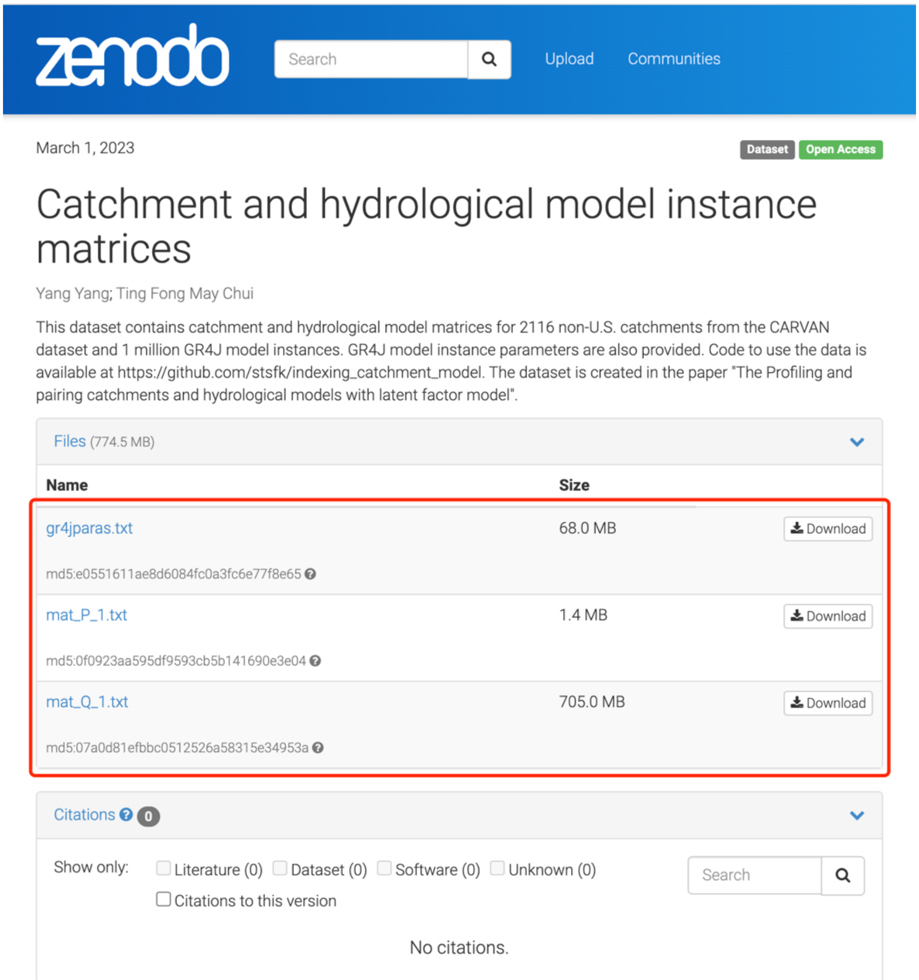
Tutorial for "Profiling and pairing catchments and hydrological models with latent factor model"

1. Data

- The source code used in this tutorial can found in: https://github.com/stsfk/indexing_catchment_model/blob/main/tutorial.R

- R programming language is used, which can be downloaded at: https://www.r-project.org. The recommended IDE for R is RStudio, which can be downloaded at: https://posit.co/download/rstudio-desktop/.

- The data can be downloaded at: https://doi.org/10.5281/zenodo.7687554. The three files are needed for this experiments.



2. Experiment steps

1. Copy the “mat_Q_1.txt”, “mat_P_1.txt”, “gr4jparas.txt”, and “tutorial.R” to the working folder of R.

2. Open “tutorial.R” in RStudio, run line 1 through line 16 to install packages used in the experiments.

```
1  # setting experiment environment ---------------------------------------
2
3  if (!require("pacman")) {
4    install.packages("pacman")
5  }
6
7  pacman::p_load(tidyverse,
8                 lubridate,
9                 zeallot,
10                recosystem,
11                GA,
12                ModelMetrics,
13                doParallel,
14                Rfast,
15                airGR,
16                cowplot)
17
18 set.seed(12345)
19
```

3. Run line 20 through 34 to load the data. The L0123001 catchment data is loaded from the airGR R package using “data(L0123001)”.

```
20 # data -------------------------------------------------------------------
21
22 # load gr4j parameters
23 paras <- read.table("./gr4jparas.txt", header = FALSE, sep = " ") %>% data.matrix()
24
25 n_instances <- nrow(paras)
26
27 # load PQ data
28 P <- read.table("./mat_P_1.txt", header = FALSE, sep = " ") %>% data.matrix()
29 Q <- read.table("./mat_Q_1.txt", header = FALSE, sep = " ") %>% data.matrix()
30
31 latent_dim <- dim(Q)[2]
32
33 # load L0123001 data from airGR package
34 data(L0123001)
```

