Course notes Advanced SWAT

Willem Vervoort

20-07-2017

Aims of this part of the course:

- Perform data extraction from satellite data into a usable format for Cotter catchment
- Perform analysis of the downloaded satellite data
- Develop and calibrate a GR4J model for the Cotter catchment in combination with local station data and satellite data
- Test how different data input can affect model performance

What is the issue?

Satellite data downloaded from typical sites is not directly useable in hydrological models. Here we use the example of MODIS data, which has a nice R package to make the data usable and take you through all the necessary steps to calibrate a Hydomad (GR4J) model.
Recapping:

- Download the data
- Load the data in R
- Prepare timeseries

Only after all that can we try to use it in a model to calibrate, so here we go!

Section 1: Downloading, extracting and analysis of the satellite ET data

I have set the working directory before, you might need to do this using `setwd()`. We first load a series of packages that we will need. The new one here is `MODISTools`. This is a specific package that allows you to download different MODIS products. Make sure you have the latest version installed.

```r
# Loading the packages required
require(MODISTools)
```

```
## Loading required package: MODISTools
```

```r
require(zoo)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
require(hydromad)
```

```
## Loading required package: hydromad

## Loading required package: lattice

## Loading required package: latticeExtra

## Loading required package: RColorBrewer

## Loading required package: polynom

## Loading required package: reshape
```

```r
require(tidyverse)
```

```
## Loading required package: tidyverse

## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## expand(): tidyr, reshape
## filter(): dplyr, stats
## lag():    dplyr, stats
## layer():  ggplot2, latticeExtra
## rename(): dplyr, reshape
```

Read-in point location within the catchment

This is for the Cotter catchment, and I have extracted all the centroids of the subcatchments using ARCGIS. There is probably an equivalent operation in QGIS.

```r
xy.loc <- read.csv("data/subbasins_Cotter.csv")
```

Step 1 is to decide on the time period which we want to download. Since this is time consuming, so as demonstration we will download data for two years and any random 5 location points within the catchment. I have downloaded all the relevant data already and made available via the shared drive. We need to figure out the name of the satellite product that we want to use, you can use `GetProducts()` and check what bands are available for the product

```r
GetProducts()
```

```
##       getproductsResponse
##  [1,] Character,0
##  [2,] "MCD12Q1"
##  [3,] "MCD12Q2"
##  [4,] "MCD43A1"
##  [5,] "MCD43A2"
##  [6,] "MCD43A4"
##  [7,] "MOD09A1"
##  [8,] "MOD11A2"
##  [9,] "MOD13Q1"
## [10,] "MOD15A2"
## [11,] "MOD15A2GFS"
## [12,] "MOD16A2"
## [13,] "MOD17A2_51"
## [14,] "MOD17A3"
## [15,] "MYD09A1"
## [16,] "MYD11A2"
## [17,] "MYD13Q1"
## [18,] "MYD15A2"
```

```
GetBands(Product="MOD16A2")
```

```
## [1] "ET_1km"    "LE_1km"    "PET_1km"   "PLE_1km"   "ET_QC_1km"
```

Create directory in which to save data, in this case we will be working first with Cotter data. You can do this in windows, but you can also use an R command. I am chekcing first if it exists

```
if (!dir.exists("MODIS_data_cotter")) {
  dir.create("MODIS_data_cotter")
}
```

Download only first 5 coordinates for 2 years

I am defining a coordinate dataframe (this follows the MODISTools Vignette) for the first 5 points. The next step is to actually pull the data from the server using the `MODISSubsets()` command

We look at this manually if we cannot get the server to respond I have downloaded the data that we will use.

```
coords <- data.frame(lat=xy.loc$Lat[1:5],
                     long=xy.loc$Lon[1:5],
                     start.date=rep(2005,5),
                     end.date=rep(2006,5),
                     ID=1:5)
## run MODISSubsets
MODISSubsets(LoadDat=coords, Products = "MOD16A2",
                Bands="ET_1km", StartDate=T,
             Size=c(0,0), SaveDir="MODIS_data_cotter")
```

```
## Files downloaded will be written to MODIS_data_cotter.
## Found 5 unique time-series to download.
## Getting subset for location 1 of 5...
## Getting subset for location 2 of 5...
## Getting subset for location 3 of 5...
## Getting subset for location 4 of 5...
## Getting subset for location 5 of 5...
## Full subset download complete. Writing the subset download file...
## Done! Check the subset download file for correct subset information and download messages.
```

Convert to timeseries

We need to read the downloaded MODIS ET data into a time-series format. For this we will create a function to read MODIS data into time-series format. This function will go to the directory we specify and find all the files that have the extension ".asc". (Or any other extension that you specify). In class we will have a look at what these files look like. In the below function, take a good look at the naming of the columns in `Store`, as we will be using this later.

```r
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 including the QA data
 # 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)
}

# and we can run this function and check the data
Cotter_ET <- MODIS_ts(MODISdir = "MODIS_data_cotter")
# Chk the data
head(Cotter_ET)
```

```
##   Year JDay   ET Point       Date
## 1 2005    1 36.2     1 2005-01-01
## 2 2005    9 33.3     1 2005-01-09
## 3 2005   17 36.5     1 2005-01-17
## 4 2005   25 40.8     1 2005-01-25
## 5 2005   33 28.9     1 2005-02-02
## 6 2005   41 31.1     1 2005-02-10
```

So this extracts for 5 data points the full time series over 2 years, but note that we only have data every 8 days. Note the column with Julian dates (Jdate). As we discussed, MODIS only supplies data every 8 days even though it is measured more frequently.

