Journal of Research in Health Sciences
ISSN: 2228-7809 | https://jrhsjournal.com

Revised estimates of chronic disease burden using multisource epidemiological data Running Title: New chronic disease burden estimates

Dr. K. Yamamoto[1], Dr. L. Müller[2]*
[1] Department of Public Health and Epidemiology, Kyoto University School of Medicine, Kyoto 606-8501, Japan
[2] Institute of Biomedical Engineering and Health Informatics, Technical University of Munich, Munich 80333, Germany

R script: Cross validation and Variable importance of Cd, DH, D

```
# 00 Spilt initial data to OR_Cd, OR_DH, OR_D #
OR_data<- read.csv("NEAsia_plotdata.csv")
str(OR_data)
OR_Cd<-OR_data[,c(3,4:5,6:30,33:35,39,41:46)];str(OR_Cd)
OR_DH<-OR_data[,c(4,6:30,33:35,39,41:46)];str(OR_DH)
OR_D<-OR_data[,c(5,6:30,33:35,39,41:46)];str(OR_D)

write.csv(OR_Cd,"OR_Cd.csv",row.names=FALSE)
write.csv(OR_DH,"OR_DH.csv",row.names=FALSE)
write.csv(OR_D,"OR_D.csv",row.names=FALSE)

#####################################################################
# 01 Cross validation ----Cd
library(parallel)
library(foreach)
library(doParallel)

# read csv file #
ORCd.data<- read.csv("OR_Cd.csv")
str(ORCd.data)

# Deterc the number of cores of your computer
no_cores <- detectCores()
set.seed(132)
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORCd.data),
size=floor(.1*nrow(ORCd.data)), replace=F)))

registerDoParallel(no_cores[1]-1)
data.list = list()
```

```
CV_Cd <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                 i=1:iter, # Number of iterations
                 .combine = rbind, # you will add the outputs of each iteration as...
                 .multicombine=TRUE,
                 .packages=c("randomForest", "caret") # libraries that you will need in foreach function
) %dopar% {
  # Create training and testing datasets
  train_rf <- ORCd.data[-sample, ]
  test_rf  <- ORCd.data[ sample, ]
  # Train the model
  rf <- randomForest(Cd ~ ., data = train_rf,ntree=1000,mtry=13,nodesize=3)
  n1 = paste0(i, 'a')
  n2 = paste0(i, 'b')
  n3 = paste0(i, 'c')
  n4 = paste0(i, 'e')

  # 1 predict in testing dataset
  data.list[[n1]]= data.frame(sample, iteration=i)

  # 2 predict in testing dataset
  pred_RF <-  predict(rf, test_rf, type="response")
  data.list[[n2]]=data.frame(obs=test_rf$Cd, pred=pred_RF, iteration=i)

  # 3 variable importance
  data.list[[n3]]=data.frame(t(apply(importance(rf,type=2,scale=FALSE), 2, function(x) x/sum(importance(rf,type=2,scale=FALSE))*100)),iteration = i)
  # 4 static
  data.list[[n4]]=data.frame(R2= R2(pred_RF, test_rf$Cd),
                             RMSE = RMSE(pred_RF,test_rf$Cd),
                             MAE  = MAE(pred_RF,test_rf$Cd),
                             iteration = i)
  return(data.list)

}

stopImplicitCluster()
# 1) data.sample.RF_Cd
data.sample.RF_Cd<-
as.data.frame(cbind(CV_Cd[[1]]$sample,CV_Cd[[2]]$sample,CV_Cd[[3]]$sample,CV_Cd[[4]]$sample,CV_Cd[[5]]$sample,CV_Cd[[6]]$sample,CV_Cd[[7]]$sample,CV_Cd[[8]]$sample,CV_Cd[[9]]$sample,CV_Cd[[10]]$sample,CV_Cd[[11]]$sample,CV_Cd[[12]]$sample,CV
```

```
_Cd[[13]]$sample,CV_Cd[[14]]$sample,CV_Cd[[15]]$sample,CV_Cd[[16]]$sample,CV_Cd
[[17]]$sample,CV_Cd[[18]]$sample,CV_Cd[[19]]$sample,CV_Cd[[20]]$sample))

colnames(data.sample.RF_Cd) <- paste0("sample", i=1:20)

# 2) Predictions.RF_Cd
Predictions.RF_Cd<-
as.data.frame(rbind(CV_Cd[[21]],CV_Cd[[22]],CV_Cd[[23]],CV_Cd[[24]],CV_Cd[[25]],CV
