Predicting the spatial distribution of soil organic carbon stock in Swedish boreal forest using remotely sensed and site-specific variables

Kpade O. L. Hounkpatin[a1], Johan Stendahl[a], Mattias Lundblad[a] , Erik Karltun[a],

[a] Department of Soil and Environment, Swedish University of Agricultural Sciences, P.O. Box 7014, SE-75007, Uppsala, Sweden

Supplementary Information (SI)

[1] Correspondence : ozias.hounkpatin@slu.se

Figure SI 1: Prediction interval plot of the local and global random forest models for the humus layer, mineral soil and total SOC stock from the local and global Random

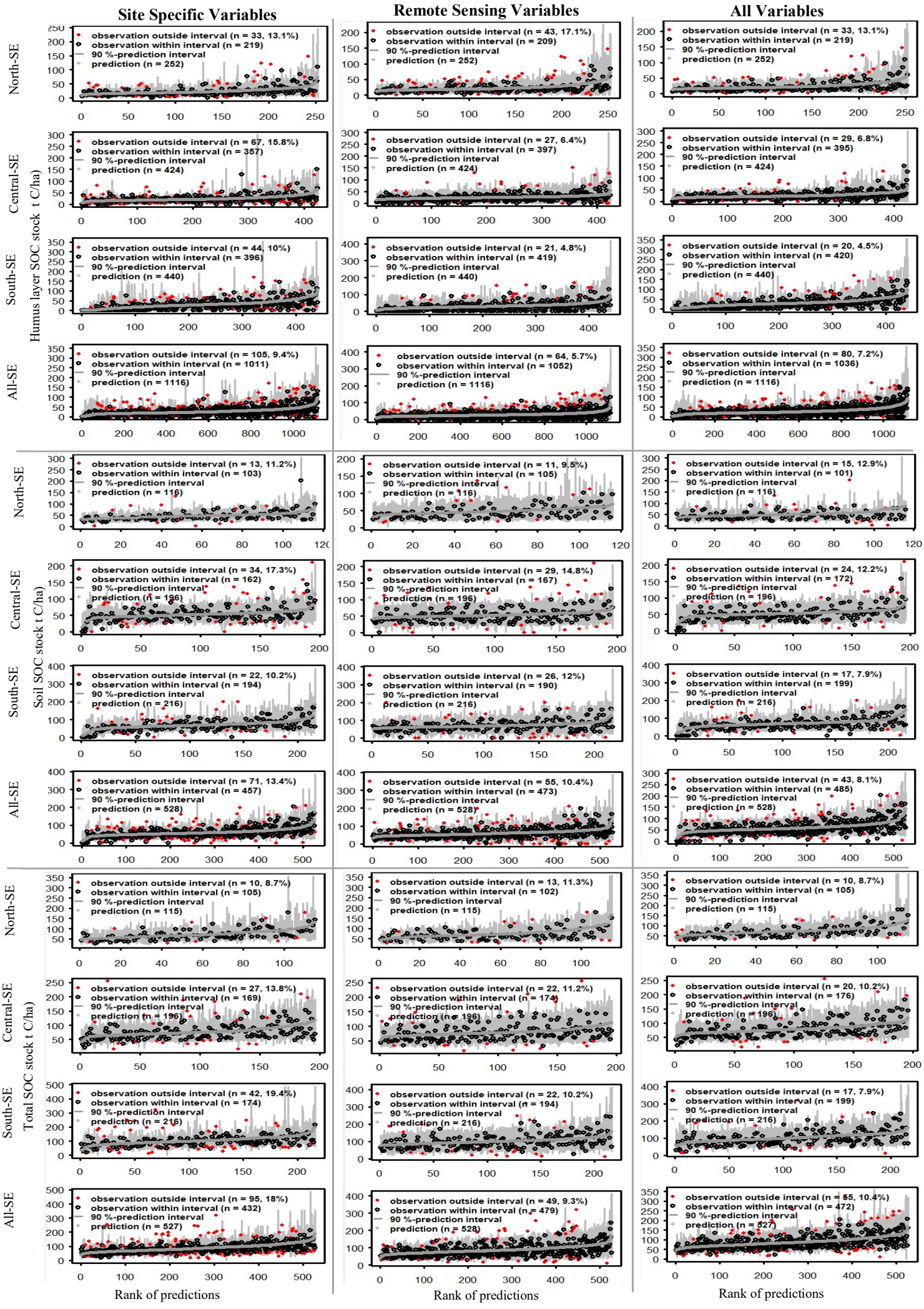


Figure SI 2: Correlation matrix of all variables with the humus layer SOC stock dataset over Sweden. Refer to table 1 for the definition of the abbreviation

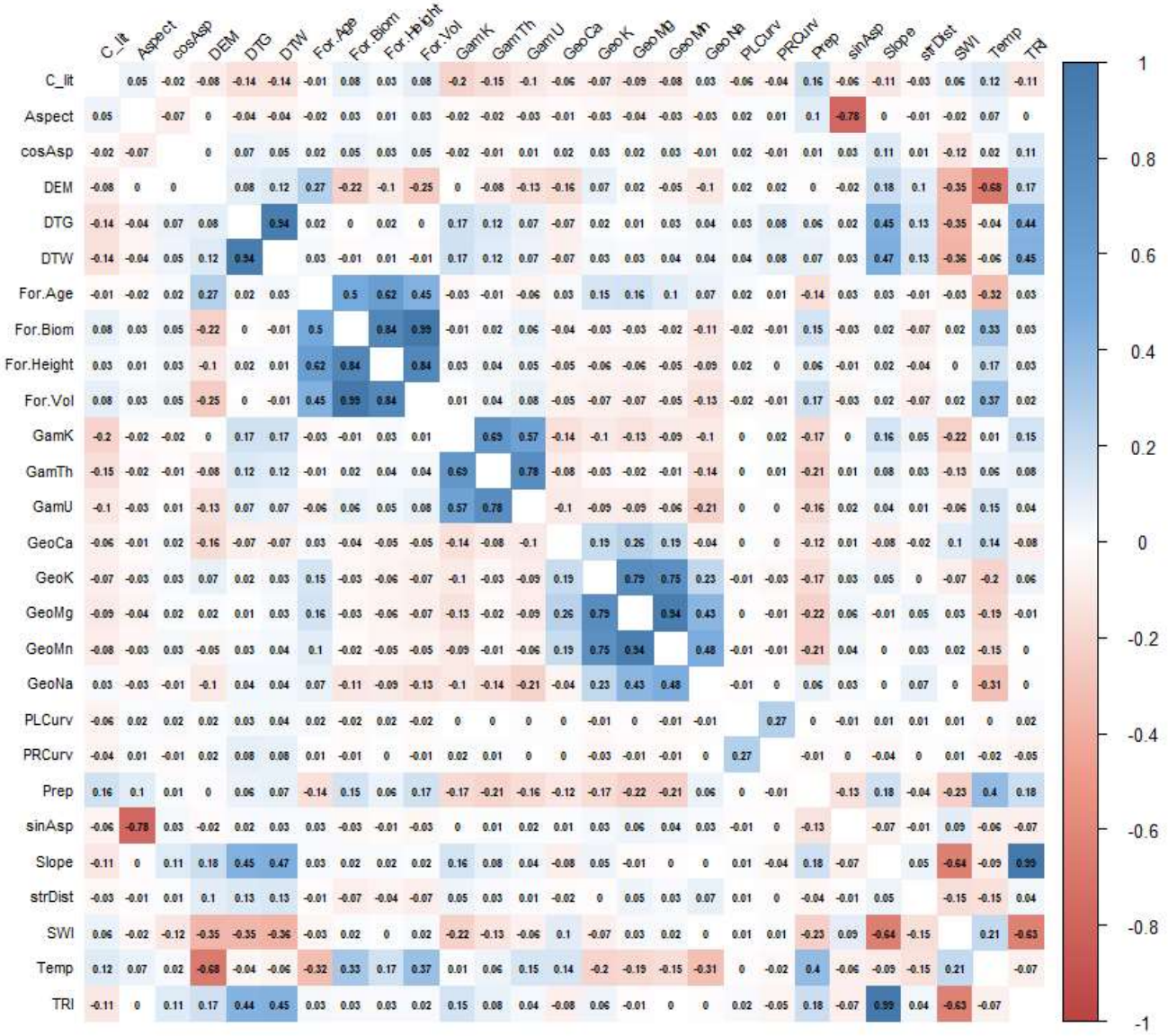


Figure SI 3: Correlation matrix of all variables with the mineral soil SOC stock dataset over Sweden. Refer to table 1 for the definition of the abbreviation