Visualise the data

We now want to visualize the 8 daily ET over two years. This is now quite straight forward, recognising thet the column Point indicates the different grid cells/pixels. Adjust colours to show this.

```r
plot(subset(Cotter_ET, Point == 1)[,c("Date","ET")],
     type = "b", pch = 16, col = "blue",
     xlab = "8-day time series", ylab = "8-day ET in mm", ylim=c(0,50))
points(subset(Cotter_ET, Point == 2)[,c("Date","ET")],
     type = "b", pch = 16, col = "red")
points(subset(Cotter_ET, Point == 3)[,c("Date","ET")],
       type = "b", pch = 16, col = "grey60")
points(subset(Cotter_ET, Point == 4)[,c("Date","ET")],
       type = "b", pch = 16, col = "black")
points(subset(Cotter_ET, Point == 5)[,c("Date","ET")],
       type = "b", pch = 16, col = "green")
```

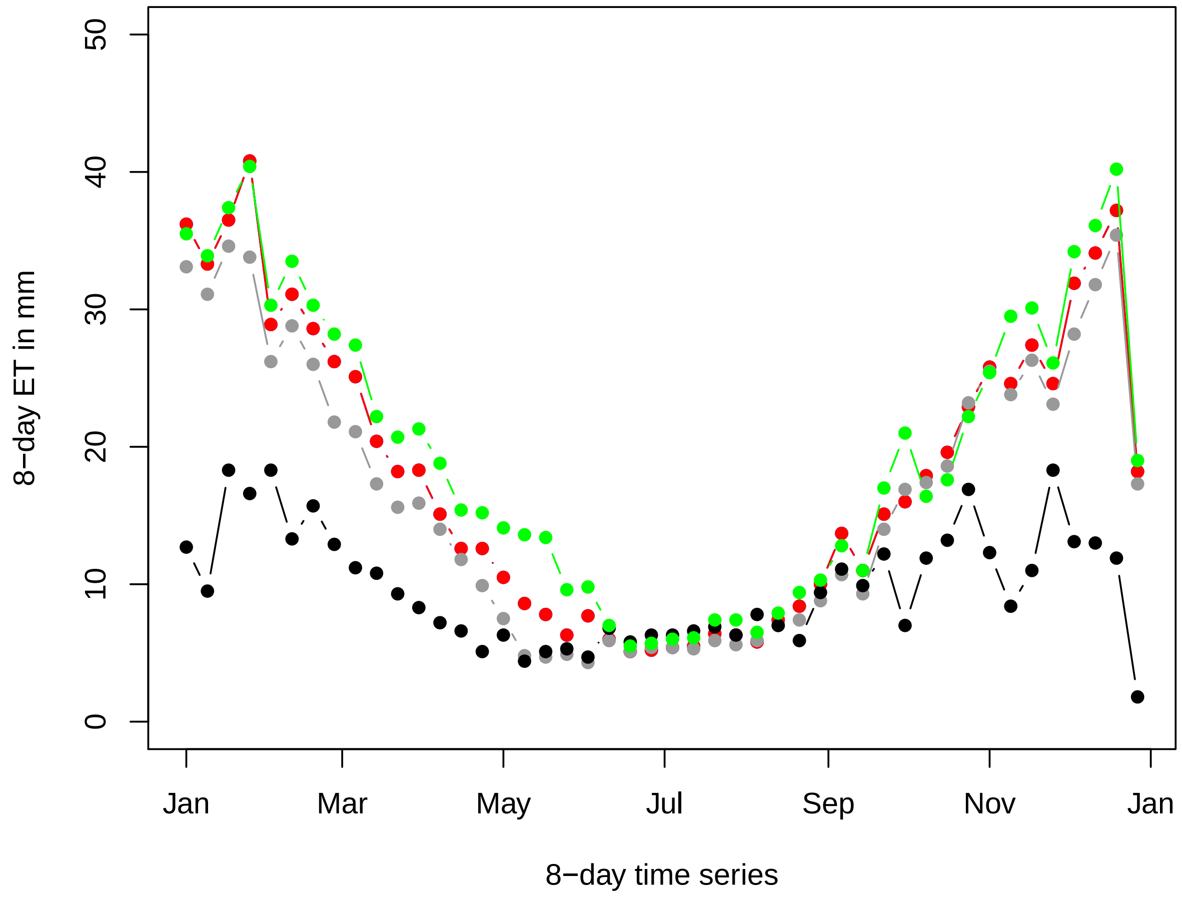

Figure 1: Timeseries of ET at the different points

We could also make a simple histogram across all the points extracted.

```r
hist(Cotter_ET$ET, xlab="MODIS ET")
```

