Supplementary Material

Distributed dynamic modelling of suspended sediment mobilization and transport from small agricultural catchments

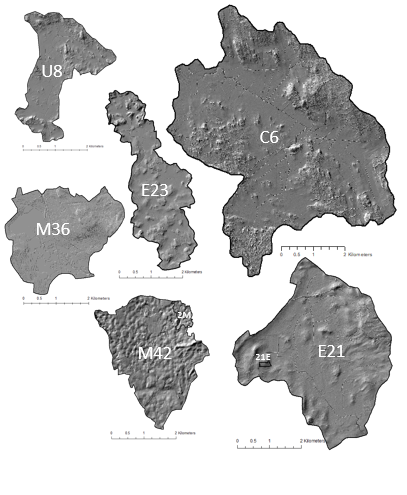
Sara Sandström\*, Hampus Markensten, Martyn N. Futter, Katarina Kyllmar, David W. O’Connell, Kevin Bishop & Faruk Djodjic

**\* Correspondence:** Sara Sandström: sara.sandstrom@slu.se

# Statistical analysis

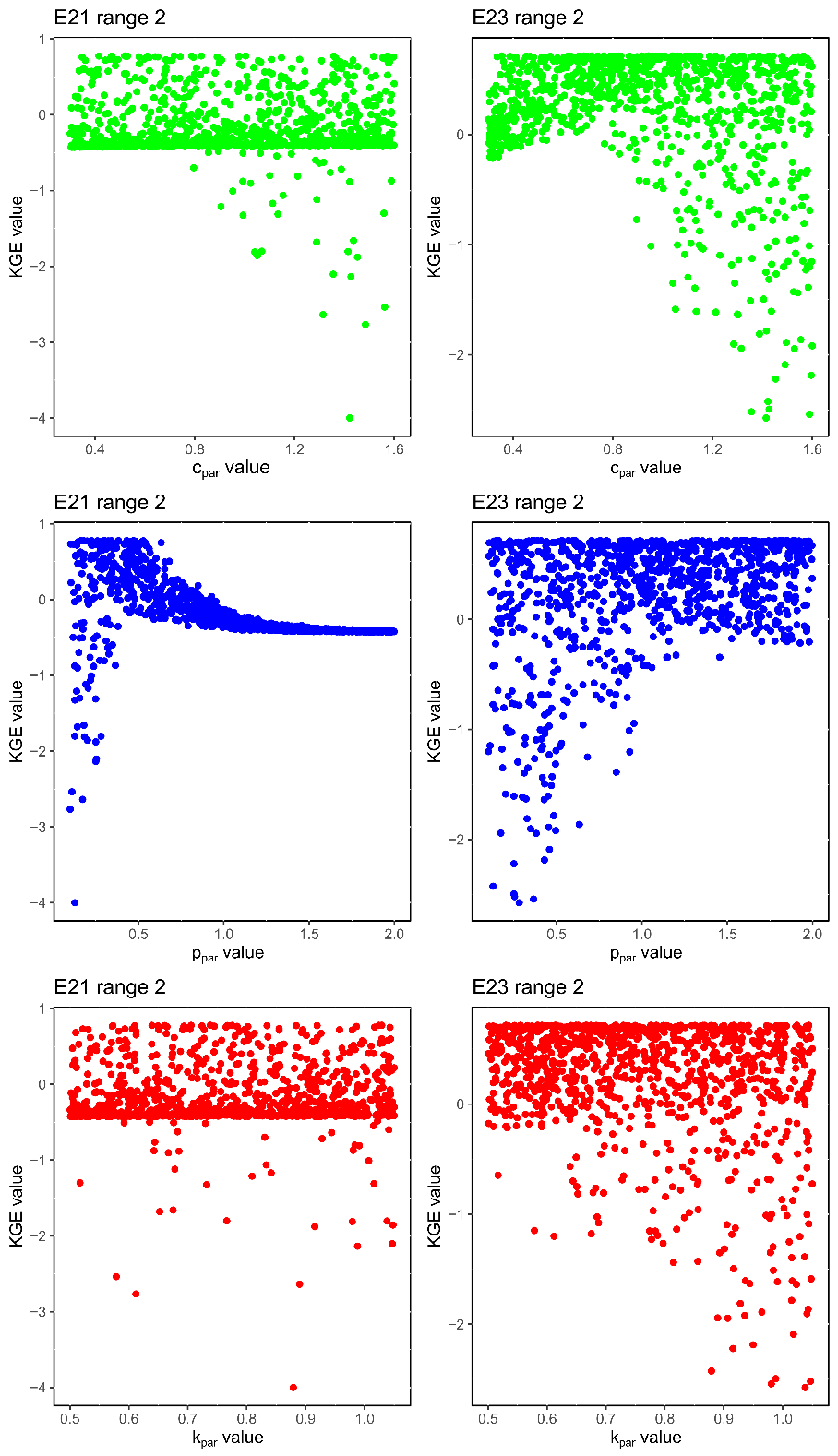
Linear regression between simulated SS and measured and calculated PP transport generally gave good agreement. Using all catchments in the same regression resulted in an explained variation of 62% (R2=0.62, p<0.001) (Figure A3), while separate regressions for each catchment ranged gave R2 value ranging from 0.40 (M36) to 0.83 (E23).

