# Optimizating SWAT performance using a Python tool

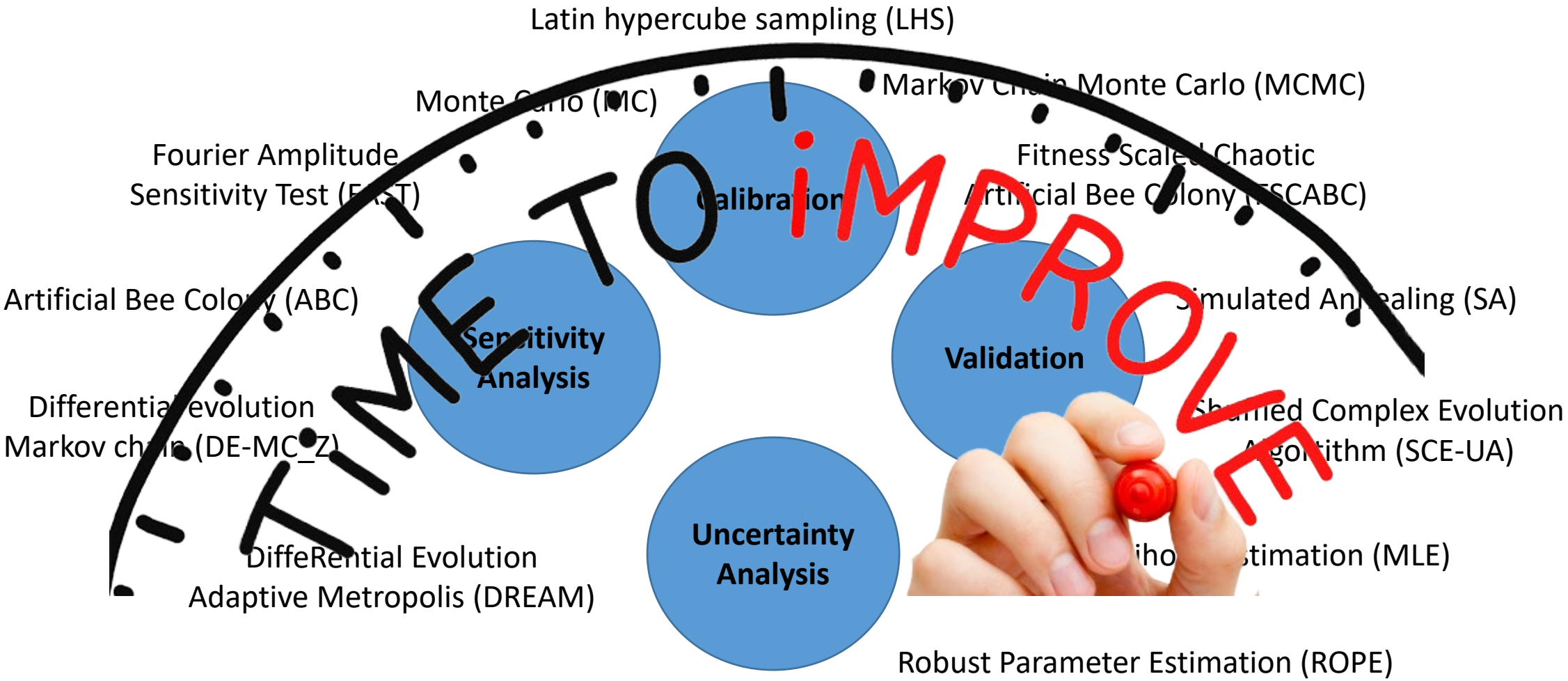
Carla Camargos*, Tobias Houska, Lutz Breuer

*Camargos.S.Carla@umwelt.uni-giessen.de*

Justus-Liebig University Giessen, Germany

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607000.

