Hydrological calibration by DREAM

DC

08/01/2022

Here I demonstrate two methods to calibrate a hydrological model by Differential Evolution Adaptive Metropolis (DREAM) algorithm using the dream package.I will demonstrate the method using the Sacremanto Soil Moisture Accounting model.

First we load the essential package.

```
library(dream)
library(hydromad)
library(knitr)
```

We need to define the control parameters for DREAM algorithm (like in Matlab). Below is just a customised example. For illustration I choose a short run of 5000 iterations.

```
control.DREAM <- list(
  # ndim is the dimension of the problem
  ndim=13, DEpairs=3,nCR=3,
  ndraw=5000,steps=10, eps=5e-2,
  outlierTest='IQR_test',  # outlier handling method
  pCR.Update=TRUE,thin.t=5,
  boundHandling="fold",
  burnin.length=Inf,  # for compatibility with matlab code
  REPORT=1e3,  # reduce frequency of progress reports,
  nseq=13
)
```

- **Method 1: change the model to a hydromad object and then use the fitbyDream functionality to fit the model.**

```
set.seed(22)
data(Cotter)  # load the data for Cotter catchment as an example
x <- Cotter[1:2000]
# specify the model structure by hydromad() function
modx <- hydromad(x, sma = "sacramento")
# fit the model simply using the fitByDream function,
# will aim to maximise the log-posterior density
foo <- fitByDream(modx, control = control.DREAM)
```

```
## Warning in handleBounds(x.new, lower, upper, control$boundHandling): Bounds
## violated after correction, using random
```

- **Method 2: Keep the original model object type, but define the objective function (log-likelihood) manually for DREAM algorithm to optimise.**

```
# Arguments should be the parameter vector, and any additional parameters to the model.
do_dream_sac <-function(pars,datamat){
  # parameters to be update, one by one
  uztwm <- pars[1]; uzfwm <- pars[2] ; uzk<- pars[3];
  pctim<- pars[4];  adimp<- pars[5] ; zperc<- pars[6];
  rexp <- pars[7]; lztwm <- pars[8]; lzfsm<- pars[9];
  lzfpm <- pars[10]; lzsk <- pars[11];
  lzpk <- pars[12];  pfree <- pars[13]

  ### Fit the model
  thisMod<-sacramento.sim(DATA =datamat ,uztwm = uztwm,uzfwm=uzfwm,
                          uzk=uzk,pctim = pctim,adimp =  adimp,
                          zperc   =zperc,rexp = rexp,
                          lztwm = lztwm,lzfsm =lzfsm,lzfpm =lzfpm,
                          lzsk = lzsk, lzpk = lzpk,pfree =pfree)
  spinup <- hydromad.getOption("warmup")  # default is 100
  Qobs <- datamat[-c(1:spinup),"Q"]
  Qsim <- thisMod[-c(1:spinup)]

  # Calculate the log-likelihood
 log_lik <- -0.5 * sum((Qobs - Qsim)^2, na.rm = TRUE)
 return(log_lik)

}
# we can check how hydromad defines log likelihood
hydromad.getOption("loglik")
```

```
## function (Q, X, ...)
## -0.5 * sum((Q - X)^2, na.rm = TRUE)
## <bytecode: 0x7fd6c2891758>
## <environment: 0x7fd6c2891330>
```

Now we're ready to do the calibration of the original model.

```
set.seed(44)  # we can try to set a different seed
# Obtain the parameter range
parlist <- as.list(hydromad.getOption("sacramento"))
trial1 <- dream(do_dream_sac, pars = parlist,
      # note here we need to give the input data for model to run
                  FUN.pars = list(datamat = x),
                  func.type = "logposterior.density",
                  control = control.DREAM
)
```

```
## R.stats:

##  fun.evals     uztwm     uzfwm       uzk     pctim     adimp     zperc      rexp     lztwm     lzfsm
##   4940.000     1.729     1.250     1.135     1.472     1.394     1.327     1.348     1.805     1.926
```