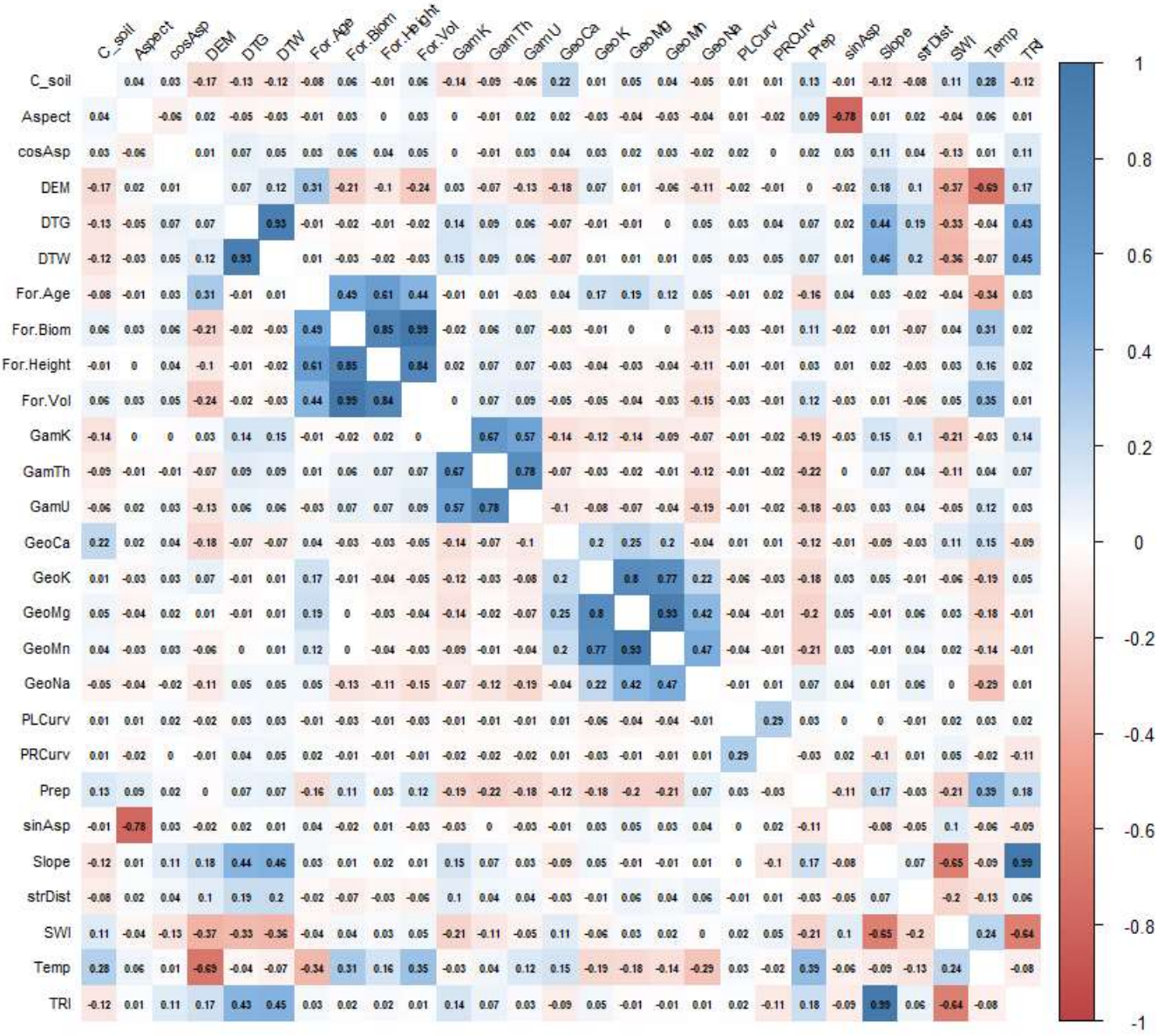


Figure SI 4: Correlation matrix of all variables with the total soil SOC stock dataset over Sweden. Refer to table 1 for the definition of the abbreviation