# Motivation


Latin hypercube sampling (LHS)
Monte Carlo (MC)
Markov Chain Monte Carlo (MCMC)
Fourier Amplitude
Sensitivity Test (FAST)
Fitness Scaled Chaotic
Artificial Bee Colony (FSCABC)
Calibration
Artificial Bee Colony (ABC)
Sensitivity Analysis
Validation
Simulated Annealing (SA)
Differential evolution
Markov chain (DE-MC_Z)
Shuffled Complex Evolution
Algorithm (SCE-UA)
TIME TO iMPROVE
Uncertainty Analysis
DiffeRential Evolution
Adaptive Metropolis (DREAM)
Maximum Likelihood Estimation (MLE)
Robust Parameter Estimation (ROPE)

SPOTPY

- Free
- Open source
- Parallel processing

**1. Parameter distributions:**
Uniform, (log-) Normal, Chi-square, Exponential, Gamma, Wald and Weibull

**2. Simulation / Evaluation**

**3. Objective Function:**
(P)Bias, (log-) NSE, r, $r^2$, Covariance; MSE, dMSE, MAE, RMSE, RRMSE, AI, KGE, hydrological signatures and likelihood functions

**4. Parameter estimation methods:**

| Sensitivity Analysis | Uncertainty Analysis | Non-Bayesian Calibration | Bayesian Calibration |
|---|---|---|---|
| FAST | GLUE | MLE, SA, SCE-UA, ABC, FSCABC | MC, LHS, MCMC, DE-$MC_Z$, DREAM, ROPE |

**5. Database:**
ram, csv, sql

**6. Analyser:**
Posterior analyses, Best model run, Trace plots, Parameter interaction, Convergence diagnostics

*http://fb09-pasig.umwelt.uni-giessen.de/spotpy/*

# SPOTPY + SWAT

```
class spot_setup(object):
    AWC = spotpy.parameter.Uniform(low=-0.9,high=0.3)
    ksat = spotpy.parameter.Uniform(low=-0.9,high=0.6)
    CHN = spotpy.parameter.Uniform(low=0.001,high=0.25)
    CHK = spotpy.parameter.Uniform(low=0.1,high=150)
    ALPHA_BF = spotpy.parameter.Uniform(low=0.001,high=0.99)
    GW_DELAY = spotpy.parameter.Uniform(low=1,high=50)
    RCHRG_DP = spotpy.parameter.Uniform(low=0,high=1)
    CN2 = spotpy.parameter.Uniform(low=-0.9,high=0.15)
    ESCO = spotpy.parameter.Uniform(low=0,high=1)

    def __init__(self):
        self.observeddata_fname = 'observed'+os.sep+'discharge.txt'
        self.observeddata = np.loadtxt(self.observeddata_fname)
        self.nr_of_observations = len(self.observeddata)
```

# SPOTPY + SWAT

```
def simulation(self,vector):
    manipulators = {}

    ### here all parameters from the gw-file are assigned in the dictionary for calibration
    gwfiles = [i for i in files if i.endswith(".gw")]
    gw = []
    for i in gwfiles:
        gw.append(gwManipulator(i, ["GW_DELAY","ALPHA_BF","GW_REVAP","GWQMN","RCHRG_DP","REVAPMN"], core_nr))
    manipulators["gw"] = gw

    for d in gw:
        d.setChangePar("GW_DELAY",vector.delay,"s")
        d.setChangePar("ALPHA_BF",vector.alpha,"s")
        d.setChangePar("RCHRG_DP",vector.RCHRG_DP,"s")
        d.finishChangePar(core_nr)
```

# SPOTPY + SWAT

```
sub.call(['./SWAT_rev664'])
subbasins = [1] #subbasin number where output should be extracted
results = (read.rchOutputManipulator(["FLOW_OUT"],subbasins,"indi",False,1,core_nr))
results = []
for subb in subbasins:
    with open('SWATsensitivity_FLOW_OUT_000'+str(subb)+'.rch', 'rb') as fh:
                for line in fh:
                    pass
                last = line
    vals = last.split(' ')
    oneresult=[]
    for val in vals:
     oneresult.append(float(val))
    results.append(oneresult)
```

