Course Notes Advanced SWAT: creating SWAT-CUP input

Willem Vervoort

22-07-2017

Functions to prepare SWAT-CUP files for the calibration and validation routine

This file demonstrates the use different auxillary functions to create input data for SWAT calibration in SWAT-CUP.
There are 4 functions.

- `MODIS_ts()` to transform the MODIS data to a timeseries, we have seen this function already in the calibration of the rainfall runoff model. Here we use the same fucntion.
- `readfun()` a small function to read in a SWAT-CUP observation file.
- `organiseFun()` a function to organise the observed data into the correct SWAT-CUP format of "number", "Identifier", "value", see the SWAT-CUP manual page 29.
- `writeFun()` a function to write the different segments of the observed.txt file including the header and assigning the correct weight to the different parts of the observed data.
- `swatcup_ETformat()` the final function that encapsulates the earlier functions and generates thedifferent files depending on the inputs given.

MODIS_ts

This function read the directory indicated by *MODISdir* and looks for files with the extension pattern given by *patt*. The output is a timeseries of values stacked for all the points that are available. There are 5 columns in the output:

- Year
- JDay (Julian Day)
- value (of actual ET in mm)
- Point (a number in the catchment)
- Date (the actual date in Y-m-d)

```
MODIS_ts <- function(MODISdir="MODIS",patt=".asc"){

  # read in all the file names
  x1 <- list.files(path=MODISdir, pattern=patt)

  # each "asc" file stores all the values in time for 1 location
```

```
  # the number of rows is important as this is all the time steps
  # divide nrows by 2 as second part is QC data
  n <- nrow(read.csv(paste(MODISdir,x1[1],sep="/"),header=F))/2
  # Create storage for the data, Jdate is Julian date
  Store <- data.frame(Year = numeric(length=n),
                      JDay = numeric(length=n),
                      ET = numeric(length=n),
                      Point = numeric(length = n))

  # Create a list to store the different pixels (each with a Store)
  Store1 <- list()
  # run a loop over the list of file names
  for (i in 1:length(x1)) {
    Mdata <- read.csv(paste(MODISdir,x1[i],sep="/"),header=F)
    # do some substringing
    Store[,1] <- as.numeric(substr(Mdata[1:n,8],2,5))
    Store[,2] <- as.numeric(substr(Mdata[1:n,8],6,8))
    Store[,3] <- Mdata[1:n,11]/10
    # 0.1 scaling factor (see MODIS read_me)
    Store[,4] <- i
    Store1[[i]] <- Store

  }
  # converting from list back to a data.frame
  ts.data <- do.call(rbind,Store1)
  # Now make the date from the Year and Jdate
  ts.data$Date <- as.Date(paste(ts.data$Year,ts.data$JDay,
                                sep = "/"), "%Y/%j")

  return(ts.data)
}

# demonstrate
# Create a single file with all the MODIS ET data for all points
ET_Data <- MODIS_ts("MODIS/Cotter")
# show the data
head(ET_Data)
```

```
##   Year JDay   ET Point       Date
## 1 2000    1 29.1     1 2000-01-01
## 2 2000    9 36.9     1 2000-01-09
## 3 2000   17 28.3     1 2000-01-17
## 4 2000   25 25.9     1 2000-01-25
## 5 2000   33 25.9     1 2000-02-02
## 6 2000   41 25.9     1 2000-02-10
```

readfun

This is a simple auxillary function to read in a txt file (*filename*) as actual txt and read in only a specific number of lines (n).

```
readfun <- function(filename, n=n) {
  foo <- file(filename,"r+")
```

```
  foo_bar <- readLines(foo, n = n)
  close(foo)
  return(foo_bar)
}

# demonstrate
header <- readfun("inputdata/observed.txt", 12)
str(header)
```

```
##  chr [1:12] "4     : number of observed variables" ...
```

organiseFun

This auxillary function reorganises the ET data in the input data frame (*df_in*) between the start date (*st.date*) and end date (*end.date*) in the correct format for SWAT-CUP. the companion function organiseFlow does the same thing for flow data

```
organiseFun <- function(df_in, st.date, end.date) {
  # simple downscale the ET, can be improved
  difdays <- diff(df_in$JDay)
  difdays <- replace(difdays,difdays==-360,5)
  df_in$ET <- df_in$ET/c(5,difdays)
  # generate the SWAT_CUP id
  df_in$cup_ID <- paste("ET_", df_in$Point[1], "_",
                        df_in$JDay, "_", df_in$Year, sep = "")
  # generate the full date series
  full_dates <- seq.Date(as.Date(st.date), as.Date(end.date), by = 1)
  full_dates <- data.frame(n = 1:length(full_dates), Dates = full_dates)
  # subsetting the data
  df_in2 <- subset(df_in, Date >= as.Date(st.date) & Date <= as.Date(end.date))
  # match full_dates with df_in2
  serial <- full_dates[full_dates[,2] %in% df_in2$Date,]

  # re arranging the dataframe
  df_out <- data.frame(Serial = serial$n,
                       cup_ID = df_in2$cup_ID, ET = round(df_in2$ET,3))
  return(df_out)
}

# demonstrate
test <- organiseFun(ET_Data[ET_Data$Point==1,], "2010-01-01",
                    "2012-12-31")
head(test)
```

```
##   Serial       cup_ID    ET
## 1      1  ET_1_1_2010 7.340
## 2      9  ET_1_9_2010 4.525
## 3     17 ET_1_17_2010 4.500
## 4     25 ET_1_25_2010 4.562
## 5     33 ET_1_33_2010 3.625
## 6     41 ET_1_41_2010 3.125
```

