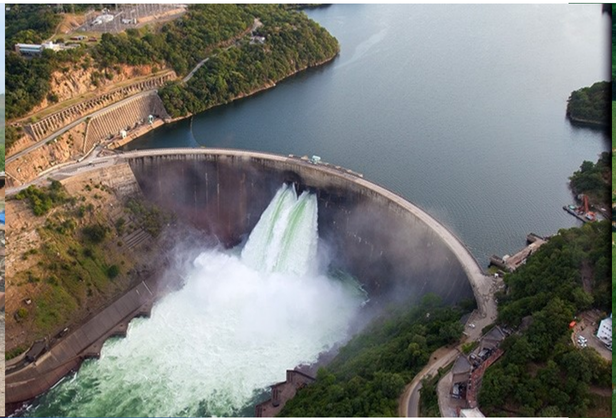
Exercise 9: Calibration

Peter Burek, Mikhail Smilovic, Luca Guillaumot
International Institute for Applied Systems Analysis
Research Scholars at
Water Program





Calibration

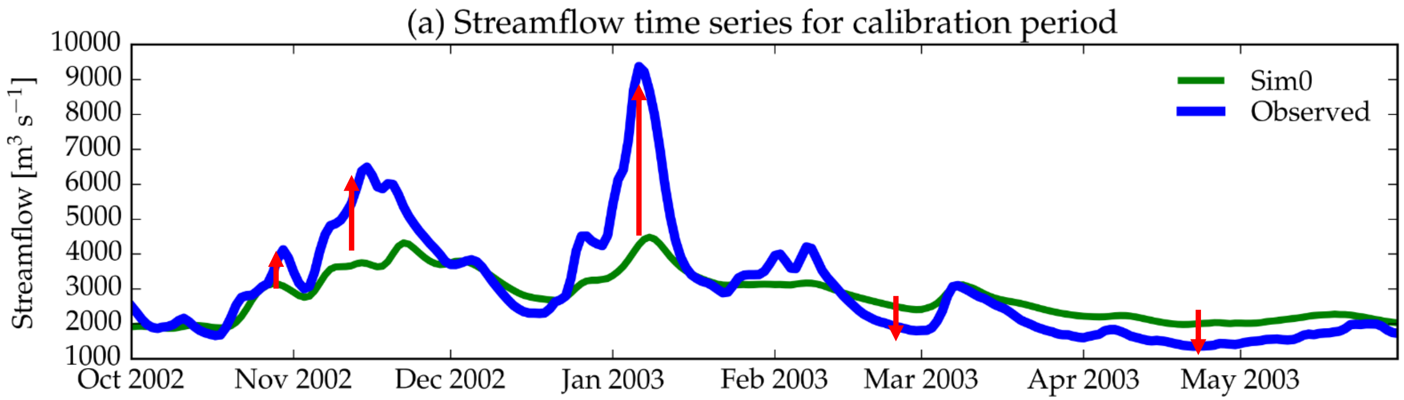
0. What you need
1. Introduction into calibration
2. Doing a test run
3. A fast calibration
4. Visualization of the results