Linear regression between measured SS and measured PP transport during the validation period gave good agreement with all catchments used in the same regression (R2=0.62, p<0.001, y=0.0011x + 0.37).

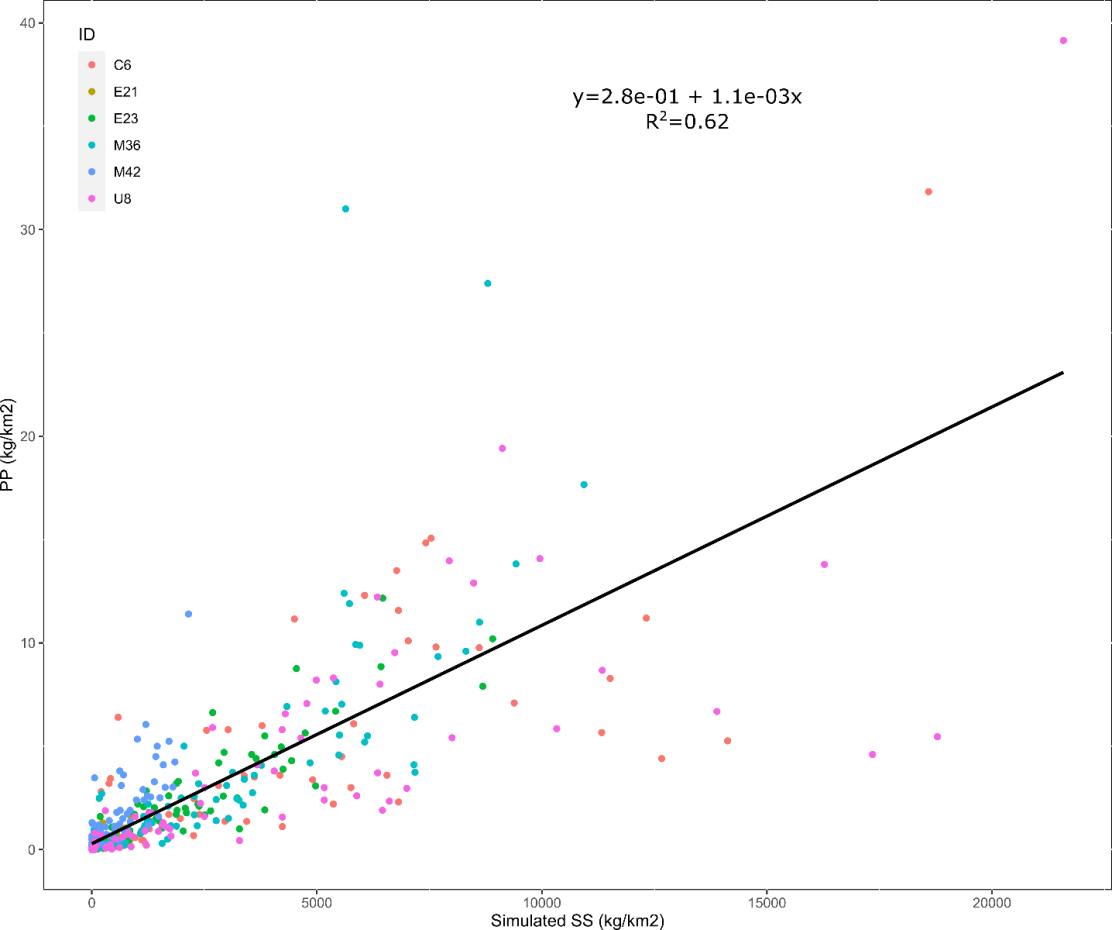
******

**Figure S1.** Hillshade maps, based on digital elevation models (DEM) describing the topography of all catchments in the study. The catchments names are marked in white in each catchment, and the used observation fields are also marked in the catchments.