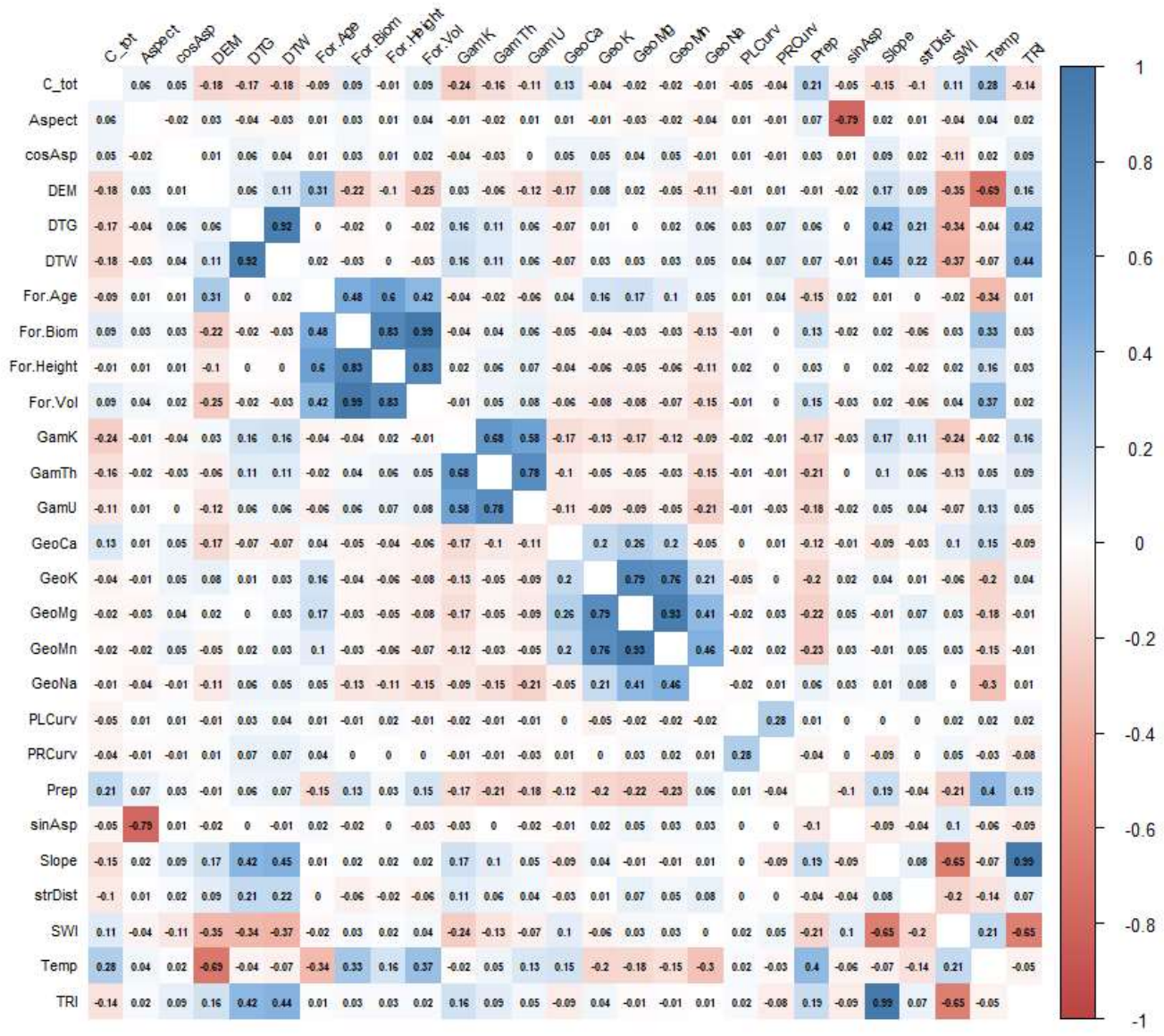


Figure SI 5: Variogram for the SOC stock and regression residuals of the local and global Random Forest models for the humus layer, mineral soil and total SOC stock. Log: log transformation, sqrt: square root transformation, Sph: spherical model, Ste: Matern, M. Stein's parameterization.

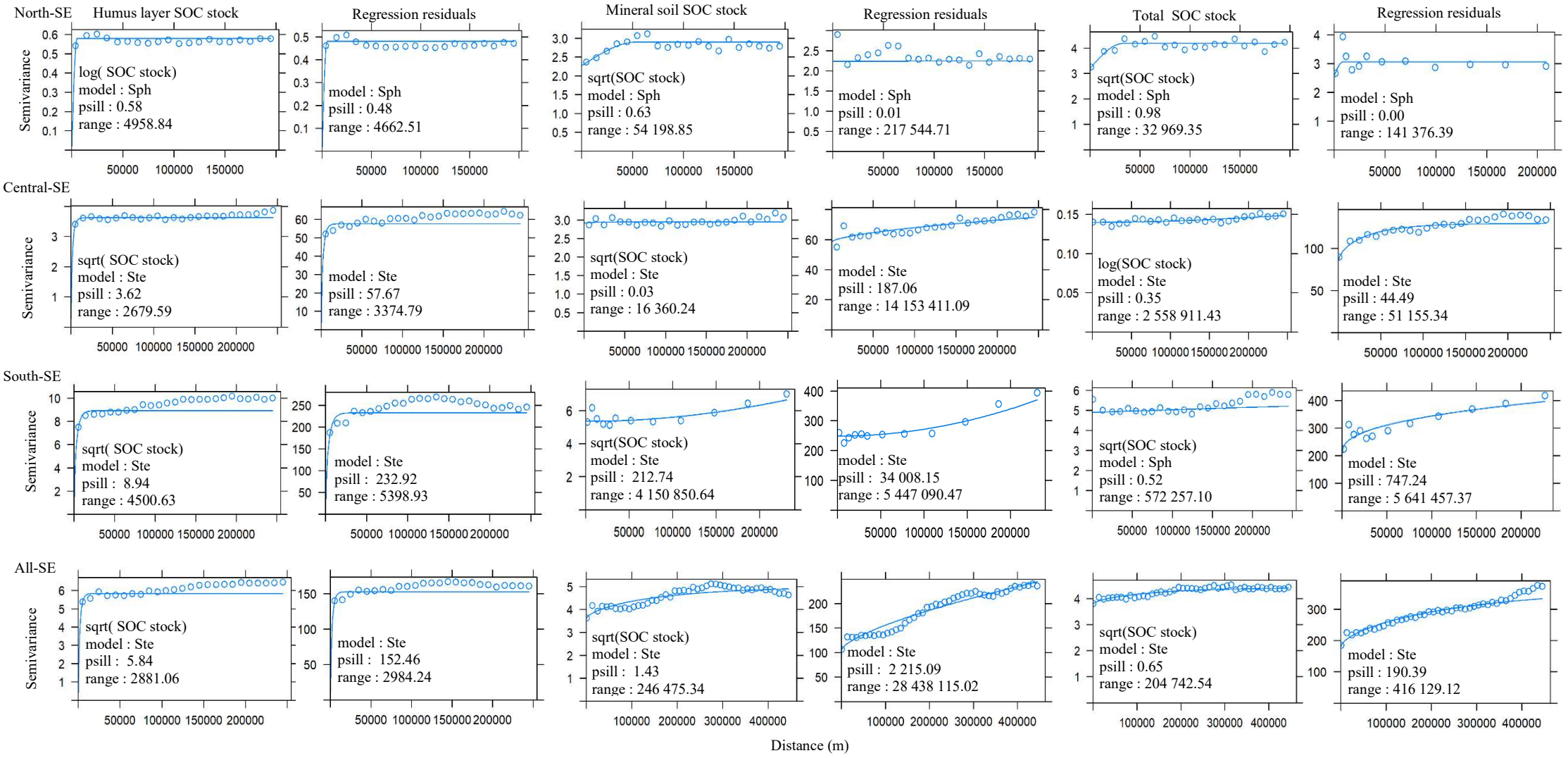


Figure SI 6: Box-plot for humus layer SOC stock by soil moisture, soil type, vegetation type, parent material and soil texture. Lines within the boxes give the median, red point within the boxes is the mean, boxes the 25th and 75th percentile (Refer to Table 2 for definition of the classes of each variable)