_Cd[[26]],CV_Cd[[27]],CV_Cd[[28]],CV_Cd[[29]],CV_Cd[[30]],CV_Cd[[31]],CV_Cd[[32]
],CV_Cd[[33]],CV_Cd[[34]],CV_Cd[[35]],CV_Cd[[36]],CV_Cd[[37]],CV_Cd[[38]],CV_Cd
[[39]],CV_Cd[[40]]))

# 3) VIM_Cd
VIM_Cd<-as.data.frame(rbind(CV_Cd[[41]],CV_Cd[[42]],CV_Cd[[43]],CV_Cd[[44]]
                            ,CV_Cd[[45]],CV_Cd[[46]],CV_Cd[[47]],CV_Cd[[48]]
                            ,CV_Cd[[49]],CV_Cd[[50]],CV_Cd[[51]],CV_Cd[[52]]
                            ,CV_Cd[[53]],CV_Cd[[54]],CV_Cd[[55]],CV_Cd[[56]]
                            ,CV_Cd[[57]],CV_Cd[[58]],CV_Cd[[59]],CV_Cd[[60]]))

# 4) Static_C
STATIC.RF_Cd<-
as.data.frame(rbind(CV_Cd[[61]],CV_Cd[[62]],CV_Cd[[63]],CV_Cd[[64]],CV_Cd[[65]],
CV_Cd[[66]],CV_Cd[[67]],CV_Cd[[68]],CV_Cd[[69]],CV_Cd[[70]],CV_Cd[[71]],CV_Cd[[
72]]  ,CV_Cd[[73]],CV_Cd[[74]],CV_Cd[[75]],CV_Cd[[76]],CV_Cd[[77]],CV_Cd[[78]],CV
_Cd[[79]],CV_Cd[[80]]))
write.csv(data.sample.RF_Cd,"data.sample.RF_Cd.csv", row.names = FALSE)
write.csv(Predictions.RF_Cd,"Predictions.RF_Cd.csv", row.names = FALSE)
write.csv(VIM_Cd,"VIM_Cd.csv", row.names = FALSE)
write.csv(STATIC.RF_Cd,"STATIC.RF_Cd.csv", row.names = FALSE)

###############
library(caret)
library(xgboost)

ORCd.data <- read.csv("OR_Cd.csv")
data.sample.RF<-read.csv("data.sample.RF_Cd.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){

   sample <- data.sample.RF[,i]

   y <- ORCd.data[, 1]
```

```
x <- ORCd.data[, -1]

train_data      <- as.matrix(x[-sample, ])
train_labels <- as.numeric(y[-sample])

test_data      <- as.matrix(x[ sample, ])
test_labels <- as.numeric(y[ sample])

train    <- list(train_data, train_labels); names(train) <- c("data", "label")
test      <- list(test_data,    test_labels);    names(test)    <- c("data", "label")

dtrain <- xgb.DMatrix(data = train$data, label = train$label)
dtest    <- xgb.DMatrix(data = test$data,    label = test$label)


wlist <- list(train = dtrain, test = dtest)

XGB <- xgb.train(data = dtrain, watchlist = wlist,
                            max_depth =5,
                            eta = 0.1,
                            nrounds =50)


# PREDICTING WITH XGBoost MODEL

pred_XGB <-    predict(XGB, test$data)

# Saving results in vectors and data frames

Stimated <- data.frame(test_labels, pred_XGB, i)
STATIC<- data.frame( R2 = R2(pred_XGB, test_labels),
                                  RMSE = RMSE(pred_XGB, test_labels),
                                  MAE = MAE(pred_XGB,test_labels),i)
if(i == 1){

   Predictions.XGB <- Stimated
   STATIC.XGB<- STATIC

}else{

   Predictions.XGB <- rbind.data.frame(Predictions.XGB, Stimated)
   STATIC.XGB<- rbind.data.frame( STATIC.XGB,STATIC)
}
```

```
}

# You need this information to know which is the best model (XGB )
write.csv(Predictions.XGB, "Predictions.XGB_Cd.csv", row.names = FALSE)
write.csv(STATIC.XGB, "STATIC.XGB_Cd.csv", row.names = FALSE)

#####lm####

ORCd.data <- read.csv("OR_Cd.csv")
data.sample.RF<-read.csv("data.sample.RF_Cd.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){

   sample <- data.sample.RF[,i]

   train_lm <- (ORCd.data)[-sample, ] # Select 90% of the data to train the model
   test_lm <- (ORCd.data)[ sample, ] # Select 10% of the data to test the model

   # TRAINING lm

   LM <- lm(Cd ~., data=train_lm, singular.ok = TRUE)

   # PREDICTING WITH RANDOM FOREST MODEL

   pred_LM <-    predict(LM, test_lm)

