```
# Author: Pedro Batista
# Contributor: Peter Fiener
# This script runs a version of WaTEM/SEDEM with R and SAGA
# for the Baldegg catchment
# Here I am mostly interested in sediment connectivity
# Grass strips, roads, field borders

# Load packages and clear enviroment
library(raster)
library(tidyverse)
library(foreach)
library(RSAGA)
library(rgdal)
library(ggplot2)
library(sp)
library(dplyr)
library(doParallel)

# Clear de environment
rm(list = ls())

# Set working directory
# make sure file names are the same or make the necessary changes
setwd("E:/Baldegg_WS_Simulations")

# Load masks matrices
Mask_roads <- readGDAL("road_mask_0.sdat")@data$band1
# This is for calculating sediment loads per sub-catchment
Mask_matrix <- readGDAL("sub_mask.sdat")@data$band1

# Define sub-catchment matrices
# Cells within the sub-catchments have values of 1
```

```
# Outside cells have NA values
Ron <- ifelse(Mask_matrix != 4, NA, Mask_matrix)
Ron <- ifelse(Ron == 4, 1, Ron)

Stagbach <- ifelse(Mask_matrix != 1, NA, Mask_matrix)
Stagbach <- ifelse(Stagbach == 1, 1, Stagbach)

Hohibach <- ifelse(Mask_matrix != 3, NA, Mask_matrix)
Hohibach <- ifelse(Hohibach == 3, 1, Hohibach)

Spittlisbach <- ifelse(Mask_matrix != 5, NA, Mask_matrix)
Spittlisbach <- ifelse(Spittlisbach == 5, 1, Spittlisbach)

Mulibach <- ifelse(Mask_matrix != 2, NA, Mask_matrix)
Mulibach <- ifelse(Mulibach == 2, 1, Mulibach)

# Set clusters for parallel processing
cl <- makeCluster(4)
registerDoParallel(cl)

# Set up environment for RSAGA
env <- rsaga.env(path = "C:/Users/pedro/saga_2.2.2_x64") #Path to SAGA
env$version #using SAGA version 2.2.2 for compatibility (code seems to work with other versions)

# Prepare simulation
N.sim <- 200 # number of iterations
cell.size <- 2 # this is used to calculate sediment yield
start_time <- Sys.time() # get start time
WS_sims <- foreach(i = 1:N.sim,
                   .combine = rbind,
                   .packages = c("RSAGA", "dplyr", "rgdal"),
                   .verbose = T
```

```
        ) %dopar%
{

# Create PTEF map based on the parcel map
# Sample PTEF for pastures and forests
PTEF <- runif(1, 0, 1) # Uniform distribution between 0-1

# With Grass Strips (GS)
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "parcel.sgrd",
                RESULT = paste("ptef_GS_", i, ".sgrd", sep = ""),
                METHOD = 0,
                OLD = 10000,
                NEW = 1 - PTEF,
                SOPERATOR = 3, #>=
                OTHEROPT = T,
                OTHERS = 1,
                NODATAOPT = T,
                NODATA = -99999), env = env)

# No Grass Strips (NS)
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "parcel.sgrd",
                RESULT = paste("ptef_NS_", i, ".sgrd", sep = ""),
                METHOD = 1,
                MIN = 1000,
                MAX = 15000,
                RNEW = 1 - PTEF,
                ROPERATOR = 1, #>=
                OTHEROPT = T,
                OTHERS = 1,
                NODATAOPT = T,
```

```
                NODATA = -99999), env = env)


# Catchment area (Multiple flow direction)
# Flow accumulation is weighted according to the PTEF grid
# Output is upslope contributing area (m2), not number of cells
# GS
rsaga.geoprocessor(lib = "ta_hydrology", 0,
            list(ELEVATION = "sf_dem.sgrd",
                WEIGHT = paste("ptef_GS_", i, ".sgrd", sep = ""),
                CAREA = paste("carea_GS_", i, ".sgrd", sep = ""),
                METHOD = 4, #multiple flow direction
                LINEAR_DO = F), env = env) #threshold to linear flow


# NS
rsaga.geoprocessor(lib = "ta_hydrology", 0,
            list(ELEVATION = "sf_dem.sgrd",
