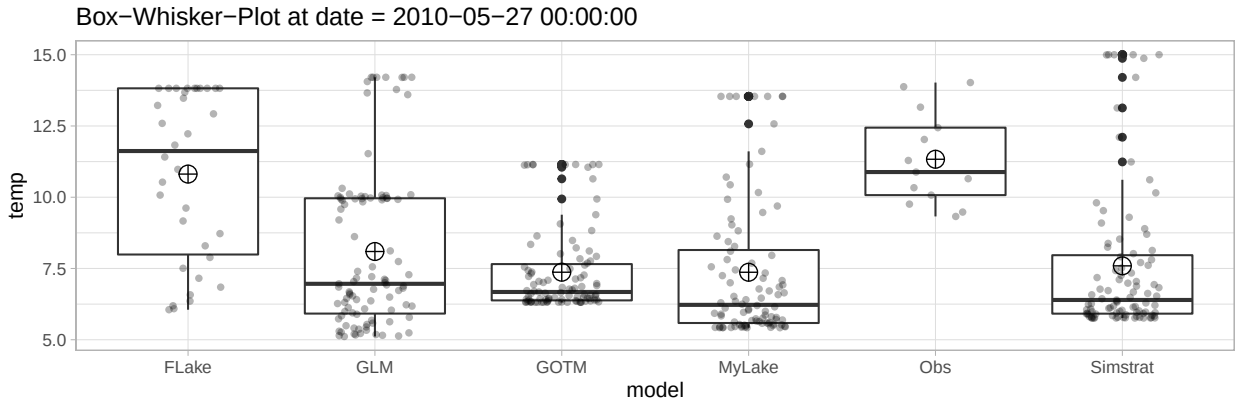
```
library(ggpubr)
p1 <- plot_ensemble("output/ensemble_output.nc", model = c("FLake", "GLM",
                                                           "GOTM", "Simstrat",
                                                           "MyLake"),
                    var = "temp", depth = 2.5,
                    residuals = TRUE)

ggarrange(p1[[1]] + theme_light(),
          p1[[2]] + theme_light(),ncol = 1, nrow = 2)
```



The function can also be used to plot depth profiles at selected dates and additionally box plots of the variable can be created too:

```
p2 <- plot_ensemble("output/ensemble_output.nc", model = c("FLake", "GLM",
                                                           "GOTM", "Simstrat",
                                                           "MyLake"),
                    var = "temp", date = "2010-05-27 00:00:00",
                    boxwhisker = TRUE, residuals = FALSE)

ggarrange(p2[[1]] + theme_light(),
          p2[[2]] + theme_light(), ncol = 1, nrow = 2)
```



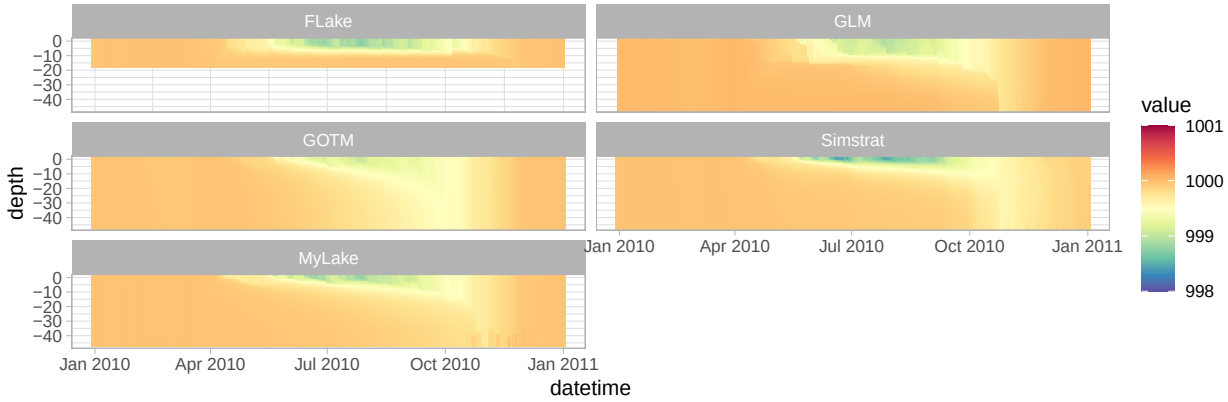


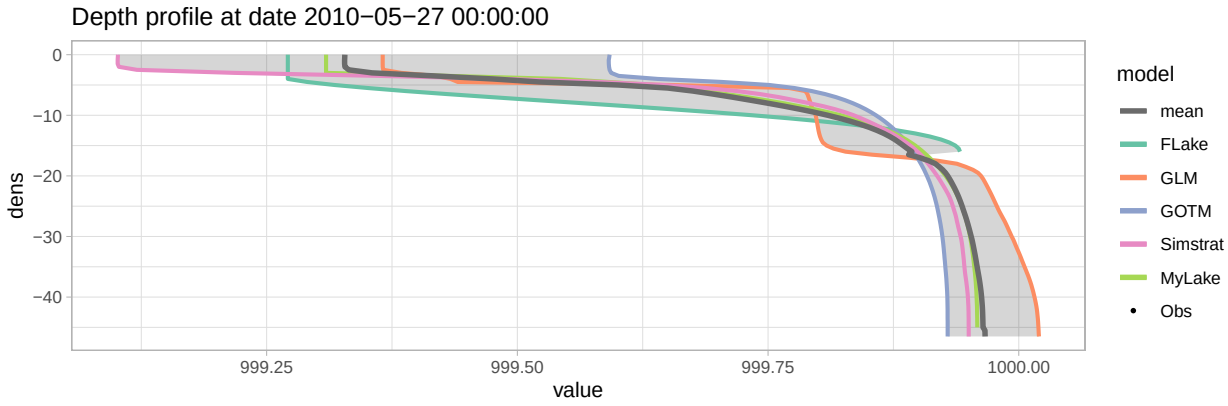
LakeEnsemblr can also gather other output variables from the model, such as density or salinity, in the created netcdf output file. To do so you just need to add them to the LakeEnsemblR.yaml file in the output section

```
yaml <- read_yaml(ler_yaml)
yaml$output$variables
yaml$output$variables <- c("temp", "ice_height", "dens")
write_yaml(yaml, ler_yaml)

run_ensemble("LakeEnsemblR.yaml",
             model = c("FLake", "GLM", "GOTM", "Simstrat", "MyLake"),
             parallel = TRUE,
             add = FALSE)
```

Now we can plot this new variable in the netcdf file using e.g. the `plot_heatmap()` or the `plot_ensemble()` function:

```
  p3 <- plot_heatmap("output/ensemble_output.nc", var = "dens") +
    theme_light() + scale_colour_gradientn(limits = c(998, 1001),
                                           colours = rev(RColorBrewer::brewer.pal(11, "Spectral")))
  p4 <- plot_ensemble("output/ensemble_output.nc", model = c("FLake", "GLM",
                                                             "GOTM", "Simstrat",
                                                             "MyLake"),
                      var = "dens", date = "2010-05-27 00:00:00") +
    theme_light()

  ggarrange(p3, p4, ncol = 1, nrow = 2)
```





Plotting text output

There are no functions in LER to plot the text output, but you could use this code, for example:

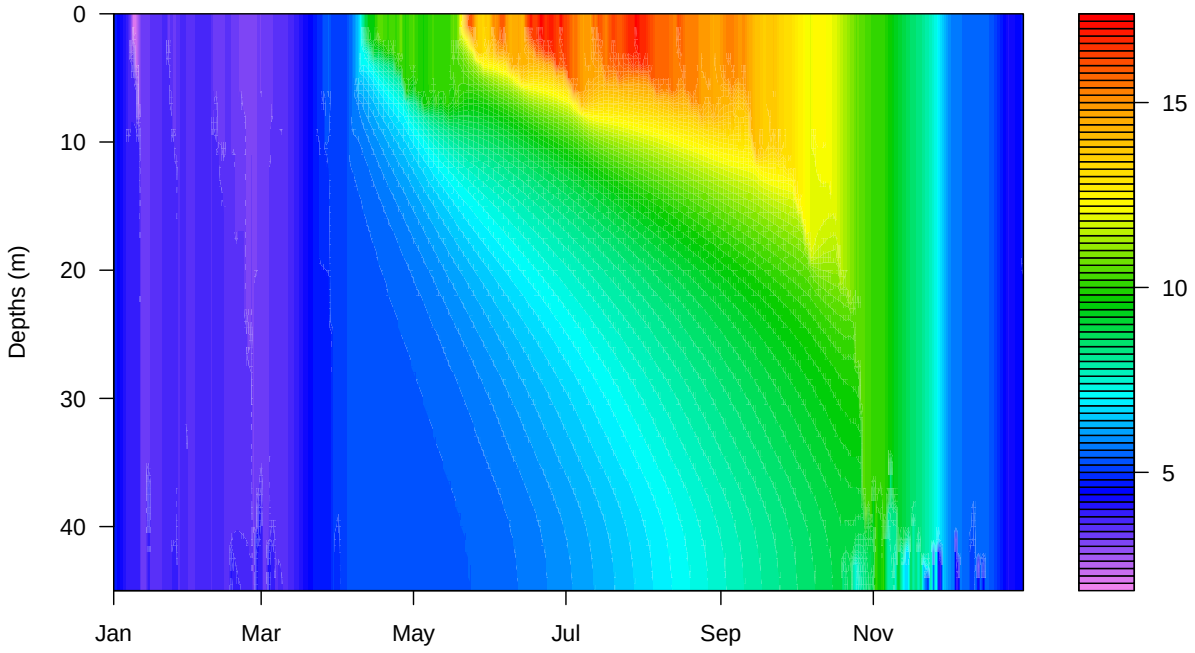
```
plot_model <- "MyLake" # Model names are case-sensitive
plot_depth <- 5 # In our example, output is given every 0.5 m
# Read in the data
wtr <- read.csv(paste0("./output/Feeagh_", plot_model, "_temp.csv"))
wtr$datetime <- as.POSIXct(wtr$datetime)

head(wtr) # data frame is in rLakeAnalyzer format
```

```
##     datetime wtr_0 wtr_0.5 wtr_0.9 wtr_1 wtr_1.5 wtr_2 wtr_2.5 wtr_3 wtr_3.5
## 1 2010-01-01  4.78    4.78    4.78  4.78    4.78  4.78    4.78  4.78    4.78
## 2 2010-01-02  4.64    4.64    4.64  4.64    4.64  4.64    4.64  4.64    4.64
## 3 2010-01-03  4.45    4.45    4.45  4.45    4.45  4.45    4.45  4.45    4.45
## 4 2010-01-04  4.29    4.29    4.29  4.29    4.29  4.29    4.29  4.29    4.29
## 5 2010-01-05  4.14    4.14    4.14  4.14    4.14  4.14    4.14  4.14    4.14
## 6 2010-01-06  3.94    3.94    3.94  3.94    3.94  3.94    3.94  3.94    3.94
##   wtr_4 wtr_4.5 wtr_5 wtr_5.5 wtr_6 wtr_6.5 wtr_7 wtr_7.5 wtr_8 wtr_8.5 wtr_9
## 1  4.78    4.78  4.78    4.78  4.78    4.78  4.78    4.78  4.78    4.78  4.78
## 2  4.64    4.64  4.64    4.64  4.64    4.64  4.64    4.64  4.64    4.64  4.64
## 3  4.45    4.45  4.45    4.45  4.45    4.45  4.45    4.45  4.45    4.45  4.45
## 4  4.29    4.29  4.29    4.29  4.29    4.29  4.29    4.29  4.29    4.29  4.29
## 5  4.14    4.14  4.14    4.14  4.14    4.14  4.14    4.14  4.14    4.14  4.14
## 6  3.94    3.94  3.94    3.94  3.94    3.94  3.94    3.94  3.94    3.94  3.94
```

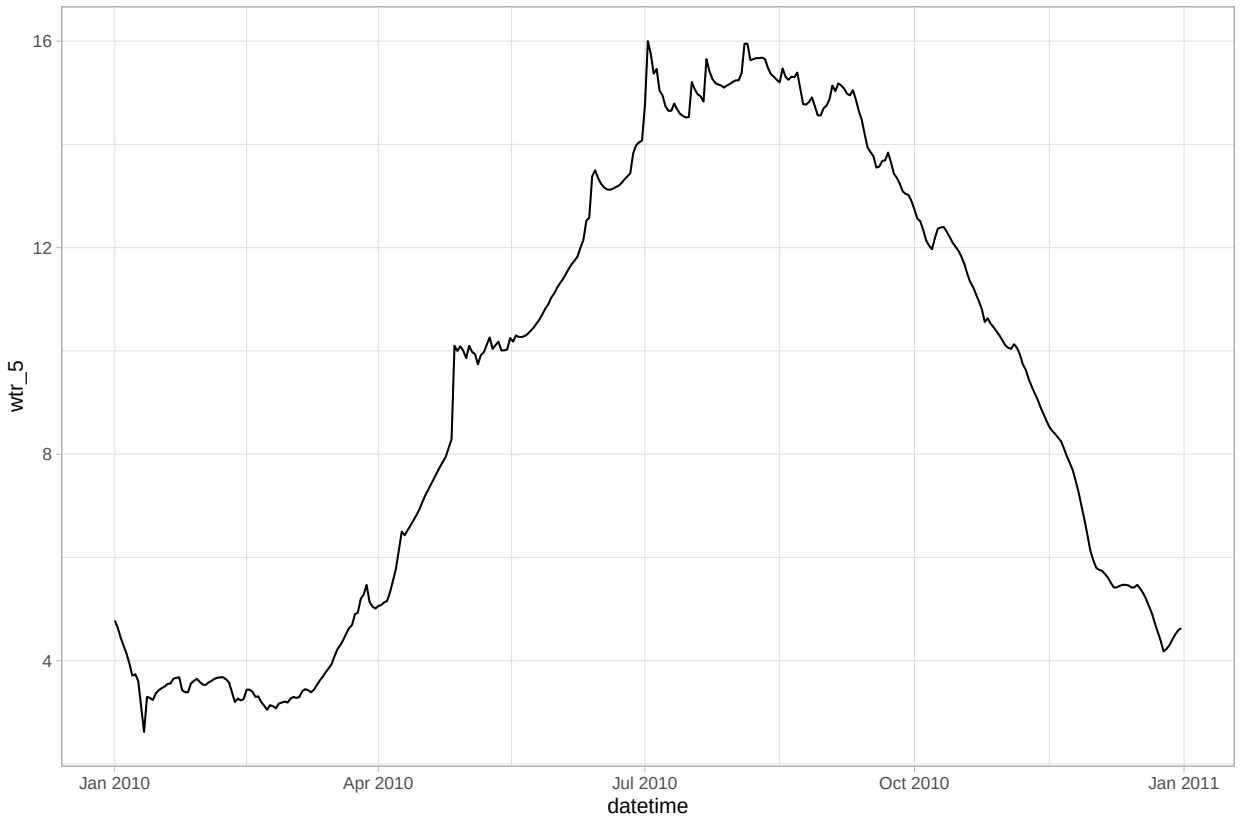
```
##   wtr_9.5 wtr_10 wtr_10.5 wtr_11 wtr_11.5 wtr_12 wtr_12.5 wtr_13 wtr_13.5
## 1    4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78
## 2    4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64
## 3    4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45
## 4    4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29
## 5    4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14
## 6    3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94
##   wtr_14 wtr_14.5 wtr_15 wtr_15.5 wtr_16 wtr_16.5 wtr_17 wtr_17.5 wtr_18
## 1   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78
## 2   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64
## 3   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45
## 4   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29
## 5   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14
## 6   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94
##   wtr_18.5 wtr_19 wtr_19.5 wtr_20 wtr_20.5 wtr_21 wtr_21.5 wtr_22 wtr_22.5
## 1     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78
## 2     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64
## 3     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45
## 4     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29
## 5     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14
## 6     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94
##   wtr_23 wtr_23.5 wtr_24 wtr_24.5 wtr_25 wtr_25.5 wtr_26 wtr_26.5 wtr_27
## 1   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78
## 2   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64
## 3   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45
## 4   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29
## 5   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14
## 6   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94
##   wtr_27.5 wtr_28 wtr_28.5 wtr_29 wtr_29.5 wtr_30 wtr_30.5 wtr_31 wtr_31.5
## 1     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78
## 2     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64
## 3     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45
## 4     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29
## 5     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14
## 6     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94
##   wtr_32 wtr_32.5 wtr_33 wtr_33.5 wtr_34 wtr_34.5 wtr_35 wtr_35.5 wtr_36
## 1   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78
## 2   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64
## 3   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45
## 4   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29
## 5   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14
## 6   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94
##   wtr_36.5 wtr_37 wtr_37.5 wtr_38 wtr_38.5 wtr_39 wtr_39.5 wtr_40 wtr_40.5
## 1     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78
## 2     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64
## 3     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45
## 4     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29
## 5     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14
## 6     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94
##   wtr_41 wtr_41.5 wtr_42 wtr_42.5 wtr_43 wtr_43.5 wtr_44 wtr_44.5 wtr_45
## 1   4.78     4.78   4.78     4.78   4.78     4.78   4.78     4.78   4.78
## 2   4.64     4.64   4.64     4.64   4.64     4.64   4.64     4.64   4.64
## 3   4.45     4.45   4.45     4.45   4.45     4.45   4.45     4.45   4.45
## 4   4.29     4.29   4.29     4.29   4.29     4.29   4.29     4.29   4.29
```

```
## 5   4.14     4.14   4.14     4.14   4.14     4.14   4.14     4.14   4.14
## 6   3.94     3.94   3.94     3.94   3.94     3.94   3.94     3.94   3.94
```