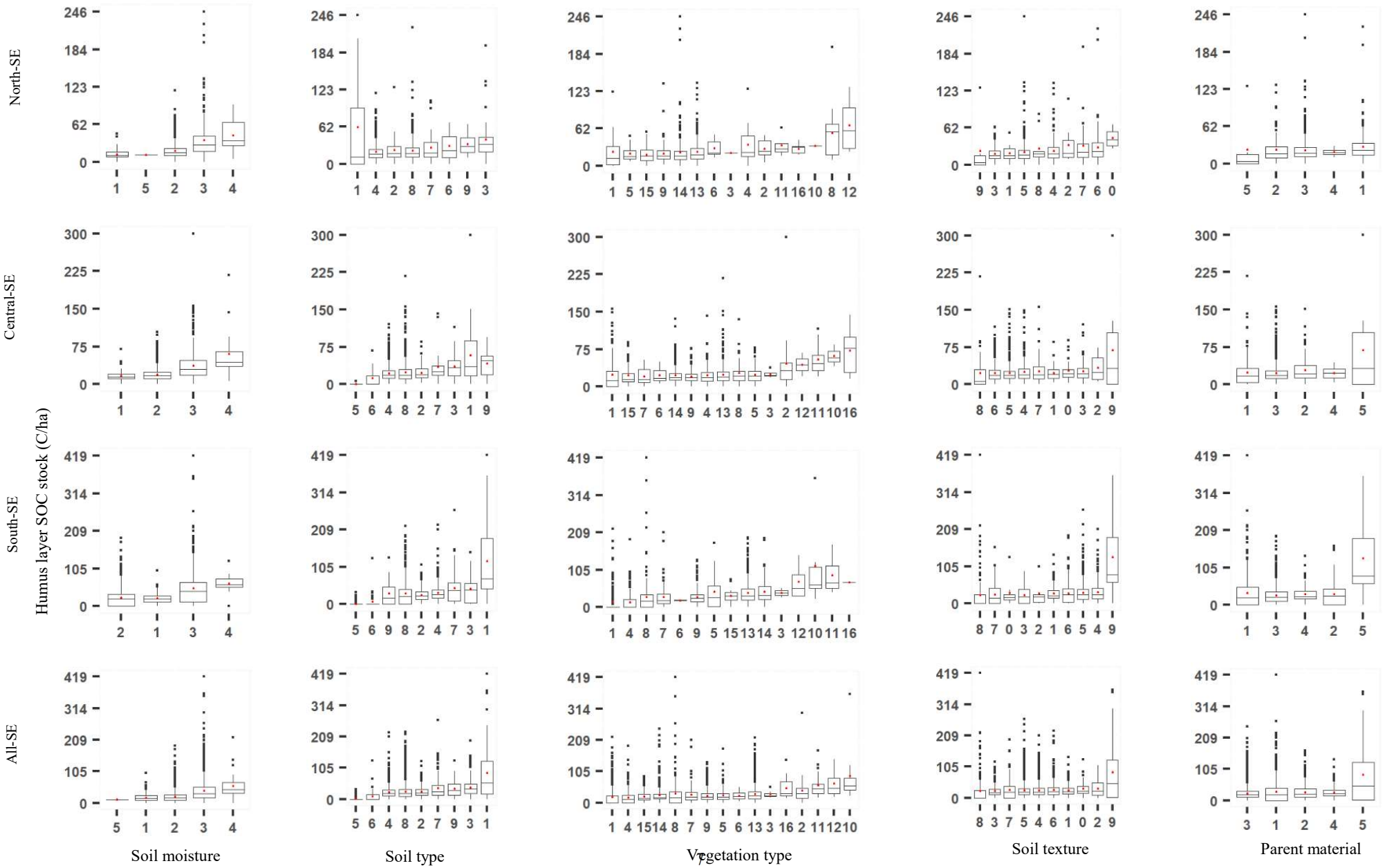


Figure SI 7: Box-plot for the mineral soil SOC stock by soil moisture, soil type, vegetation type, soil texture and parent material. Lines within the boxes give the median, red point within the boxes is the mean, boxes the 25th and 75th percentile (Refer to Table 2 for definition of the classes of each variable)

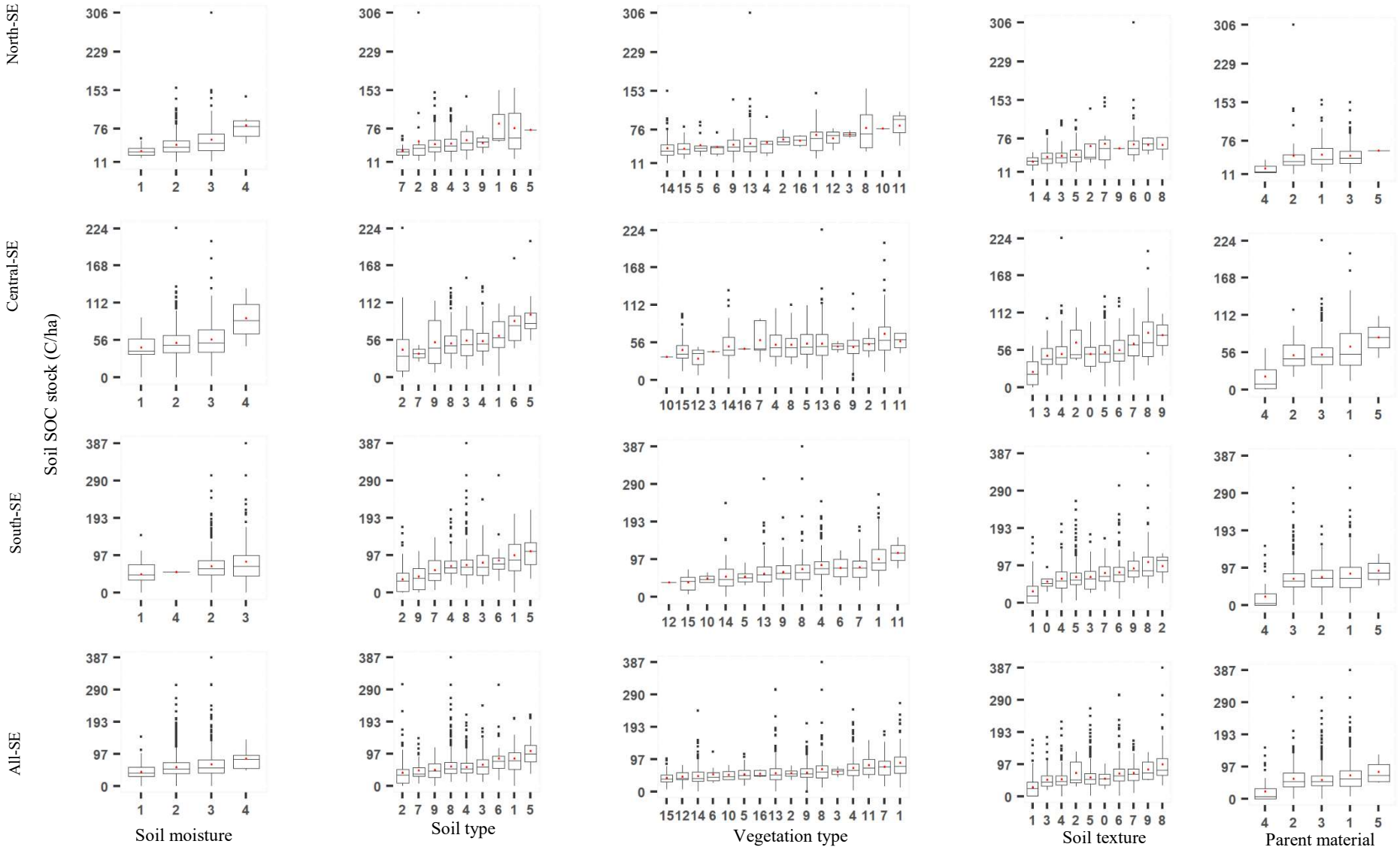


Figure SI 8: Box-plot for total SOC stock by soil moisture, soil type, vegetation type, soil texture and parent material. Lines within the boxes give the median, red point within the boxes is the mean, boxes the 25th and 75th percentile (Refer to Table 2 for definition of the classes of each variable)

