**WATExR tool suggested workflow**

*LJB 18/06/2019*

**Aims:**

1. **Automated workflow for us to be able to run the hindcast experiment to assess:**
   1. model skill: run model driven by best possible observed met data
   2. model skill, driven using reanalysis data (ERA-interim). Model output from this will be assumed to be ‘true’ and used to assess skill of seasonal forecast (to provide longer time series of observed water quality than observations alone)
   3. combined skill of model and seasonal forecast: run model with ERA-interim for warmup and seasonal forecast

For each of these, involves running model for each season in the hindcast period.

Date of the hindcast period still to be decided on. Take as starting point 1981-2010

1. **Operational workflow to predict water quality in the coming season**

Our first priority is the hindcast experiment to assess the skill of the models and seasonal forecasts, but we should try to make any code we produce for this as clean as possible so it can be incorporated within the operational workflow.

I would suggest we start by making a set of Jupyter notebooks which independently do the required steps, so we can easily visualise output. Later, we can convert the contents of these notebooks into python functions, which can be called from a master script (either a notebook which we could ‘GUI-ise’ using e.g. ipyWidgets, or a script in QGIS)

Development environment: ICRA WATExR GitHub repository, Norway\_Morsa folder: <https://github.com/icra/WATExR/tree/master/Norway_Morsa>

Eventually: use the DSToolkit (on NIVA’s JupyterHub)? But for now installing our own packages not well supported (e.g. Climate4R)

Suggested starting point, ~8 notebooks (4 for the hindcast experiment, 4 very similar ones for the operational forecasting):

**NB1a\_hindcast: Run the Climate4R scripts to download met data for the hindcast experiment**

The Climate4R package is on R. The WATExR-specific scripts are in the Github repository: <https://github.com/icra/WATExR/tree/master/R>. See the readme there for a description of the different files. @sixtoherrera and @miturbide on Mattermost are very helpful and quick at troubleshooting, ask them if there are any issues. Flick back up through the discussions too for problems people have had and potential solutions.

We want to:

* Download historic observed meteorological data:
  + Hindcast experiment 1:
    - met.no’s 1km x 1km gridded data (done already, saved here, for example GitHub\WATExR\Norway\_Morsa\Data\Meteorological\Obs\_metno\Obs\_Metno\_1km\_gridded\_Morsa.csv
    - For wind, weather station data from Rygge airport:

GitHub\WATExR\Norway\_Morsa\Data\Meteorological\Obs\_metno\Obs\_Rygge\_MeanWindSpeed.csv

* + Hindcast experiment 2: ERA-interim
  + Check out how big the bias is in ERA-Interim data compared to met.no data
  + Bias correct ERA-interim data using observed data (a function in the Climate4R package does this. Need to double-check that any spatial resampling that also goes on in the bias correct function is ok. Perhaps get Sixto to adapt it for us so it works with a single time series rather than a gridded dataset)
  + If have time: also download ERA5 data (from Copernicus, not on the Santander server yet). How is the bias? Ok to use without bias correction?
  + Rearrange into nice Python structure for accessing later
* Download seasonal forecast met data for the hindcast period:
  + Seasonal forecast: For now use System4, though we’ll probably use System 5 too. Has 15 or 25 members (can’t remember)
  + Variables of interest: precipitation, min and max temp (to calculate PET), daily mean temp, wind speed, (anything else that MyLake needs?). Climate4R scripts calculate PET if you run those bit of the code.
  + For each season in the hindcast period: Select lead time 0. Download for four seasons, each four months long, for period 1981-2010:
    - Spring: download Feb, March, Apr, May
    - Summer: download May, June, July, Aug
    - Autumn: download August, Sep, Oct, Nov
    - Winter: download Nov, Dec, Jan, Feb
  + Rearrange data into some kind of nice python structure that we can work with. E.g. something like a daily\_met\_dict with key [season, year, variable], which returns a pandadas dataframe with datetime index and one column for each ensemble member.
* Bias correct seasonal forecast data using met.no data and calculate PET if not already done
* QC: Time series plots and summary stats to check all looks ok.
* Have a go at doing a tercile plot comparing observed data and seasonal forecast data (Climate4R package has a function for this).
* Save daily time series for access by other notebooks

**NB2a\_hindcast: Source any chem/ecol data, post-process the data and generate features:**

All these features are worked out in notebook xxx, so code can be taken from here

* Water chemistry and ecology data:
  + Read in observed chemistry and ecology data, stored here: GitHub\WATExR\Norway\_Morsa\Data\Observed\_Chem\_Ecol
  + Calculate mean water chemistry for the previous summer (concentrations of TP, chl-a, cyanobacterial biovolume, lake colour)
* Post-processing of met data. For now just calculate a couple of things to get us going..
  + Data to warm up BBN: Observed data only (met.no data or ERA-Interim data from NB1). For each season/year:
    - Sum of winter precipitation for the winter before the current month
    - Mean monthly air temp
    - Monthly precipitation sum
  + Data for forecasting: Want all the below to work for any of the met.no data, ERA-Interim (and ERA5 in the future), seasonal forecast data. Could have a user-option describing the dataset you want to use, and if it’s the seasonal forecast data then the data structures for storage of results will have to take ensemble member into account too.