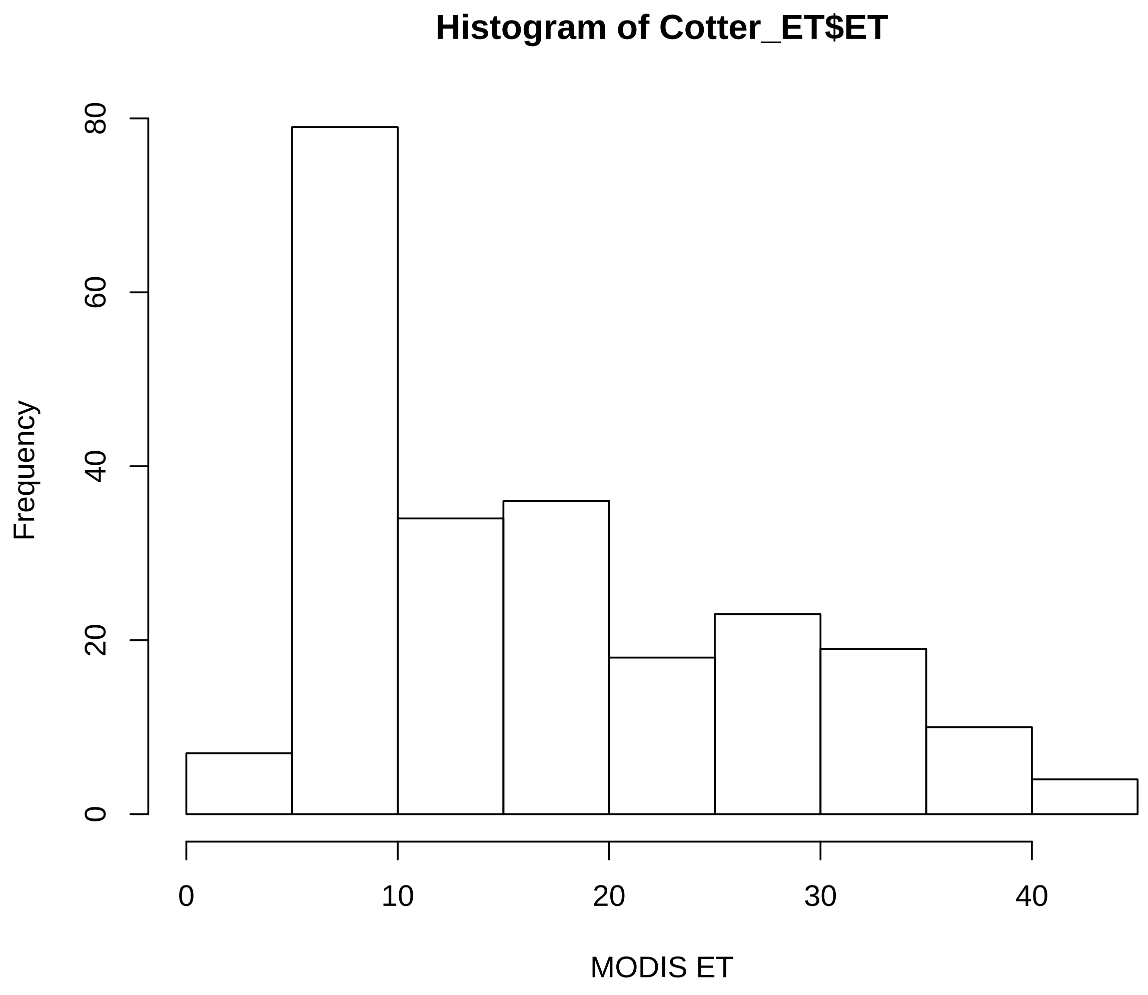

Figure 2: Histogram of MODIS ET across the different points

Aggregating to mean ET for the catchment

We can also agrregate the ET data and obtain average ET for the entire catchment (Note that we are using only 5 points in this case so this will actually not be representative of the catchment). We will use the function `aggregate()` again

```r
ET.mean <- aggregate(Cotter_ET[,3],list(JDay=Cotter_ET$JDay,
                          Year=Cotter_ET$Year), FUN = mean,na.rm=T)
# Check the data
head (ET.mean)
```

```
##   JDay Year     x
## 1    1 2005 30.74
## 2    9 2005 28.22
## 3   17 2005 32.66
## 4   25 2005 34.48
## 5   33 2005 26.52
## 6   41 2005 27.56
```

```r
# Create a date column from Jdate
ET.mean$Date <- as.Date(paste(ET.mean$Year,ET.mean$JDay,
                              sep = "/"), "%Y/%j")
# Now, make a plot of the mean 8 daily ET
plot(ET.mean$Date,ET.mean$x, xlab = "Time (8-daily)",
     ylab = "Basin average ET")
```