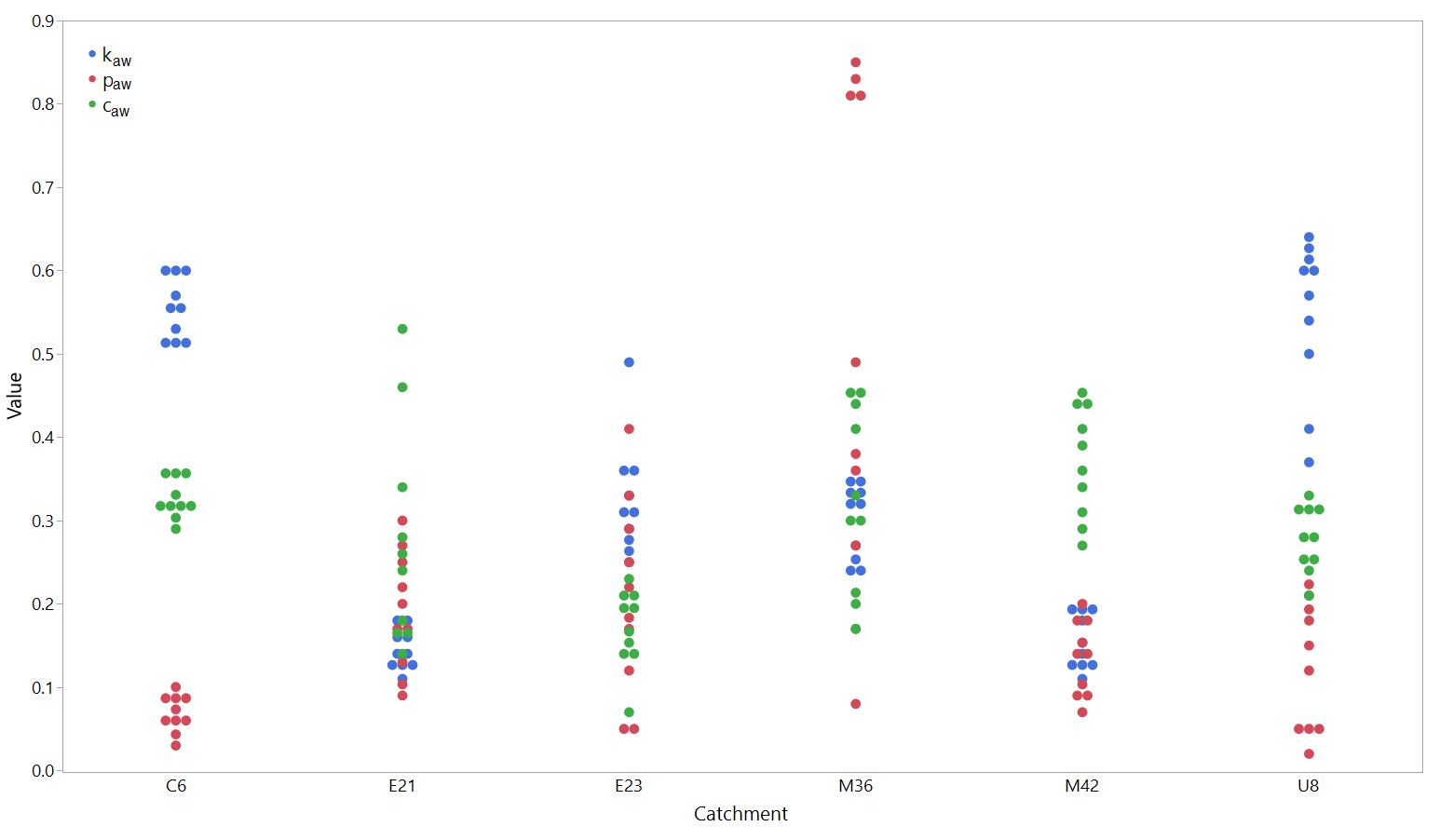
**Figure S2**. Example of relationship between KGE values for the same parameter settings in the calibration and validation periods.



**Figure S3**. Dotty plots with second parameter range for the three pseudo parameters cpar, ppar and kpar (1000 Monte Carlo runs) for catchments E21 and E23. Parameter value on the x-axis and corresponding KGE value on the y-axis.



**Figure S4.** Linear regression between simulated monthly transport of suspended solids (SS, kg km-2 month-1) for the validation period and measured and calculated transport of particulate phosphorus (PP, kg km-2 month-1) for the same period. Points are coloured by catchment, black solid line represents the regression line (regression equation given in the diagram).