   # Saving results in vectors and data frames

   Stimated <- data.frame(test_lm$Cd, pred_LM,i)
   STATIC<- data.frame( R2 = R2(pred_LM, test_lm$Cd),
                        RMSE = RMSE(pred_LM, test_lm$Cd),
                        MAE = MAE(pred_LM,test_lm$Cd),i)
   if(i == 1){

      Predictions.LM <- Stimated
      STATIC.LM<- STATIC

   }else{

      Predictions.LM <- rbind.data.frame(Predictions.LM, Stimated)
      STATIC.LM<- rbind.data.frame( STATIC.LM,STATIC)
   }
```

```
}

# You need this information to know which is the best model (LM )
write.csv(Predictions.LM, "Predictions.LM_Cd.csv", row.names = FALSE)
write.csv(STATIC.LM, "STATIC.LM_Cd.csv", row.names = FALSE)


#####################################################################
# 02 Cross validation ----DH
library(parallel)
library(foreach)
library(doParallel)

# 02 Cross validation ----DH
# read csv file
ORDH.data <- read.csv("OR_DH.csv")
str(ORDH.data)
# Deterc the number of cores of your computer
no_cores <- detectCores()
set.seed(132)
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORDH.data), size=floor(.1*nrow(ORDH.data)), replace=F)))

registerDoParallel(no_cores[1]-1)
data.list = list()

CV_DH <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                 i=1:iter, # Number of iterations
                 .combine = rbind, # you will add the outputs of each iteration as...
                 .multicombine=TRUE,
                 .packages=c("randomForest", "caret") # libraries that you will need in foreach function
) %dopar% {
  # Create training and testing datasets
  train_rf <- ORDH.data[-sample, ]
  test_rf  <- ORDH.data[ sample, ]
  # Train the model
  rf <- randomForest(DH ~ ., data = train_rf,ntree=1000,mtry=13
                     ,nodesize=2)

  # ,ntree=1000,mtry=11 ,nodesize=2), # r2=0.489, rmse=2.77
```

```
  n1 = paste0(i, 'a')
  n2 = paste0(i, 'b')
  n3 = paste0(i, 'c')
  n4 = paste0(i, 'e')

  # 1 predict in testing dataset
  data.list[[n1]]= data.frame(sample, iteration=i)

  # 2 predict in testing dataset
  pred_RF <-    predict(rf, test_rf, type="response")
  data.list[[n2]]=data.frame(obs=test_rf$DH, pred=pred_RF, iteration=i)

  # 3 variable importance
  data.list[[n3]]=data.frame(t(apply(importance(rf,type=2,scale=FALSE),2,function(x)
x/sum(importance(rf,type=2,scale=FALSE))*100)),iteration = i)

  # 4 static
  data.list[[n4]]=data.frame(R2      = R2(pred_RF, test_rf$DH),
                             RMSE = RMSE(pred_RF,test_rf$DH),
                             MAE    = MAE(pred_RF,test_rf$DH),
                             iteration = i)
  return(data.list)

}

stopImplicitCluster()

# 1) data.sample.RF_DH
data.sample.RF_DH<-
as.data.frame(cbind(CV_DH[[1]]$sample,CV_DH[[2]]$sample,CV_DH[[3]]$sample,CV_D
H[[4]]$sample,CV_DH[[5]]$sample,CV_DH[[6]]$sample,CV_DH[[7]]$sample,CV_DH[[8]]
$sample,CV_DH[[9]]$sample,CV_DH[[10]]$sample,CV_DH[[11]]$sample,CV_DH[[12]]$s
ample,CV_DH[[13]]$sample,CV_DH[[14]]$sample,CV_DH[[15]]$sample,CV_DH[[16]]$sa
mple,CV_DH[[17]]$sample,CV_DH[[18]]$sample,CV_DH[[19]]$sample,CV_DH[[20]]$sa
mple))

colnames(data.sample.RF_DH) <- paste0("sample", i=1:20)

# 2) Predictions.RF_DH
Predictions.RF_DH<-
as.data.frame(rbind(CV_DH[[21]],CV_DH[[22]],CV_DH[[23]],CV_DH[[24]],CV_DH[[25]],
CV_DH[[26]],CV_DH[[27]],CV_DH[[28]],CV_DH[[29]],CV_DH[[30]],CV_DH[[31]],CV_
DH[[32]],CV_DH[[33]],CV_DH[[34]],CV_DH[[35]],CV_DH[[36]],CV_DH[[37]],CV_DH[[