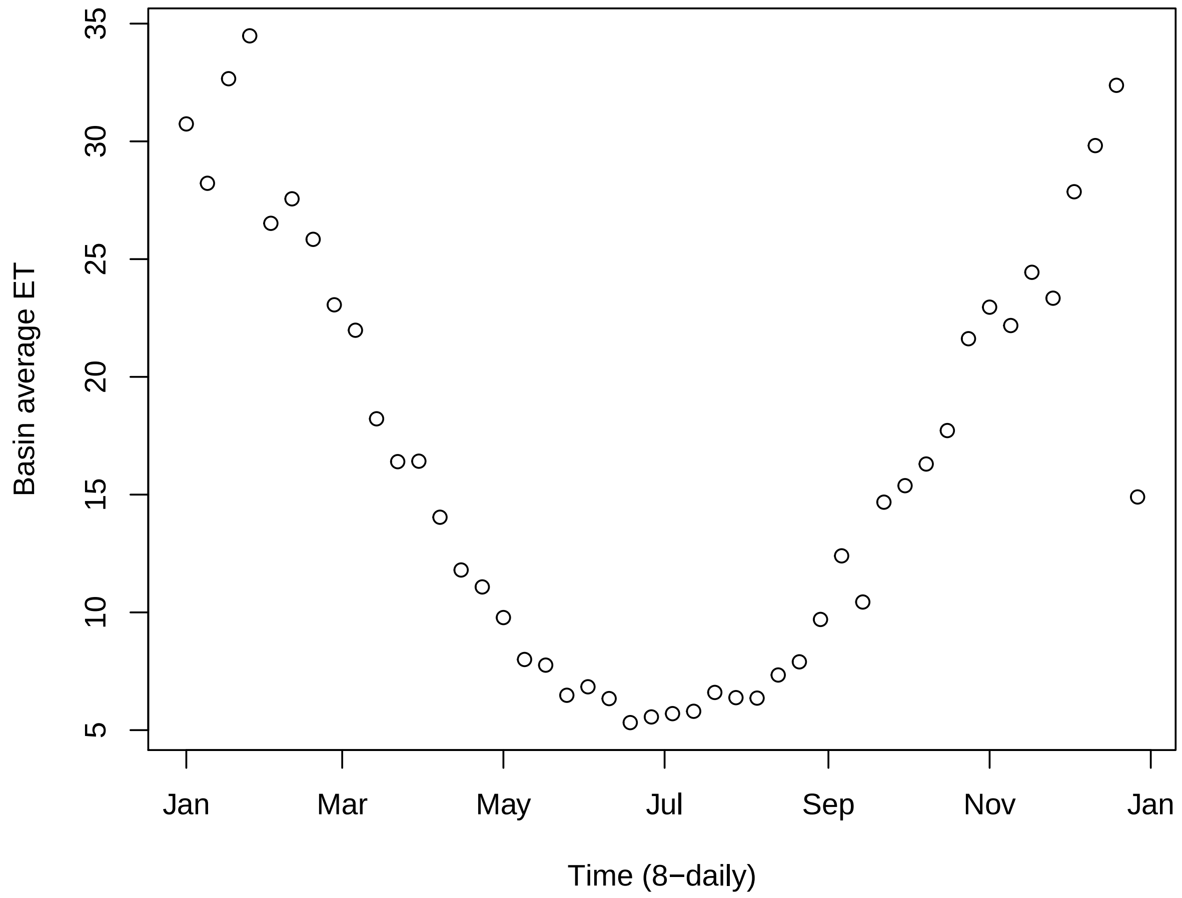

Figure 3: Average ET for the five points

Task 1: Repeat the analysis for Santa Lucia catchment, pick any two years between 2001 and 2014 and select any random 5 points within the Santa Lucia catchment.

Section 2: Develop and Calibrate a model with Satellite ET data for the Cotter Catchment

Load ET data

In this case we are going to use data that I have downloaded before for all the pixels in the catchment. This takes to long for the course, so I have made a separate file with the data for all the pixels in the catchment. This file is on the shared drive

```r
# Read-in the ET data already downloaded for the Cotter catchment
Cotter_ET <- MODIS_ts(MODISdir = "MODIS/cotter")

# In here, first convert to wide format, use tidyverse
Cotter_ET_w <- spread(Cotter_ET, key=Point, value=ET)

# Converting date to date format
Cotter_ET_w$Date <- as.Date(Cotter_ET_w$Date)
names(Cotter_ET_w)
```

```
##  [1] "Year" "JDay" "Date" "1"    "2"    "3"    "4"    "5"    "6"    "7"
## [11] "8"    "9"    "10"   "11"   "12"   "13"   "14"   "15"   "16"   "17"
## [21] "18"   "19"   "20"   "21"   "22"   "23"   "24"   "25"
```

```r
# Calculating the catchment average ET
Cotter_avgET  <- data.frame(Date = Cotter_ET_w$Date,
                 ETa = apply(Cotter_ET_w[,4:ncol(Cotter_ET_w)],
                             1, mean, na.rm = T))

# Converting to a zoo format to work with hydromad
Cotter_MODISET <- zoo(Cotter_avgET$ETa,
                      order.by=Cotter_avgET$Date)
# we can plot to see the data
plot(Cotter_MODISET, xlab="Date",
     ylab="8-day summed MODIS ET (mm)")
```

Define and calibrate GR4J

We will again use the "GR4J" model in hydromad package. The approach will be similar to week 3 but we will also test calibration with satellite derived ET. This basically repeats some of the analysis Vervoort et al (2014) did with IHACRES. To make things easier, Joseph Guillaume and I have written a series of functions in R to help do this in hydromad. These will be incorporated in hydromad in the future, but at the moment, it just exists as separate files. Load these first using `source()`, and they are stored in the "functions" folder in my system, but they are also on the U drive and on the LMS.

```r
source("functions/leapfun.R")
source("functions/ETa.merge.R")
source("functions/plot.ET.R")
source("functions/ETfit.objectives.R")
```

Then we need to load the data and merge this with the ET data. The `ETa.merge()` function needs to be used rather than the classical `zoo.merge()` because the MODIS ETa data is only every 8 days and this needs to be merged with the daily flow, rainfall and maxT data.

I have prepared a full hydromad dataset for the Cotter catchment earlier. This is `Cotter.Rdata`.

