GLEON 21 GSA Ecological Forecasting Workshop Hands On Activity

2019-11-04

```r
library(dplyr)
## Warning: package 'dplyr' was built under R version 3.6.1
```

In this exercise we will apply an Ensemble Kalman Filter (EnKF) algorithm to a lake dataset to forecast dissolved organic carbon (DOC) concentrations. The dataset includes DOC loads to the lake, lake volume, water discharge, and in-lake DOC mass.

First, let's source some functions that will help us with the EnKF analysis.

```r
# source important functions
source('Code/EnKF_functions.R')
source('Code/lake_doc_model.R')
```

The process model we are using to make predictions of in-lake DOC is extremely simple and calibrates only one parameter, the decay rate of DOC (day^{-1}). This decay rate includes all sources of DOC degredation, such as microbial decay and photomineralization, and is not a function of any drivers. We could change the form of the model to make the decay rate a function of temperature or light, for example, but for now we will keep it simple to better understand the dynamics of the EnKF.

Process model

```r
#' @param doc_load DOC loaded to lake in mol C day^-1
#' @param doc_lake mass of DOC in lake in mol C
#' @param water_out volume of water leaving the lake through stream / groundwater in m^3 day^-1
#' @param lake_vol volume of the lake in m^3
#' @param decay Decay rate of DOC in fraction day^-1

predict_lake_doc = function(doc_load, doc_lake, water_out, lake_vol, decay){

  cur_doc_conc = doc_lake / lake_vol # mol C / m^3

  doc_predict = doc_lake - water_out * cur_doc_conc + doc_load - doc_lake * decay  # mol C

  return(list(doc_predict = doc_predict, decay = decay))
}
```

As the name suggests, the *Ensemble* Kalman Filter utilizes a collection of model runs (ensembles) to make forecasts of the system state (in our example, lake DOC) and uses these ensembles to represent the error statistics as a sample covariance matrix. When a new observation is made, the EnKF updates the model states (DOC) and parameters (decay rate of DOC) based on the relative confidence in the observations and the model estimates. The number of ensembles needed for an analysis should be large enough to allow for reasonable estimates of the mean and covariance, and typically around 100 ensembles are usually used. For our simple model, we don't have to worry about computational costs so we could run more than 100 ensembles but let's just stick to 100 for now.

```r
n_en = 100 # number of ensembles
```

The ensembles allow for uncertainty in parameters, initial conditions, and driver data to be incorporated into the forecast by drawing from a distribution of each respective dataset. Since we are not fitting parameters for driver uncertainty or initial conditions, these sources of uncertainty remain constant throughout the model run. We set parameter uncertainty initially, however, since we are updating parameters in the EnKF, our uncertainty in the parameters we're fitting will become reduced. Let's set our uncertainty based on coefficient of variation for each source.

```r
# coefficient of variation (CV) as a representation of uncertainty
param_cv = 0.2 # DOC decay uncertainty
driver_cv = c(0.2, 0.2, 0.0) # CV of driver data for DOC Load, Discharge out, and Lake volume, respecti
init_cond_cv = .1
```

We have uncertainty in our observations of the amount of DOC in the lake. We also do not fit a parameter for observation uncertainty so we set the uncertainty based on our best guess of uncertainty in our observations. We should take into account uncertainty in the concentration measurement and lake volume estimate to accurately estimate our total observation uncertainty.

```r
# coefficient of variation (CV) as a representation of uncertainty
obs_cv = 0.1
```

The parameter that we are estimating in our model, the DOC decay rate, needs to be initialized as well. Let's set our initial decay rate to 0.005 day^{-1} based on some previous knowledge of the particular lake we are studying and also based on typical literature values. Parameter initial conditions, or priors, can also be estimated as the posterior of a seperate model or with a formal expert solicitation.

```r
decay = 0.005 # decay rate day^-1
```

Finally, we need a start and end date for our model run. The entire timeseries of the dataset is from 2014-06-04 to 2014-09-20 so let's utilize the full timeseries for our first run.