38]],CV_DH[[39]],CV_DH[[40]]))
```

```
# 3) VIM_DH
VIM_DH<-as.data.frame(rbind(CV_DH[[41]],CV_DH[[42]],CV_DH[[43]],CV_DH[[44]]
                            ,CV_DH[[45]],CV_DH[[46]],CV_DH[[47]],CV_DH[[48]]
                            ,CV_DH[[49]],CV_DH[[50]],CV_DH[[51]],CV_DH[[52]]
                            ,CV_DH[[53]],CV_DH[[54]],CV_DH[[55]],CV_DH[[56]]
                            ,CV_DH[[57]],CV_DH[[58]],CV_DH[[59]],CV_DH[[60]]))

# 4) Static_DH
STATIC.RF_DH<-
as.data.frame(rbind(CV_DH[[61]],CV_DH[[62]],CV_DH[[63]],CV_DH[[64]],CV_DH[[65]],
CV_DH[[66]],CV_DH[[67]],CV_DH[[68]],CV_DH[[69]],CV_DH[[70]],CV_DH[[71]],CV_
DH[[72]],CV_DH[[73]],CV_DH[[74]],CV_DH[[75]],CV_DH[[76]],CV_DH[[77]],CV_DH[[
78]],CV_DH[[79]],CV_DH[[80]]))

write.csv(data.sample.RF_DH,"data.sample.RF_DH.csv", row.names = FALSE)
write.csv(Predictions.RF_DH,"Predictions.RF_DH.csv", row.names = FALSE)
write.csv(VIM_DH,"VIM_DH.csv", row.names = FALSE)
write.csv(STATIC.RF_DH,"STATIC.RF_DH.csv", row.names = FALSE)

#############################################################################
library(caret)
library(xgboost)

ORDH.data <- read.csv("OR_DH.csv")
data.sample.RF<-read.csv("data.sample.RF_DH.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){
   sample <- data.sample.RF[,i]
   y <- ORDH.data[, 1]
   x <- ORDH.data[, -1]
   train_data      <- as.matrix(x[-sample, ])
   train_labels <- as.numeric(y[-sample])
   test_data      <- as.matrix(x[ sample, ])
   test_labels <- as.numeric(y[ sample])

   train    <- list(train_data, train_labels); names(train) <- c("data", "label")
   test     <- list(test_data,    test_labels);    names(test)    <- c("data", "label")

   dtrain <- xgb.DMatrix(data = train$data, label = train$label)
   dtest    <- xgb.DMatrix(data = test$data,    label = test$label)
```

```
    wlist <- list(train = dtrain, test = dtest)

    XGB <- xgb.train(data = dtrain, watchlist = wlist,
                                max_depth =5,
                                eta = 0.1,
                                nrounds =50)

    # PREDICTING WITH XGBoost MODEL

    pred_XGB <-    predict(XGB, test$data)

    # Saving results in vectors and data frames

    Stimated <- data.frame(test_labels, pred_XGB, i)
    STATIC<- data.frame( R2 = R2(pred_XGB, test_labels),
                                    RMSE = RMSE(pred_XGB, test_labels),
                                    MAE = MAE(pred_XGB,test_labels),i)

    if(i == 1){

       Predictions.XGB <- Stimated
       STATIC.XGB<- STATIC

    }else{

       Predictions.XGB <- rbind.data.frame(Predictions.XGB, Stimated)
       STATIC.XGB<- rbind.data.frame( STATIC.XGB,STATIC)
    }
}

# You need this information to know which is the best model (XGB )
write.csv(Predictions.XGB, "Predictions.XGB_DH.csv", row.names = FALSE)
write.csv(STATIC.XGB, "STATIC.XGB_DH.csv", row.names = FALSE)

#####lm####
ORDH.data <- read.csv("OR_DH.csv")
data.sample.RF<-read.csv("data.sample.RF_DH.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){

    sample <- data.sample.RF[,i]
```

```
  train_lm <- (ORDH.data)[-sample, ] # Select 90% of the data to train the model
  test_lm <- (ORDH.data)[ sample, ] # Select 10% of the data to test the model

  # TRAINING lm

  LM <- lm(DH ~., data=train_lm, singular.ok = TRUE)

  # PREDICTING WITH RANDOM FOREST MODEL

  pred_LM <-    predict(LM, test_lm)