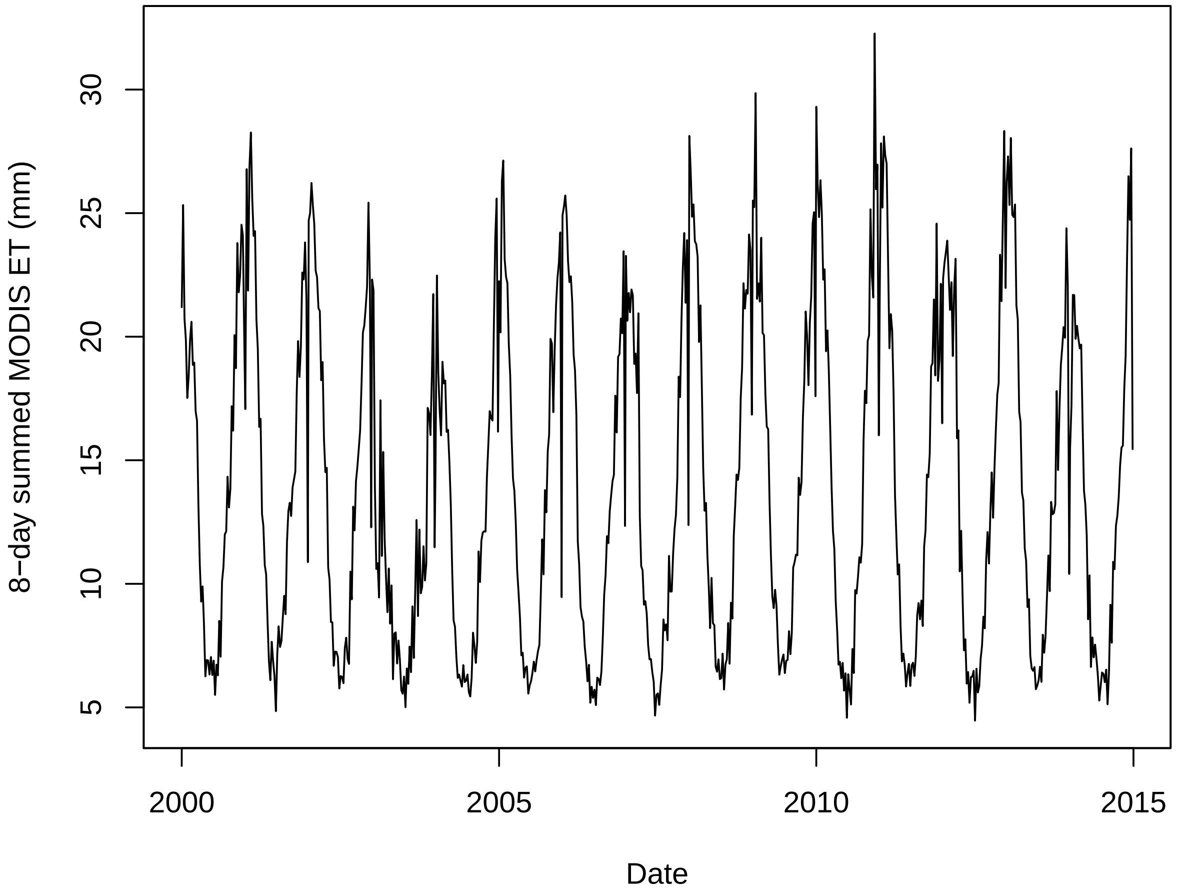

Figure 4: Cotter Catchment average MODIS ET timeseries

```r
load("data/Cotter.Rdata") # this is the flow data
Cotter$Q <- convertFlow(Cotter$Q, from="ML", area.km2=250)

# discard the data before 2000
Cotter <- window(Cotter, start="2000-01-01")

Cotter_Sat <- ETa.merge(Flowdata=Cotter,ETdata=Cotter_MODISET)
# Make a plot
xyplot(Cotter_Sat)
```

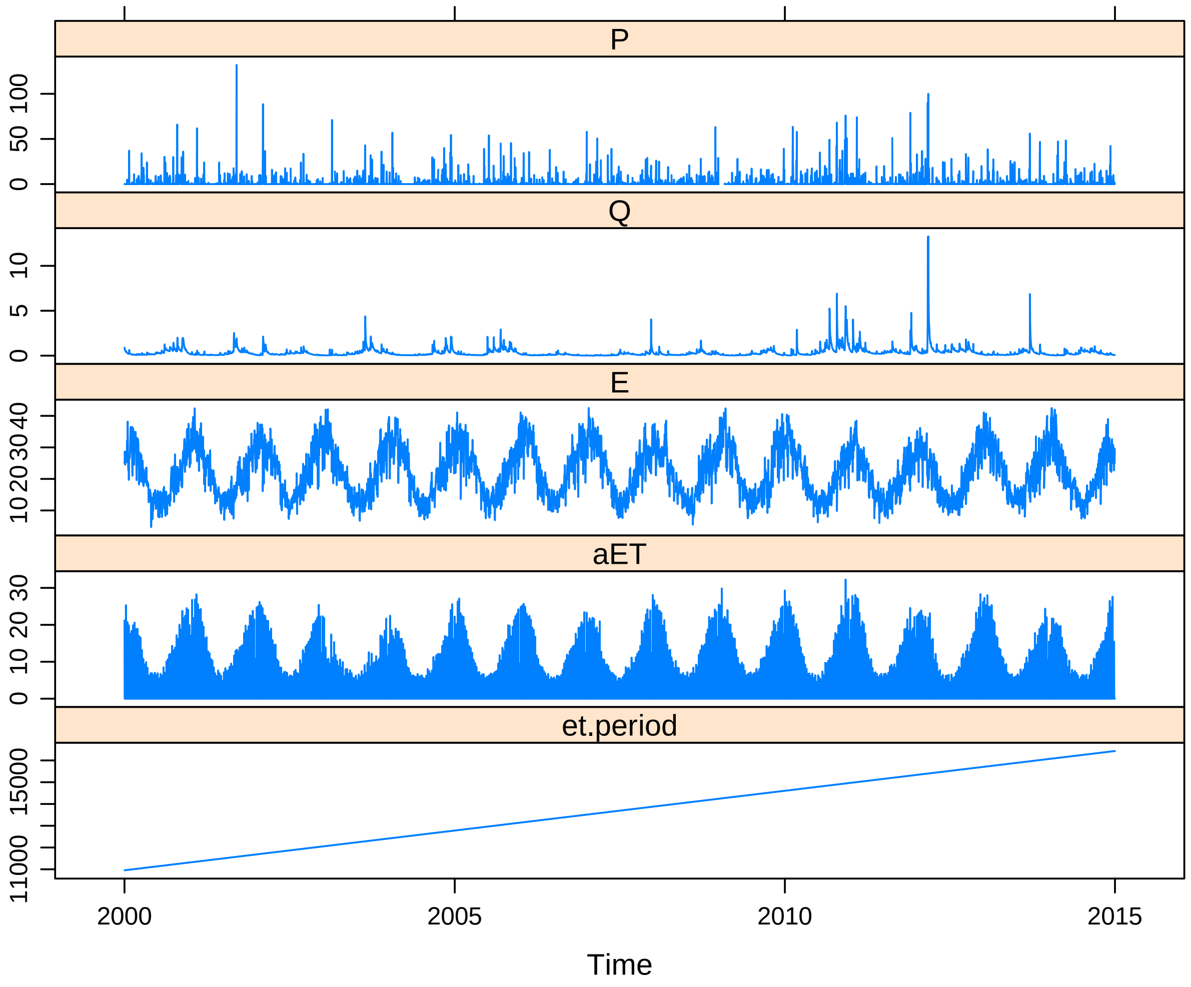

Figure 5: Merged dataset for the Cotter Catchment

Calibrating GR4J

Now we will first setup and calibrate the model with station data as we have done in Week3 prac then we will calibrate the same model in combination with the satellite ET data. This will enable us to see how model estimation is influenced by adding extra observable flux. Subset the data for the calibration period and load