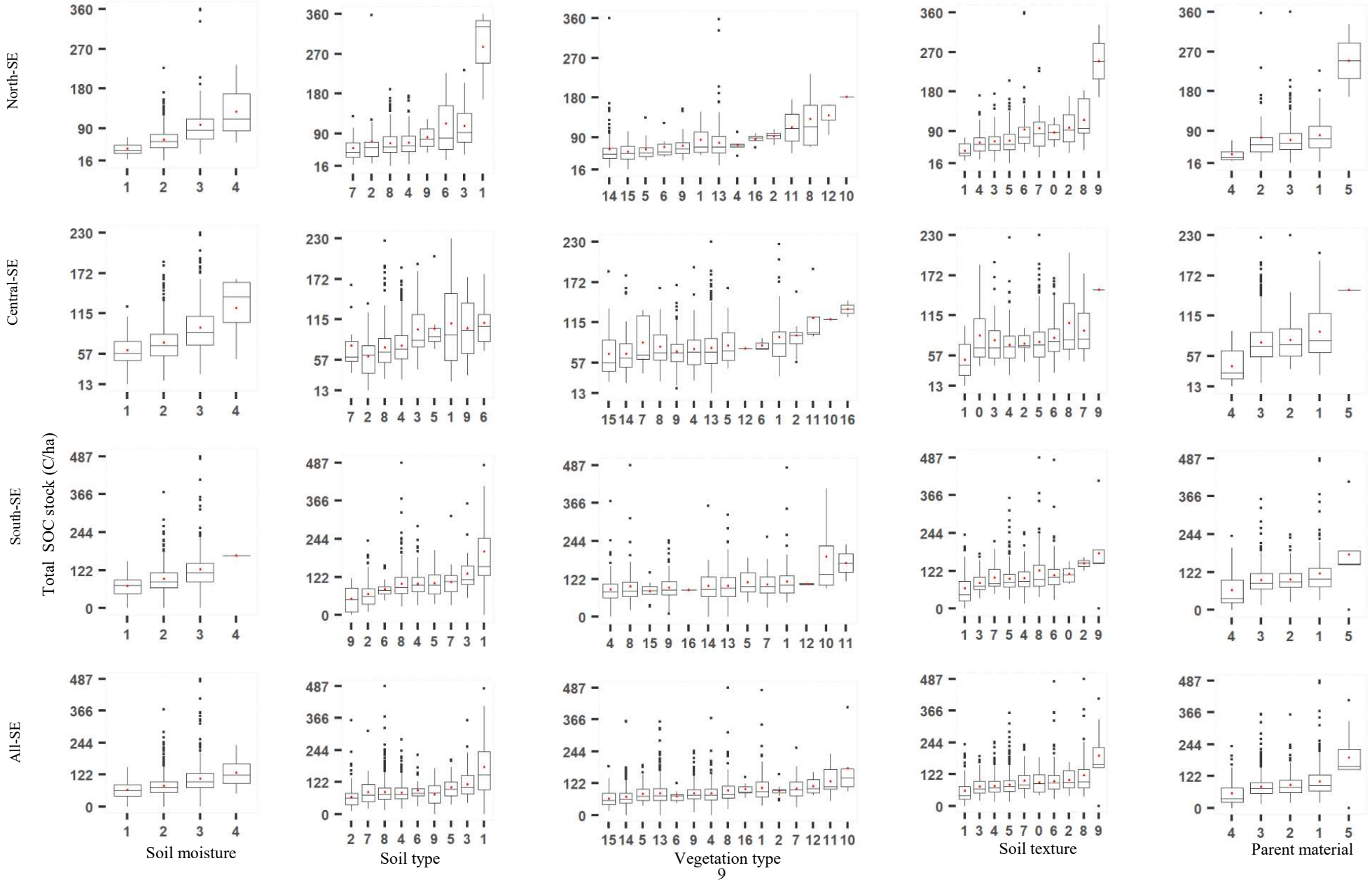
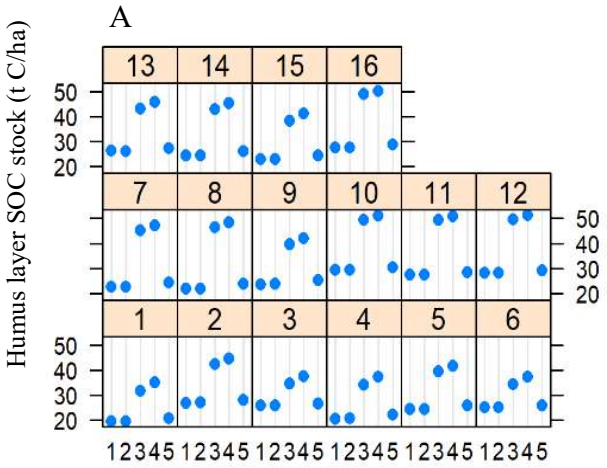
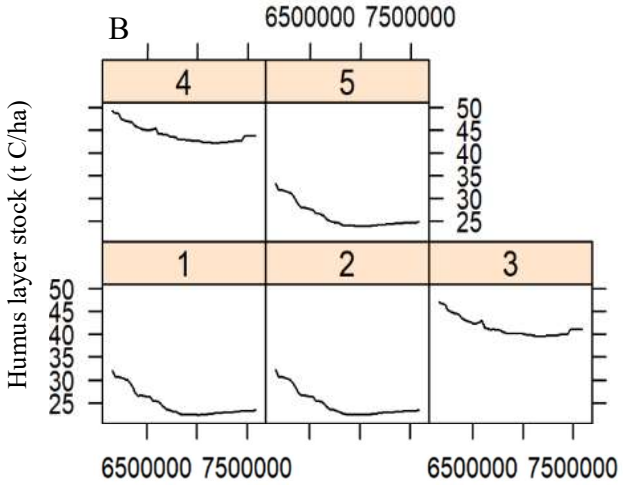


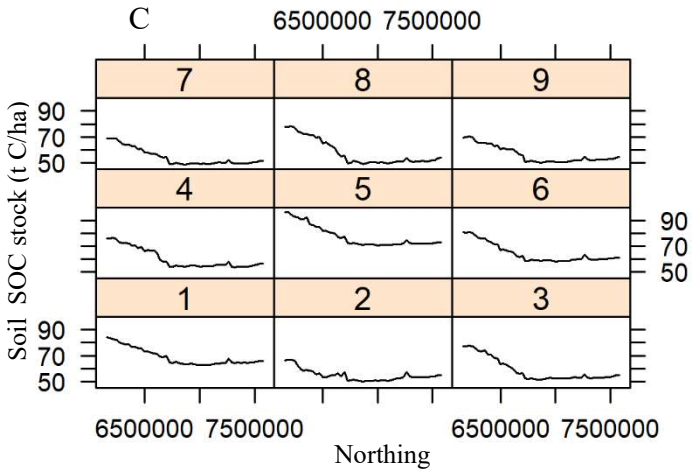
Figure SI 9: Partial dependence plots. Refer to Table 1 for the definition of the abbreviation and numbers for the classes of the variables.



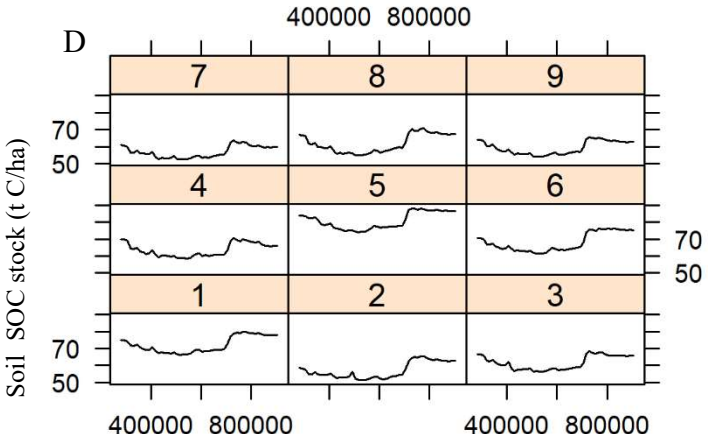
A: Partial plot between the humus layer SOC stock, soil moisture and vegetation type (RF Global model with all variables)