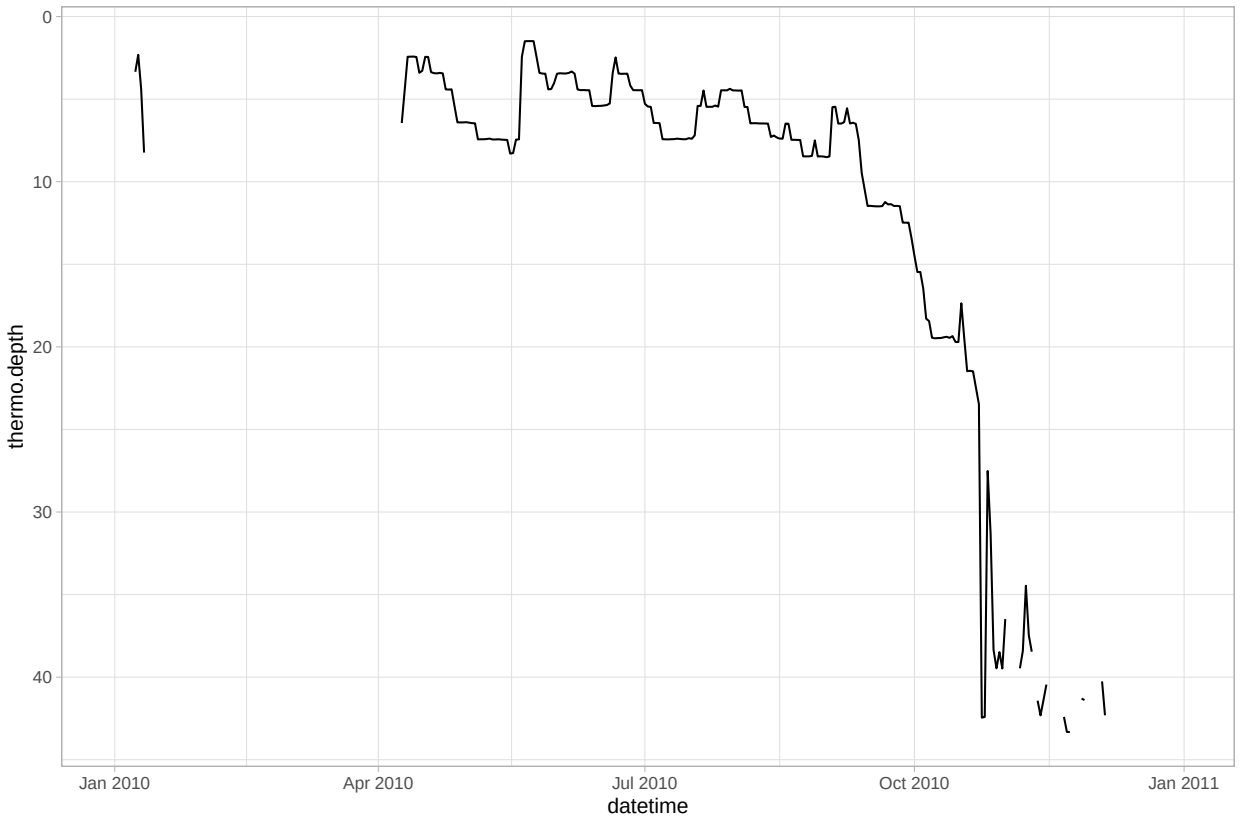
```
rLakeAnalyzer::wtr.heat.map(wtr)
```



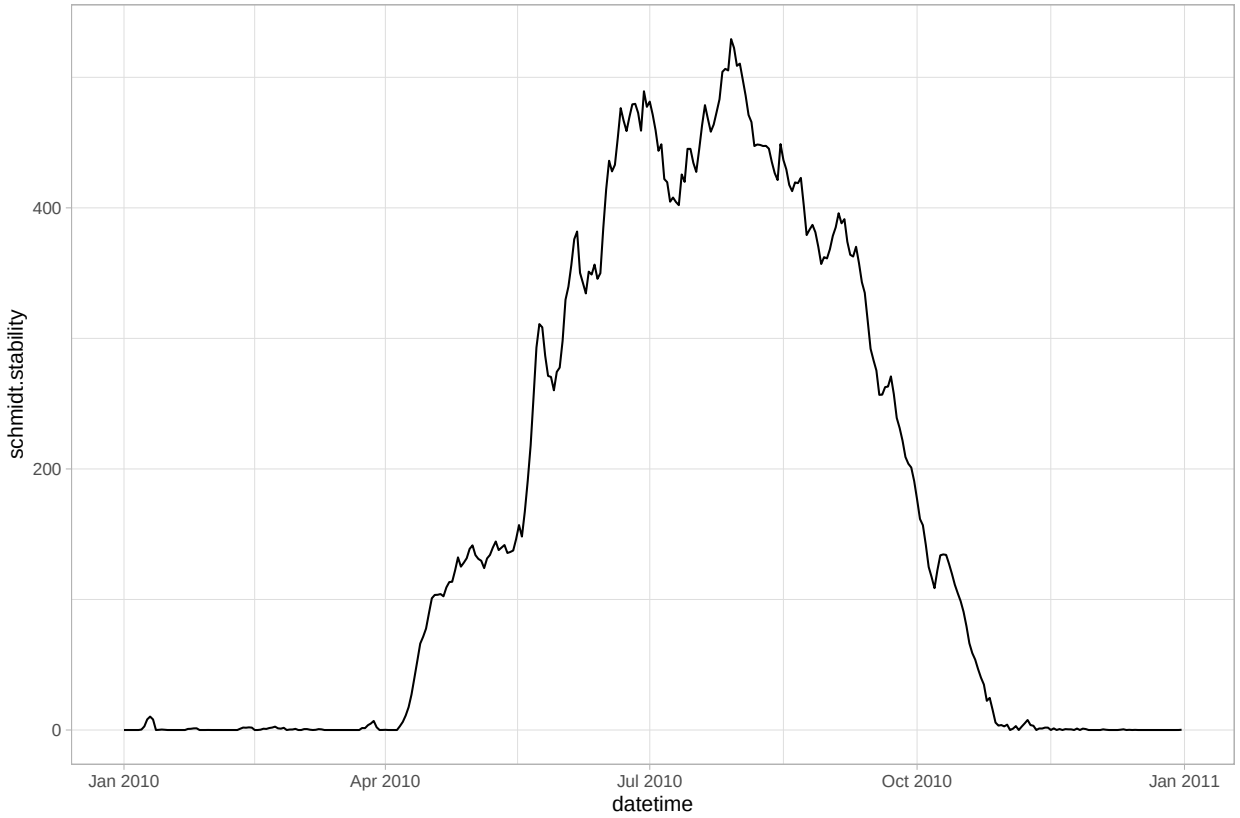

```
# Plot
ggplot(wtr)+
  geom_line(aes_string(x = "datetime", y = paste0("wtr_", plot_depth)))+
  theme_light()
```




```
# Calculate & plot thermocline depth
td <- ts.thermo.depth(wtr)
ggplot() +
  geom_line(data = td, aes(datetime, thermo.depth)) +
  scale_y_reverse() +
  theme_light()
```




```
# Calculate & plot Schmidt stability
# Load in hypsograph data to calculate Schmidt Stability
bathy <- read.csv(yaml$location$hypsograph)
colnames(bathy) <- c("depths", "areas")

sch_stab <- ts.schmidt.stability(wtr, bathy)

ggplot() +
  geom_line(data = sch_stab, aes(datetime, schmidt.stability)) +
  theme_light()
```