                WEIGHT = paste("ptef_NS_", i, ".sgrd", sep = ""),
                CAREA = paste("carea_NS_", i, ".sgrd", sep = ""),
                METHOD = 4, #multiple flow direction
                LINEAR_DO = F), env = env) #threshold to linear flow


# Reclassify catchment area for LS factor
# If contributing area >= 240 m² (flow 120 m lenght), 240, else keep original value
# GS
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = paste("carea_GS_", i, ".sgrd", sep = ""),
                RESULT = paste("reclass_GS_", i, ".sgrd", sep = ""),
                OLD = 240,
                NEW = 240,
                SOPERATOR = 4), env = env)
# NS
rsaga.geoprocessor(lib = "grid_tools", 15,
```

```
            list(INPUT = paste("carea_NS_", i, ".sgrd", sep = ""),
                RESULT = paste("reclass_NS_", i, ".sgrd", sep = ""),
                OLD = 240,
                NEW = 240,
                SOPERATOR = 4), env = env)
# Calculate LS factor
# GS
rsaga.geoprocessor(lib = "ta_hydrology", 22,
            list(SLOPE = "slope_rad.sgrd",
                AREA =  paste("reclass_GS_", i, ".sgrd", sep = ""),
                LS =  paste("LS_GS_", i, ".sgrd", sep = ""),
                METHOD = 1), env = env) #Desmet & Govers 1996

# GS
rsaga.geoprocessor(lib = "ta_hydrology", 22,
            list(SLOPE = "slope_rad.sgrd",
                AREA =  paste("reclass_NS_", i, ".sgrd", sep = ""),
                LS =  paste("LS_NS_", i, ".sgrd", sep = ""),
                METHOD = 1), env = env) #Desmet & Govers 1996

# Calculate Soil Loss
# Sample R and K values from uniform distributions
R <- runif(1, 950, 1350) * 10^-4
K <- runif(1, 0.025, 0.04) * 10^3

# Create R factor grid with a single lumped value
# Reclassifying any grid of the area, in this case the c_fac_lookup grid
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "cfac_lookup.sgrd",
                RESULT = paste("R_", i, ".sgrd", sep = ""),
                OLD = -1,
                NEW = R,
```

```
                SOPERATOR = 4), env = env)

# Create K factor grid
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "cfac_lookup.sgrd",
                RESULT = paste("K_", i, ".sgrd", sep = ""),
                OLD = -1,
                NEW = K,
                SOPERATOR = 4), env = env)

# Sample C factor values for each land use
C_arable <- runif(1, 0.01, 0.5)
C_grass <- runif(1, 0.001, 0.009)
C_orchard <- runif(1, 0.001, 0.2)
C_vineyard <- runif(1, 0.1, 0.6)
C_forest <- runif(1, 0.0001, 0.003)

# Prepare data with/without grass strips
C_vector_GS <- c(C_forest, C_grass, C_orchard, C_arable, C_vineyard)
C_vector_NS <- c(C_forest, C_orchard, C_arable, C_vineyard)

# Create lookup table with GS
C_lookup_GS <- read.table("LookupTable_GS.txt", header = T) %>% #reads lookup base table
  mutate(new = C_vector_GS)

write.table(C_lookup_GS, paste("LookupTable_GS_", i, ".txt", sep = ""), row.names = F,
          sep = "\t") #writes a new one with sampled C values

# Create lookup table without GS
C_lookup_NS <- read.table("LookupTable_NS.txt", header = T) %>% #reads lookup base table
  mutate(new = C_vector_NS)
```

```
write.table(C_lookup_NS, paste("LookupTable_NS_", i, ".txt", sep = ""), row.names = F,
        sep = "\t")

# Create C factor grid GS
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "cfac_lookup_GS.sgrd",
                RESULT = paste("C_GS_", i, ".sgrd", sep = ""),
                METHOD = 3,
                RETAB_2 = paste("LookupTable_GS_", i, ".txt", sep = ""),
                TOPERATOR = 1,
                F_MIN = "mininum",
                F_MAX = "maximum",
                F_CODE = "new"), env = env)