  # Saving results in vectors and data frames

  Stimated <- data.frame(test_lm$DH, pred_LM,i)
  STATIC<- data.frame( R2 = R2(pred_LM, test_lm$DH),
                       RMSE = RMSE(pred_LM, test_lm$DH),
                       MAE = MAE(pred_LM,test_lm$DH),i)
  if(i == 1){

    Predictions.LM <- Stimated
    STATIC.LM<- STATIC

  }else{

    Predictions.LM <- rbind.data.frame(Predictions.LM, Stimated)
    STATIC.LM<- rbind.data.frame( STATIC.LM,STATIC)
  }
  }

# You need this information to know which is the best model (LM )
write.csv(Predictions.LM, "Predictions.LM_DH.csv", row.names = FALSE)

write.csv(STATIC.LM, "STATIC.LM_DH.csv", row.names = FALSE)

#############################################################################
# 03 Cross validation ----D
library(parallel)
library(foreach)
library(doParallel)

# 03 Cross validation ----D
# read csv file
ORD.data <- read.csv("OR_D.csv")
str(ORD.data)
```

```
# Deterc the number of cores of your computer
no_cores <- detectCores()
# How many iterations do you want?
set.seed(132)
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORD.data), size=floor(.1*nrow(ORD.data)), replace=F)))

registerDoParallel(no_cores[1]-1)
data.list = list()

CV_D <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                i=1:iter, # Number of iterations
                .combine = rbind, # you will add the outputs of each iteration as...
                .multicombine=TRUE,
                .packages=c("randomForest", "caret") # libraries that you will need in foreach function
) %dopar% {
  # Create training and testing datasets
  train_rf <- ORD.data[-sample, ]
  test_rf  <- ORD.data[ sample, ]
  # Train the model
  rf <- randomForest(D ~ ., data = train_rf,ntree=1000,mtry=14
                     ,nodesize=2)
  n1 = paste0(i, 'a')
  n2 = paste0(i, 'b')
  n3 = paste0(i, 'c')
  n4 = paste0(i, 'e')

  # 1 predict in testing dataset
  data.list[[n1]]= data.frame(sample, iteration=i)

  # 2 predict in testing dataset
  pred_RF <-  predict(rf, test_rf, type="response")
  data.list[[n2]]=data.frame(obs=test_rf$D, pred=pred_RF, iteration=i)

  # 3 variable importance
  data.list[[n3]]=data.frame(t(apply(importance(rf,type=2,scale=FALSE), 2, function(x) x/sum(importance(rf,type=2,scale=FALSE))*100)),iteration = i)

  # 4 static
  data.list[[n4]]=data.frame(R2    = R2(pred_RF, test_rf$D),
```

```
                                    RMSE = RMSE(pred_RF,test_rf$D),
                                    MAE    = MAE(pred_RF,test_rf$D),
                                    iteration = i)
  return(data.list)
}

stopImplicitCluster()
# 1) data.sample.RF_D
data.sample.RF_D<-
as.data.frame(cbind(CV_D[[1]]$sample,CV_D[[2]]$sample,CV_D[[3]]$sample,CV_D[[4]]$s
ample,CV_D[[5]]$sample,CV_D[[6]]$sample,CV_D[[7]]$sample,CV_D[[8]]$sample,CV_D[
[9]]$sample,CV_D[[10]]$sample,CV_D[[11]]$sample,CV_D[[12]]$sample,CV_D[[13]]$sam
ple,CV_D[[14]]$sample,CV_D[[15]]$sample,CV_D[[16]]$sample,CV_D[[17]]$sample,CV_
D[[18]]$sample,CV_D[[19]]$sample,CV_D[[20]]$sample))

colnames(data.sample.RF_D) <- paste0("sample", i=1:20)

# 2) Predictions.RF_D
Predictions.RF_D<-
as.data.frame(rbind(CV_D[[21]],CV_D[[22]],CV_D[[23]],CV_D[[24]],CV_D[[25]],CV_D[[
26]],CV_D[[27]],CV_D[[28]],CV_D[[29]],CV_D[[30]],CV_D[[31]],CV_D[[32]],CV_D[[33]
],CV_D[[34]],CV_D[[35]],CV_D[[36]],CV_D[[37]],CV_D[[38]],CV_D[[39]],CV_D[[40]]))

# 3) VIM_D
VIM_D<-as.data.frame(rbind(CV_D[[41]],CV_D[[42]],CV_D[[43]],CV_D[[44]]