Both the netcdf output and the text output are structured in a standardised format, which makes further analysis or plotting easy. Only water temperature and ice thickness are currently generated in the "output" folder. Some models have more output variables (e.g. energy dissipation rate in GOTM). This output is still available in the model-specific folders, and can be accessed if you'd like.

6. Calibration

In the previous section we ran the models using default or pre-set (in the "model_parameters" section) parameters, and with observed meteorological forcing. However, often you will want to apply some form of calibration to your model.

The LER config file also has a "calibration" setting, which we will use here. Let's use the settings that are present in this file.

Essentially, the calibration section has two parts: "met" and "model-specific". In the "met" section, you can scale wind speed, shortwave radiation and longwave radiation, which will be calibrated separately for each model. In the other sections, you can provide the name of each model and calibrate model-specific parameters.

Have a look at the calibration section to see which parameters we're calibrating and between what ranges.

We call the calibration almost the same way as the normal ensemble run, but with the `cali_ensemble()` function. We add the "parallel = TRUE" statement, which will calibrate each model on a separate core, speeding up the process. We're using the "MCMC" (Markov Chain Monte Carlo) method here. To save some time, we will only do 10 iterations for each model (this is way too few for a proper calibration, in which case several thousands of iterations are more appropriate). The speed of running each model differs quite a lot, with FLake being the fastest and MyLake being the slowest. You can follow progress by looking at the csv files in the "cali" folder that are being created during the calibration.

PS: you need to run `export_config()` before starting the calibration. You have done that earlier in this script.

Depending on your PC, this function could take a few minutes to run. If you want to speed it up, you can remove MyLake from the model argument.

```
  cali_result <- cali_ensemble("LakeEnsemblR.yaml",
                               model = c("FLake", "GLM", "GOTM", "Simstrat", "MyLake"),
                               num = 10,
                               cmethod = "MCMC",
                               parallel = TRUE)
```

You can access the best parameters for each model in cali_result, e.g.for GLM

```
  cali_result[["GLM"]][["bestpar"]]
```

```
##          wind_speed                 swr mixing/coef_mix_hyp
##           1.1099176           0.8091587           0.9565655
```

For cmethod = "LHC", the different parameter sets and goodness-of-fit metrics will instead be put in tables in the "cali" folder. The reason for this is that sometimes one parameter set may have the lowest RMSE, and another the highest Pearson's r.

A third possible calibration method is cmethod = "modFit", this method uses the `modFit()` function of the FME package which provides different "classical" algorithms for constrained fitting of a model to data. For details see the `FME::modFit()` help page.

There is not yet a way of automatically enter the optimal fit of a calibration in the LER config file. Find the optimal values for each model in cali_result and enter them in the config file (in the "scaling_factors" and "model_parameters" sections. Then run export_config and run_ensemble again, to get a calibrated model run in "output/ensemble_output.nc".

First manually enter calibration values into "LakeEnsemblR.yaml", then run:

```
export_config("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                             "Simstrat", "MyLake"))
run_ensemble("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                            "Simstrat", "MyLake"))
```

7. Adding ensemble members to the NetCDF

Running multiple models is one way of creating an ensemble. Another option is to run the same model multiple times, but with different forcings, initial conditions, and/or parameter values. And a combination of these is possible as well, of course.

LakeEnsemblR allows multiple runs of the same models to be stored in the output netCDF file, by the "add" argument in the `run_ensemble()` function. Let's try it out by increasing the light extinction coefficient. We can speed up export config by switching off everything else except `extinction` which will only export the updated light extinction setting.

```
  # Change the light attenuation coefficient
yaml <- read_yaml("LakeEnsemblR.yaml")
yaml$input$light$Kw <- 2.5
```

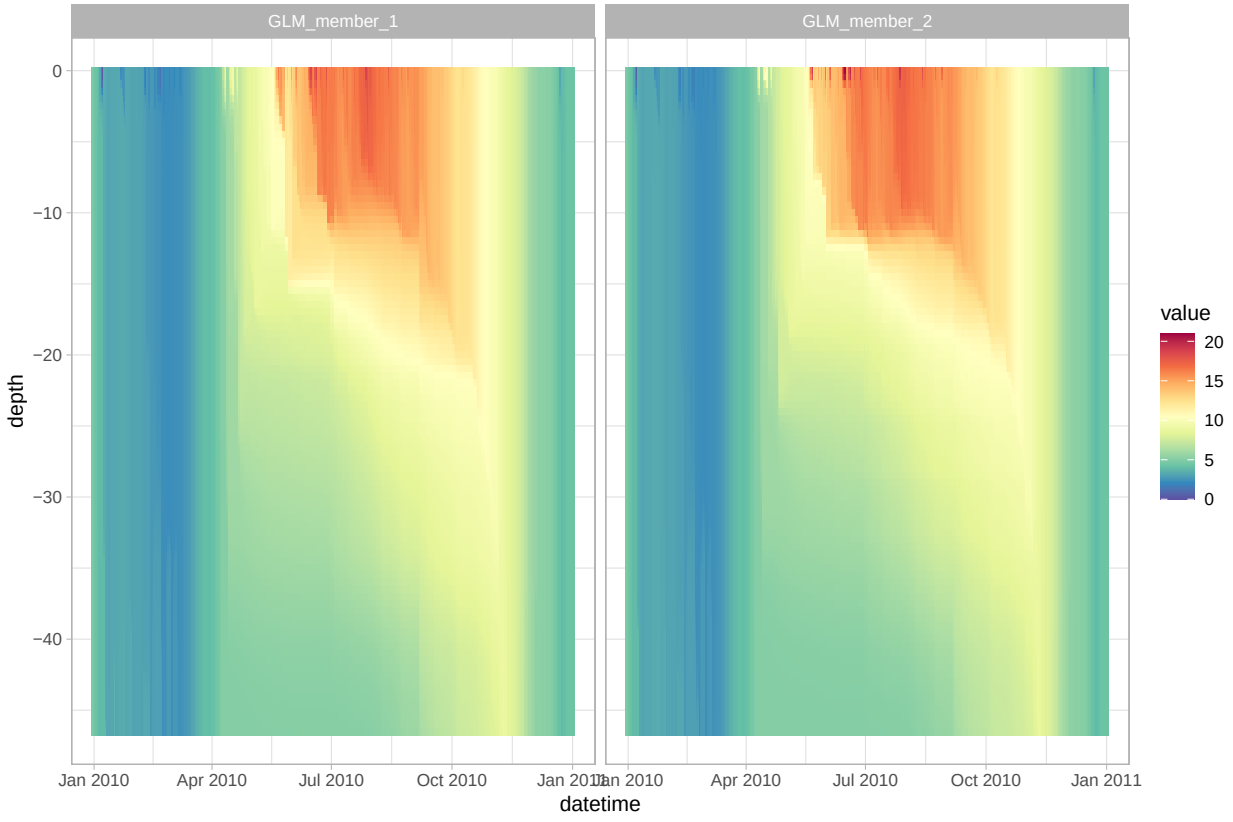
```
write_yaml(yaml, "LakeEnsemblR.yaml")

# Now run export_config and run_ensemble again, but add "add = TRUE"
export_config("LakeEnsemblR.yaml", model = c("FLake", "GLM", "GOTM",
                                             "Simstrat", "MyLake"),
              dirs = FALSE, time = FALSE, location = FALSE, output_settings = FALSE, meteo = FALSE,
              init_cond = FALSE, extinction = TRUE, inflow = FALSE, model_parameters = FALSE)

run_ensemble("LakeEnsemblR.yaml",
             model = c("FLake", "GLM", "GOTM", "Simstrat", "MyLake"),
             parallel = TRUE,
             add = TRUE)
```

