**WATExR tool suggested workflow**

*LJB 01/07/2019, updated 07/08/19*

**Briefe overview:**

Aim to predict lake concentration of total phosphorus (TP), chlorophyll-a (chl-a), lake colour and biovolume of cyanobacteria. Predictions will be made using a Bayesian Belief Network (BBN), driven using observed water chemistry and ecology from the previous summer, and observed or forecasted meteorological variables. Predictions will be made for the 3 months after the current month. For Vansjø, the WFD class is decided on based on the condition of the lake during the months May - October. So most likely splitting of the year into seasons will be something like:

|  |  |  |  |
| --- | --- | --- | --- |
| **Season** | **Lake forecast produced in month** | **Forecast produced for months** | **Met data to parameterise & drive the BN model** |
| Early summer | April | May, June, July | Previous winter (Oct-March): reanalysis data  May - July: seasonal forecast |
| Late summer | July | Aug, Sep, Oct | May-June: reanalysis data  July: seasonal data  Aug - Oct: seasonal data |
| Winter | Oct | Nov, Dec, Jan | Nov - Jan: seasonal forecast |
| Spring | Jan | Feb, March, April | Feb - Apr: seasonal forecast |

Water quality/ecology predictions will only be produced for Early and Late summer. Then, we are interested in variables (to start with):

* Total P concentration
* Lake colour
* Chl-a concentration
* Cyanobacteria biovolume
* Overall WFD status (calculated as a function of the other variabbles)
* (Other possibles: PTI)

Predictions will only be produced for Vanemfjorden, the western basin in Lake Vansjø (often referred to as Van2).

**Coding aims:**

1. **Automated workflow for us to be able to run several hindcast experiments:**

* **Task 1**: assess lake model forecasting skill: run model driven by best possible observed met data (met.no data for temp and precipitation, Rygge station data for wind speed)
* **Task 2**: generate pseudo observations for the period 1981-2010: drive model using bias-corrected reanalysis data (ERA-interim). Model output from this will be assumed to be ‘true’ (to provide longer time series of observed water quality than observations alone)
* **Task 3**: Assess seasonal met forecasting skill in the context of lake forecasting: run model with ERA-interim for warmup and seasonal forecast S4 for target season, and compare output to pseudo-observations from Task 2.

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

Date of the hindcast period:

* Task 1: 1989-2018
* Task 2 & 3: 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):

**Hindcast experiment**

**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.

Leah got observations.R and seasonalForecast.R working for Vansjø last November in Magdeburg. Files are in the WATExR\_NIVA google drive. Changes compared to master scripts:

* Login details (not provided in the GitHub scripts): loginUDG("WATExR", "1234567890")
* Vansjø-specific stuff: changing the lat and long variables (domain extent and location of the lake), plus a few other bits and pieces
* Some more comments to explain what bits are doing

Scripts will probably have changed a bit now, and we want more variables than just in these example scripts, but they should be useful anyway.

Notebook aims:

* Download historic observed meteorological data:
  + For Task 1: Download local observed met data
    - met.no’s 1km x 1km gridded data for the Morsa catchment, update to include summer 2018 data (I guess summer 2019 won’t be possible yet?). (Data to mid 2018 is saved here: GitHub\WATExR\Norway\_Morsa\Data\Meteorological\Obs\_metno\Obs\_Metno\_1km\_gridded\_Morsa.csv)
    - Combine with wind data from Rygge airport:

(up to date to 2018; saved here: GitHub\WATExR\Norway\_Morsa\Data\Meteorological\Obs\_metno\Obs\_Rygge\_MeanWindSpeed.csv)

* Put into same format as other data so can be read in by the rest of the notebooks
  + For Task 2: Download global observed/reanalysis met data
    - EWEMBI and ERA-interim data. ERA-interim data is here: <http://meteo.unican.es/tds5/dodsC/interim/interim075_WATExR.ncml>

EWEMBI is also on the unican server.