it into model.
First, we will run the model without the satellite ET data

```r
# Data period for calibration
data_cal <- window(Cotter, start = "2005-01-01",end = "2010-12-31")

# Data for validation period
data_val <- window(Cotter, start = "2011-01-01",end = "2014-12-31")

# Define the model, important to define return_state=T
Cotter_mod <- hydromad(DATA=data_cal,
                   sma = "gr4j", routing = "gr4jrouting",
                   x1 = c(500,2500), x2 = c(-30,20), x3 = c(5,500),
                   x4 = c(0.5,10), etmult=c(0.01,0.5),
                   return_state=TRUE)

# Using shuffled complex evolution algorithm for fitting
Cotter_fit<- fitBySCE(Cotter_mod,
                     objective= hmadstat("r.squared"))

# Extract the coefficients and the summary
summary(Cotter_fit)
```

```
##
## Call:
## hydromad(DATA = data_cal, x1 = 1835.04, x2 = -3.91653, x3 = 74.241,
##     x4 = 1.14383, etmult = 0.0700084, return_state = TRUE, sma = "gr4j",
##     routing = "gr4jrouting")
##
## Time steps: 2091 (55 missing).
## Runoff ratio (Q/P): (0.304 / 1.802) = 0.1687
## rel bias: 0.007694
## r squared: 0.7012
## r sq sqrt: 0.5921
## r sq log: 0.4399
##
## For definitions see ?hydromad.stats
```

```r
# plot
xyplot(Cotter_fit, with.P = TRUE)
```

```
## Warning in formals(fun): argument is not a function
```

Recalibrate with MODIS ET data

Now calibrate the model with the satellite ET data. This requires a few little tricks, one of these is the use a specific objective function (as we discussed in the lecture) that combines the Eta and the Q function. There is a “weighting” factor in this function, which we initially will just set to 0.5 (equal weighting between ETa and Q).

```r
# remake the calibration data
data_modis_cal <- window(Cotter_Sat, start = "2005-01-01",end = "2010-12-31")

# also make the validation data
data_modis_val <- window(Cotter_Sat, start = "2011-01-01",end = "2014-12-31")
```