# SPOTPY + SWAT

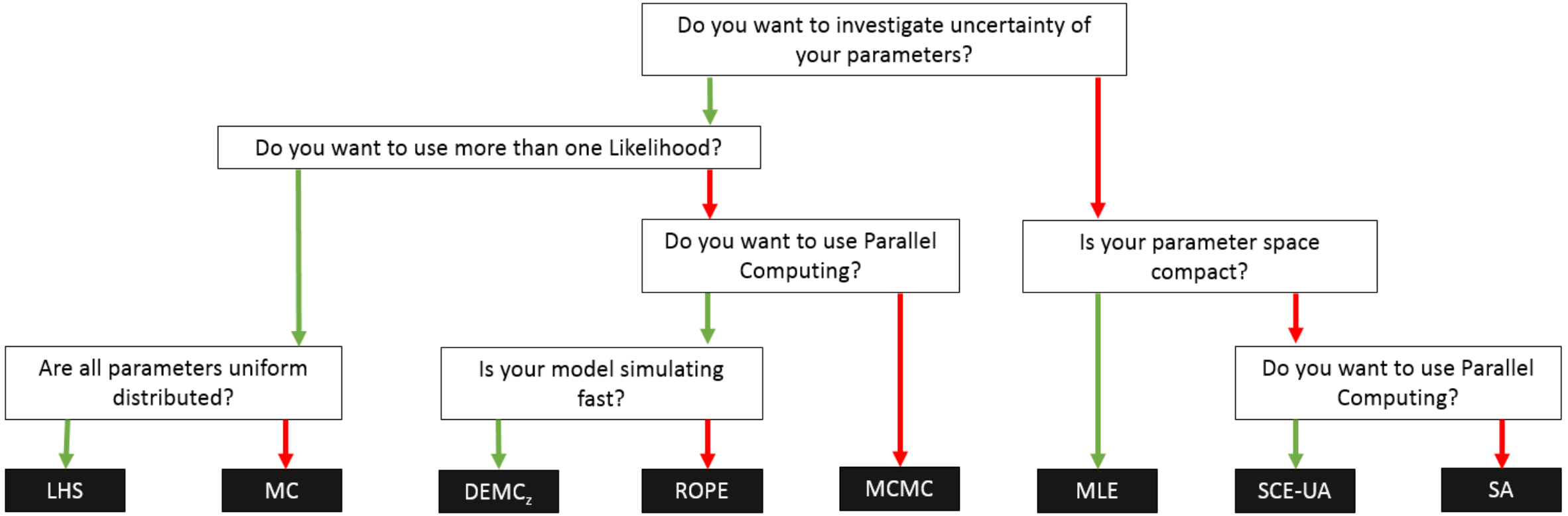
```
def evaluation(self):
    observationdatalists = []
    observationdatalists.append(self.observeddata)
    return observationdatalists

def objectivefunction(self,simulation,evaluation):
    indexs=[]
    for obs in evaluation:
        index=[]
        for i in range(len(obs)):
            if not obs[i] == -9999: #used for missing observation data
                index.append(i)
        indexs.append(index)
    sub1 = np.array(simulation[0])
    sub = np.array(sub1)[indexs[0]]
    sub_logp = spotpy.objectivefunctions.log_p(evaluation[0][indexs[0]],sub)
    return sub_logp
```

**3. Objective Function:**
(P)Bias, (log-) NSE, r, $r^2$, Covariance; MSE, dMSE, MAE, RMSE, RRMSE, AI, KGE, hydrological signatures and likelihood functions