* + - Bias-correct ERA-interim data using EWEMBI (using a function in the Climate4R package.  [See: https://github.com/SantanderMetGroup/downscaleR/wiki/bias-correction-of-seasonal-forecasts’](https://github.com/SantanderMetGroup/downscaleR/wiki/bias-correction-of-seasonal-forecasts))
  + Rearrange into nice Python structure for accessing later
  + Out of interest, compare bias in EWEMBI and ERA-interim data with met.no data
* Download seasonal forecast met data for the hindcast period and bias correct:
  + System4 data. Has 15 or 25 members (can’t remember), which are different initializations
  + Variables of interest: precipitation, min and max temp (to calculate PET), daily mean temp, wind speed, (anything else that MyLake/GOTM needs?). Climate4R scripts calculate PET, so run those bits of the code.
  + Download seasonal data. Unfortunately, we’re going to need different seasons for the two models that we’re going to run the hindcast experiment for (GOTM and the Bayesian Network). For both model, for each season in the hindcast period: Select lead time 0. Download for four seasons, for period 1981-2010:
    - GOTM:
      * Spring (March-May): download Feb, March, Apr, May
      * Summer (Jun-Aug): download May, June, July, Aug
      * Autumn (Sep-Nov): download August, Sep, Oct, Nov
      * Winter (Dec-Feb): download Nov, Dec, Jan, Feb
    - Bayesian network:
      * Early summer: May-Jul
      * Late summer: July-Oct (yes, one extra month here compared to the other seasons, see table at the top of the doc)
      * Winter: Nov-Jan
      * Spring: Feb-Apr
  + Bias correct seasonal forecast data using EWEMBI data and calculate PET if not already done (or bias-correct PET too?)
  + 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.
* QC: Time series plots and summary stats to check all looks ok.
* Tercile plot comparing observed and seasonal seasonal forecast data for temp, precipitation, wind speed (Climate4R package has a function for this).
* Save into a nice structure for access by other notebooks. E.g. a dictionary with one key per dataset (observed, reanalysis, seasonal)? **See NB2a point 1** for what’s coming next for the met data. Good to have this in mind when saving it.

**NB2a\_hindcast: Generate explanatory variables to predict water quality**

All these features are worked out in notebook <https://github.com/icra/WATExR/blob/master/Norway_Morsa/Model_Development/Notebooks/01_Make_data_matrix.ipynb>, and notebook 03 in the same folder, so code can be reused from there.

1. Post-processing of met data. This list is preliminary to get us going (exact features will be decided on by late September 2019, once the final BBN structure is decided on)
   1. To provide input met data for Tasks 1 and 2:

For both observed met data and bias-corrected ERA-Interim data:

For each season/year:

* + 1. Sum of winter precipitation for the winter before the current season
    2. Mean seasonal air temp
    3. Seasonal precipitation sum
    4. Something wind-related. e.g. count of number of days per season when daily mean wind speed was less than 3 m/s.
  1. Data to provide input to forecast late summer lake status with the BN (Task 3):

Create a realistic patched dataset, which represents what would be available to us when producing forecasts in July of lake status in Aug-Oct:

* + - Patch together bias-corrected ERA-interim data from May and June with seasonal forecast data from July, to create one continuous data series (May-July)
    - Calculate a(i) – a(iv) on this early summer met season dataset (which will be used to predict the lake in late summer).
  1. For the seasonal forecast data:
     + Drop July from the late summer season set of downloads
     + Then, for all seasons/years/ensemble members, convert to seasonal frequency and calculate a(i)-a(iv)
  2. Whatever Francois needs for GOTM, which will include daily temp, rainfall and PET for SimplyQ. The seasonal split will be different
  3. Similar to (d) SimplyQ requirements, but this time for SimplyP for comparing to the BBN results, so the start and end dates for the different seasons will be different to the GOTM experiment.

1. Water chemistry and ecology data:
   * Read in observed chemistry and ecology data, stored here (one file for cyanobacteria, everything else is in the other file: GitHub\WATExR\Norway\_Morsa\Data\Observed\_Chem\_Ecol

We’re interested in variables: **TP and chl-a concentration, cyanobacterial biovolume, lake colour**. Everything else can be dropped.

* + Join data, drop anything outside the hindcast period
  + Calculate mean seasonal water chemistry and ecology (to compare to simulations), so have a dataframe with an index something like (year, season)
  + Calculate means for the previous summer (explanatory variable)

1. Store all target and explanatory variables in a nice big dataframe

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

Leah to do

1. Define the BBN structure
2. Read in the historic features used to create the BBN?
3. 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):

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

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

* 1. Run BBN and save the output for all vars of interest (cyanobacterial biovolume, TP concentration, chl-a concentration, colour)
  2. Drop any warmup months

For each season and year:

* 1. Calculate an ensemble median for each var of interest

**NB4a\_hindcast: Calculate model performance and visualise results**

These steps are much less certain than the previous ones, needs more thinking through. Contributions welcome! Probably also a case of just getting started and solving problems as they arise

1. Observed data processing:
   1. Read in historic monthly observed water quality/ecology data from NB2
   2. Reclassify into ‘High’, ‘Good’, ‘Moderate’ categories output by the BBN. Search for ‘WFD class boundaries’ in <https://github.com/icra/WATExR/blob/master/Norway_Morsa/Model_Development/Notebooks/01_Make_data_matrix.ipynb> for some example code and boundaries for some of the variables as a starting point.