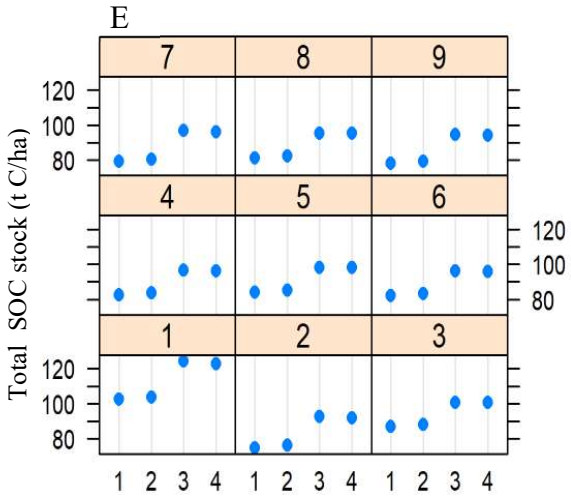
B: Partial plot between the humus layer SOC stock, soil moisture and Northing (RF Global model with all variables)



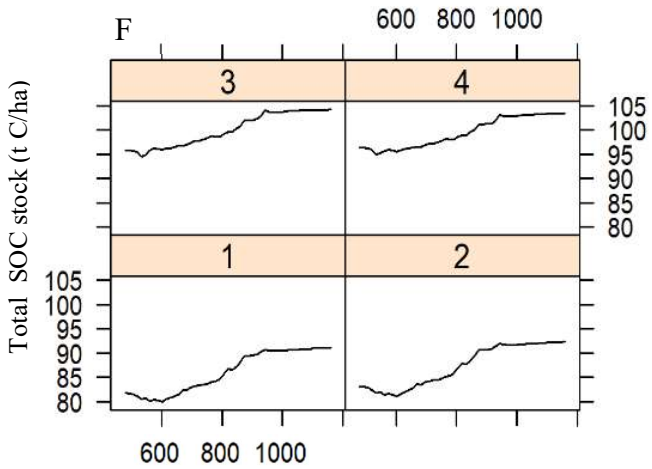
C: Partial plot between the mineral SOC stock, Northing and soil type (RF Global model with site specific variables)



D: Partial plot between the mineral SOC stock, Easting and soil type (RF Local model with site specific variables)

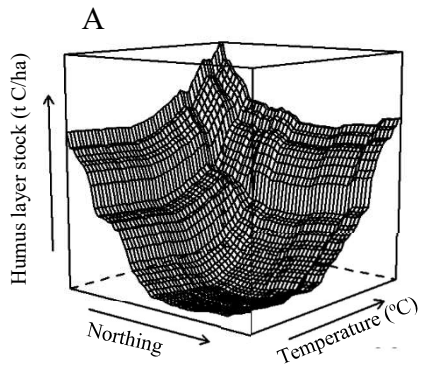


E: Partial plot between the total SOC stock, soil moisture and soil type (RF Global model all variables)

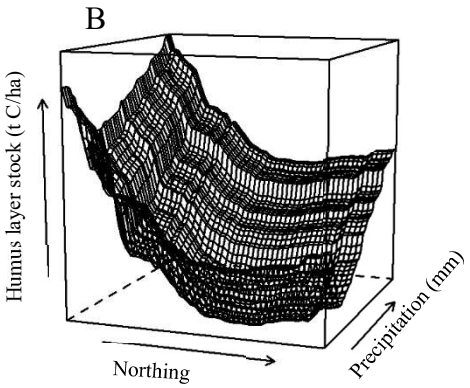


F: Partial plot between the total SOC stock, precipitation and soil moisture (RF Global model all variables)

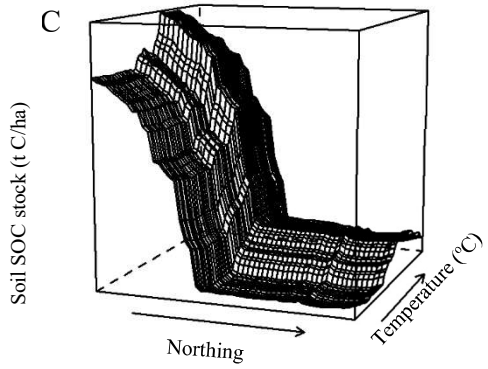
Figure SI 10: Partial dependence plots between SOC stock, northing, temperature and precipitation.



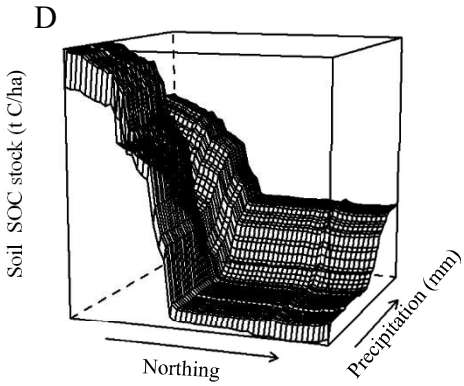
A: Partial plot between the humus layer SOC stock, Northing and Temperature (RF Global model with all variables)



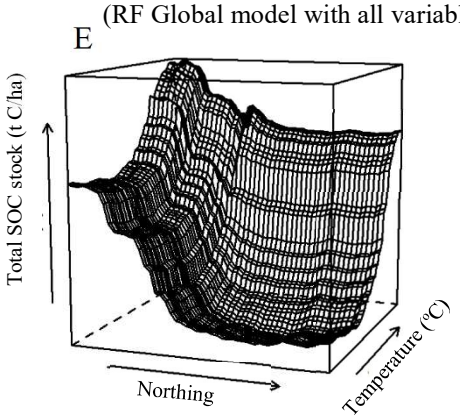
B: Partial plot between the humus layer SOC stock, Northing and precipitation (RF Global model with all variables)



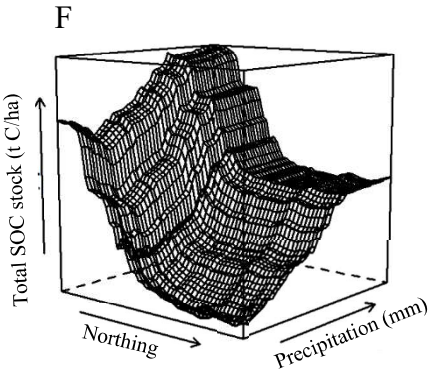
C: Partial plot between the mineral SOC stock, Northing and Temperature (RF Global model with all variables)



D: Partial plot between the mineral SOC stock, Northing and precipitation (RF Global model with all variables)



E: Partial plot between the total SOC stock, Northing and Temperature (RF Global model with all variables)



F: Partial plot between the total SOC stock, Northing and precipitation (RF Global model with all variables)