# Create C factor grid NS
rsaga.geoprocessor(lib = "grid_tools", 15,
            list(INPUT = "cfac_lookup_NG.sgrd",
                RESULT = paste("C_NS_", i, ".sgrd", sep = ""),
                METHOD = 3,
                RETAB_2 = paste("LookupTable_NS_", i, ".txt", sep = ""),
                TOPERATOR = 1,
                F_MIN = "mininum",
                F_MAX = "maximum",
                F_CODE = "new"), env = env)


# Create soil loss grid GS
# Stream cells will have NoData values
rsaga.grid.calculus(list(paste("LS_GS_", i, ".sgrd", sep = ""),
                 paste("R_", i, ".sgrd", sep = ""),
                 paste("K_", i, ".sgrd", sep = ""),
                 paste("C_GS_", i, ".sgrd", sep = ""),
```

```
                "snet_mask_NA.sgrd"), #input grids
              paste("ER_GS_", i, ".sgrd", sep = ""),
              ~  a * b * c * d * e, env = env) #a and are the input grids

# Create soil loss grid NS
# Stream cells will have NoData values
rsaga.grid.calculus(list(paste("LS_NS_", i, ".sgrd", sep = ""),
                  paste("R_", i, ".sgrd", sep = ""),
                  paste("K_", i, ".sgrd", sep = ""),
                  paste("C_NS_", i, ".sgrd", sep = ""),
                  "snet_mask_NA.sgrd"), #input grids
              paste("ER_NS_", i, ".sgrd", sep = ""),
              ~  a * b * c * d * e, env = env) #a and are the input grids

# Ktc values will be sampled from uniform distributions
KtcHigh <- round(runif(1, 1, 200)) #choose the range -- in this case min = 1 and max = 200
KtcLow <- round(runif(1, 1, ifelse(
  KtcHigh < 100, KtcHigh - 1, 100))
  ) # choose the range -- in this case min = 1 and max = KtcHigh - 1
# parcel connectivity for border cells
Parcel.Connectivity <- runif(1, 0, 1) # transport capacitiy at border cells will be 1 - PCon

# Create parcel connectivity grids
# GS
rsaga.geoprocessor(lib = "grid_tools", 15,
              list(INPUT = "pcon_GS.sgrd", # this grid identifies the forest or grassland cells
                  # that border arable fields
                  RESULT = paste("pcon_GS_", i, ".sgrd", sep = ""),
                  OLD = 0,
                  NEW = 1 - Parcel.Connectivity,
                  SOPERATOR = 0), env = env)
# NS
```

```
rsaga.geoprocessor(lib = "grid_tools", 15,
                   list(INPUT = "pcon_NS.sgrd",
                        RESULT = paste("pcon_NS_", i, ".sgrd", sep = ""),
                        OLD = 0,
                        NEW = 1 - Parcel.Connectivity,
                        SOPERATOR = 0), env = env)

# Create Ktc grid GS
# High and low Ktc values will be assigned according to your c factor grid
rsaga.geoprocessor(lib = "grid_tools", 15,
                   list(INPUT = paste("C_GS_", i, ".sgrd", sep = ""),
                        RESULT = paste("Ktc_GS_", i, ".sgrd", sep = ""),
                        METHOD = 1,
                        MAX = 0.01, # Areas with C factor values above this value will receive KtcHigh
                        RNEW = KtcLow, # Remaining will receive KtcLow
                        OTHEROPT = T,
                        OTHERS = KtcHigh), env = env)

# Calculate the slope term of the TC equation (LS - aSir)
rsaga.grid.calculus(list(paste("LS_GS_", i, ".sgrd", sep = ""),
                         "slope_mm.sgrd", "snet_mask_NA.sgrd"), #i nput grids
                    paste("term_GS_", i, ".sgrd", sep = ""), # output grid
                    ~ (a - (4.12 * b ^ 0.8)) * c, env = env) # a, b, c and d are the input grids

# Then reclassify if negative (e.g. for cells in which the LS factor is ~0)
rsaga.geoprocessor(lib = "grid_tools", 15,
                   list(INPUT = paste("term_GS_", i, ".sgrd", sep = ""),