For all seasons/years/ensemble members, convert to monthly frequency and calculate lags:

* + - Monthly precipitation sum
    - Monthly mean temperature
    - Lag 1 month for both pptn sum and mean temp (so the value from the previous month is stored against the current month)

**NB3a\_hindcast: Set up and run the Bayesian Belief Network (BBN) and compare to obs**

Leah to do

* Define the BBN structure
* Read in the historic features used to create the BBN?
* For each kind of met data (met.no, ERA-Interim, S4/S5):

For each season, year (and for S4/5 each member in the seasonal forecast ensemble):

* + Using the features generated in NB2, update the BBN contingency tables:

N.B. Think carefully about which datasets are used for which months…

* + Run BBN and save the output for all vars of interest (cyanobacterial biovolume, TP concentration, chl-a concentration, colour)
  + Drop any months that aren’t considered in final seasonal analysis (e.g. for spring have Feb – May, but drop Feb from forecast results).

For each season and year:

* + Calculate an ensemble median for each var of interest

**NB4a\_hindcast: Visualise results – hindcast experiment**

* Read in historic observed water quality/ecology data from NB2
* Read in forecast data from NB3. As this data isn’t available yet, create dummy data. Something like a dictionary with key (season, year, result variable), returns dataframe with columns for each ensemble member and 3 rows (‘High’, ‘Good’, ‘Moderate’). The values in the cells are a probability of being in that row (sum of probabilities across rows adds up to 1).
* (Also want tercile plots & stats comparing seasonal forecast and met.no observed precipitation, temperature and wind speed)
* For each season:
  + Generate tercile plots comparing observed and simulated data (using the Climate4R visualise script)
  + Calculate some kind of goodness-of-fit statistic – see Climate4R visualise again.

--------------------------------------------------------------------------------------------------------------------------------

Basially a repeat of the previous 4 notebooks, but this time the aim is to produce operational seasonal water quality forecasts:

**NB1b\_forecast: Run the Climate4R scripts to download met data for the operational forecast**

4 times a year:

* Start date is the next month. Set the target season (Spring, summer, autumn or winter)
* Download historic met data to update the model:
  + Will be using ERA5 data, but discussions still ongoing as to the source of this. Hopefully Copernicus
  + Download daily precipitation and average temperature (maybe wind speed?) for the previous year
* Download seasonal forecast met data for the coming season

**NB2b\_forecast: Source any chem/ecol data, post-process the data and generate features:**

Just a couple of differences to NB2a (mostly simplifications):

* Water chemistry and ecology data:
  + Read in observed chemistry and ecology data from the previous summer. Probably best done from NIVABase (see James’ nivapy python package for convenience functions to do this) if we’re only installing locally; or perhaps from vannmiljø if that’s easy (I don’t think it is). The safest thing might be to extract the data we want from NIVABase locally once a year (e.g. in February) and store it in a csv next to the scripts:
    - From Station ID xxx
    - Variables: xxx
    - Time period: previous ~18 months
  + Calculate mean water chemistry for the previous summer (concentrations of TP, chl-a, cyanobacterial biovolume, lake colour)
* Post-processing of met data. For now just do a couple of things to get us going, e.g.
  + Observed met data:
    - Sum of winter precipitation
    - Mean monthly air temp
    - Monthly precipitation sum
  + Seasonal forecast data: for all ensemble members, convert to monthly frequency and calculate lags:
    - Monthly precipitation sum
    - Monthly mean temperature
    - Lag 1 month for both pptn sum and mean temp (so the value from the previous month is stored against the current month)

All these things are worked out in notebook xxx, so code can be taken from here

**NB3b\_forecast: Set up and run the Bayesian Belief Network (BBN)**

Leah to do

* Define the BBN structure
* Read in the historic features used to create the BBN, updating with any more recent data
* For each month in the coming season and ensemble member in the seasonal forecast:
  + Run BBN using features generated in NB2 and save the output for all vars of interest (cyanobacterial biovolume, TP concentration, chl-a concentration, colour)
  + Drop any months that aren’t considered in final seasonal analysis (e.g. for spring have Feb – May, but drop Feb from forecast results).
  + Calculate an ensemble median for each var of interest

**NB4b\_forecast: Visualise results – operational forecast, including pdf generation**

As for NB4a, but also:

* Forecast for the coming season is compared to observations for previous seasons (probability of being below average, average or above average?)
* Auto-generate a pdf summarising results (map of catchment, tercile plot, summary stat, skill score result from hindcast experiment, interpretative text).
* This should use the elements in here as inspiration (but dropping all the time series plots): K:\Prosjekter\Ferskvann\O-17323 WATExR\05\_IntegratedTool\Norway\_Plugin\_Design\_2018-11.pptx

----------------------------------------------------------------------------------------------------------------------------

Once the ‘b’ series of notebooks are done, we can think about moving a lot of the code into functions, and having a single place that we run to do everything from. This place could be:

* **Plan A**: A Python script, run from within a QGIS plugin. The plugin design would look a bit like the sketch here, but without the time series plots:

K:\Prosjekter\Ferskvann\O-17323 WATExR\05\_IntegratedTool\Norway\_Plugin\_Design\_2018-11.pptx

I don’t see any life for the Plugin after the project, where as the code itself could be useful. So I don’t want us to spend long on this. If however we can put something quick together in little time then that would be optimal.

* Plan B: Alternatively, we could do a Jupyter notebook and ‘GUI’ it using e.g. ipython widgets (basically just add a big red ‘Run’ button, plus maybe some other options)