Asset SI 11: R code for SOC stock prediction

```
#-----------------------------------Script R SOC stock modelling ----------------------------------#
##############################################################################
#Author : Kpade Ozias L. Hounkpatin
#Paper : Predicting the spatial distribution of soil organic carbon stock in Swedish forests using remotely
#sensed and site-specific variables
#Code: The following code applies for litter layer, mineral soil and total soil carbon stock

# Load necessary Packages
library(quantregForest)
library(rgdal)
library(raster)
library(spatial.tools)
library(caret)
library(randomForest)
library(imputeTS)
library(stringr)
library(sp)
library(xgboost)
library(plyr)
library(dplyr)
library(parallel)
library(tidyverse)
library(Metrics)
library(ithir)
library(gridExtra)
library(doParallel)
library(pastecs)
library(utils)
library(broom) # for tidy()
library(knitr) # for kable()
library(stargazer)
library(pdp)
#Set options
options(digits = 2,scipen=999)
###Set my working directory
setwd("yourpath")

#-----------------------------------------------------------------------------------------------------------------------#
#-----Load data, Deal with NA values & remove problematic column-----------------------------------------#
#-----------------------------------------------------------------------------------------------------------------------#
#---Load the table
litterdata<- read.table("./path/litterdata.txt", header = T, sep = "\t")
#-- Change data class for site specific variables
#cor.train <- lapply(cor.train[:], as.factor), #insert where factor variables start and end in [:]
#cor.test<- lapply(cor.test[:], as.factor), #insert where factor variables start and end in [:]
#Transform into spatial soildata frame
coordinates(litterdata) <- ~EastC+NorthC
#Project the soildata into EPSG : 3006
crs<-CRS("+init=epsg:3006")
proj4string(litterdata) <- crs
#-Load the predictors
rlist<-list.files("path", pattern="tif$", full.names=TRUE)#Load in the covariates
covStack <- stack(rlist)#Stack all the covariates
#Extract multi value from covariates---TRAIN
litterdata_cov<- raster::extract(covStack, litterdata,sp = 1, method = "simple")
```

```
#-------------------------------------------------------------------------------------------------------------------------------#
#-----------------------train set, Validation set-----------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------------#
## Specify the percentage that will be used for training. the remaining will be used for testing
set.seed(1)
split <- 0.8
#### Training samples  for soil moisture
trainIndex <- createDataPartition(c(litterdata_cov["C_lit"], recursive=T), p=split, list = F)
pcDat<- liptData[trainIndex,] # Training set
ptDat<-liptData[-trainIndex,] # Validation set

#-------------------------------------------------------------------------------------------------------------------------------#
#------------------CHECKING HIGHLY CORRELATED COVARIATES----------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------------#
#Only for numeric variables
# corstarsl: https://github.com/kyuni22/ksmv/blob/master/functions/corstarsl.R
corstarsl <- function(x){
  require(Hmisc)
  x <- as.matrix(x)
  R <- rcorr(x)$r
  p <- rcorr(x)$P
  ## define notions for significance levels; spacing is important.
  mystars <- ifelse(p < .001, "***", ifelse(p < .01, "** ", ifelse(p < .05, "* ", " ")))
  ## trunctuate the matrix that holds the correlations to two decimal
  R <- format(round(cbind(rep(-1.11, ncol(x)), R), 2))[,-1]
  ## build a new matrix that includes the correlations with their apropriate stars
  Rnew <- matrix(paste(R, mystars, sep=""), ncol=ncol(x))
  diag(Rnew) <- paste(diag(R), " ", sep="")
  rownames(Rnew) <- colnames(x)
  colnames(Rnew) <- paste(colnames(x), "", sep="")
  ## remove upper triangle
  Rnew <- as.matrix(Rnew)
  Rnew[upper.tri(Rnew, diag = TRUE)] <- ""
  Rnew <- as.data.frame(Rnew)
  ## remove last column and return the matrix (which is now a data frame)
  Rnew <- cbind(Rnew[1:length(Rnew)-1])
  return(Rnew)
}
corr.data<-corstarsl(pcDat[sapply(pcDat,is.numeric)])
#Save correlation data
write.table(corr.data, file = str_c("./path/litterSOC/RFEdata_LitterSOC.txt"), sep = "\t",row.names=F)
descrCorr <- cor(pcDat[sapply(pcDat,is.numeric)])
highcorr <- caret::findCorrelation(descrCorr, cutoff = 0.80,names = TRUE)
cor.train<-pcDat %>% dplyr::select(-highcorr)
cor.test<-ptDat%>% dplyr::select(-highcorr)

#-------------------------------------------------------------------------------------------------------------------------------#
#------------------Feature Engineering_RFE-------------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------------#
xtrain <- cor.train[, 2:ncol(cor.train)]
xtest <-cor.test[, 2:ncol(cor.test)]
ytrain <- cor.train[, 1]
ytest <- cor.test[, 1]
rfe.ctrl = rfeControl(functions = rfFuncs,
                      method = "cv",
                      number = 50,
                      allowParallel = TRUE,
                      verbose = TRUE)
set.seed(1)
RF_rfe<- rfe(x=xtrain, y=ytrain,rfeControl = rfe.ctrl)
```

```
modelFile <-paste("./path",sep = "", "rfemodel_LitterSOC", ".rds")
saveRDS(object = RF_rfe, file = modelFile)
# Get names of the eliminated variables
RFNotsel<-xtrain %>% dplyr::select(-one_of(predictors(RF_rfe)[1:length(predictors(RF_rfe))]))
#Get final rfe train dataset
se_RF_train<-xtrain %>% dplyr::select(one_of(predictors(RF_rfe)[1:length(predictors(RF_rfe))]))
RF_train<-cbind(cor.train$C_lit,se_RF_train)
names(RF_train)[1] <- "C_lit" #Rename First column
write.table(RF_train, file = str_c("./path/train_rfe_LitterSOC.txt"), sep = "\t",row.names=F)
#Get final rfe test dataset
se_RF_test<-xtest %>% dplyr::select(one_of(predictors(RF_rfe)[1:length(predictors(RF_rfe))]))
RF_test<-cbind(cor.test$C_lit,se_RF_test)
names(RF_test)[1] <- "C_lit" #Rename First column
write.table(RF_test, file = str_c("./path/test_rfe_LitterSOC.txt"), sep = "\t",row.names=F)

#-----------------------------------------------------------------------------------------------------------------------#
#--------------------TUNE RF model-----------------------------------------------------------------------------------#
#-----------------------------------------------------------------------------------------------------------------------#
RFgrid <- expand.grid(.mtry = 2:ncol(RF_train))
control <- trainControl(method="repeatedcv", number=10,
                        repeats = 3,
                        verboseIter = TRUE,
                        returnData = "FALSE",
                        returnResamp = "all",
                        allowParallel = TRUE)
set.seed(1)
Tune_RF<- train(x=xtrain,y=ytrain,method = "rf",
                trControl=control,tuneGrid = RFgrid)
#Save the model
modelFile <-paste("./path",sep = "", "tunemodel_LitterSOC", ".rds")
saveRDS(object = Tune_RF, file = modelFile)
RF_tune_Table<-as.data.frame(Tune_RF$bestTune)

#-----------------------------------------------------------------------------------------------------------------------#
#------------------CROSS VALIDATION with RF-------------------------------------------------------------------------#
#-----------------------------------------------------------------------------------------------------------------------#
control <- trainControl(method="repeatedcv", number=10,
                        repeats=5,