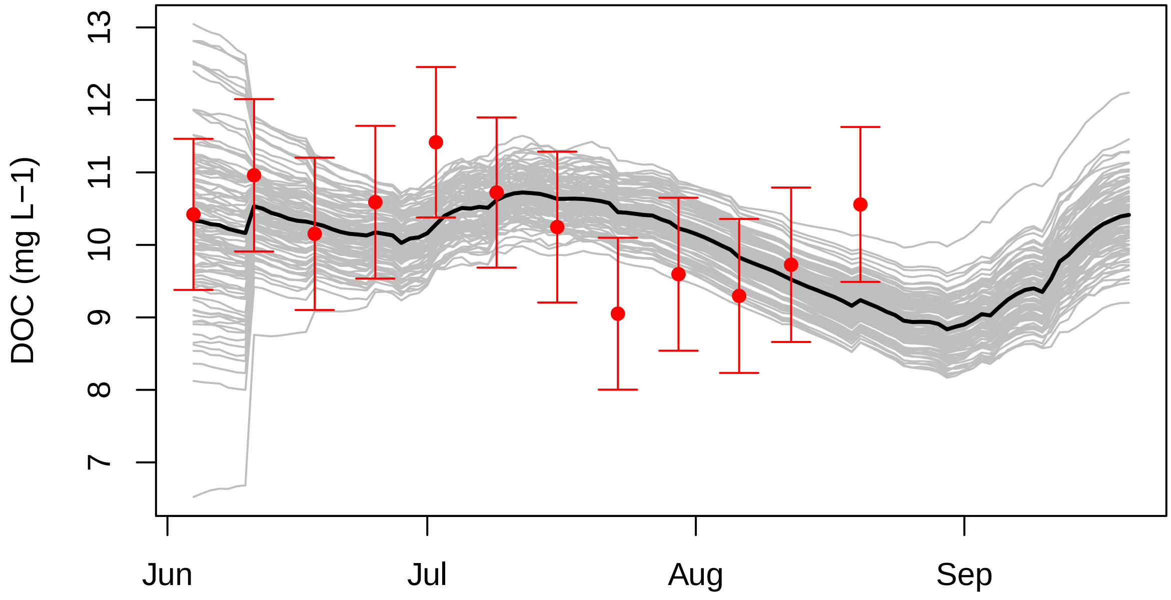
```r
start = '2014-06-04'
stop = '2014-09-20'
```

Now let's run our model while assimilating observations with the EnKF.

```r
est_out = EnKF(n_en = n_en,
           start = start,
           stop = stop,
           obs_file = 'Data/lake_c_data.rds',
           driver_file = 'Data/lake_c_data.rds',
           decay_init = decay,
           obs_cv = obs_cv,
           param_cv = param_cv,
           driver_cv = driver_cv,
           init_cond_cv = init_cond_cv)
```

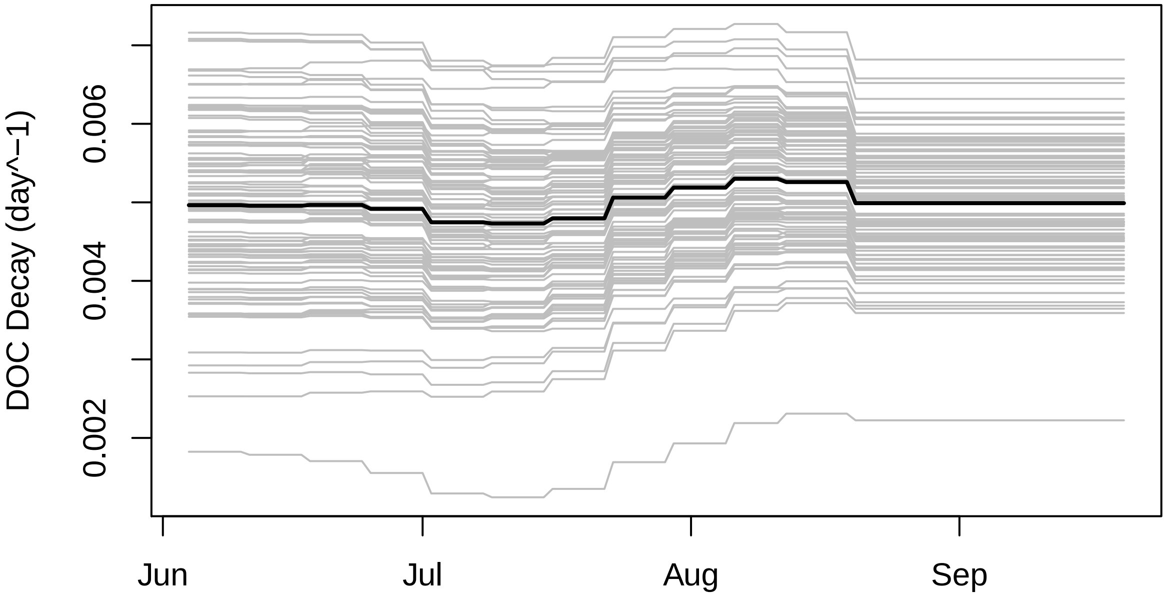
And define some functions that allow us to plot the estimated and observed DOC concentration through time as well as the decay rate estimate through time. Each ensemble member prediction is given by the grey lines and the mean of the ensemble predictions is the black line. The observed DOC are the red points with error bars indicating 1 standard deviation from the mean.

```r
# plotting
plot_doc = function(est_out){
  mean_doc_est = apply(est_out$Y[1,,] / est_out$drivers[,3,] * 12, 1, FUN = mean)
  plot(mean_doc_est ~ est_out$dates, type ='l',
       ylim = range(c(est_out$Y[1,,] / est_out$drivers[,3,] * 12,
                      est_out$obs[1,,] / apply(est_out$drivers[,3,], 1, FUN = mean) * 12),
                    na.rm = T),
       col = 'grey', ylab = 'DOC (mg L-1)', xlab = '')
  for(i in 2:n_en){
    lines(est_out$Y[1,,i] / est_out$drivers[,3,i] * 12 ~ est_out$dates,
          col = 'grey')
  }
  lines(mean_doc_est ~ est_out$dates, col = 'black', lwd =2 )
  points(est_out$obs[1,,] / apply(est_out$drivers[,3,], 1, FUN = mean) * 12 ~
           est_out$dates, pch = 16, col = 'red')
  arrows(est_out$dates, est_out$obs[1,,] / apply(est_out$drivers[,3,], 1, FUN = mean) * 12 -
           est_out$state_sd / apply(est_out$drivers[,3,], 1, FUN = mean) * 12,
         est_out$dates, est_out$obs[1,,] / apply(est_out$drivers[,3,], 1, FUN = mean) * 12 +
           est_out$state_sd / apply(est_out$drivers[,3,], 1, FUN = mean) * 12,
         code = 3, length = 0.1, angle = 90, col = 'red')
}
plot_doc(est_out)
```