                        RESULT = paste("term_final_GS_", i, ".sgrd", sep = ""),
                        METHOD = 0,
                        OLD = 0,
                        NEW = 0,
                        SOPERATOR = 2,
```

```
                    OTHEROPT = F), env = env)


# Create Transport Capacity grid
# road mask is used to give zero TC to roads
# stream mask is used to give streams NoData values
# parcel connectivity will reduce the transport capacity based on the sampled value
rsaga.grid.calculus(list(paste("Ktc_GS_", i, ".sgrd", sep = ""),
                    paste("K_", i, ".sgrd", sep = ""),
                    paste("R_", i, ".sgrd", sep = ""),
                    paste("term_final_GS_", i, ".sgrd", sep = ""),
                    "snet_mask_NA.sgrd", "road_mask_0.sgrd",
                    paste("pcon_GS_", i, ".sgrd", sep = "")), #input grids
                  paste("TC_R0_GS_", i, ".sgrd", sep = ""), #output grid
                  ~a * b * c * d * e * f * g, env = env) #a, b, c and d are the input grids


# Create Transport Capacity grid
# road mask is used to give very high TC values to roads
# stream mask is used to give streams NoData values
rsaga.grid.calculus(list(paste("Ktc_GS_", i, ".sgrd", sep = ""),
                    paste("K_", i, ".sgrd", sep = ""),
                    paste("R_", i, ".sgrd", sep = ""),
                    paste("term_final_GS_", i, ".sgrd", sep = ""),
                    "snet_mask_NA.sgrd", "road_mask_high.sgrd",
                    paste("pcon_GS_", i, ".sgrd", sep = "")), #input grids
                  paste("TC_RVH_GS_", i, ".sgrd", sep = ""), #output grid
                  ~a * b * c * d * e * f * g, env = env) #a, b, c and d are the input grids


# Create accumulation function grids for roads with zero TC
# This will route sediments based on the erosion grid (supply) and the TC grid (control)
rsaga.geoprocessor(lib = "grid_analysis", 18,
                   list(SURFACE = "sf_dem.sgrd",
                     INPUT = paste("ER_GS_", i, ".sgrd", sep = ""),
```

```
                CONTROL = paste("TC_R0_GS_", i, ".sgrd", sep = ""),
                FLUX = paste("SedFlux_R0_GS_", i, ".sgrd", sep = ""), # These are sediment transport
rates
                STATE_OUT = paste("Deposition_R0_GS_", i, ".sgrd", sep = ""), # These are deposition
rates
                OPERATION = 1,
                LINEAR = F), env = env)


 # Create accumulation function grids for roads with very high TC
 # This will route sediments based on the erosion grid (supply) and the TC grid (control)
 rsaga.geoprocessor(lib = "grid_analysis", 18,
            list(SURFACE = "sf_dem.sgrd",
                INPUT = paste("ER_GS_", i, ".sgrd", sep = ""),
                CONTROL = paste("TC_RVH_GS_", i, ".sgrd", sep = ""),
                FLUX = paste("SedFlux_RVH_GS_", i, ".sgrd", sep = ""), #These are sediment transport
rates
                STATE_OUT = paste("Deposition_RVH_GS_", i, ".sgrd", sep = ""), #These are deposition
rates
                OPERATION = 1,
                LINEAR = F), env = env)


 # Combine soil loss and deposition grids
 # the mean of the resulting grid is equal to area specific sediment yield
 erosion_GS <- readGDAL(paste("ER_GS_", i, ".sdat", sep = ""))@data$band1 #read as matrix
 deposition_R0_GS <- readGDAL(paste("Deposition_R0_GS_", i, ".sdat", sep = ""))@data$band1
#read as matrix
 deposition_RVH_GS <- readGDAL(paste("Deposition_RVH_GS_", i, ".sdat", sep = ""))@data$band1
#read as matrix
 erdep_R0_GS <- (deposition_R0_GS - erosion_GS) #kg/m2
 erdep_RVH_GS <- (deposition_RVH_GS - erosion_GS) #kg/m2