0. What you need

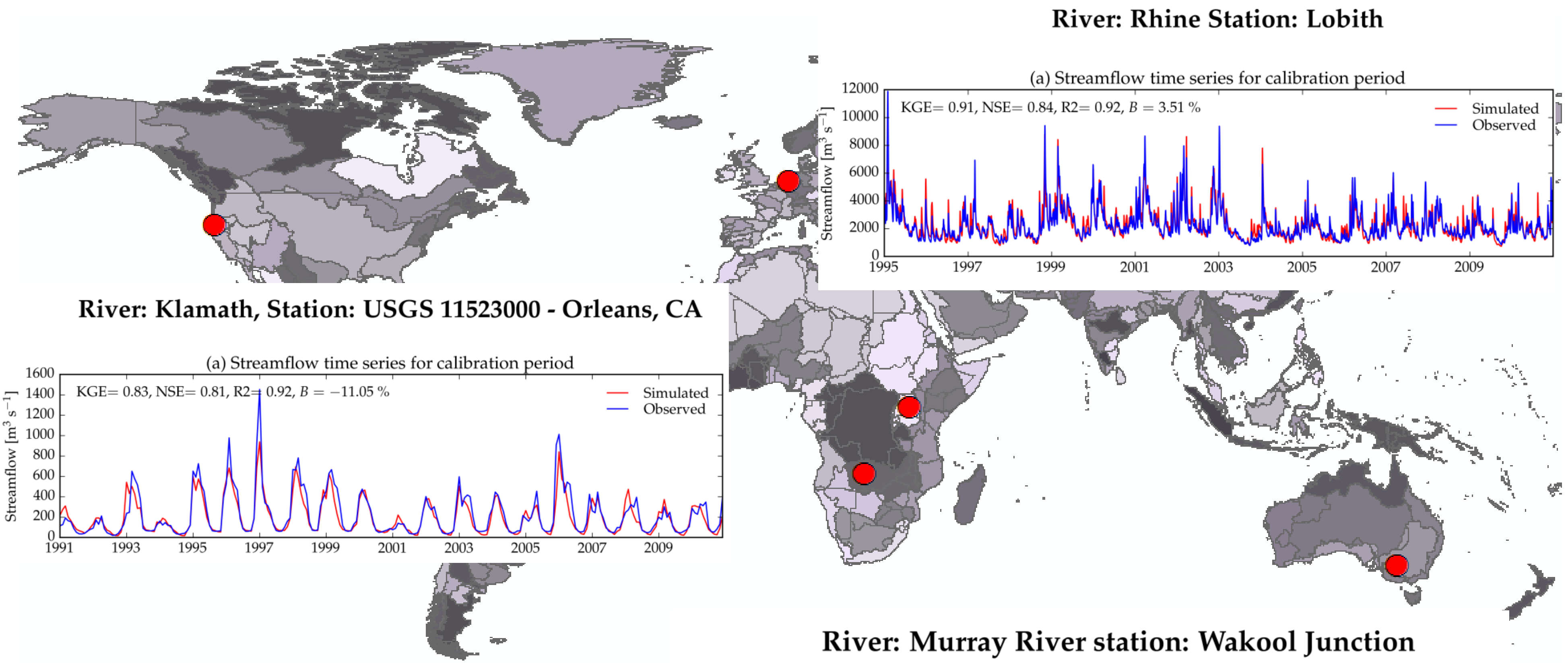
- Python 3.7 or 3.8 CWatM running
- additional libraries: Deap
  pip install deap or conda install deap
- and the libraries: pandas and matplotlib

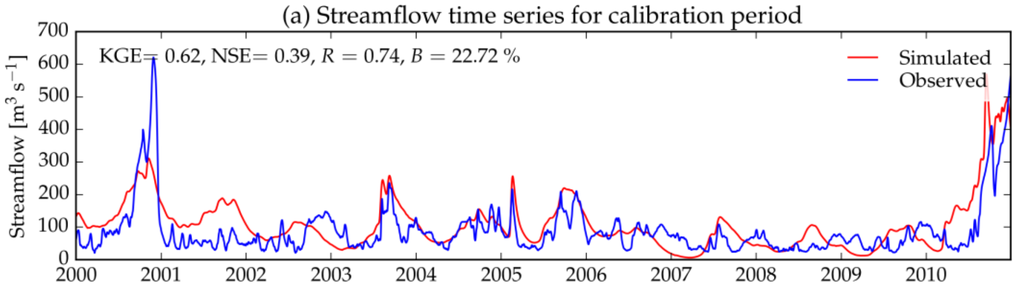
Calibration

Finding a parameter set which represents the observed discharge data



Calibration





Calibration:
- Daily run of 12 to 20 years
- Compared to daily or monthly observed discharge
- Objective function: KGE'