                        verboseIter = TRUE,
                        returnData = "FALSE",
                        returnResamp = "all",
                        allowParallel = TRUE)
RFgrid<-expand.grid(mtry = RF_tune_Table$mtry)
set.seed(7)
fit_RF <- train(x=RF_train[,2:ncol(RF_train)],
                y=RF_train$C_lit,method = "rf",
                trControl=control,
                tuneGrid = RFgrid,
                importance = TRUE)
modelFile <-paste("./path", sep="","CVmodelRF_LitterSOC",".rds")
saveRDS(object = fit_RF, file = modelFile)
#Get and Save resampling statistics
CV_RF_Stat.desc<-as.data.frame(stat.desc(fit_RF$resample,basic=FALSE))
write.table(CV_RF_Stat.desc, file = str_c("./path/Stat.desc_LitterSOC.txt"), sep = "\t",row.names=F)
#Predict for external validation
C_lit.pred.RF<- as.data.frame(predict(fit_RF, xtest))
#Get external validation statistics and save to disk
obs.C_lit.pred.RF<-as.data.frame(cbind(ytest, C_lit.pred.RF))
names(obs.C_lit.pred.RF)[1]<-"obs"
names(obs.C_lit.pred.RF)[2]<-"pred"
```

```
RF.goof<-as.data.frame(goof(obs.C_lit.pred.RF$obs,obs.C_lit.pred.RF$pred))
#Write table for Results EV for RF
RF.goof$mae<-mae(obs.C_lit.pred.RF$obs,obs.C_lit.pred.RF$pred)
write.table(RF.goof, file = str_c("./path/goofRF_LitterSOC.txt"), sep = "\t",row.names=F)

#-------------------------------------------------------------------------------------------------------------------------#
#-----------------Variable Importance-----------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------#
VI.RF<-varImp(fit_RF)
RF.VImp<-as.data.frame(VI.RF[1])
names(RF.VImp)[1]<-"VImp.RF"
RF.VImp<-as_tibble(setNames(cbind(rownames(RF.VImp), RF.VImp, row.names = NULL), c("Factors",
"Importance")))
RF.VImp<-as.data.frame(RF.VImp[order(-RF.VImp$Importance),])
write.table(RF.VImp, file = str_c("./path/VarImpCVRF_LitterSOC.txt"), sep = "\t")

#-------------------------------------------------------------------------------------------------------------------------#
#-----------------Density Plots-----------------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------#
RF_SOC.test<-cbind(RF_test$C_lit,C_lit.pred.RF)
RF_SOC.test<-setNames(RF_SOC.test, c("C_lit","RF_pred"))
write.table(RF_SOC.test, file = str_c("path/Obs_PredRF.data_LitterSOC.txt"), sep = "\t",row.names=F)
# RF_density plots with actual and predicted SOC
RF_density<-RF_SOC.test[,c(1,length(RF_SOC.test))]
RF_density$Type1<-"Actual"
RF_density$Type2<-"PredictedC"
RF_d.soil<-RF_density[,c(1,3)]
RF_d.pred<-RF_density[,c(2,4)]
RF_d.soil<-setNames(RF_d.soil, c("SOC",         "Type"))
RF_d.pred<-setNames(RF_d.pred, c("SOC",       "Type"))
RF_density.data<-(as.data.frame(rbind(RF_d.soil,RF_d.pred)))
RF_predType<-as.data.frame(ddply(RF_density.data, "Type", summarise, rating.pred=mean(SOC)))
write.table(RF_predType, file = str_c("./path/densityRF_predType_LitterSOC.txt"), sep = "\t",row.names=F)
write.table(RF_density.data, file = str_c("./path/densityRF_data_LitterSOC.txt"), sep = "\t",row.names=F)
# Density plots with semi-transparent fill
N10RF_densityP_lit<-ggplot(RF_density.data, aes(x=SOC, fill=Type))+geom_density(alpha=0.4)+
  geom_vline(data=RF_predType, aes(xintercept=rating.pred,  colour=Type),linetype="dashed")+
          labs(y = "Density", x = "SOC stock",title = "litter SOC stock",
          linetype="dashed", size=1,show.legend=FALSE)+
          scale_color_manual(values=c("#999999", "#E69F00"))+
          scale_y_continuous(breaks=c(0,0.035, 0.065))+
  theme(axis.title.x = element_blank())+
  theme(legend.position = "none")+
  theme(axis.text=element_text(size=9, face="bold"),axis.title=element_text(size=10, face="bold"),#,face="bold"
       plot.title = element_text(size = 14, face = "bold",hjust = 0.5))+
  theme(panel.background = element_rect(fill = "white", colour = "#f7f9fb",size = 2, linetype = "solid"))

#-------------------------------------------------------------------------------------------------------------------------#
#-----------------Compute Partial dependance----------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------#
# Compute partial dependence data
var1<-as.character(RF.VImp["1",1])
var2<-as.character(RF.VImp["2",1])
var3<-as.character(RF.VImp["3",1])
RF_train<-as.data.frame(RF_train)
pd <- pdp::partial(fit_RF, pred.var = c(var1, var2),train=RF_train, rug=TRUE)
# Default PDP
pdp1 <- plotPartial(pd)
# Add contour lines and use a different color palette
rwb <- colorRampPalette(c("red", "white", "blue"))
```

```
pdp2 <- plotPartial(pd, contour = TRUE, col.regions = rwb,rug=TRUE,train=RF_train)
# 3-D surface
pdp3 <- plotPartial(pd, levelplot = FALSE, zlab = "C_lit", screen = list(z = -20, x = -80))
#plot(pdp3)
tiff("./path/3Dpdp_Var1_Var2__LitterSOC.tiff", height = 3, width = 4, units = 'in', res=300)
dev.off()
tiff("./path/grid_pdp_Var1_Var2_LitterSOC.tiff", height = 3, width = 6, units = 'in', res=300)
grid.arrange(pdp1, pdp2, pdp3,ncol = 3)
dev.off()
#-----------------------------------------------For other variables
p1d <- pdp::partial(fit_RF, pred.var = c(var1, var3),train=RF_train, rug=TRUE)
# Default PDP
pdp11<- plotPartial(p1d)
# Add contour lines and use a different color palette
rwb <- colorRampPalette(c("red", "white", "blue"))
pdp12 <- plotPartial(p1d, contour = TRUE, col.regions = rwb,rug=TRUE,train=RF_train)
# 3-D surface
pdp13 <- plotPartial(p1d, levelplot = FALSE, zlab = "C_lit", colorkey = FALSE,
                screen = list(z = -40, x = -80))
tiff("./path/3Dpdp_Var1_Var3CVRF_LitterSOC.tiff", height = 3, width = 4, units = 'in', res=300)
plot(pdp13)
dev.off()
tiff("./path/grid_pdp_Var1_Var3CVRF_LitterSOC.tiff", height = 3, width = 6, units = 'in', res=300)
grid.arrange(pdp11, pdp12, pdp13,ncol = 3)
dev.off()

#-------------------------------------------------------------------------------------------------------------------------------#
#------------------Prediction interval coverage probability ----------------------------------------------------------------#
#-------------------------------------------------------------------------------------------------------------------------------#
C_lit.quantRF <- quantregForest(x = RF_train[,2:(ncol(RF_train))],
                           y = RF_train$C_lit, mtry=RF_tune_Table$mtry,
                           ntree=500)
## predict 0.01, 0.02,..., 0.99 quantiles for validation data
C_lit.pred.distribution <- predict(C_lit.quantRF,
                                   newdata = RF_test,
                                   what = seq(0.01, 0.99, by = 0.01))
# Coverage probabilities plot
# create sequence of nominal probabilities
ss <- seq(0,1,0.01)
# compute coverage for sequence
t.prop.inside <- sapply(ss, function(ii){
  boot.quantile <-  t( apply(C_lit.pred.distribution, 1, quantile,
                             probs = c(0,ii) ) )[,2]
  return(sum(boot.quantile <= RF_test$C_lit)/nrow(RF_test))
})
plot.data<-as.data.frame(cbind(ss,t.prop.inside[length(ss):1]))
names(plot.data)[2]<-"Int"
plot<-ggplot(plot.data, aes(x = ss, y = Int))+
  geom_line() +
  geom_abline(color="red",linetype="dashed", size=0.3)+
  labs( x="Confidence level",y = "Coverage probabilities")+
  theme(
    axis.title.x = element_blank(), axis.title.y= element_blank(),
    panel.background = element_rect(fill = "white", colour = "#f7f9fb",size = 2, linetype = "solid"),
    axis.text=element_text(size=6, face="bold"),axis.title=element_text(size=12),#,face="bold"
    plot.background = element_rect(fill = "white")
  )
```