4. Run line 40 through 51 to configure the run option of GR4J model from airGR package. The run period is from “1990-01-01” to “1999-12-31”.

```r
# Model calibration -------------------------------------------------------
# The code in this section is from the "Get Started with airGR" user guide included in the
# airGR package.

InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%Y-%m-%d") == "1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d") == "1999-12-31"))

RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run,
                               IniStates = NULL, IniResLevels = NULL, IndPeriod_WarmUp = NULL)

InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, VarObs = "Q", Obs = BasinObs$Qmm[Ind_Run])
```

5. Run line 53 to 64 to set calibration options. The search ranges are specified in MARRMoT (https://github.com/wknoben/MARRMoT).

```r
53 CalibOptions <-
54   CreateCalibOptions(
55     FUN_MOD = RunModel_GR4J,
56     FUN_CALIB = Calibration_Michel,
57     SearchRanges = matrix(c(1, -10, 1, 1, 2000, 15, 300, 15), nrow = 2, byrow = T)
58   )
59
60 OutputsCalib <- Calibration_Michel(InputsModel = InputsModel, RunOptions = RunOptions,
61                                    InputsCrit = InputsCrit, CalibOptions = CalibOptions,
62                                    FUN_MOD = RunModel_GR4J)
63
64 calibrated_para <- OutputsCalib$ParamFinalR # calibrated parameter
65 calibrated_nse <- OutputsCalib$CritFinal
```

6. Run line 69 to line 82 to define two functions. “p_rmse” computes the RMSE of the association $r_{i,j}$ predicted for a catchment latent factor vector p_i. The “para_gof” returns the NSE value resulting from the “para”,

```r
69 p_rmse <- function(p, Q, rating) {
70   # This function computes the RMSE of predicted ratings of models specified by Q
71   ModelMetrics::rmse(predicted = p %*% t(Q) %>% as.vector(),
72                      actual = rating)
73 }
74
75 para_gof <- function(para){
76   # compute the gof metric for the parameter set "para"
77   # InputsModel and RunOptions are defined in the global environment
78
79   OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = para)
80   OutputsCrit <- ErrorCrit(InputsCrit = InputsCrit, OutputsModel = OutputsModel, verbose = FALSE)
81   OutputsCrit$CritValue
82 }
```

7. Run line 84 to line 97 to define function “para_ids_gof”, which derive the ratings (normalized NSE from 0 to 10) for the model instances specified by “selected_para_ids”, for which the model parameters are stored in “paras”.

```
para_ids_gof <- function(selected_para_ids){
  # get the ratings for the model instances specified by selected_para_ids
  selected_paras <- paras[selected_para_ids,]

  nses <- foreach(i=1:nrow(selected_paras)) %do%
    para_gof(selected_paras[i,]) %>%
    unlist()

  tibble(
    para_id = selected_para_ids,
    nse = nses,
    rating = 1 / (2 - nses) * 10
  )
}
```

8. Run line 99 to 130 to define function "prepare_Qs_for_retrieval", which splits Q into "Q_probed", "Q_train", and "Q_val". "Q_probed" is associated with sampled model instances, whose $r_{i,j}$ with the catchment is defined through hydrological modelling. "Q_probed" is further split into "Q_train", and "Q_val" for deriving the optimal number of iterations in genetic algorithm (GA).

```
99  prepare_Qs_for_retrieval  <- function(n_probed, train_portion){
100   # This function split Q into Q_probed, Q_train, Q_val
101   # weights of the links between the look-up catchment and the models associated with Q_probed is known
102   # Q_probed is further split into Q_train and Q_val for deriving the optimal number of iterations in GA
103 
104   # n_probed: the number of links between Q and the look-up catchment with known weights
105 
106   selected_para_ids <- sample(1:n_instances, n_probed) %>% sort()
107   rating_probed <- para_ids_gof(selected_para_ids) %>% pull(rating)
108   Q_probed <- Q[selected_para_ids,]
109 
110   # train and validation split
111   n_train <- round(length(selected_para_ids)*train_portion)
112 
113   sample_id <- sample(seq_along(selected_para_ids), size = n_train) %>% sort()
114   model_id_train <- selected_para_ids[sample_id]
115   model_id_val <- selected_para_ids[-sample_id]
116   Q_train <- Q[model_id_train,]
117   Q_val <- Q[model_id_val,]
118   rating_train <- rating_probed[sample_id]
119   rating_val <- rating_probed[-sample_id]
120 
121   # output
122   list(
123     Q_probed = Q_probed,
124     rating_probed = rating_probed,
125     Q_train = Q_train,
126     rating_train = rating_train,
127     Q_val = Q_val,
128     rating_val = rating_val
129   )
130 }
```