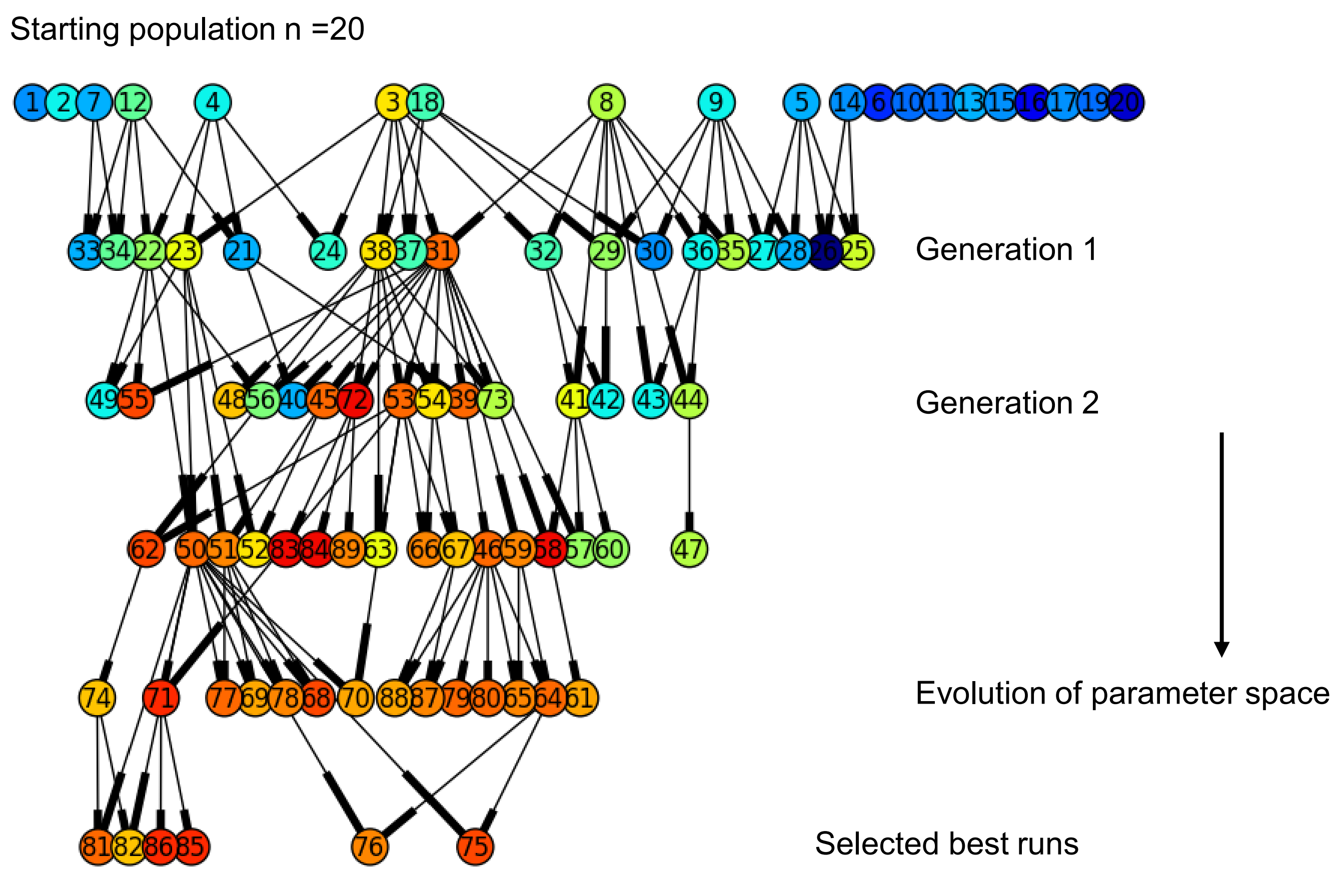
KGE': modified Kling-Gupta efficiency
NSE: Nash-Sutcliffe Efficiency
R2: Correlation coefficient
B: Bias

Calibration

Calibration is using an evolutionary computation framework in Python called DEAP (Fortin et al., 2012).