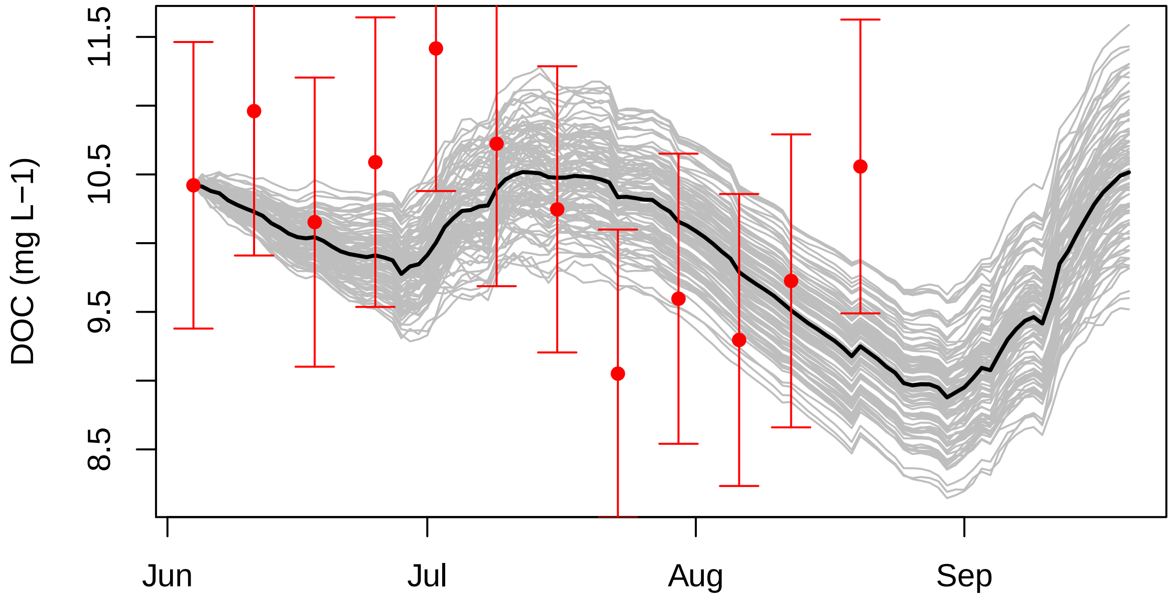
Notice that a lot of model prediction uncertainty is reduced after assimilating a single observation (i.e. grey lines get closer to each other). Our simple model does a pretty good job of estimating DOC as the mean of our observations are within 1 standard deviation for most of the observations. The model uncertainty also does not get very large when observations aren't assimilated (i.e. forecast step) which likely reflects the simple structural form of our model, relatively slow processes affecting DOC concentration in this lake, and our measured driver data (as opposed to forecasted driver data which would increase uncertainty in each forecast step).

```
plot_decay = function(est_out){
  mean_decay_est = apply(est_out$Y[2,,], 1, FUN = mean)
  plot(mean_decay_est ~ est_out$dates, type ='l',
       ylim = range(est_out$Y[2,,]),
       col = 'grey', ylab = 'DOC Decay (day^-1)', xlab ='')
  for(i in 2:n_en){
    lines(est_out$Y[2,,i] ~ est_out$dates, col = 'grey')
  }
  lines(mean_decay_est ~ est_out$dates, col = 'black', lwd = 2)
}
plot_decay(est_out)
```