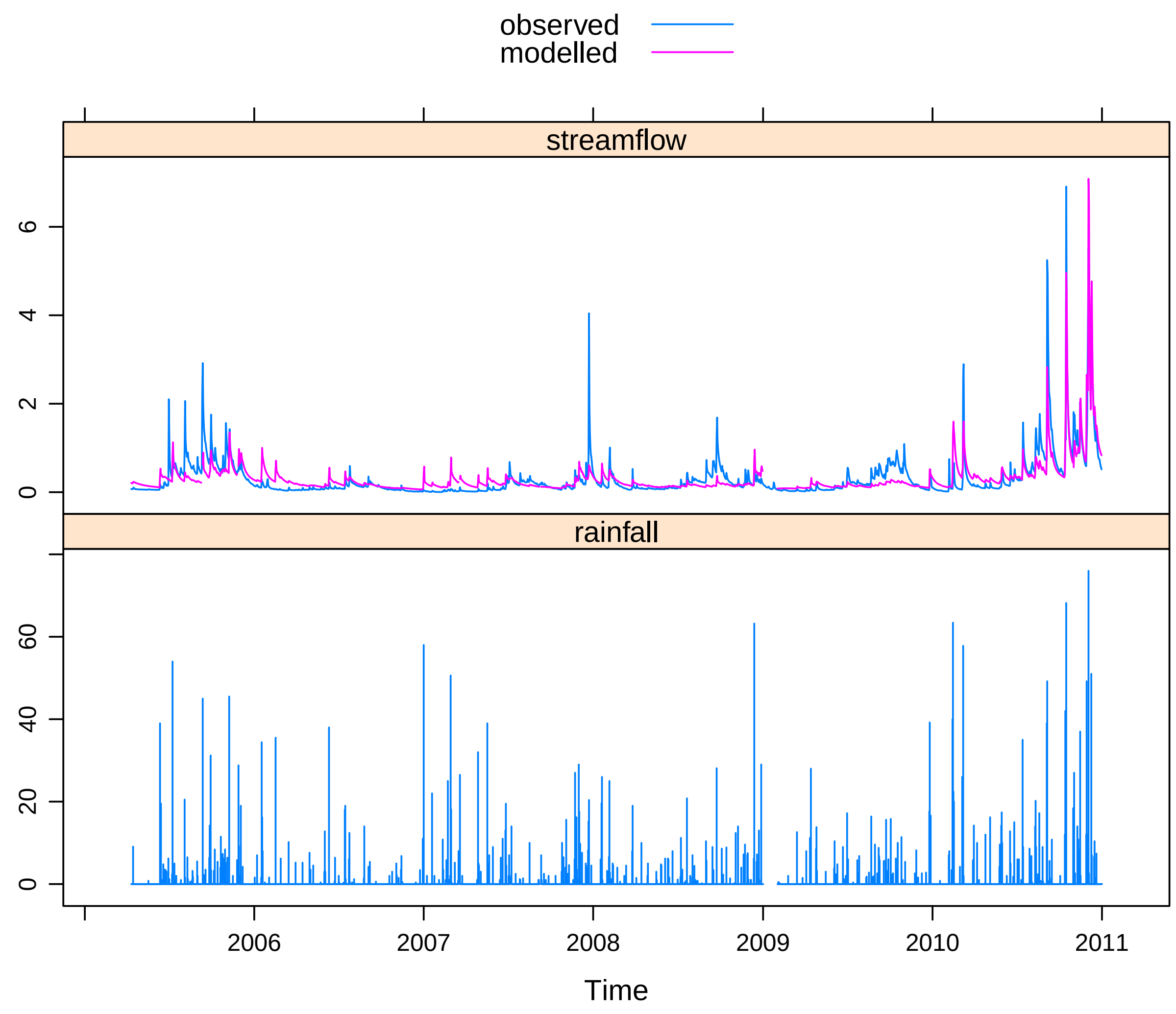

Figure 6: GR4J model fitted with local station data

```r
# Because we have rebuilt data.cal, redefine the model
Cotter_mod_Modis <- hydromad(DATA=data_modis_cal,
                    sma = "gr4j", routing = "gr4jrouting",
                    x1 = c(500,3000), x2 = c(-30,20),
                    x3 = c(5,500), x4 = c(0.5,10),
                    etmult=c(0.01,0.5),
                    return_state=TRUE)


# fit both ET and Q using special objective function
Cotter_Fit_Modis <- fitBySCE(Cotter_mod_Modis,
                        objective=~hmadstat("JointQandET")(Q,X,w=0.5,
                               DATA=DATA,model=model,objf = hmadstat("viney")))

# check the model fit
summary(Cotter_Fit_Modis)
```

```
##
## Call:
## hydromad(DATA = data_modis_cal, x1 = 3000, x2 = 0.706379, x3 = 14.7814,
##     x4 = 1.11577, etmult = 0.156018, return_state = TRUE, sma = "gr4j",
##     routing = "gr4jrouting")
##
## Time steps: 2091 (55 missing).
## Runoff ratio (Q/P): (0.304 / 1.802) = 0.1687
## rel bias: -0.0005019
## r squared: 0.443
## r sq sqrt: 0.3303
## r sq log: 0.1718
##
## For definitions see ?hydromad.stats
```

```r
##
# Plotting the results
xyplot(Cotter_Fit_Modis, with.P = TRUE)
```

```
## Warning in formals(fun): argument is not a function
```

We can now also look at how it predicts actual ET and how this compares to the observed ET. Use the `plot.ET()` function.

```r
plot.ET(caldata=data_modis_cal,Cotter_Fit_Modis)
```

Model comparison

We can now compare the model performance and whether this has chnaged for the two calibration methods.

```r
coef(Cotter_fit)
```

```
##            x2            x3            x4            x1        etmult
##   -3.91652615   74.24095884    1.14383021 1835.04038280    0.07000844
```

```r
coef(Cotter_Fit_Modis)
```

```
##           x2           x3           x4           x1       etmult
##    0.7063793   14.7814329    1.1157739 2999.9999999    0.1560180
```