DEAP implemented the evolutionary algorithm NSGA-II (Deb et al., 2002)

It is used here as single objective optimization.



Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné, "DEAP: Evolutionary Algorithms Made Easy", Journal of Machine Learning Research, vol. 13, pp. 2171-2175

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE Transactions on Evolutionary Computation 6(2): 182-197.

Calibration

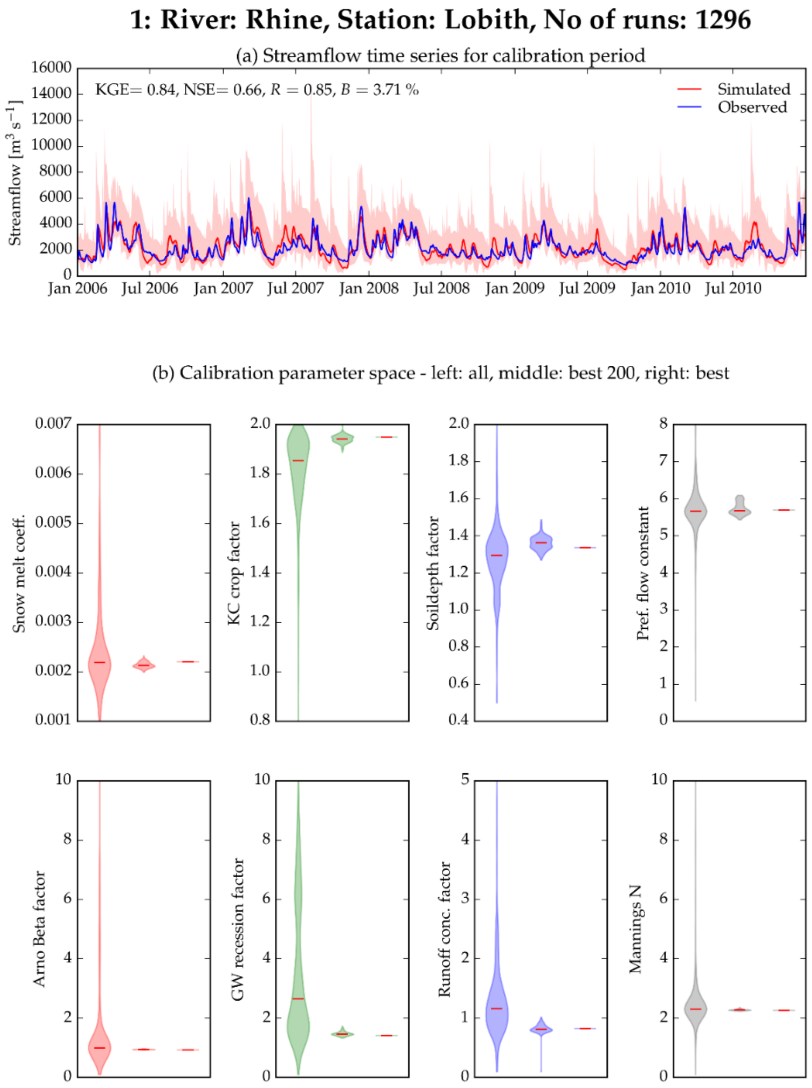
Discharge:
Daily (or monthly) pairs of observed and simulated discharge at gauging stations

Objective function:
Modified version of the Kling-Gupta Efficiency (Kling et al., 2012),

$$KGE' = 1 - \sqrt{(r-1)^2 + (\beta-1)^2 + (\gamma-1)^2}$$

$$\text{where: } \beta = \frac{\mu_s}{\mu_o} \text{ and } \gamma = \frac{CV_s}{CV_o} = \frac{\sigma_s/\mu_s}{\sigma_o/\mu_o}$$

Where:
r as the correlation coefficient between simulated and observed discharge (dimensionless), β as the bias ratio (dimensionless) and γ as the variability ratio. CV is the coefficient of variation, μ is the mean streamflow [$m^3\ s^{-1}$] and σ is the standard deviation of the streamflow [$m^3\ s^{-1}$]. KGE’, r, β and γ have their optimum at unity.