In PyNcView you can now also change the "member" counter, to view multiple ensemble members of the same model

You can plot multiple members of the same model by using the "dim" argument. Change dim_index to plot different models (1 = FLake, 2 = GLM, etc.)

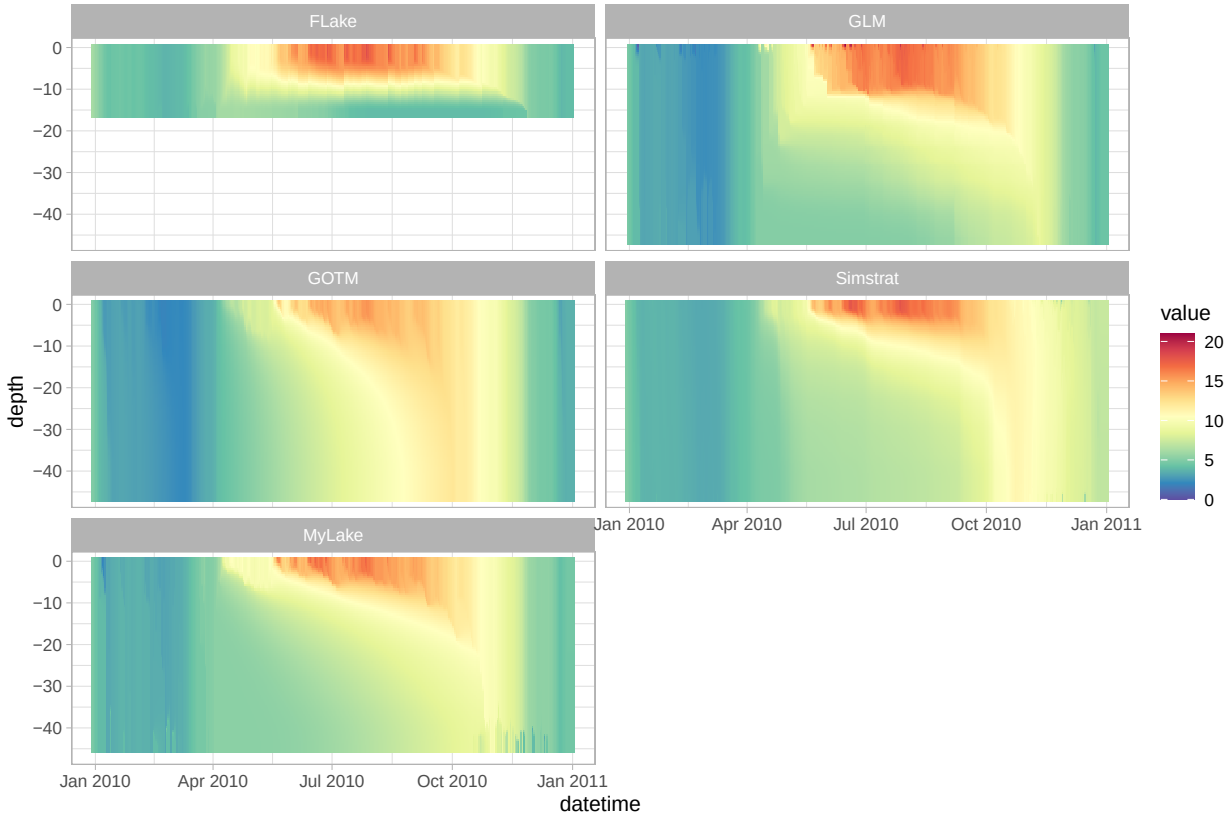
```
plot_heatmap("output/ensemble_output.nc", dim = "member", dim_index = 2) +
  scale_colour_gradientn(limits = c(0, 21), colours = rev(RColorBrewer::brewer.pal(11, "Spectral"))) +
  theme_light()
```

```
## Extracted temp from output/ensemble_output.nc

## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



```
plot_heatmap("output/ensemble_output.nc", dim = "model", dim_index = 2) +
  scale_colour_gradientn(limits = c(0, 21), colours = rev(RColorBrewer::brewer.pal(11, "Spectral"))) +
  theme_light()
```

```
## Extracted temp from output/ensemble_output.nc
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



This also works with the `plot_ensemble()` function or the `calc_fit()` function.

8. Post-processing with LakeEnsemblR

There are a variety of functions in the package that allow you to quickly analyse the standardised output of LER. There are functions for calculating model performance metrics and key indices such as stratification timing, ice on/off dates and max/min surface and bottom temperatures.

```
  out_res <- analyse_ncdf(ncdf = "output/ensemble_output.nc",
                          model = c("FLake", "GLM", "GOTM","Simstrat", "MyLake"))
```

This will return a list with the following entries:

```
  names(out_res)
```

```
## [1] "out_df"    "stats"     "strat"     "obs_temp"
```

```
  print(out_res[["strat"]])
```

```
##      model year    TsMax TsMaxDay     TsMin TsMinDay    TbMax TbMaxDay    TbMin
## 1      obs 2010 17.67833      179 3.4849999       13 12.76064      292 3.463672
## 2    FLake 2010 17.17080      210 3.7078300       57  6.47294      330 3.707830
## 3      GLM 2010 18.78964      210 0.2590204        9  8.93522      315 2.343762
## 4     GOTM 2010 15.54534      210 1.9107183       53 12.10592      276 2.147178
## 5 Simstrat 2010 18.93700      172 3.4189999       54 11.79100      301 3.186400
## 6   MyLake 2010 17.31264      209 1.9327056        8 10.06199      305 3.169321
##   TbMinDay MaxStratDur MeanStratDur TotStratDur StratStart StratEnd StratFirst
## 1       13         184       184.00         184        105      289        105
## 2       57         212       212.00         212        106      318        106
## 3       57         194        50.00         200        117      311        103
## 4       70         133        34.75         139        136      269        116
## 5       15         192       192.00         192        106      298        106
## 6       56         205        53.75         215        100      305        100
##   StratLast MeanIceDur MaxIceDur TotIceDur IceOn IceOff FirstFreeze LastThaw
## 1       289         NA        NA        NA    NA     NA          NA       NA
## 2       318         NA        NA        NA    NA     NA          NA       NA
## 3       311         NA        NA        NA    NA     NA          NA       NA
## 4       269         NA        NA        NA    NA     NA          NA       NA
## 5       298         NA        NA        NA    NA     NA          NA       NA
## 6       327         NA        NA        NA    NA     NA          NA       NA
##   HiceMax HiceMaxDay
## 1       0         NA
## 2       0         NA
## 3       0         NA
## 4       0         NA
## 5       0         NA
## 6       0         NA
```

Calculate model performance statistics, e.g. RMSE (Root Mean Square Error) or Pearson's r