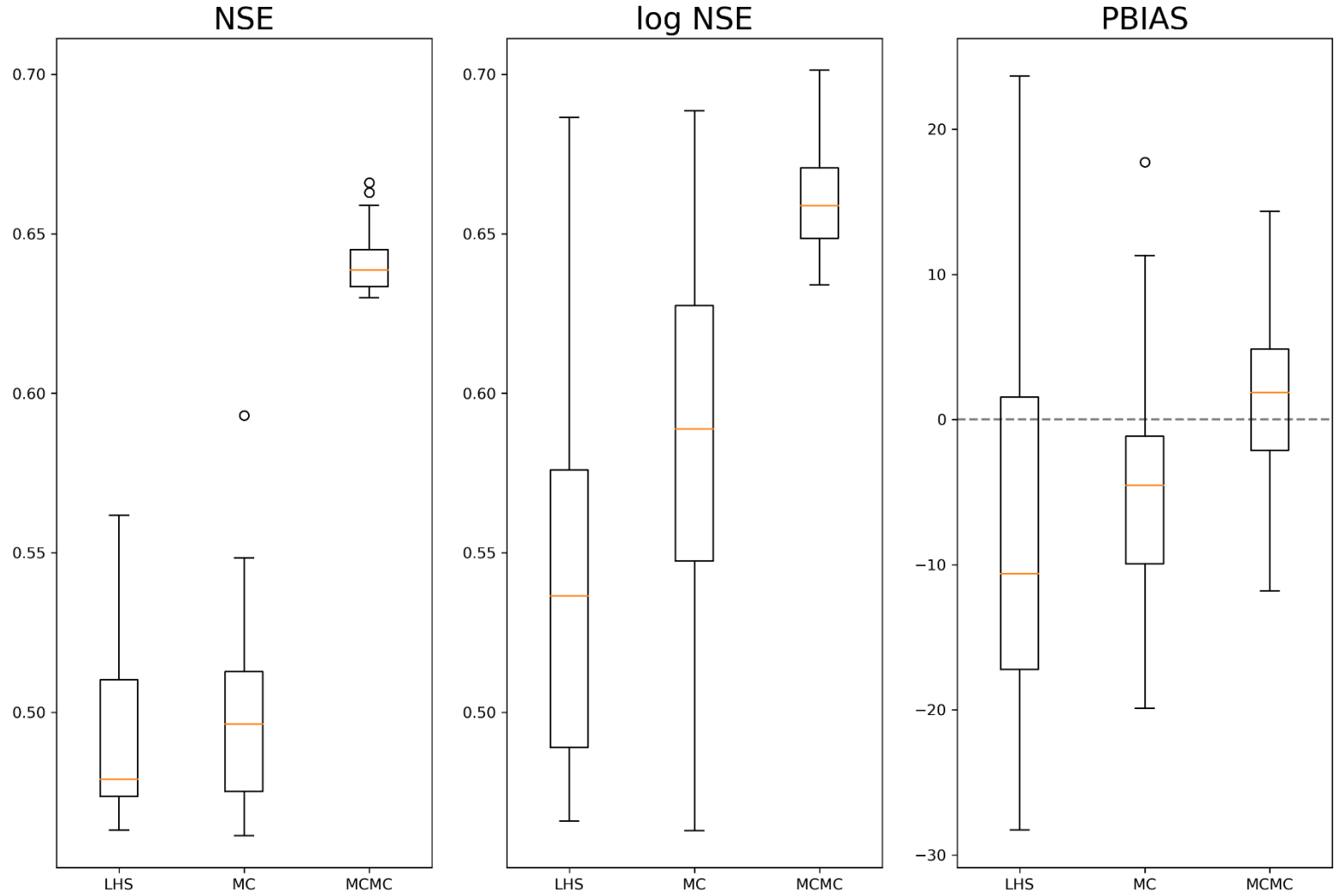
# SPOTPY + SWAT

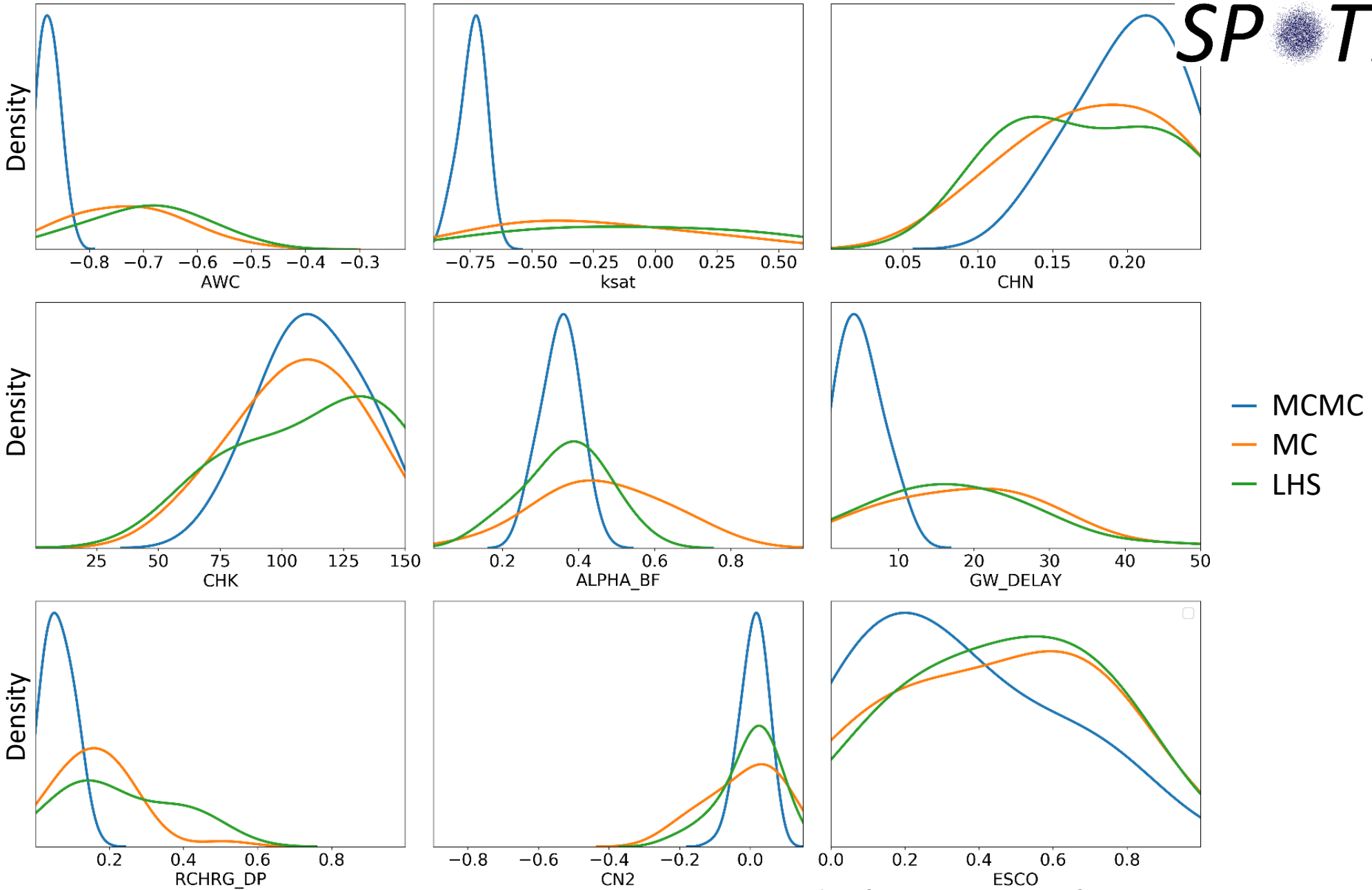
```
starter = spot_setup(parallel)
sampler = spotpy.algorithms.mcmc(starter,dbname='SWAT-discharge_mcmc',dbformat='csv', alt_objfun=None,parallel=parallel)
sampler.sample(repetitions=15000, nChains=3)
```