We can examine the outputs from these two methods

```
coef(trial1)
```

```
##        uztwm        uzfwm          uzk        pctim        adimp        zperc
## 3.065546e+00 1.026035e+02 1.625063e-01 2.138221e-02 1.250216e-01 4.844459e+01
##         rexp        lztwm        lzfsm        lzfpm         lzsk         lzpk
## 4.651107e+00 1.511941e+02 5.203109e+02 7.542746e+02 1.840199e-01 9.975254e-03
##        pfree
## 4.666714e-01
```

```
coef(foo)
```

```
##        uztwm        uzfwm          uzk        pctim        adimp        zperc
##   1.36093968  84.18879462   0.35110561   0.02380243   0.12604594  47.67853556
##         rexp        lztwm        lzfsm        lzfpm         lzsk         lzpk
##   1.69671481 136.16024400 553.37877252 977.10158761   0.20413632   0.01884324
##        pfree
##   0.38186893
```

```
# Algorithmic summary of DREAM run
summary(foo$fit.result)
```

```
##
## Exit message:  Maximum function evaluations reached
## Num fun evals: 2080
## Time (secs):   25.3
## Final R.stats:
##  uztwm:  2.496345
##  uzfwm:  1.874992
##  uzk:    1.480294
##  pctim:  2.383285
##  adimp:  2.817849
##  zperc:  1.485537
##  rexp:   1.810074
##  lztwm:  1.715141
##  lzfsm:  1.668869
##  lzfpm:  2.379319
##  lzsk:   2.401830
##  lzpk:   4.242050
##  pfree:  2.302482
##
## CODA summary for last 50% of MCMC chains:
##
## Iterations = 81:156
## Thinning interval = 5
## Number of chains = 13
## Sample size per chain = 16
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
```

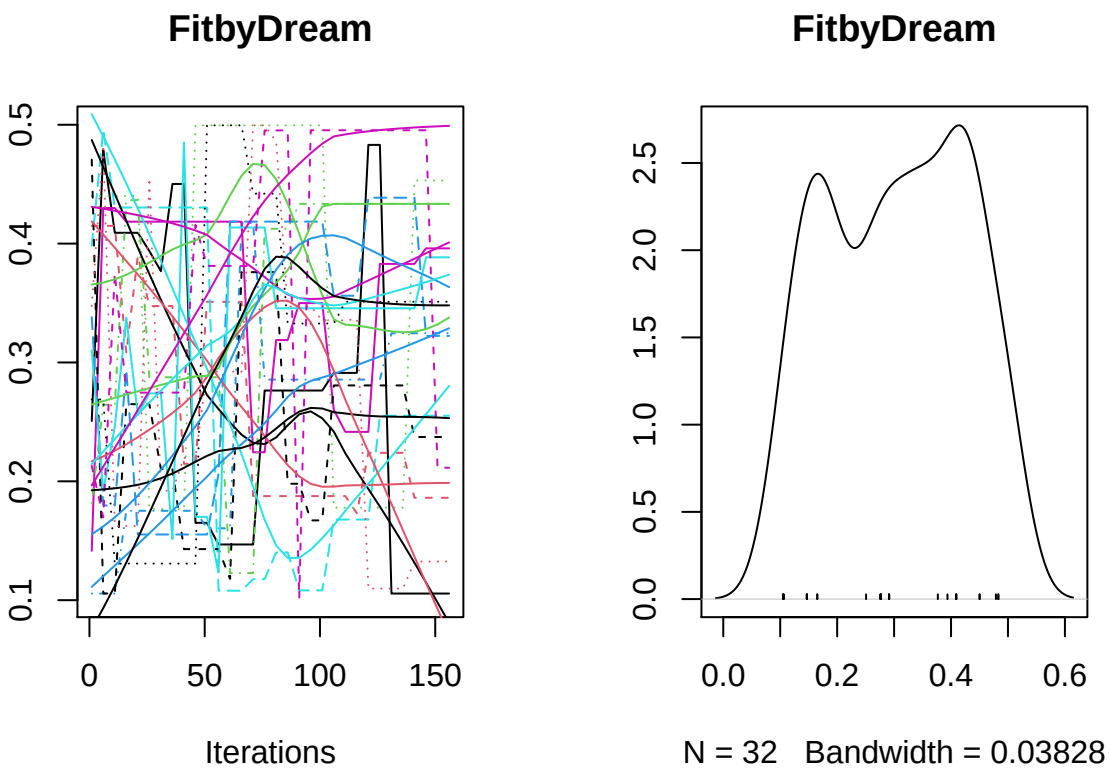
```
##            Mean        SD  Naive SE Time-series SE
## uztwm   3.45752   1.66473 1.154e-01       0.131198
## uzfwm  67.54258  35.17418 2.439e+00       2.502439
## uzk     0.31303   0.11321 7.850e-03       0.012341
## pctim   0.03609   0.01404 9.733e-04       0.001357
## adimp   0.14540   0.09650 6.691e-03       0.009288
## zperc 124.85043  73.94951 5.127e+00       8.637143
## rexp    2.22425   1.33005 9.222e-02       0.146389
## lztwm 281.26210 112.68225 7.813e+00      12.375188
## lzfsm 486.55576 251.08889 1.741e+01      29.340205
## lzfpm 623.10148 207.08730 1.436e+01      17.163690
## lzsk    0.13050   0.07493 5.195e-03       0.005968
## lzpk    0.05059   0.05193 3.601e-03       0.001918
## pfree   0.49176   0.07159 4.964e-03       0.004961
##
## 2. Quantiles for each variable:
##
##             2.5%       25%       50%       75%     97.5%
## uztwm 1.361e+00   2.27733   3.28277   4.39623   7.83370
## uzfwm 8.604e+00  35.92098  68.99642  89.15296 132.63800
## uzk   1.056e-01   0.22398   0.32867   0.40014   0.49538
## pctim 1.476e-02   0.02380   0.03502   0.04570   0.06419
## adimp 8.935e-03   0.05720   0.14931   0.18799   0.38768
## zperc 1.835e+01  48.47864 130.64130 185.01422 237.39645
## rexp  5.037e-02   1.45209   2.02691   3.26657   4.89271
## lztwm 9.770e+01 209.02200 266.50918 378.94798 476.20242
## lzfsm 1.086e+02 295.45720 459.64797 638.88863 937.41928
## lzfpm 2.714e+02 456.51009 631.62725 818.96406 977.10159
## lzsk  1.344e-02   0.06113   0.12828   0.20414   0.23740
## lzpk  4.421e-03   0.01615   0.02969   0.08151   0.21347
## pfree 3.720e-01   0.42850   0.49619   0.55211   0.59692
##
##
## Acceptance Rate
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   7.692   8.056  12.846  46.154
```

```
summary(trial1)
```