                           ,CV_D[[45]],CV_D[[46]],CV_D[[47]],CV_D[[48]]
                           ,CV_D[[49]],CV_D[[50]],CV_D[[51]],CV_D[[52]]
                           ,CV_D[[53]],CV_D[[54]],CV_D[[55]],CV_D[[56]]
                           ,CV_D[[57]],CV_D[[58]],CV_D[[59]],CV_D[[60]]))

# 4) Static_D
STATIC.RF_D<-as.data.frame(rbind(CV_D[[61]],CV_D[[62]],CV_D[[63]],CV_D[[64]]
                                 ,CV_D[[65]],CV_D[[66]],CV_D[[67]],CV_D[[68]]
                                 ,CV_D[[69]],CV_D[[70]],CV_D[[71]],CV_D[[72]]
                                 ,CV_D[[73]],CV_D[[74]],CV_D[[75]],CV_D[[76]]
                                 ,CV_D[[77]],CV_D[[78]],CV_D[[79]],CV_D[[80]]))
write.csv(data.sample.RF_D,"data.sample.RF_D.csv", row.names = FALSE)
write.csv(Predictions.RF_D,"Predictions.RF_D.csv", row.names = FALSE)
write.csv(VIM_D,"VIM_D.csv", row.names = FALSE)
write.csv(STATIC.RF_D,"STATIC.RF_D.csv", row.names = FALSE)

##############################################################################
####
library(caret)
```

```
library(xgboost)

ORD.data <- read.csv("OR_D.csv")
data.sample.RF<-read.csv("data.sample.RF_D.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){

   sample <- data.sample.RF[,i]

   y <- ORD.data[, 1]
   x <- ORD.data[, -1]

   train_data      <- as.matrix(x[-sample, ])
   train_labels <- as.numeric(y[-sample])

   test_data      <- as.matrix(x[ sample, ])
   test_labels <- as.numeric(y[ sample])

   train    <- list(train_data, train_labels); names(train) <- c("data", "label")
   test      <- list(test_data,    test_labels);    names(test)    <- c("data", "label")

   dtrain <- xgb.DMatrix(data = train$data, label = train$label)
   dtest    <- xgb.DMatrix(data = test$data,    label = test$label)


   wlist <- list(train = dtrain, test = dtest)

   XGB <- xgb.train(data = dtrain, watchlist = wlist,
                              max_depth =5,
                              eta = 0.1,
                              nrounds =50)

   # PREDICTING WITH XGBoost MODEL

   pred_XGB <-    predict(XGB, test$data)

   # Saving results in vectors and data frames

   Stimated <- data.frame(test_labels, pred_XGB, i)
   STATIC<- data.frame( R2 = R2(pred_XGB, test_labels),
                                  RMSE = RMSE(pred_XGB, test_labels),
                                  MAE = MAE(pred_XGB,test_labels),i)
```

```
  if(i == 1){

    Predictions.XGB <- Stimated
    STATIC.XGB<- STATIC

  }else{

    Predictions.XGB <- rbind.data.frame(Predictions.XGB, Stimated)
    STATIC.XGB<- rbind.data.frame( STATIC.XGB,STATIC)

  }

}

# You need this information to know which is the best model (XGB )
write.csv(Predictions.XGB, "Predictions.XGB_D.csv", row.names = FALSE)

write.csv(STATIC.XGB, "STATIC.XGB_D.csv", row.names = FALSE)
mean(STATIC.XGB$R2);mean(STATIC.XGB$RMSE)


#########################################################################################
#####
######lm#####

ORD.data <- read.csv("OR_D.csv")
data.sample.RF<-read.csv("data.sample.RF_D.csv")

n <- 20 # Number of iterations

for(i in c(1:n)){

  sample <- data.sample.RF[,i]
  train_lm <- (ORD.data)[-sample, ] # Select 90% of the data to train the model
  test_lm <- (ORD.data)[ sample, ] # Select 10% of the data to test the model

  # TRAINING lm

  LM <- lm(D ~., data=train_lm, singular.ok = TRUE)