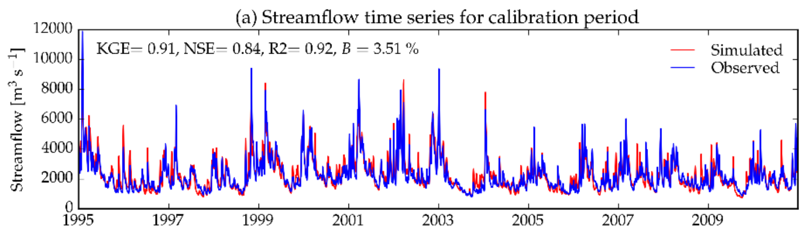
Parameter space for 8 parameter

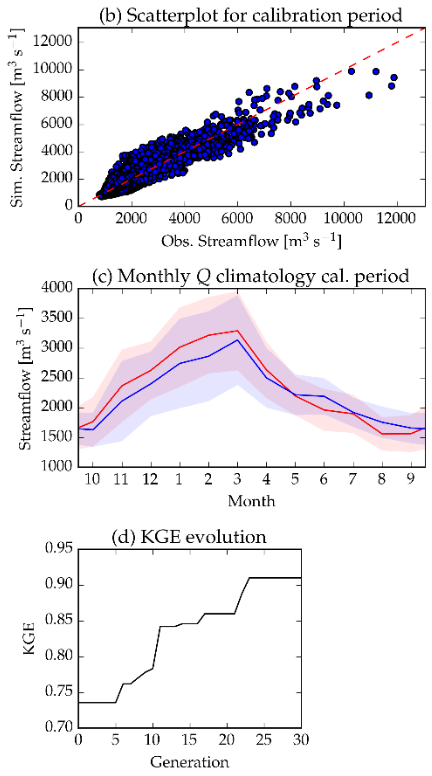
Gupta, H. V., H. Kling, K. K. Yilmaz and G. F. Martinez (2009). “Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling.” Journal of Hydrology 377(1-2): 80-91
Kling, H., M. Fuchs and M. Paulin (2012). “Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios.” Journal o Hydrology 424-425: 264-277

Calibration

Calibration 1995 - 2010

River: Rhine Station: Lobith





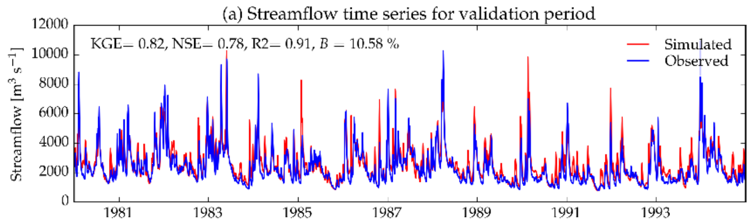
	Obs.	Sim.
KGE		0.910
NS		0.840
NSlog		0.795
R2		0.923
Bias		3.51%
RMSE		450
MAE		333
Mean	2258	2337
Min	788	729
5 %	1136	1046
50 %	1956	2076
95 %	4387	4542
99 %	6451	6163
Max	11885	9889