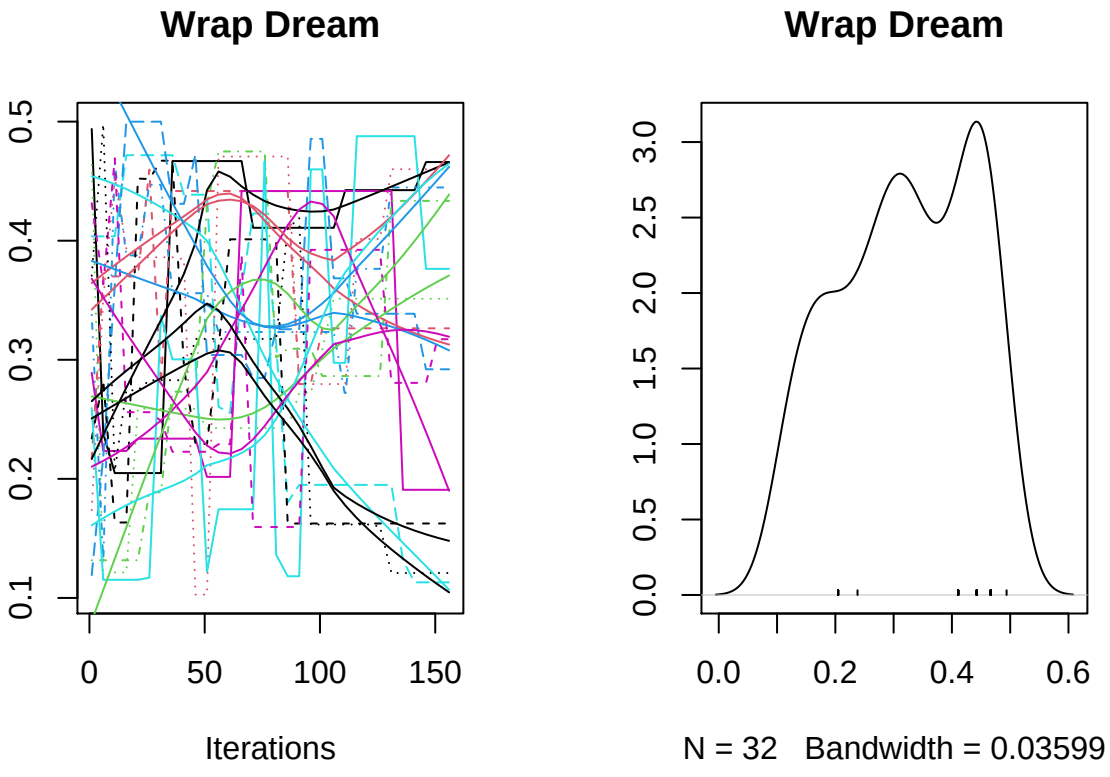
```
##
## Exit message:  Maximum function evaluations reached
## Num fun evals: 2080
## Time (secs):   16.7
## Final R.stats:
##  uztwm:  1.386962
##  uzfwm:  1.546420
##  uzk:    1.518715
##  pctim:  1.764694
##  adimp:  1.966790
##  zperc:  1.541919
##  rexp:   1.320419
##  lztwm:  1.613233
##  lzfsm:  1.735403
##  lzfpm:  1.907219
```

```
##  lzsk:   1.942286
##  lzpk:   1.946647
##  pfree:  2.775357
##
## CODA summary for last 50% of MCMC chains:
##
## Iterations = 81:156
## Thinning interval = 5
## Number of chains = 13
## Sample size per chain = 16
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean        SD  Naive SE Time-series SE
## uztwm   3.45655   5.18054 3.592e-01       0.429187
## uzfwm  67.55259  37.40552 2.594e+00       4.024753
## uzk     0.31913   0.11211 7.773e-03       0.012843
## pctim   0.03865   0.01349 9.352e-04       0.001639
## adimp   0.17026   0.11153 7.733e-03       0.010414
## zperc 121.74121  59.84058 4.149e+00       6.189505
## rexp    2.83036   1.46756 1.018e-01       0.219959
## lztwm 333.98704 116.22306 8.059e+00      12.793966
## lzfsm 604.67259 258.14713 1.790e+01      25.973735
## lzfpm 605.00017 171.90379 1.192e+01      15.482178
## lzsk    0.12662   0.07371 5.111e-03       0.007478
## lzpk    0.06139   0.05883 4.079e-03       0.005660
## pfree   0.47055   0.06572 4.557e-03       0.003483
##
## 2. Quantiles for each variable:
##
##            2.5%       25%       50%       75%     97.5%
## uztwm 1.001e+00   1.64309   2.56903   3.80877   7.45917
## uzfwm 1.618e+01  36.13107  65.10228  90.58942 134.01351
## uzk   1.187e-01   0.19493   0.32639   0.43342   0.48737
## pctim 1.598e-02   0.03063   0.03566   0.04735   0.07189
## adimp 6.361e-03   0.06553   0.17024   0.26557   0.39345
## zperc 3.042e+01  61.93563 123.49589 176.48808 226.83516
## rexp  1.547e-01   1.43318   2.97722   4.06754   4.65111
## lztwm 9.184e+01 275.15844 342.35253 437.40860 489.20885
## lzfsm 1.524e+02 385.65027 620.58046 827.20875 990.53110
## lzfpm 3.580e+02 446.73640 630.08057 747.96859 894.52785
## lzsk  1.998e-02   0.05010   0.15073   0.18402   0.24889
## lzpk  7.272e-03   0.01151   0.03591   0.10407   0.20181
## pfree 2.989e-01   0.44745   0.48043   0.51471   0.56810
##
##
## Acceptance Rate
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   7.692  11.373  15.385  46.154
```

```
# check the parameter distribution, using multiple chains
plot(foo$fit.result$Sequences[,3],type="l",main="FitbyDream")
```




```
plot(trial1$Sequences[,3],type="l",main="Wrap Dream")
```