  # PREDICTING WITH RANDOM FOREST MODEL

  pred_LM <-    predict(LM, test_lm)
```

```
  # Saving results in vectors and data frames

  Stimated <- data.frame(test_lm$D, pred_LM,i)
  STATIC<- data.frame( R2 = R2(pred_LM, test_lm$D),
                       RMSE = RMSE(pred_LM, test_lm$D),
                       MAE = MAE(pred_LM,test_lm$D),i)
  if(i == 1){

    Predictions.LM <- Stimated
    STATIC.LM<- STATIC

  }else{

    Predictions.LM <- rbind.data.frame(Predictions.LM, Stimated)
    STATIC.LM<- rbind.data.frame( STATIC.LM,STATIC)
  }
}

# You need this information to know which is the best model (LM )
write.csv(Predictions.LM, "Predictions.LM_D.csv", row.names = FALSE)

write.csv(STATIC.LM, "STATIC.LM_D.csv", row.names = FALSE)
```

2. Variable importance for DH, D, C, W

```
# 1 DH--- variable importance plot
VI <- read.csv("VIM_DH.csv",header=T)
VI<-VI[,-ncol(VI)]
str(VI)

VI2 <- as.data.frame(matrix(NA, nrow=37, ncol=4))
colnames(VI2) <- c("Variable","Mean","SD","Type")
VI2[,1] <- colnames(VI)[1:37]
VI2[,4] <- "lightgreen"
VI2[c(5,9,4,1,36,2),4] <- "lightgreen"
for (i in c(1:nrow(VI2))){
  VI2[i,2] <- mean(VI[,i])
  VI2[i,3] <- sd(VI[,i])
}

library(ggplot2)

ggplot(VI2)+
  aes(x=reorder(Variable, Mean), y=Mean, fill=Type)+
  geom_bar(stat="identity")+
  geom_errorbar(aes(ymin=(Mean-
SD),ymax=(Mean+SD)),lwd=0.3,width=0.5,color="gray20")+
  scale_fill_identity()+
  theme_bw()+
  coord_flip()+
  ylab("Variable Importance-DH")+
  theme(axis.title.y = element_blank(),text=element_text(family="serif",colour="black"))

# 2 D--- variable importance plot
VI <- read.csv("VIM_D.csv",header=T)
VI<-VI[,-ncol(VI)]
str(VI)

VI2 <- as.data.frame(matrix(NA, nrow=37, ncol=4))
colnames(VI2) <- c("Variable","Mean","SD","Type")
VI2[,1] <- colnames(VI)[1:37]
VI2[,4] <- "lightgreen"
VI2[c(36,30,2,3,1,15),4] <- "lightgreen"
for (i in c(1:nrow(VI2))){
  VI2[i,2] <- mean(VI[,i])
  VI2[i,3] <- sd(VI[,i])
```

```
}

library(ggplot2)

ggplot(VI2)+
   aes(x=reorder(Variable, Mean), y=Mean, fill=Type)+
   geom_bar(stat="identity")+
   geom_errorbar(aes(ymin=(Mean-
SD),ymax=(Mean+SD)),lwd=0.3,width=0.5,color="gray20")+
   scale_fill_identity()+
   theme_bw()+
   coord_flip()+
   ylab("Variable Importance-D")+
   theme(axis.title.y = element_blank(),text=element_text(family="serif",colour="black"))
```

 # 3 C--- variable importance plot

```
VI <- read.csv("VIM_C.csv",header=T)
VI<-VI[,-ncol(VI)]
str(VI)

VI2 <- as.data.frame(matrix(NA, nrow=39, ncol=4))
colnames(VI2) <- c("Variable","Mean","SD","Type")
VI2[,1] <- colnames(VI)[1:39]
VI2[,4] <- "lightgreen"
VI2[c(2,1,17,5,34,32),4] <- "lightgreen"
for (i in c(1:nrow(VI2))){
   VI2[i,2] <- mean(VI[,i])
   VI2[i,3] <- sd(VI[,i])
}

library(ggplot2)

ggplot(VI2)+
   aes(x=reorder(Variable, Mean), y=Mean, fill=Type)+
   geom_bar(stat="identity")+
   geom_errorbar(aes(ymin=(Mean-
SD),ymax=(Mean+SD)),lwd=0.3,width=0.5,color="gray20")+
   scale_fill_identity()+
   theme_bw()+
   coord_flip()+
   ylab("Variable Importance-C")+
   theme(axis.title.y = element_blank(),text=element_text(family="serif",colour="black"))
```

 # 4 W--- variable importance plot

```
VI <- read.csv("VIM_W.csv",header=T)
VI<-VI[,-ncol(VI)]
str(VI)

VI2 <- as.data.frame(matrix(NA, nrow=39, ncol=4))
colnames(VI2) <- c("Variable","Mean","SD","Type")
VI2[,1] <- colnames(VI)[1:39]
VI2[,4] <- "lightgreen"
VI2[c(2,1,17,5,25,38),4] <- "lightgreen"
for (i in c(1:nrow(VI2))){