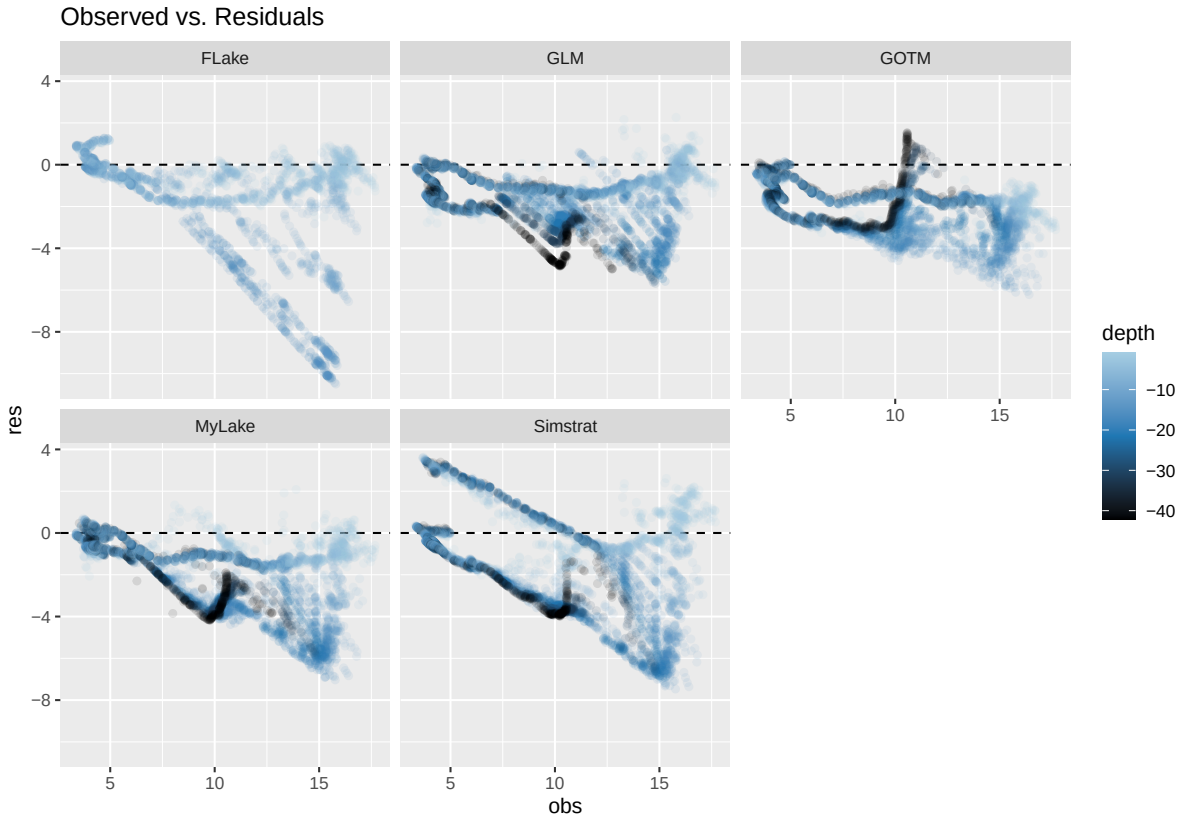
```
calc_fit(ncdf = "output/ensemble_output.nc", model = c("FLake", "GLM", "GOTM",
                                                       "Simstrat", "MyLake"))
```

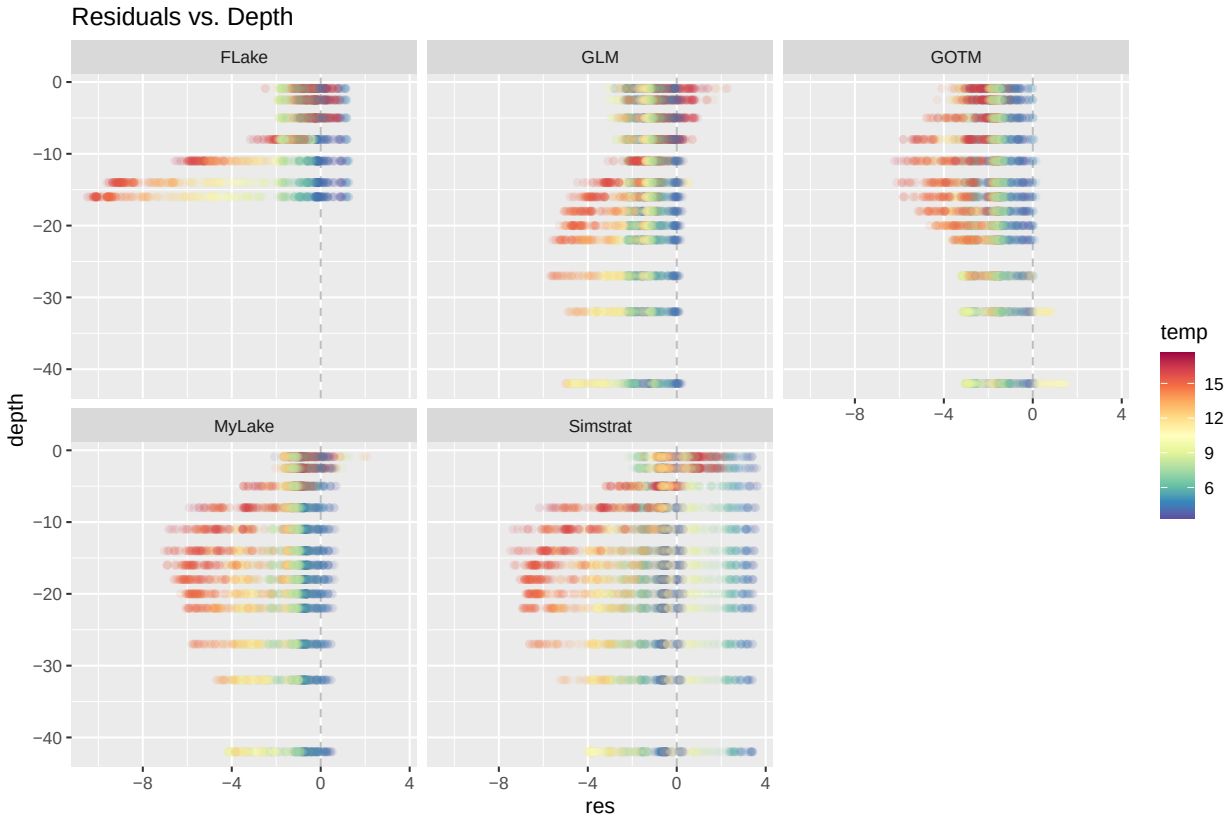
```
## $FLake
##       rmse       nse         r      bias      mae      nmae
## 1 3.303119 0.4772237 0.8070985 -1.871917 2.016814 0.1765731
##
## $GLM
##      rmse      nse         r      bias      mae      nmae
## 1 2.13118 0.737847 0.9529726 -1.715194 1.753024 0.2043534
##
## $GOTM
##       rmse       nse        r      bias      mae      nmae
## 1 2.382865 0.6724347 0.962442 -2.089218 2.114291 0.2466362
##
## $Simstrat
##      rmse      nse         r      bias      mae      nmae
## 1 2.80509 0.546066 0.8254992 -1.521165 2.181163 0.2343622
##
```