 sedload_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1, total load,
TCroads = 0, roads as sinks
 sedload_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1, total load,
TCroads = very high
 saveRDS(erdep_R0_GS, file = (paste("ErDep_R0_GS_", i, ".rds", sep = ""))) #saving as RDS
```

```
  saveRDS(erdep_RVH_GS, file = (paste("ErDep_RVH_GS_", i, ".rds", sep = ""))) #saving as RDS


  # Calculate total load for roads as hydrological shortcuts

  erdep_RT_GS <- erdep_R0_GS * Mask_roads #deposited sediment at roads becomes zero,
increasing total total load to same extent

  sedload_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1

  saveRDS(erdep_RT_GS, file = (paste("ErDep_RT_GS", i, ".rds", sep = "")))


  # Calculate sediment loads per sub-catchment (road as sinks)

  Load.Ron_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr

  Load.Spittlisbach_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm = T)
#ton/yr

  Load.Mulibach_R0_GS <- sum(((erdep_R0_GS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # Calculate sediment loads per sub-catchment (roads as )

  Load.Ron_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr

  Load.Spittlisbach_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm
= T) #ton/yr

  Load.Mulibach_RVH_GS <- sum(((erdep_RVH_GS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # Calculate sediment loads per sub-catchment (roads as short-cuts)

  Load.Ron_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr
```

```
  Load.Spittlisbach_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm = T)
#ton/yr

  Load.Mulibach_RT_GS <- sum(((erdep_RT_GS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # Repeat everything without grass-strips

  # Create Ktc grid NS

  # High and low Ktc values will be assigned according to your c factor grid

  rsaga.geoprocessor(lib = "grid_tools", 15,

                     list(INPUT = paste("C_NS_", i, ".sgrd", sep = ""),

                          RESULT = paste("Ktc_NS_", i, ".sgrd", sep = ""),

                          METHOD = 1,

                          MAX = 0.01, #Areas with C factor values above this value will receive KtcHigh

                          RNEW = KtcLow, #Remaining will receive KtcLow... change these values as you will

                          OTHEROPT = T,

                          OTHERS = KtcHigh), env = env)


  # Calculate the slope term of the TC equation (LS - aSir)

  rsaga.grid.calculus(list(paste("LS_NS_", i, ".sgrd", sep = ""), "slope_mm.sgrd",
"snet_mask_NA.sgrd"), #input grids

                      paste("term_NS_", i, ".sgrd", sep = ""), #output grid

                      ~ (a - (4.12 * b ^ 0.8)) * c, env = env) #a, b, c and d are the input grids

  # then reclassify if negative

  rsaga.geoprocessor(lib = "grid_tools", 15,

                     list(INPUT = paste("term_NS_", i, ".sgrd", sep = ""),

                          RESULT = paste("term_final_NS_", i, ".sgrd", sep = ""),

                          METHOD = 0,

                          OLD = 0,

                          NEW = 0,

                          SOPERATOR = 2,

                          OTHEROPT = F), env = env)


  # Create Transport Capacity grid
```

```
 # road mask is used to give zero TC to roads
 # stream mask is used to give streams NoData values
 rsaga.grid.calculus(list(paste("Ktc_NS_", i, ".sgrd", sep = ""),
                          paste("K_", i, ".sgrd", sep = ""),
                          paste("R_", i, ".sgrd", sep = ""),
                          paste("term_final_NS_", i, ".sgrd", sep = ""),
                          "snet_mask_NA.sgrd", "road_mask_0.sgrd",
                          paste("pcon_NS_", i, ".sgrd", sep = "")), #input grids