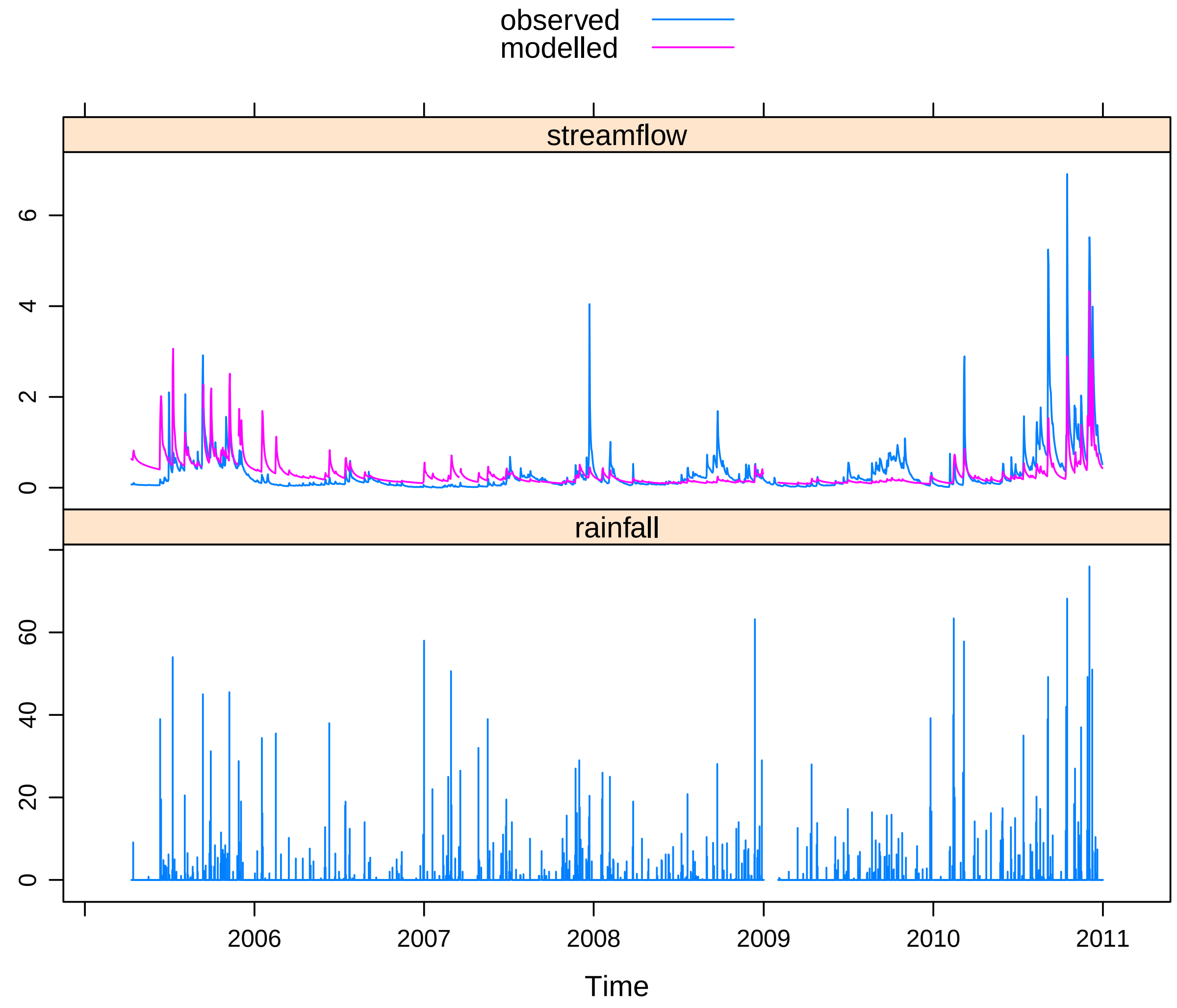

Figure 7: GR4J Model fitted with local station and Satellite ET data

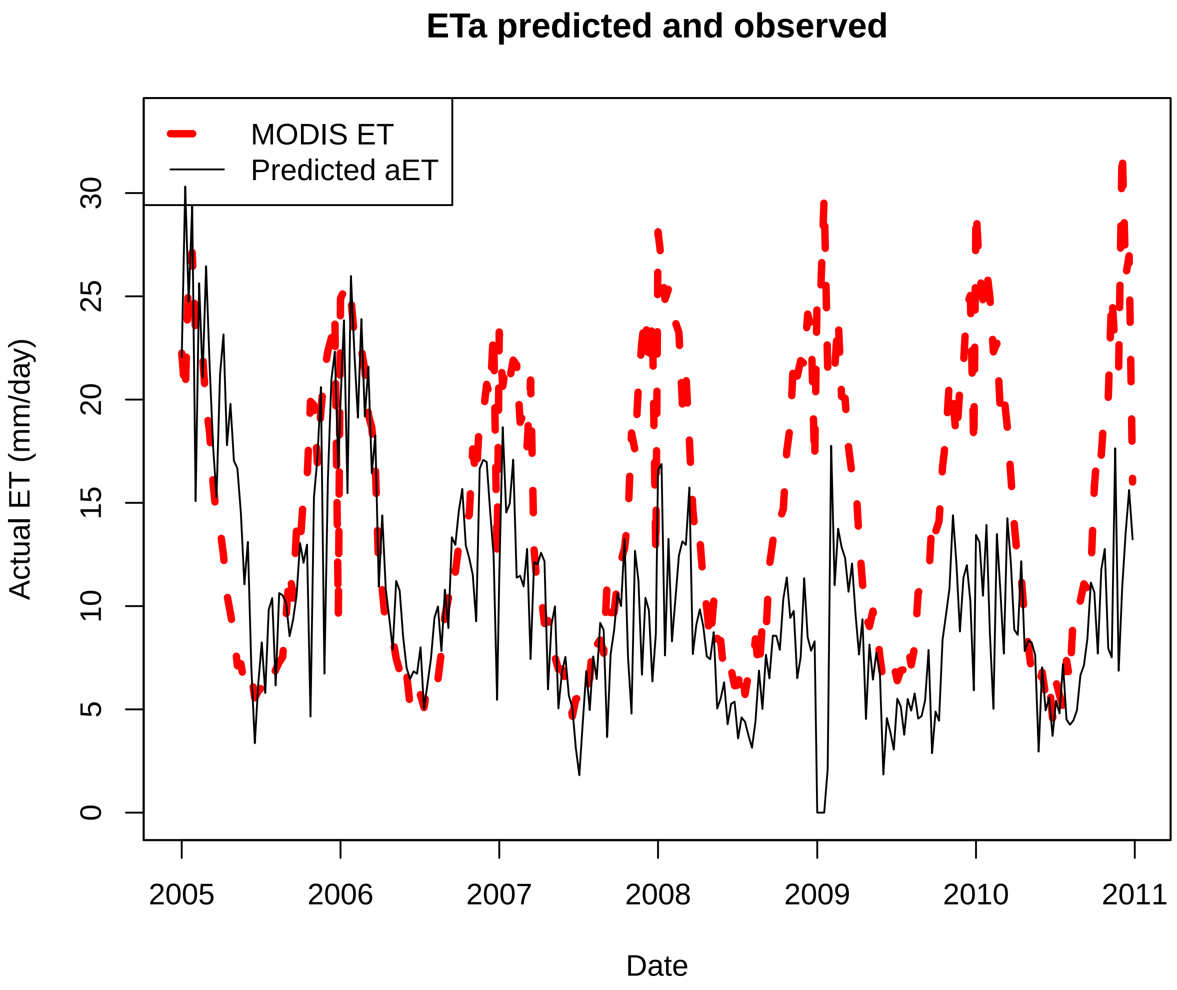

Figure 8: Plot showing observed vs predicted actual ET

```r
objFunVal(Cotter_fit)
```

```
## [1] 0.6416808
```

```r
objFunVal(Cotter_Fit_Modis)
```

```
## [1] 0.348592
```

And then do the same for validation and make a runlist.

```r
# updating the model data for the validation
sim_val <- update(Cotter_fit, newdata = data_val)
sim_val_modis <- update(Cotter_Fit_Modis, newdata = data_modis_val)

# runlist
allMods <- runlist(calibration=Cotter_fit, validation=sim_val,
                   calibrationET=Cotter_Fit_Modis,
                   validationET= sim_val_modis)

# Get the summary results
round(summary(allMods),2)
```

```
##                rel.bias r.squared r.sq.sqrt r.sq.log
## calibration        0.01      0.70      0.59     0.44
## validation         0.02      0.80      0.62     0.47
## calibrationET      0.00      0.44      0.33     0.17
## validationET       0.24      0.64      0.38     0.22
```

Comment on the model performance:

- Does fitting on ET help?
- What other comparisons could we make to test the behaviour?
- Can we improve the model performance?
- How could we manipulate the emphasis of the calibration on ET relative to Q?

Task 2 (optional):

- Perform the model calibration and validation for Santa Lucia catchment and explain your results with graphs and statistical summary.
- How does use of satellite ET data benefit in model calibration?