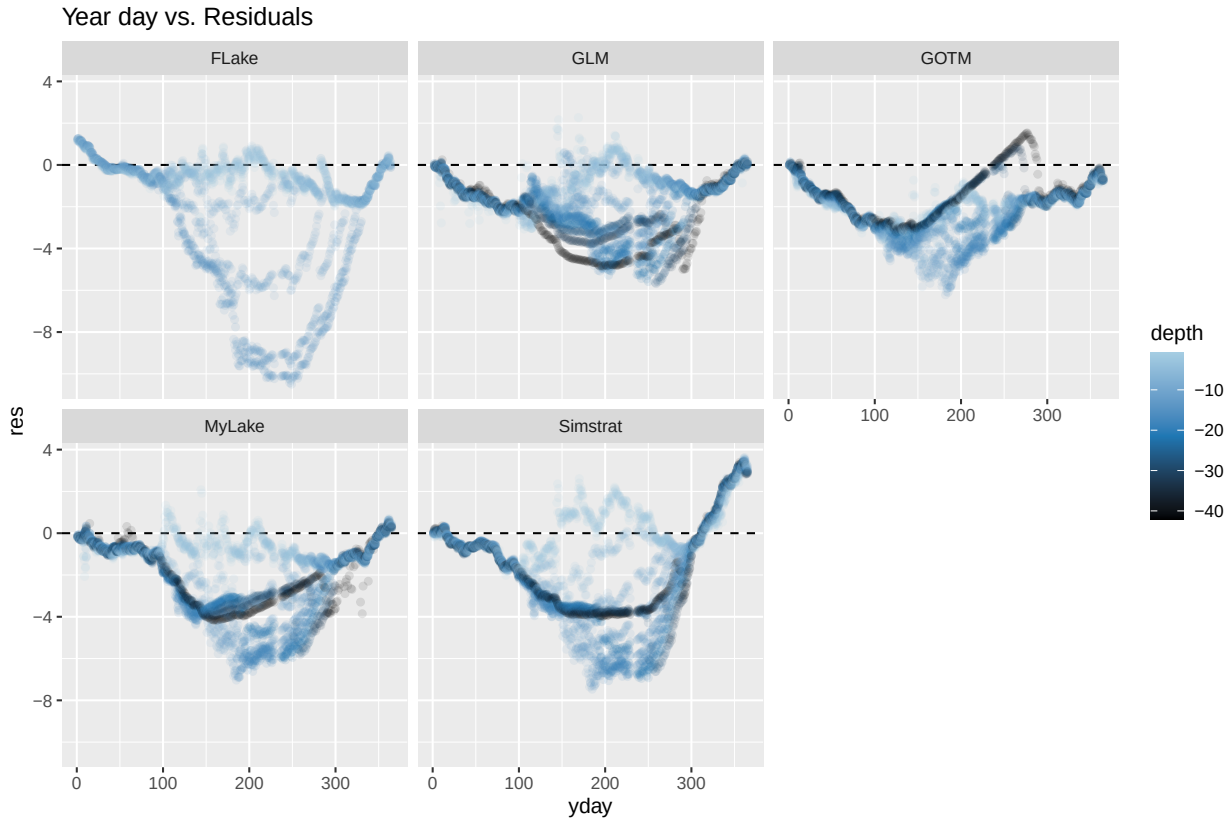
```
## $MyLake
##       rmse       nse         r      bias      mae      nmae
## 1 2.551914 0.6243087 0.9206151 -1.928489 1.962926 0.1950404
##
## $ensemble_mean
##       rmse       nse         r    bias      mae      nmae
## 1 2.388701 0.6708281 0.9344541 -1.8447 1.913075 0.2019023
```

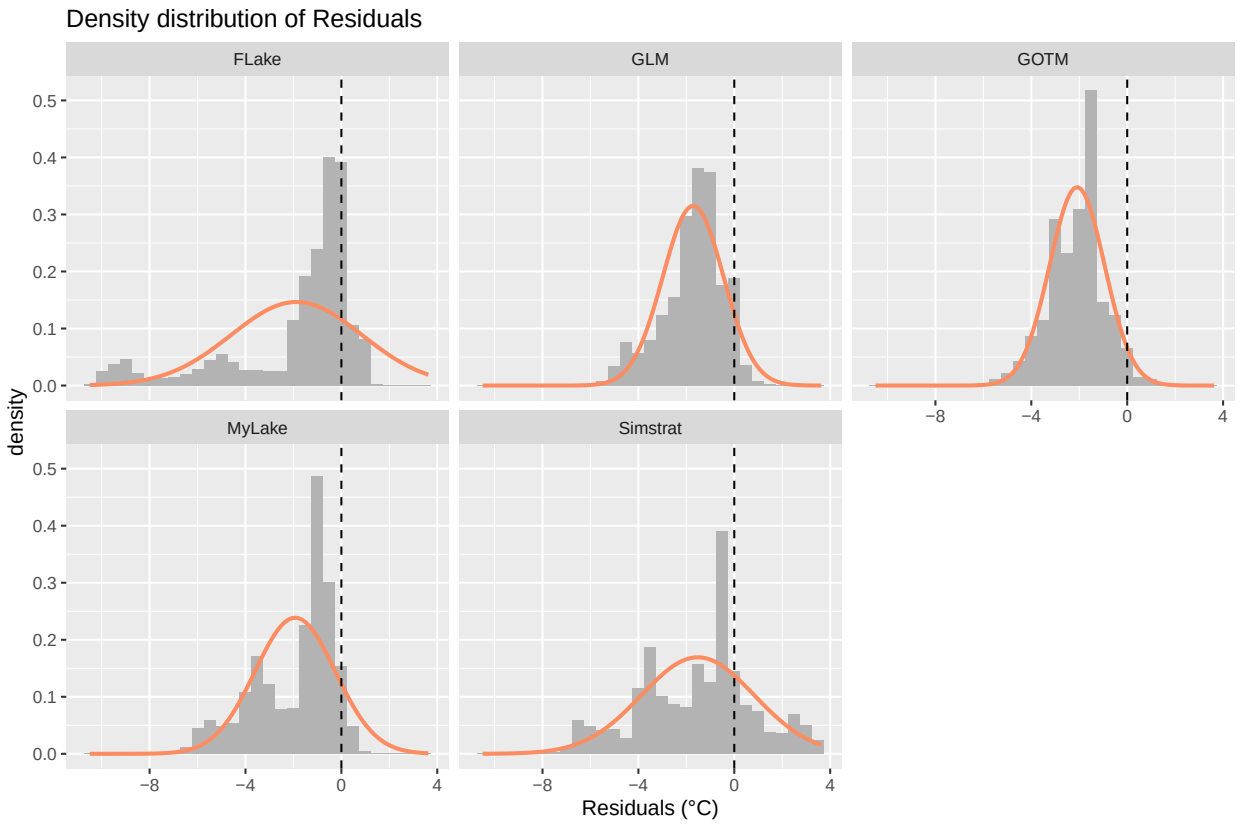
The `plot_resid()` function shows multiple plots that are helpful to discover where (time and space) the models perform well or poorly.

```
plot_resid(ncdf = "output/ensemble_output.nc", var = "temp")
```




Residuals vs. Depth
FLake
GLM
GOTM
MyLake
Simstrat
depth
res
temp
15
12
9
6
0
−10
−20
−30
−40
−8
−4
0
4


Year day vs. Residuals
FLake
GLM
GOTM
MyLake
Simstrat
res
yday
depth
−10
−20
−30
−40
4
0
−4
−8
0
100
200
300



Loading data into R

For further analysis the data can be loaded into R. The default format is a list format, with each model being its own object in a list. This format is the same as is used in `rLakeAnalyzer` so allows for application of such functions to the data.