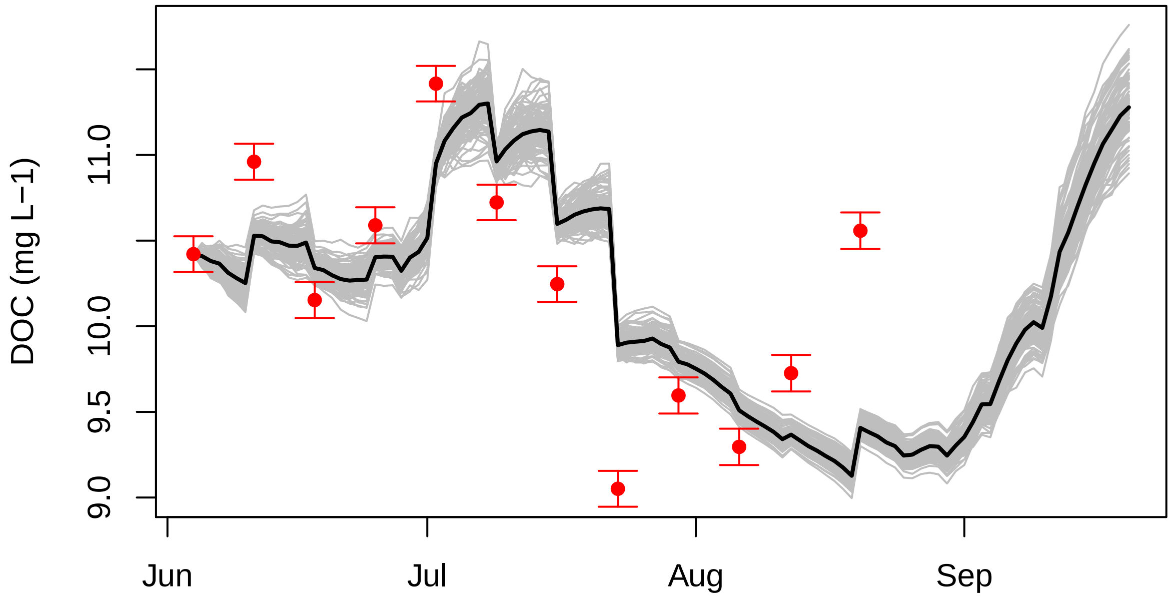
Our initial decay rate mean (0.005 day^{-1}) was fairly close to the final estimated mean and there was little variation in the mean throughout the model run. The decay rate becomes constrained as more observations are assimilated, however, there is still uncertainty in our decay rate at the end of the model run. Assimilating an observation of the decay rate (e.g. bottle incubation) would help us constrain this parameter even more as well as constrain our estimates of lake DOC.

```
est_out_zero_init_cv = EnKF(n_en = n_en,
                            start = start,
                            stop = stop,
                            obs_file = 'Data/lake_c_data.rds',
                            driver_file = 'Data/lake_c_data.rds',
                            decay_init = decay,
                            obs_cv = obs_cv,
                            param_cv = param_cv,
                            driver_cv = driver_cv,
                            init_cond_cv = 0)

plot_doc(est_out_zero_init_cv)
```



When we assume we know the starting conditions perfectly (init_cond_cv = 0), it is hard for the model to incorporate the first few observations into the model since the model is so confident in itself and essentially ignores the observations.

```
est_out_low_obs_error = EnKF(n_en = n_en,
                             start = start,
                             stop = stop,
                             obs_file = 'Data/lake_c_data.rds',
                             driver_file = 'Data/lake_c_data.rds',
                             decay_init = decay,
                             obs_cv = 0.01,
                             param_cv = param_cv,
                             driver_cv = driver_cv,
                             init_cond_cv = 0)

plot_doc(est_out_low_obs_error)
```



Unrealistically low observation uncertainty coupled with no uncertainty in initial conditions creates some strange dynamics in the model forecasts where the model is trying to fit to the highly-confident observations, which creates some 'jumps' in estimated lake DOC.

- How does uncertainty change if you increase or decrease the parameter uncertainty?
- How do the forecasts change if you increase or decrease the initial decay rate?
- How would you change the process model to better capture the DOC dynamics? Would you need additional drivers and/or parameters?