SMART code development notes

Yixin, May 2018

1. Changes to inputs:

* “input\_dataset”:
  + A “.mat” file containing all input precipitation and SM data
  + Specifically, the data file needs to contain the following variables with exactly the specified variable name:
    - “prec\_orig”:
      * Original observed precipitation data
      * Dimension: [npixel, ntime]
    - “prec\_true”:
      * True precipitation
      * This variable is not really used in any calculation; but is in the original code so is kept
      * Dimension: [npixel, ntime]
    - “prec\_for\_tuning\_lambda”:
      * Precipitation data for tuning lambda, if specified
      * Dimension: [npixel, ntime]
    - “sm\_ascend”:
      * Ascending soil moisture
      * Dimension: [npixel, ntime]
    - “sm\_descend”:
      * Descending soil moisture
      * Dimension: [npixel, ntime]
    - “sm\_error”:
      * Soil moisture observation error standard deviation, same unit as sm\_ascend and sm\_descend
      * Dimension: [npixel, ntime]
* “start\_time” and “end\_time”:
  + The first and last time step of SMART run
  + Format: “YYYY-MM-DD HH-MM”
  + Example: “1980-01-01 00-00”, “1980-12-31-21-00”
* “time\_step”:
  + Time step length in hour for all input data (precipitation, SM); SM observations are again assumed to be at the end of each corresponding precipitation time step
* “window\_size”:
  + Number of timesteps in a rainfall-correction window to aggregate
  + For example, if “time\_step” = 3 and want to aggregate to 3-day windows, then “window\_size” = 24
* “API\_range” (only used if API is varying seasonally):
  + Was hardcoded in the original code; changed to be an input variable
* “phi”:
  + Indicates the level of autocorrelation of precipitation multiplier
  + phi is the parameter for the AR(1) process that generates the underlying normally distributed variable for multiplier; phi=0 for white multiplier; phi=1 for completely auto-correlated multiplier
* “if\_rescale”:
  + Whether to rescale input SM meas (options: 0 or 1)
  + If 1, then rescale as normal; if 0, reset “sm\_observed\_trans” and “R\_API” to original unscaled values
* “lambda\_tuning\_target”:
  + If lambda\_flag == 999 (i.e., tuning lambda), what target objective function to use
  + Options:
    - “rmse” - the original SMART objective function
    - “corrcoef” – tune to maximum correlation coefficient
* “sep\_sm\_orbit”:
  + Options: 0 or 1
  + 1 for separately rescaling ascending and descending SM (this only makes sense for subdaily run); 0 for combining ascending and descending soil moisture products and SM observations appearing on the same timestep will be averaged
* Note on input data and timestep consistency:
  + All the input precipitation and SM data are assumed to start from “start\_time” and at “time\_step”;
  + All input data must be *at least* the length of whatever is specified by “start\_time”, “end\_time” and “time\_step”;
  + Input data can be longer than “start\_time” to “end\_time”; the input data after “end\_time” will be ignored
* “API\_mean” and “bb”:
  + Now both of these two inputs can either be a number (as before) or the path of the “.mat” file that contains an input 1D vector. If the former, spatially-constant parameter value will be used; if the latter, the input vector specifies the parameter value for each pixel.
  + Specifically, if input a vector of variable values, the variable name for API\_mean should be “API\_COEFF\_tuned”, and the variable name for bb should be “bb\_tuned”. Both variable dimensions should be [npixelnum].

2. I commented out the “spatial masking” if condition – can keep back in

3. I moved the code for rescaling into a separate function (“rescale()”). now in the “analysis()” function, if sep\_sm\_orbit = 1, rescale ascending & descending SM completely separately calling rescale(), then merge the rescaled products together into one time series; if sep\_sm\_orbit = 0, first merge ascending & descending products together (average if appearing on the same timestep), then apply rescaling.

4. In gamma-coefficient calculation:

* “DOY” (only used if API is varying seasonally) now considers non-daily timestep. After the change, DOY(k) is still the day of year of the kth timestep.

5. When rescaling SM and SM error, the original code does one of the following:

1) Uses API values of all time steps to calculate mean & standard deviation. This is the case for:

* + Seasonal 2nd-moment-matching (transform\_flag = 2);
  + Long-term 2nd-moment-matching (transform\_flag = 3);
  + Rescaling SM error to R;
  + Seasonal 2nd-moment-matching with seasonal std. (transform\_flag = 5)

2) Uses API values at SM-obs-available timesteps. This is the case for:

* + Long-term cdf-matching (transform\_flag = 1);
  + Seasonal cdf-matching (transform\_flag = 4)

Now, I changed everything to follow *the second approach* (i.e., use API values at SM-obs-available timesteps to calculate all mean & standard deviation values). This is done by the following specific changes:

* “API\_DOY” was originally calculated based on API values at all timesteps. Now it is calculated based on only timesteps when SM observations are available;
* Changed the section of calculating long-term mean and standard deviation to use SM-obs-available timesteps only

6. Enabled ensemble rainfall correction:

* *Each ensemble member of increment is applied to correct the corresponding perturbed rainfall ensemble member*

7. Rescale corrected rainfall in the end to have the same long-term mean as observation. This rescaling is applied to:

* The deterministic corrected rainfall
* Each ensemble member of corrected rainfall individually (for ensemble filter options)

8. Now the code saves ensemble increments and rainfall results

9. I commented out the validation calculation in the end

10. Change the identification of missing sm\_obs time points from “-1” to “NAN”

11. When filter\_flag = 6 (EnKS), set an upper limit for the back-fill period (currently set to 10 days). This is to prevent back filling for too long when there is a big gap in the SM measurement data

12. Originally in SMART:

1. The independent rainfall (for tuning lambda) is first bias-corrected toward original rainfall, before tuning lambda
2. Then, during tuning lambda, RMSE calculation does not consider the final rescaling (i.e., optimize not-bias-corrected corrected rainfall v.s. bias-corrected independent rainfall)
3. Finally, the corrected rainfall is bias-corrected toward original rainfall

The above steps result in inconsistent RMSE calculation post-SMART (i.e., bias-corrected rainfall v.s. not-bias-corrected independent rainfall) compared to optimal lambda tuned by SMART.

Now I updated the SMART code so that lambda tuning procedure is using exactly the same objective function (RMSE) as we will calculate after SMART. Specifically:

* Delete Step 1 above (i.e., do not bias-correct the independent rainfall in SMART before tuning lambda)
* Modify Step 2 above (i.e., in “fraction\_tune()”, include bias correction when calculating RMSE)