                     paste("TC_R0_NS_", i, ".sgrd", sep = ""), #output grid
                     ~a * b * c * d * e * f * g, env = env) #a, b, c and d are the input grids

 # Create Transport Capacity grid
 # road mask is used to give very high TC values to roads
 # stream mask is used to give streams NoData values
 rsaga.grid.calculus(list(paste("Ktc_NS_", i, ".sgrd", sep = ""),
                          paste("K_", i, ".sgrd", sep = ""),
                          paste("R_", i, ".sgrd", sep = ""),
                          paste("term_final_NS_", i, ".sgrd", sep = ""),
                          "snet_mask_NA.sgrd", "road_mask_high.sgrd",
                          paste("pcon_NS_", i, ".sgrd", sep = "")), #input grids
                     paste("TC_RVH_NS_", i, ".sgrd", sep = ""), #output grid
                     ~a * b * c * d * e * f * g, env = env) #a, b, c and d are the input grids

 # Create accumulation function grids for roads with TC = 0
 # This will route sediments based on the erosion grid (supply) and the TC grid (control)
 rsaga.geoprocessor(lib = "grid_analysis", 18,
                    list(SURFACE = "sf_dem.sgrd",
                         INPUT = paste("ER_NS_", i, ".sgrd", sep = ""),
                         CONTROL = paste("TC_R0_NS_", i, ".sgrd", sep = ""),
                         FLUX = paste("SedFlux_R0_NS_", i, ".sgrd", sep = ""), #These are sediment transport
rates
```

```
                STATE_OUT = paste("Deposition_R0_NS_", i, ".sgrd", sep = ""), #These are deposition
rates
                OPERATION = 1,
                LINEAR = F), env = env)

 # Create accumulation function grids for roads with TC = very high
 # This will route sediments based on the erosion grid (supply) and the TC grid (control)
 rsaga.geoprocessor(lib = "grid_analysis", 18,
             list(SURFACE = "sf_dem.sgrd",
                INPUT = paste("ER_NS_", i, ".sgrd", sep = ""),
                CONTROL = paste("TC_RVH_NS_", i, ".sgrd", sep = ""),
                FLUX = paste("SedFlux_RVH_NS_", i, ".sgrd", sep = ""), #These are sediment transport
rates
                STATE_OUT = paste("Deposition_RVH_NS_", i, ".sgrd", sep = ""), #These are deposition
rates
                OPERATION = 1,
                LINEAR = F), env = env)

 # Combine soil loss and deposition grids
 # the mean of the resulting grid is equal to area specific sediment yield
 erosion_NS <- readGDAL(paste("ER_NS_", i, ".sdat", sep = ""))@data$band1
 deposition_R0_NS <- readGDAL(paste("Deposition_R0_NS_", i, ".sdat", sep = ""))@data$band1
 deposition_RVH_NS <- readGDAL(paste("Deposition_RVH_NS_", i, ".sdat", sep = ""))@data$band1
 erdep_R0_NS <- (deposition_R0_NS - erosion_NS) #kg/m2
 erdep_RVH_NS <- (deposition_RVH_NS - erosion_NS) #kg/m2
 sedload_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1
 sedload_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1
 saveRDS(erdep_R0_NS, file = (paste("ErDep_R0_NS_", i, ".rds", sep = "")))
 saveRDS(erdep_RVH_NS, file = (paste("ErDep_RVH_NS_", i, ".rds", sep = "")))

 #  Load for short-cut scenario
 erdep_RT_NS <- erdep_R0_NS * Mask_roads
 sedload_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000), na.rm = T) #ton yr-1
```

```
  saveRDS(erdep_RT_NS, file = (paste("ErDep_RT_NS", i, ".rds", sep = "")))


  # calculate sediment loads (roads as sinks)

  Load.Ron_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr

  Load.Spittlisbach_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm = T)
#ton/yr

  Load.Mulibach_R0_NS <- sum(((erdep_R0_NS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # calculate sediment loads (roads as...)