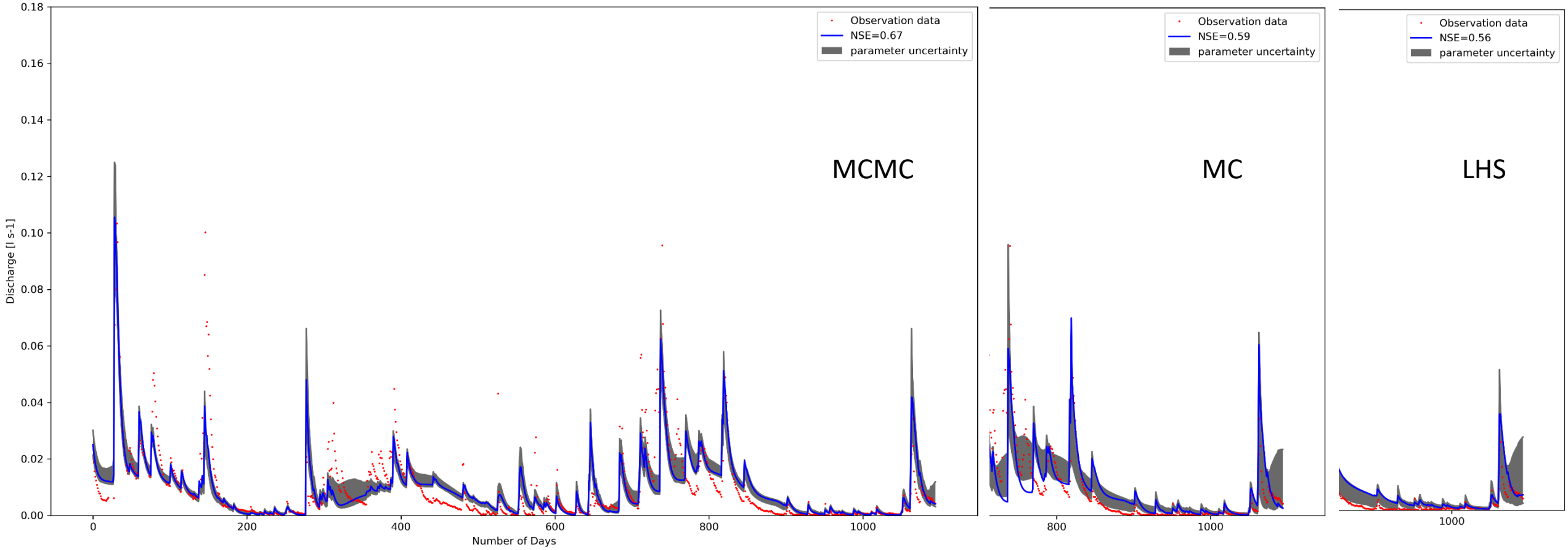

*http://fb09-pasig.umwelt.uni-giessen.de/spotpy/Tutorial/8-Algorithm_guide/*

# Comparing methodologies

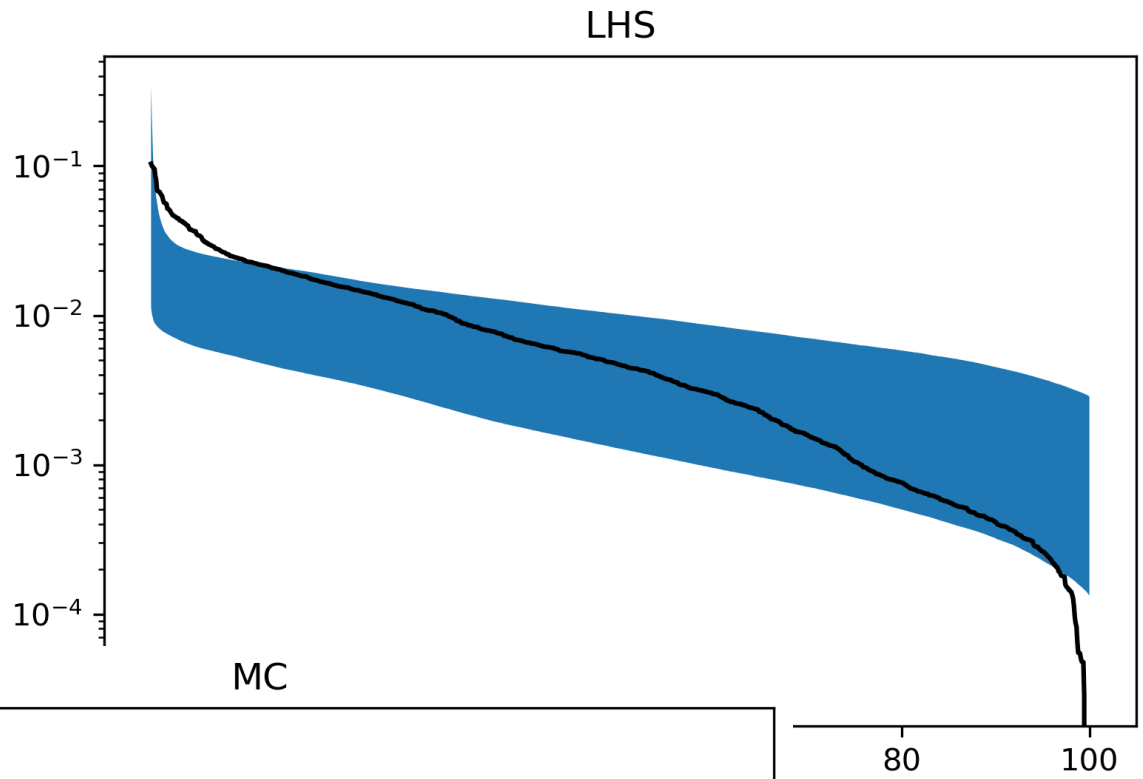


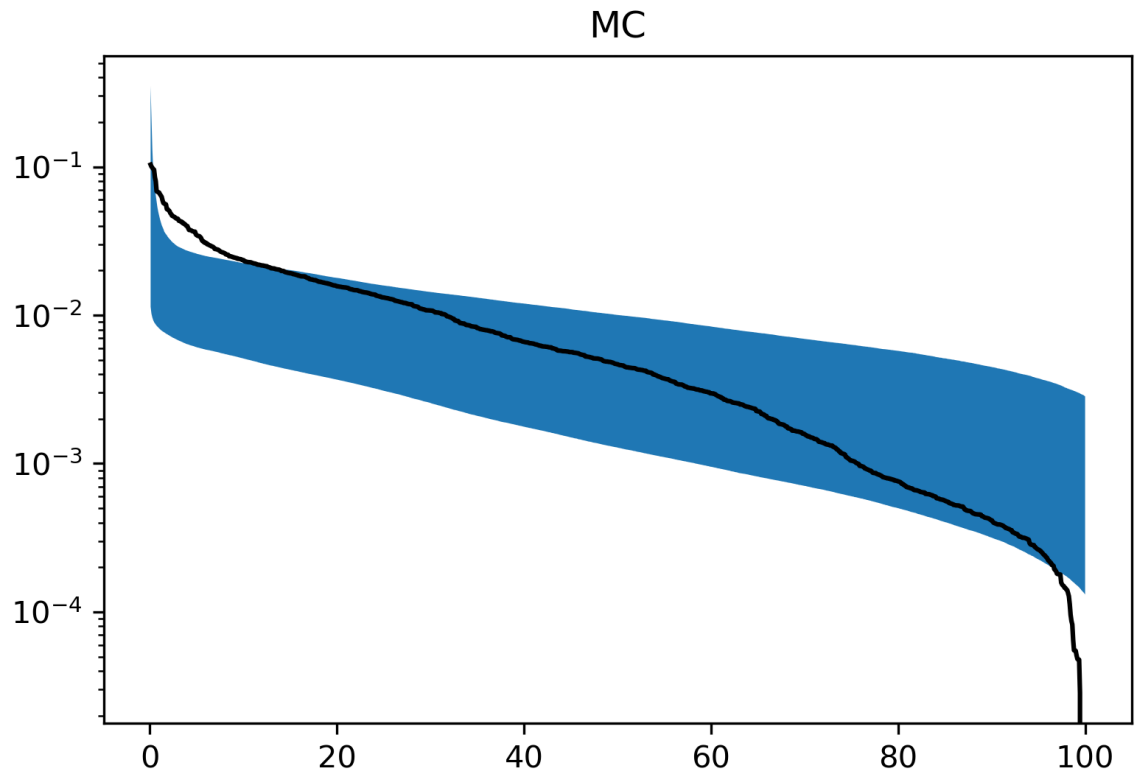