The function `organiseFlow()` does the same thing for flow data

```r
organiseFlow <- function(df_in, st.date, end.date) {

  df_in$JDay <- format.Date(df_in$Date, "%j")
  df_in$Year <- format.Date(df_in$Date, "%Y")
  df_in$Date <- as.Date(df_in$Date)
  df_in$cup_ID <- paste("FLOW_OUT_",df_in$JDay,"_",df_in$Year, sep = "")

  # subsetting the data
  df.sub <- subset(df_in, Date >= as.Date(st.date) &
                     Date <= as.Date(end.date))
  df.sub$Serial <- 1:nrow(df.sub)
  # Removing the NA values as SWAT-CUP does not deal with NA values
  df.sub <- na.omit(df.sub)

  # re arranging the dataframe
  df.sub <- data.frame(Serial = df.sub$Serial,
                       cup_ID = df.sub$cup_ID, Flow = df.sub[,2])
  return(df.sub)
}

# Demonstrate
# read in flow data
flowdata <- readRDS(file="inputdata/Discharge_data_2000_2017.RDS")
head(flowdata)
```

```
##         Date Flow_cumec_Weejasper_410024 Flow_cumec_Gingira_410730
## 1 2000-01-01                      19.439                     2.369
## 2 2000-01-02                      15.735                     2.077
## 3 2000-01-03                      13.361                     1.885
## 4 2000-01-04                      11.750                     1.706
## 5 2000-01-05                      10.518                     1.566
## 6 2000-01-06                       9.496                     1.445
```

```r
colnames(flowdata)[1] <- "Date"

test <- organiseFlow(flowdata[,c(1,3)],"2010-01-01", "2010-12-31")
head(test)
```

```
##   Serial            cup_ID  Flow
## 1      1 FLOW_OUT_001_2010 0.283
## 2      2 FLOW_OUT_002_2010 0.264
## 3      3 FLOW_OUT_003_2010 0.255
## 4      4 FLOW_OUT_004_2010 0.218
## 5      5 FLOW_OUT_005_2010 0.205
## 6      6 FLOW_OUT_006_2010 0.194
```

writeFun

This function writes the text back to th input files for SWAT-CUP. The following input is needed:

- outfile, this is the name of the specific SWAT-CUP input file that you wish to write
- df_write, this is a dataframe, generally there result from the previous `organiseFun()`

- header, this is the header of the original SWAT-CUP file and which needs to be added to each observed data set

- FLOW, a boolean switch, whether or not flow data should be included in the file
- np, the total number of observed variables, this is generated automatically from the input data
- i, the counter for the specific subbasin, this is generated automatically from the input data

- weight this is the weight assigned to the flow data, or can be a vector of weights, this is imported from the overlying function (defined next below).

```
writeFun <- function(outfile, df_write, header = header,
                     Flow = FALSE,
                     np = NULL, i = NULL, weight = 0.5) {
  #browser()
  p <- grep("subbasin number",header)
  r <- grep("number of data points", header, ignore.case=T)

  # writing the rch file
  if (gregexpr("observed_rch.txt",outfile)==T) {
      header[p] <- paste("FLOW_OUT_1","   ",
                         substr(header[p], 11, nchar(header[p])),sep="")
      header[r] <- paste(nrow(df_write),substr(header[r], 6,
                                               nchar(header[r])),
                         sep = "   ")
  }

  # writing the observed.txt file
  if (gregexpr("observed.txt",outfile, fixed=T)==T & Flow == TRUE) {
    header[p] <- paste("FLOW_OUT_1", "  ",
                       substr(header[p], 11, nchar(header[p])),sep="")
    header[p + 1] <- paste(ifelse(length(weight) > 1,weight[1],weight),
                           "    ",substr(header[p + 1], 7,
                                         nchar(header[p + 1])),sep="")
    header[r] <- paste(nrow(df_write),substr(header[r], 9,
                                             nchar(header[r])),
                       sep = "   ")
  } else {
    if (gregexpr("observed.txt",outfile, fixed=T)==T & Flow == FALSE) {
      if (length(weight) > 1) w <- weight[i+1] else w <- (1-weight)/np
      header[p] <- paste("ET_", i,"   ",
                         substr(header[p], 11, nchar(header[p])),sep="")
      header[p + 1] <- paste(round(w,4), "    ",
                             substr(header[p + 1], 9,
                                    nchar(header[p + 1])),
                             sep="")
      header[r] <- paste(nrow(df_write),substr(header[r], 6,
                                               nchar(header[r])),
                         sep = "   ")
    }
  }
  # writing observed_sub.txt
  if (gregexpr("observed_sub.txt",outfile)==T) {
    header[p] <- paste("ET_", i, "  ",
                       substr(header[p], 11, nchar(header[p])),sep="")
```

```
    header[r] <- paste(nrow(df_write),substr(header[r], 6,
                                             nchar(header[r])),
                       sep = "    ")
  }
  # write to a file, but with a header
  write(header[(p-1):(r+2)], file = outfile, append = T)
  write.table(df_write,file=outfile, sep = "    ",row.names = F,
              col.names = F, quote = F,
              append = T)
}

# This can't be demonstrated directly as this write to a file
```