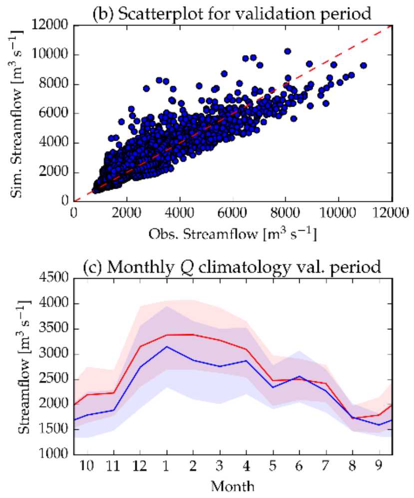
Calibration:
- Daily run of 15 years
- Compared to daily observed discharge
- Objective function: KGE’

KGE’: modified Kling-Gupta efficiency
NSE: Nash-Sutcliffe Efficiency
R2: Correlation coefficient
B: Bias

Validation 1980-1995

River: Rhine Station: Lobith





	Obs.	Sim.
KGE		0.818
NS		0.780
NSlog		0.768
R2		0.908
Bias		10.58%
RMSE		592
MAE		419
Mean	2378	2629
Min	794	768
5 %	1100	1211
50 %	2015	2321
95 %	4858	4998
99 %	7257	6902
Max	10940	10263

Calibration of river discharge Rhine / Lobith

Calibration

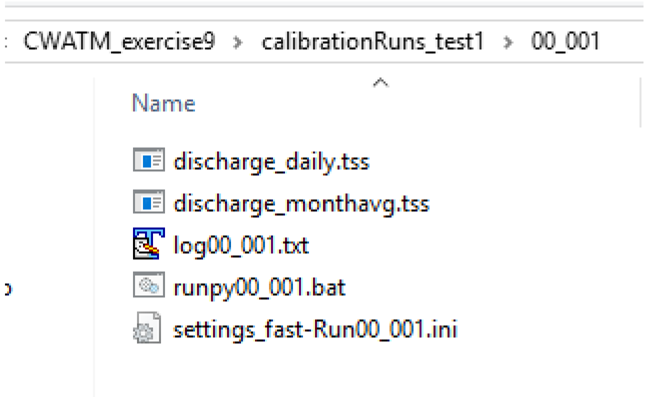
1. Running a test calibration

Please have a look at:
https://cwatm.iiasa.ac.at/calibration.html
https://cwatm.iiasa.ac.at/calibration_tutorial.html

First we check if calibration setup is ok
Start: runCalibration_test_1.bat
or type python calibration_single.py settings_test_1.txt

You need the python libraries (in addition to those you need for CWatM):
pandas, deap, matplotlib

This should produce the folder:
CWATM_exercise9\calibrationRuns_test1\00_001
With the files shown here:



If you do not see the discharge_daily.tss
Please run the runpy00_001.bat and check the errors
Have a look at:
https://cwatm.iiasa.ac.at/calibration_tutorial.html

Calibration

2. Running a fast calibration

The fast calibration is using only:

- 2 year run (normally we use ≥ 10 years)
- Initial population of 8 (normally ≥ 256)
- 2 generations (normally ≥ 20)
- 4 runs per generation (normally ≥ 16)