**Figure S5**. Variation in area-weighted average values of parameters *c*, *k* and *p* (caw, kaw, paw).

**Table S1.** Starting values for parameters *p* (soil permeability) and *k* (soil erodibility) for different land uses, and calibration ranges for pseudo parameters ppar and kpar

| **Soil texture (FAO and SGU)** | **Original p-value** | **Original k-value** |
| --- | --- | --- |
| Sand | 0.80 | 0.05 |
| Loamy sand | 0.70 | 0.1 |
| Sandy loam | 0.60 | 0.15 |
| Sandy clay loam | 0.55 | 0.20 |
| Loam | 0.55 | 0.40 |
| Silt loam | 0.10 | 0.82 |
| Silt | 0.05 | 0.95 |
| Sandy clay | 0.45 | 0.35 |
| Clay loam | 0.20 | 0.67 |
| Silty clay loam | 0.05 | 0.92 |
| Silty clay | 0.10 | 0.82 |
| Clay | 0.20 | 0.67 |
| Organic soil | 0.80 | 0.01 |
| Clay (non-arable) | 0.20 | 0.67 |
| Silt (non-arable) | 0.10 | 0.9 |
| Sand (non-arable) | 0.80 | 0.04 |
| Gravel | 0.80 | 0.02 |
| Cobbles to boulders | 0.20 | 0.02 |
| Fluvioglacial sediment, cobbles to boulders | 0.80 | 0.02 |
| Clay till | 0.30 | 0.3 |
| Till (moraine) | 0.45 | 0.1 |
| Thin soil layer | 0.10 | 0.1 |
| Rock | 0.10 | 0.01 |
| Artificial fill | 0.80 | 0.02 |
| Other | 0.80 | 0.02 |
| Water | 0.99 | 0.001 |
|  | ppar | kpar |
| Starting parameter range: | 0.5-2 | 0.5-1.05 |
| Second parameter range | 0.1-2 | 0.5-1.05 |

**Table S2**. Starting values for vegetation cover parameter *c* in different land uses and parameter ranges for calibration for pseudo parameter cpar

| **Land use** | **c-value** |
| --- | --- |
| Mire | 0.01 |
| Arable land | Varying as below |
| Open land | 0.02 |
| Water | 0.001 |
| Forest | 0.01 |
| Clear cut | 0.1 |
| Pasture land | 0.04 |
|  |  |
| **Month (arable land)** | **c-value** |
| January | 0.61 |
| February | 0.59 |
| March | 0.59 |
| April | 0.4 |
| May | 0.24 |
| June | 0.07 |
| July | 0.05 |
| August | 0.04 |
| September | 0.4 |
| October | 0.48 |
| November | 0.51 |
| December | 0.6 |
|  | **cpar** |
| Starting parameter range: | 0.5-1.6 |
| Second parameter range: | 0.3-1.6 |

**Table S3.** Values of R2 obtained for linear regression between the different area-weighted parameter values (caw, kaw, paw) in the best 10 runs (\*p<0.05, \*\*p<0.01, \*\*\* p<0.001)

| **Catchment** |  | **kaw** | **paw** | **caw** |
| --- | --- | --- | --- | --- |
| C6 | caw |  |  |  |
| E21 | caw |  | 0.75\*\* |  |
| E23 | caw | 0.37\* | 0.60\*\* |  |
| M36 | paw | 0.50\* |  | 0.94\*\*\* |
| M42 | paw | 0.66\*\* |  | 0.48\* |
| U8 | paw |  |  |  |

# Generalized modelling script used in Python

*# -\*- coding: utf-8 -\*-  
#USPED model for catchment general catchment  
#Based on scripts for static USPED models created by Faruk Djodjic  
#Further developed to dynamic and translated to python by Sara Sandström***from** pcraster.framework **import**\*  
**import** pcraster **as** pr  
**from** myconfig **import** rundir  
**import** os  
*# set working directory*os.chdir(rundir)  
*# doublecheck wd*wd = os.getcwd()  
print(wd)  
  