9. Run line 132 to line 140 to define "fn_factory", which is a function factory that compute the "-RMSE" for given "Q" and the associated "rating" (i.e., NNSE).

```r
132 fn_factory <- function(Q, rating) {
133   function(x) {
134     # This function computes the predicted
135     pred <- Rfast::eachrow(Q, x, "*")
136     pred <- Rfast::rowsums(pred, parallel = T)
137 
138     - ModelMetrics::rmse(actual = rating, predicted = pred)
139   }
140 }
```

10. Run line 142 through 208 to define "derive_p", which derives the optimal p_i for "n_probed" sample model instances, and "train_portion" specifies the portion of model samples selected for estimating the optimal number of generations in GA.

```r
142 derive_p <- function(n_probed, train_portion){}
208 
```

11. Run line 209 through 230 to define "top_n_nse" that select the "top_n" models based on predicted $r_{i,j}$ using the optimal p_i and compute the NSE.

```r
209 top_n_nse <- function(p, n_retrieved){
210 
211   pred <- p %*% t(Q) %>% as.vector()
212 
213   top_n <- tibble(
214     model_id = 1:n_instances,
215     rating = pred
216   ) %>%
217     arrange(desc(rating)) %>%
218     slice(1:n_retrieved) %>%
219     pull(model_id)
220 
221   gof <- para_ids_gof(top_n)
222 
223   # output
224   tibble(
225     para_id = top_n,
226     rating = gof$rating,
227     nse = gof$nse,
228     rank = 1:n_retrieved
229   )
230 }
```

12. Run line 232 to 242 to conduct the experiment using the functions defined earlier, where the number of sampled models is 4 times of the latent factor dimension, number of models retrieved is 200, and 80% of the population is used to find the optimal number of generations.

```
232 # Modeling ---------------------------------------------------------------
233
234 n_probed <- latent_dim * 4
235 n_retrieved <- 200
236 train_portion <- 0.8
237
238 p <- derive_p(n_probed, train_portion)
239 retrieval_result <- top_n_nse(p, n_retrieved)
240
241 retrieved_nse <- retrieval_result$nse %>% max()
242 retrieved_para <- paras[retrieval_result$para_id[which.max(retrieval_result$nse)],] %>% unname()
243
```

13. Analyse the plot the results by running line 244 through 282.

```
244 # Result analysis ---------------------------------------------------------
245
246 Q_calibrated <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = calibrated_para)$Qsim
247 Q_retrieved <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = retrieved_para)$Qsim
248 Q_actual <- BasinObs[Ind_Run,] %>% pull(Qmm)
249
250 data_plot <- tibble(
251   date = seq(from = ymd("1990-01-01"), to = ymd("1999-12-31"), by = 1),
252   calibrated = Q_calibrated,
253   retrieved = Q_retrieved,
254   actual = Q_actual
255 )
256
257 p1 <- data_plot %>%
258   gather(item, value, -date) %>%
259   ggplot(aes(date, value, color = item, linetype = item))+
260   geom_line(size = 0.25)+
261   labs(color = "",
262        linetype = "",
263        y = "Flow [mm/day]") +
264   theme_bw(base_size = 10)+
265   theme(legend.position = "top",
266         axis.title.x = element_blank())
267
268 p1 +
269   cowplot::draw_text(
270     x = ymd("1990-01-01"),
271     y = 19,
272     size = 10,
273     hjust = 0,
274     text = paste0("NSE of calibrated model = ", round(calibrated_nse,3))
275   )+
276   cowplot::draw_text(
277     x = ymd("1990-01-01"),
278     y = 17.5,
279     size = 10,
280     hjust = 0,
281     text = paste0("NSE of retrieved model = ", round(retrieved_nse,3))
282   )
```

14. The retrieved parameters are: 247.619507, 1.105701, 93.257153, 2.092982, and the calibrated parameters are: 257.237556, 1.012237, 88.234673, 2.207958. The two sets of parameters are relatively similar.

15. Here is the predicted hydrographs compared to the “observed” one using the retrieved and calibrated models.