  Load.Ron_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr

  Load.Spittlisbach_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm
= T) #ton/yr

  Load.Mulibach_RVH_NS <- sum(((erdep_RVH_NS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # calculate sediment loads (roads as short-cuts)

  Load.Ron_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000) * Ron * -1, na.rm = T) #ton/yr

  Load.Stagbach_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000) * Stagbach * -1, na.rm = T)
#ton/yr

  Load.Hohibach_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000) * Hohibach * -1, na.rm = T)
#ton/yr

  Load.Spittlisbach_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000) * Spittlisbach * -1, na.rm = T)
#ton/yr

  Load.Mulibach_RT_NS <- sum(((erdep_RT_NS * cell.size ^ 2) / 1000) * Mulibach * -1, na.rm = T)
#ton/yr


  # write sampled parameter values in a table and results
```

```
  # for each iteration

  data.frame(R = R, K = K, C_arable = C_arable, C_grass = C_grass, C_forest = C_forest, C_orchard =
C_orchard,

        C_vineyard = C_vineyard, PTEF = PTEF, KtcHigh = KtcHigh,

        KtcLow = KtcLow, Parcel.Connectivity = Parcel.Connectivity,

        sedload_R0_GS = sedload_R0_GS * -1, sedload_RVH_GS = sedload_RVH_GS * -1,

        sedload_RT_GS = sedload_RT_GS * -1,

        Load.Ron_R0_GS = Load.Ron_R0_GS, Load.Stagbach_R0_GS = Load.Stagbach_R0_GS,

        Load.Hohibach_R0_GS = Load.Hohibach_R0_GS, Load.Spittlisbach_R0_GS =
Load.Spittlisbach_R0_GS,

        Load.Mulibach_R0_GS = Load.Mulibach_R0_GS,

        Load.Ron_RVH_GS = Load.Ron_RVH_GS, Load.Stagbach_RVH_GS = Load.Stagbach_RVH_GS,

        Load.Hohibach_RVH_GS = Load.Hohibach_RVH_GS, Load.Spittlisbach_RVH_GS =
Load.Spittlisbach_RVH_GS,

        Load.Mulibach_RVH_GS = Load.Mulibach_RVH_GS,

        Load.Ron_RT_GS = Load.Ron_RT_GS, Load.Stagbach_RT_GS = Load.Stagbach_RT_GS,

        Load.Hohibach_RT_GS = Load.Hohibach_RT_GS, Load.Spittlisbach_RT_GS =
Load.Spittlisbach_RT_GS,

        Load.Mulibach_RT_GS = Load.Mulibach_RT_GS,

        sedload_R0_NS = sedload_R0_NS * -1, sedload_RVH_NS = sedload_RVH_NS * -1,

        sedload_RT_NS = sedload_RT_NS * -1,

        Load.Ron_R0_NS = Load.Ron_R0_NS, Load.Stagbach_R0_NS = Load.Stagbach_R0_NS,

        Load.Hohibach_R0_NS = Load.Hohibach_R0_NS, Load.Spittlisbach_R0_NS =
Load.Spittlisbach_R0_NS,

        Load.Mulibach_R0_NS = Load.Mulibach_R0_NS,

        Load.Ron_RVH_NS = Load.Ron_RVH_NS, Load.Stagbach_RVH_NS = Load.Stagbach_RVH_NS,

        Load.Hohibach_RVH_NS = Load.Hohibach_RVH_NS, Load.Spittlisbach_RVH_NS =
Load.Spittlisbach_RVH_NS,

        Load.Mulibach_RVH_NS = Load.Mulibach_RVH_NS,

        Load.Ron_RT_NS = Load.Ron_RT_NS, Load.Stagbach_RT_NS = Load.Stagbach_RT_NS,

        Load.Hohibach_RT_NS = Load.Hohibach_RT_NS, Load.Spittlisbach_RT_NS =
Load.Spittlisbach_RT_NS,

        Load.Mulibach_RT_NS = Load.Mulibach_RT_NS,
```

```
        Sim = i)


}
# Calculate time spent
end_time <- Sys.time()
end_time - start_time


# Save results
write.csv(x = WS_sims, "1-200.csv")
```