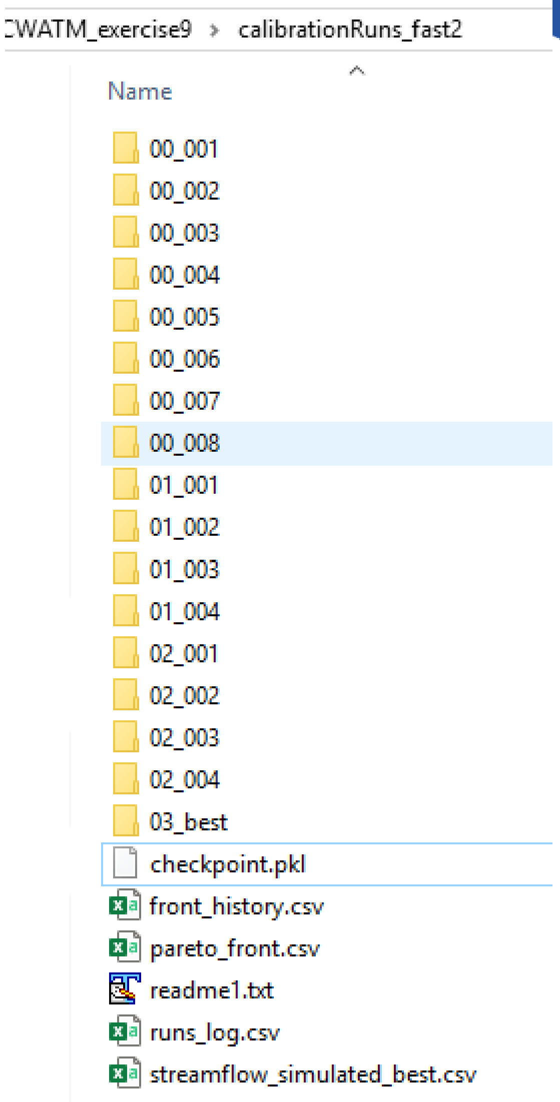
My computer as 8 nodes,
the calibration is splitting the runs for multiprocessing

Start: runCalibration_fast_2.bat
or type python calibration_single.py settings_fast_2.txt

This should produce the folder:

CWATM_exercise9\calibrationRuns_fast2

With the files shown right:



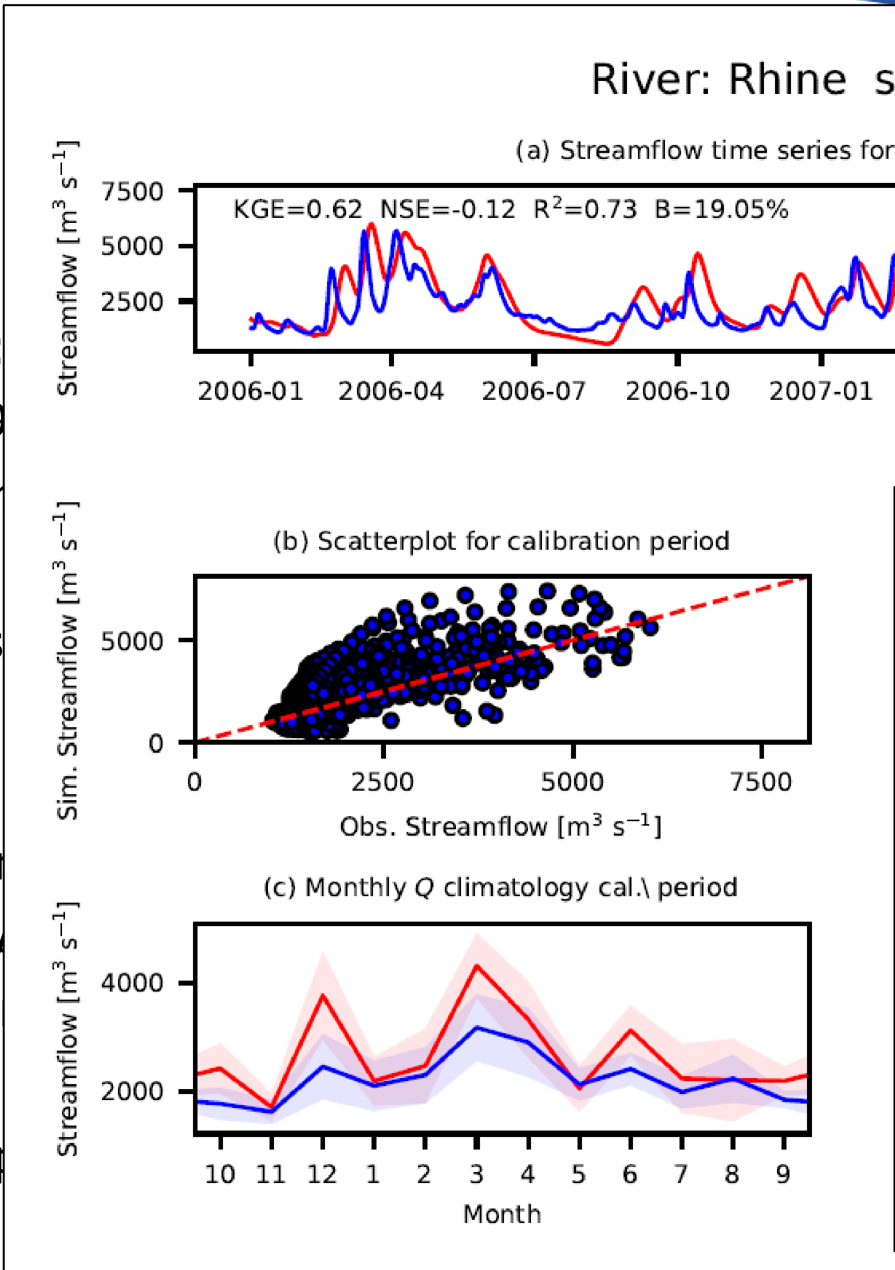
Calibration

3. Displaying results

Results for the fast calibration or in:
F:\CWATM.ECHO\CWATM_exercise9
- pareto_front.csv shows the param
- runs_log.csv shows the objective
- streamflow_simulated_best.csv is
- Each folder e.g. 02_003 hast the

To display a graphical figure of the r
plotCali_fast_2.bat or *python Cali_p*
(you need the libraries pandas and

And have a look at: cali_plot_fast2.p



	Obs.	Sim.
KGE		0.621
NS		-0.123
NSlog		0.031
R^2		0.730
Bias		19.05%
RMSE		1029
MAE		733
Mean	2235	2660
Min	1036	565
5%	1183	944
50%	1980	2410
95%	4222	5235
99%	5418	6579
Max	6028	7401

Calibration

4. Running a longer calibration

The longer calibration is using only:

- 15 year run (normally we use ≥ 10 years)
- Initial population of 8 (normally ≥ 256)
- 5 generations (normally ≥ 20)
- 8 runs per generation (normally ≥ 16)

Start: runCalibration_long_3.bat
or type python calibration_single.py settings_long_3.txt

(This can take an hour!)

This should produce the results shown in the figure

C:\WINDOWS\system32\cmd.exe

```
   run_rand_id: 03_007, KGE: 0.565
   run_rand_id: 03_004, KGE: 0.586
   run_rand_id: 03_002, KGE: 0.594
   run_rand_id: 03_006, KGE: 0.538
   run_rand_id: 03_008, KGE: 0.584
>> gen: 3, effmax_KGE: 0.616
   run_rand_id: 04_006, KGE: 0.608
   run_rand_id: 04_008, KGE: 0.602
   run_rand_id: 04_007, KGE: 0.636
   run_rand_id: 04_004, KGE: 0.613
   run_rand_id: 04_005, KGE: 0.596
   run_rand_id: 04_002, KGE: 0.581
   run_rand_id: 04_003, KGE: 0.592
   run_rand_id: 04_001, KGE: 0.619
>> gen: 4, effmax_KGE: 0.636
   run_rand_id: 05_004, KGE: 0.625
   run_rand_id: 05_005, KGE: 0.644
   run_rand_id: 05_008, KGE: 0.631
   run_rand_id: 05_007, KGE: 0.618
   run_rand_id: 05_002, KGE: 0.615
   run_rand_id: 05_003, KGE: 0.595
   run_rand_id: 05_001, KGE: 0.613
   run_rand_id: 05_006, KGE: 0.620
>> gen: 5, effmax_KGE: 0.644
>> Termination criterion ngen fulfilled.
>> Time elapsed: 686.56 s
>> Saving optimization history (front_history.csv)
>> Saving Pareto optimal solutions (pareto_front.csv)
>> Running Model using the "best" parameter set
```

Still this is not the full calibration, so results do not show best performance!