Density
AWC
ksat
CHN
CHK
ALPHA_BF
GW_DELAY
RCHRG_DP
CN2
ESCO
MCMC
MC
LHS

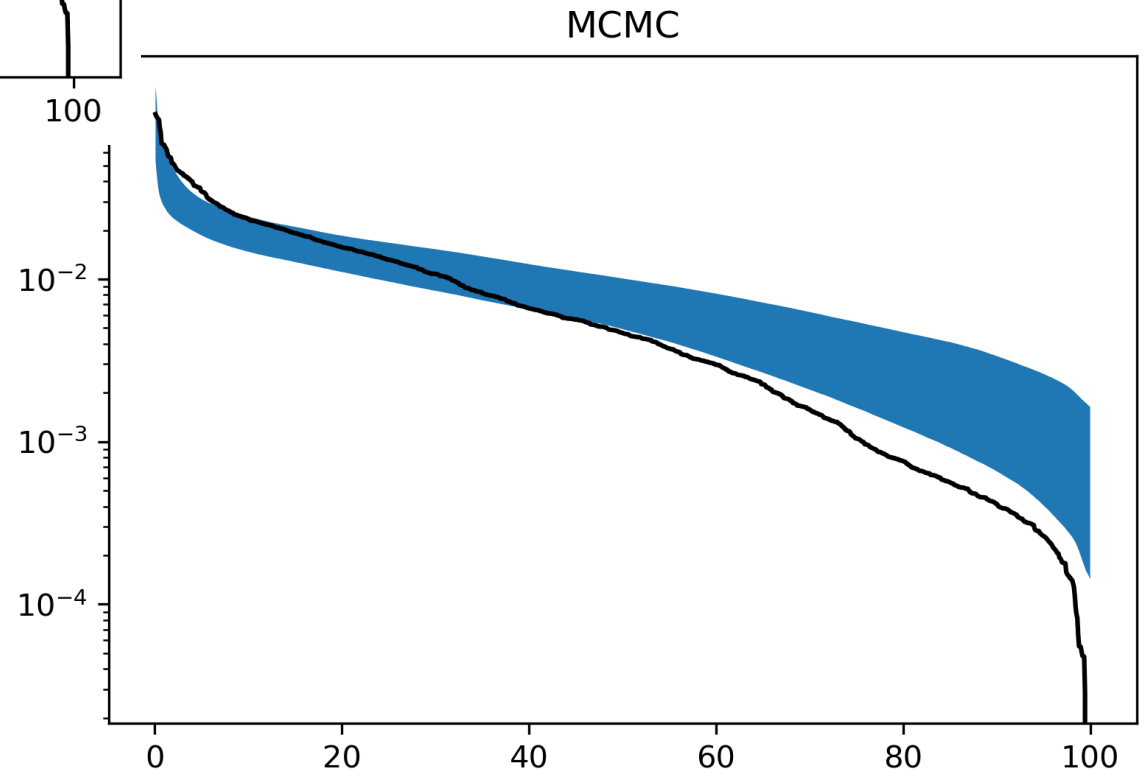
# Comparing methodologies



# Flow duration curve



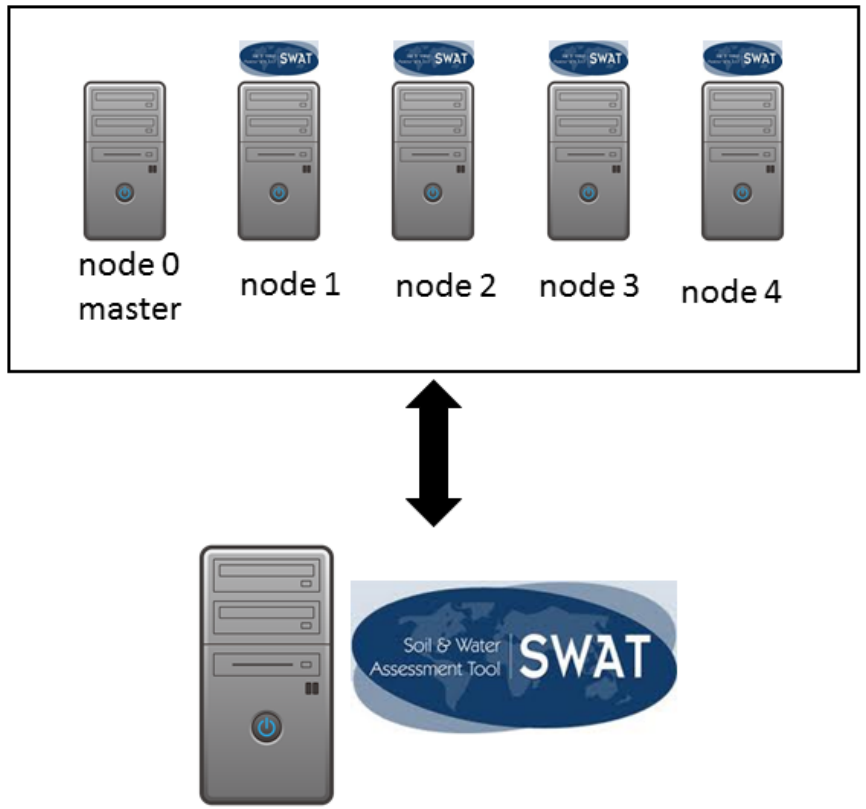



# Thank you!

More information: http://fb09-pasig.umwelt.uni-giessen.de/spotpy/

Camargos.S.Carla@umwelt.uni-giessen.de


node 0
master
node 1
node 2
node 3
node 4
Soil & Water
Assessment Tool
SWAT