*#create a class of the model, USPEDmodel***class** USPEDmodel(DynamicModel):  
 *###binding section with pcrcalc  
 #read all available maps with the function readmap* **def** \_\_init\_\_(self, cloneMap, rts , park\_, parp\_, parc\_):  
 DynamicModel.\_\_init\_\_(self)  
 *# in python pcraster, set map for area extent (catchment boundaries)  
 # what map you use is set when you call the model (see the end of the script)* pr.setclone(cloneMap)  
 self.park = park\_  
 self.parp = parp\_  
 self.parc = parc\_  
 self.rtss = rts *#flowdata/runoff (is now read when the model is called)  
 #read in maps and timeseries for use, and make calculations that only needs to be made once* **def** initial(self):  
 *#define and read in available maps that has been created before (changemaps.py)* self.dem = pr.readmap(**"dem.map"**) *#read in the DEM-file* self.flowdir = pr.readmap(**"ldd.map"**) *#flowdirection map* self.slope = pr.readmap(**"slope.map"**) *#slope map* self.slopedegre\_map = pr.readmap(**"slopedegree.map"**) *#slope map in degrees* self.slopelength\_map = pr.readmap(**"slopelength\_mod.map"**) *#slopelength map* self.profcurv\_map = pr.readmap(**"profcurv\_map.map"**) *# read profile curve map* self.plancurv\_map = pr.readmap(**"plancurv\_map.map"**) *# read plan curve map  
  
 #read timeseries for c (landuse factors)  
  
 #self.ctss = "cmonthly.tss" #landusedata* self.ctss = **"cmonthlyval.tss"** *# landusedata for validation  
  
 #read landuse map and outlet map. outlet map and landuse needs to be nominal* self.czones = pr.readmap(**"landusenom.map"**) *#read in landuse map* self.utlopp = pr.readmap(**"outletnom.map"**) *#read outletmap  
  
 #read in maps* self.soil = pr.readmap(**"soilmap.map"**) *#read in the map of soil texture* self.k\_map = pr.lookupscalar(**"klookup.tbl"**, self.soil) *#read in k values (soil erodibility) and apply to soil map* pr.report(self.k\_map, **"k\_map.map"**) *#create and save map of k values  
 # park = 1* self.k\_map = self.k\_map\*self.park  
  
 p\_map = pr.lookupscalar(**"plookup.tbl"**, self.soil) *#read in p values (exponent value for rill erosion) and apply to soil map* pr.report(p\_map, **"p\_map.map"**) *#save p map  
 # parp = 1* p\_map = p\_map\*self.parp  
  
 *# calculates rill erosion risk (?) (1.6 literature value for rill erosion, 22.13 factor to change units, 0.0.1745 factor to change units)* self.lsrill\_map = ((self.slopelength\_map / 22.13)\*\*1.6) \* (pr.sin(self.slopedegre\_map\*0.01745)\*\*(1+p\_map))  
 pr.report(self.lsrill\_map, **"lsrill\_map.map"**) *#save map of rill erosion  
  
 #initialise timeoutput* self.erotss = TimeoutputTimeseries(**"erofluxm"**, self, self.utlopp, noHeader=**False**) *#erosionflux time series  
  
 #dynamic model, all calculations that are looped* **def** dynamic(self):  
  
 *# parc = 1* self.r = pr.timeinputscalar(self.rtss, 1) *# flowdata/runoff* self.report(self.r, **"flow"**)  
 self.c\_map = pr.timeinputscalar(self.ctss, self.czones)  
 *#self.report(self.c\_map, "cmap")* self.c\_map = self.c\_map \* self.parc  
  
 *#create an erosion map* ero\_map = self.lsrill\_map \* self.k\_map \* self.c\_map \* self.r\* (1 + -1.0000000001 \* self.plancurv\_map)\* (1 + -1.0000000001 \* self.profcurv\_map)\*4 *#numbers relate to direction, 4 is 2x2, cellsize* self.report(ero\_map, **"ero\_map"**) *#save an erosion map of each timestep* eroflux\_map = pr.accuflux(self.flowdir, ero\_map) *#erosion flux* self.report(eroflux\_map, **"erofluxm"**) *#save erosion flux map  
  
 #save the sample (time step) in the time series* self.erotss.sample(eroflux\_map)  
 **return** self.erotss  
  
*####  
#To initialize the model, set the mask, modelname and number of timesteps, then run the model  
#the model can be run either by changing these parameters and uncommenting them,  
#or by initializing from a separate script  
#park = 1  
#parp = 1  
#parc = 1  
#rtss = "rval.tss" #flowdata/runoff  
#myModel = USPEDmodel("mask.map", rtss, alpha, beta, gamma) #mask to use, outline of catchment  
#dynModelFw = DynamicFramework(myModel, lastTimeStep=96, firstTimestep=1) #set timesteps to run the simulation over  
#dynModelFw.run() #run the model*