swatcup_ETformat

This function encapsulates the earlier functions and uses them to write the input files for SWAT-CUP in the right format for both ET and flow data. This function takes the following input

- df, this is a data frame with flow data or ET data, the output of `MODIS_ts()`
- df_flow this is an optional dataframe with flow data if df is ET data
- date.format is a definition of the date format in case the date format in the flow data is incorrect
- st.date: the starting date for the output
- end.date: the end date for the output
- outfile: the SWAT-CUP file you want to write
- infile: the SWAT-CUP file you use as template
- nlines: the number of lines in the header
- Flow: a boolean indicating whether or not flow data is included
- weight: a single number or a vector indicating the weight of the flow data relative to the other input data. The objective function weights will be $weight * flow + \sum weight/np * Obs_i$

```
swatcup_ETformat <- function(df, df_flow = NULL,
                             date.format = "%Y-%m-%d",
                             st.date, end.date,
                             outfile ,infile, nlines,
                             Flow = FALSE,
                             weight = 0.5){
  # colnames should be c("Year","JDay", "ET", "Point", "Date")
  # Formating to a date format
  df$Date <- as.Date(df$Date, format = date.format)
  # read in the header from the file
  header <- readfun(infile, nlines)
  # write the number of observed variables to the top
  header[1] <- paste(ifelse(Flow==FALSE ,
                            length(unique(df$Point)),
                            length(unique(df$Point)) + 1),
                     "     : number of observed variables")
  write(header[1:(grep("subbasin number",header) - 2)],
        file = outfile)
  # prepare the flow data
  if (Flow == TRUE) {
    # use organiseflow
    if (length(df_flow)>0) df_input <- df_flow else df_input <- df
    df_in2 <- organiseFlow(df_in = df_input,
```

```
                            st.date, end.date)
    # use writeFun
    writeFun(outfile = outfile,
             df_write = df_in2, header = header, Flow = Flow,
             np = NULL,
             weight = weight)
  }

   # running a loop through the number of points
  if (length(df_flow) > 0 | Flow == FALSE) {
    for (i in 1:length(unique(df$Point))) {
      df_sub <- df[df$Point==i,]

      df_in2 <- organiseFun(df_sub, st.date, end.date)

      # write the data and header
      writeFun(outfile = outfile,
               df_write = df_in2, header = header, Flow = FALSE,
               np = length(unique(df$Point)), i = i,
               weight=weight)
    }
  }
}



# Demonstrate

# write observed_sub.txt
swatcup_ETformat(ET_Data, df_flow = NULL, date.format = "%Y-%m-%d",
                           "2006-01-01", "2011-12-31",
                 "inputdata/observed_sub.txt" ,
                 "inputdata/observed_sub.txt", 6, weight= 0.1)



# write observed.txt
swatcup_ETformat(ET_Data, df_flow = flowdata[,c(1,3)],
                 date.format = "%Y-%m-%d",
                 "2006-01-01", "2011-12-31",
                 "inputdata/observed.txt" ,
                 "inputdata/observed.txt", 14, Flow = TRUE,
                 weight = 0.1)

# write observed_rch.txt
swatcup_ETformat(flowdata[,c(1,3)],df_flow=NULL,
                 date.format = "%Y-%m-%d",
                 "2006-01-01", "2011-12-31",
                 "inputdata/observed_rch.txt" ,
                 "inputdata/observed_rch.txt", 6,
                 Flow = TRUE)

# Now test putting in weights relative to the size of the subcatchment
```

```r
subbasin_data <- read.csv("inputdata/subbasins_Cotter_alldata.csv")

# calculate weights from relative areas
f_w <- 0.1 # flow weight
ET_w <- subbasin_data$Area/sum(subbasin_data$Area)*(1-f_w)
 w_in <- c(f_w, ET_w)

 # now try to write the file
swatcup_ETformat(ET_Data, df_flow = flowdata[,c(1,3)],
                  date.format = "%Y-%m-%d",
                  "2006-01-01", "2011-12-31",
                  "inputdata/observed.txt" ,
                 "inputdata/observed.txt", 14,
                 Flow = TRUE, weight = w_in)
```