Calculating Schmidt Stability

So we can calculate and plot the Schmidt stability for every model:

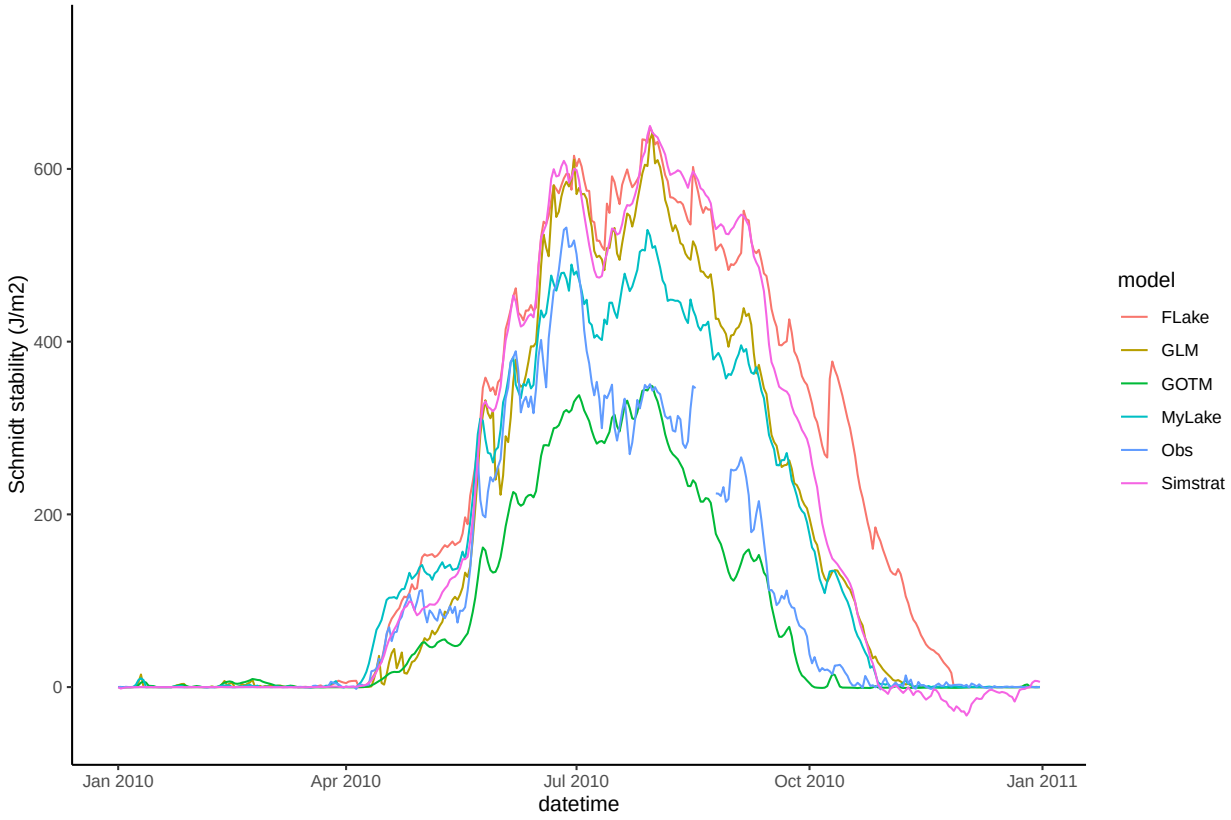
```
library(rLakeAnalyzer)
out <- load_var(ncdf = "output/ensemble_output.nc", var = "temp")
bathy <- read.csv('LakeEnsemblR_bathymetry_standard.csv')
colnames(bathy) <- c("depths", "areas")
ts.sch <- lapply(out, function(x) {
  ts.schmidt.stability(x, bathy = bathy, na.rm = TRUE)
})
```

Convert from a list object to a dataframe for plotting in ggplot

```
library(reshape)
df <- melt(ts.sch, id.vars = 1)
colnames(df)[4] <- "model"
```

Plot with ggplot

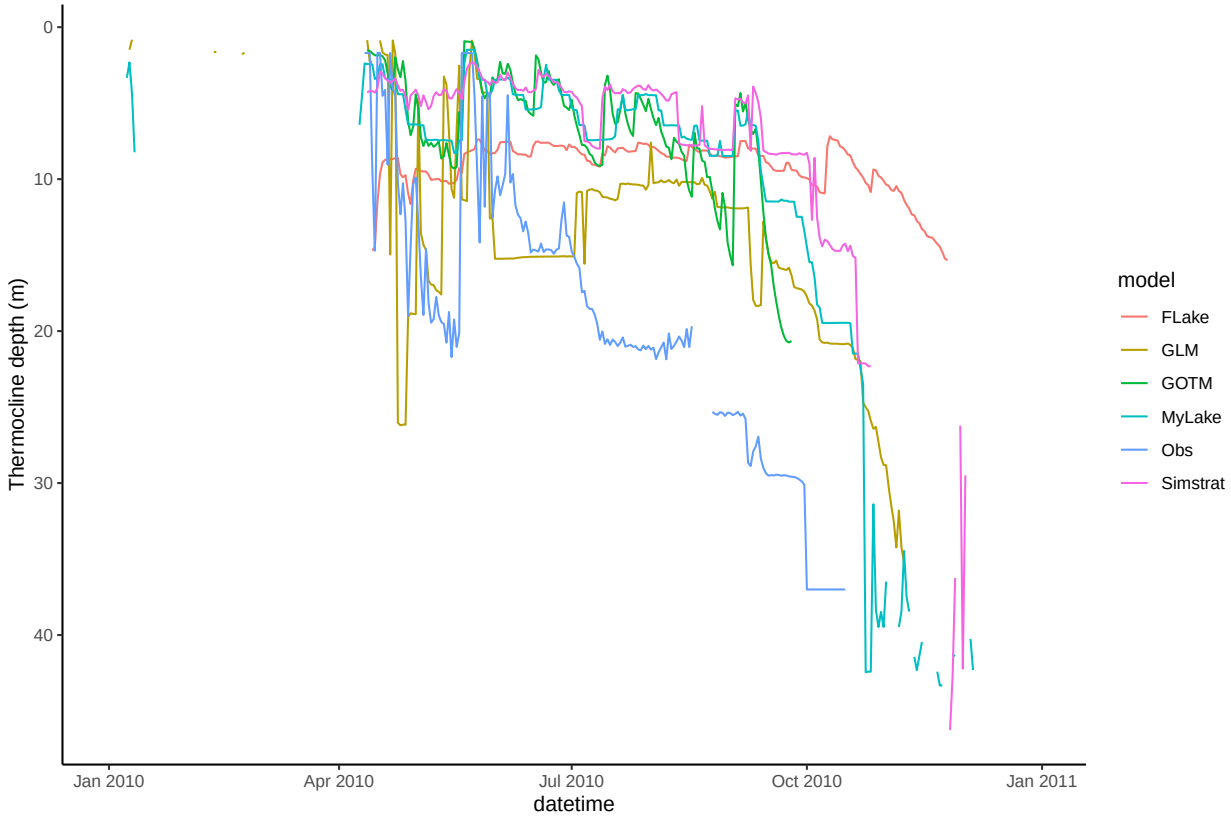
```
ggplot(df, aes(datetime, value, colour = model)) +
  geom_line() +
  labs(y = "Schmidt stability (J/m2)") +
  theme_classic() + ylim(-50, 750)
```



Calculating thermocline depth

In the same manner, we can also calculate the thermocline depth:

```
ts.td <- lapply(out, function(x) {
  ts.thermo.depth(x, Smin = 0.1, na.rm = TRUE)
})

df <- melt(ts.td, id.vars = 1)
colnames(df)[4] <- "model"

ggplot(df, aes(datetime, value, colour = model)) +
  geom_line() +
  labs(y = "Thermocline depth (m)") +
  scale_y_continuous(trans = "reverse") +
  theme_classic()
```



This workflow can be used with each of the functions within `rLakeAnalyzer`.

9. Apply LakeEnsemblR to your own data, or play around with the settings

In the next part of the workshop, you are encouraged to either apply LER to your own lake data or to one of the examples given on https://github.com/aemon-j/LER_examples. You can also try playing around with the settings for the Lough Feeagh test case (e.g. add varying light extinction, or try different calibration methods). The workshop leaders will be around to help with issues if needed. The LakeEnsemblR package provides template files for all files necessary to simulate your lake, they can be accessed by using the `get_template()` function, which will copy the template to your current working directory e.g.

```
get_template("Initial temperature profile")
```

The names of the available template files can be obtained by executing the `get_template()` function with no argument:

```
get_template()
```

```
## You need to specify what you want a template for.
## Options are:
## all
## LakeEnsemblR_config
## FLake_config
## GLM_config
```

```
## GOTM_config
## Simstrat_config
## MyLake_config
## Initial temperature profile
## Light extinction
## Inflow
## Hypsograph
## Ice height
## Meteo
## Temperature observations
```

Let us know if you have any questions!

At the end of the workshop some time is reserved to brainstorm potential uses for this package.