Clearly the posterior distributions show some differences. We can fit the model by the optimised parameter values to see the performance.

```
  # fit the model using the outcome from the second method
  pars1 <- coef(trial1)
  fit_sac <- hydromad(DATA = x,sma = "sacramento")
  fit_sac <-update(fit_sac,newpars =  pars1)

  # or, if not a hydromad object, fit directly
  modfit <-sacramento.sim(x,uztwm =         pars1[1],uzfwm=pars1[2],uzk=pars1[3],pctim = pars1[4],
   adimp =  pars1[5],zperc = pars1[6],rexp = pars1[7],
   lztwm = pars1[8],lzfsm = pars1[9],lzfpm =pars1[10],lzsk = pars1[11],
   lzpk = pars1[12],pfree = pars1[13] )

  # Evaluation of performance metrics
  hmadstat("r.squared")(Q=observed(foo),X=fitted(foo))  # Method 1
```

```
## [1] 0.553769
```

```
  hmadstat("r.squared")(Q=observed(fit_sac),
                        X=fitted(fit_sac)) # Method 2
```

```
## [1] 0.5566737
```

```
  # Note: Need to manually apply the spin-up period if
  # we fit on the original model directly
  hmadstat("r.squared")(Q=x[-c(1:100),"Q"],X=modfit[-c(1:100)])
```

```
## [1] 0.5566737
```

```
  # Use Nash-Sutcliffe Efficiency at log-scale
  hmadstat("r.sq.log")(Q=x[-c(1:100),"Q"],X=modfit[-c(1:100)])
```

```
## [1] 0.4683961
```

```
  hmadstat("r.sq.log")(Q=observed(foo),X=fitted(foo))
```

```
## [1] 0.4948943
```

We can see that different combinations of parameters but give similar fitting performance, showing equifinality of the model. We may consider bringining new constraints e.g. from remote sensing data.