WFD\_class\_dict = {'TP\_lake': {'G-M':20., 'M-P':39.},

'chl-a\_lake': {'G-M':10.5, 'M-P':20.},

'CyanoBiovol': {'G-M':1., 'M-P':2.}

}

1. Read in forecasted water quality/ecology 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).
2. For each season:
   1. For each month in the season:
      1. Generate tercile plots comparing observed and simulated data (using the Climate4R visualise script)
      2. Calculate some kind of goodness-of-fit statistic – see Climate4R visualise again for ideas?

Key questions: does model performance reduce later in the season when driven by the seasonal forecast? Better/worse for different variables? Better for certain seasons?

* 1. Calculate monthly averages (have to think about how to average probabilities quite carefully), and then repeat (tercile plot and goodness-of-fit)

1. Also want tercile plots & stats comparing seasonal forecast and met.no observed precipitation, temperature and wind speed

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

**Forecasting**

Very similar to 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:

1. Start date is the next month. Set the target season (early summer, late summer, winter, spring)
2. Download historic met data to update the model:
   1. Will be using ERA5 data, but discussions still ongoing as to the source of this. Hopefully Copernicus
   2. Download daily precipitation and average temperature and wind speed for the previous year
3. Download seasonal forecast met data for the coming season. Not sure what system we’ll be using, maybe System5?
4. Bias correct seasonal forecast data (assuming historic bias applies)

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

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

1. Water chemistry and ecology data:
   1. Read in observed lake chemistry and ecology data from the previous summer. From NIVABase (see James’ nivapy python package for convenience functions to do this if working on the network), Aquamonitor or Vannmiljø. Don’t spend long on this for now though - it wouldn’t take long to once a year (e.g. in February) extract relevant data and store it in a csv next to the scripts:
      1. From Station ID? Don’t know what the Van1 code is in Aquamonitor. I got data from Vannmiljø, Vannlokalitet\_kode: 003-30776 (Vannsjø, vanemfjorden (VAN2)
      2. Variables: (again, these are Vannmiljø- codes) P-TOT (Total P), KLFA (chl-a), FARGE (colour). Not sure about cyanobacteria, I got this data directly from lots of different people for the historic period as there are lots of gaps in the NIVABase data. There’s CYANOM, but that’s just the maximum for the season. Monthly values might be good too.
      3. Time period: previous ~18 months
   2. Calculate mean water chemistry for the previous summer (concentrations of TP, chl-a, cyanobacterial biovolume, lake colour)
2. Post-processing of met data. For now just do a couple of things to get us going, e.g.
   1. Observed met data and seasonal forecast data:
      1. Sum of winter precipitation
      2. Mean monthly air temp
      3. Monthly precipitation sum
      4. Number of days per month when wind was below 3 m/s
   2. Calculate lagged values for all the 3 monthly series (so the value from the previous month is stored against the current month)

As in NB2a, lots of code can be re-used from Leah’s Make\_Data\_Matrix notebook

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

Leah to do

1. Define the BBN structure
2. Read in the historic features used to create the BBN, updating with any more recent data
3. For each month in the coming season and ensemble member in the seasonal forecast:
   1. Run BBN using features generated in NB2 and save the output for all vars of interest (cyanobacterial biovolume, TP concentration, chl-a concentration, colour)
   2. Drop any warm-up months
   3. Calculate an ensemble median for each var of interest

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

As for NB4a, but also:

1. Forecast for the coming season is compared to observations for previous seasons (probability of being below average, average or above average?)
2. **Auto-generate a pdf summarising results** (map of catchment, tercile plot, summary stats, skill score result from hindcast experiment, interpretative text).
3. 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

It also needs to be designed together with out stakeholder, and is a really important ouput.

1. Perhaps also auto-generate a simplified summary, suitable for emailing out to e.g. farmers.

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

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:

1. **Plan A**: A Python script, run from within a QGIS plugin. Eventually, the plugin design could look 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, whereas 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.

1. 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).