   VI2[i,2] <- mean(VI[,i])
   VI2[i,3] <- sd(VI[,i])
}

library(ggplot2)

ggplot(VI2)+
   aes(x=reorder(Variable, Mean), y=Mean, fill=Type)+
   geom_bar(stat="identity")+
   geom_errorbar(aes(ymin=(Mean-
SD),ymax=(Mean+SD)),lwd=0.3,width=0.5,color="gray20")+
   scale_fill_identity()+
   theme_bw()+
   coord_flip()+
   ylab("Variable Importance-W")+
   theme(axis.title.y = element_blank(),text=element_text(family="serif",colour="black"))
```

3. Partial dependence plots for D, DH, C, W

1 Eco1----Plot ceteris paribus effect of important variables on C
Plot ceteris paribus effect of important variables on C in Eco1

```
# parallel package
library(parallel)
library(foreach)
library(doParallel)
library(randomForest)



# 1 Cross validation ----Eco1C
# read csv file
Eco1C_data <- read.csv("Eco1_C.csv")
str(Eco1C_data)
#Eco1C.data<-Eco1C_data[,c(1:3,4:5,6,10,21,36,40,52)]
Eco1C.data<-Eco1C_data
str(Eco1C.data)
head(Eco1C.data[,c(3,2,6,28,7,10)])

no_cores <- detectCores()
registerDoParallel(no_cores[1]-1)

for (var in c(3,2,6,28,7,10)){
   new_df <- colMeans(Eco1C.data)
   new_df <- as.data.frame(lapply(new_df, rep, 100))
   new_df[,var]                                                                    <-
seq(min(Eco1C.data[,var]),quantile(Eco1C.data[,var],0.999),length.out=100)

   head(new_df)
   set.seed(133)

   # How many iterations do you want?
   iter <- 20
   # Selects different plots for cross validate using different training and testing data
   d      <-      as.data.frame(sapply(1:iter,      function(x)      sample.int(n=nrow(Eco1C.data),
size=floor(.1*nrow(Eco1C.data)), replace=F)))

   PDP_Eco1C <- foreach(sample=iter(d, by="column"), # number of rows (to create training
and testing data) that you will select for each iteration
                        i=1:iter, # Number of iterations
                        .combine = rbind, # you will add the outputs of each iteration as...
```

```
                        .multicombine=TRUE,
                        .packages=c("randomForest") # libraries that you will need in foreach
function
  ) %dopar% {
    # Create training and testing datasets
    train_rf <- Eco1C.data[-sample, ]
    #test_rf    <- Eco1C.data[ sample, ]

    # Train the model
    #rf <- randomForest(C ~ ., data = train_rf) # You can add more parameters...
    rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=25 ,nodesize=3)

    # predict in testing dataset
    pred_RF <-    predict(rf, new_df, type="response")

    data.frame(pred=pred_RF, iteration=i)
    # pred_RF$iter = i
    # write.csv(pred_RF, paste("Eco1C_",var, '_', i,".csv", sep = ""), row.names = T)

  }

  PDP.Eco1C<-as.data.frame(matrix(PDP_Eco1C$pred,100,20))
  rownames(PDP.Eco1C)<-
seq(min(Eco1C.data[,var]),quantile(Eco1C.data[,var],0.999),length.out=100)
  colnames(PDP.Eco1C) <- paste0("loop", i=1:20)

  write.csv(PDP.Eco1C, paste("Eco1C_",var,".csv", sep = ""), row.names = T)

  print(var)

}
```

2 Eco2----Plot ceteris paribus effect of important variables on C
Plot ceteris paribus effect of important variables on C in Eco2

```
# parallel package
library(parallel)
library(foreach)
library(doParallel)
library(randomForest)

# 2 Cross validation ----Eco2C
# read csv file
Eco2C_data <- read.csv("Eco2_C.csv")
```

```
str(Eco2C_data)
#Eco2C.data<-Eco2C_data[,c(1:3,4:5,6,10,21,36,40,52)]
Eco2C.data<-Eco2C_data
str(Eco2C.data)
head(Eco2C.data[,c(3,2,6,28,7,10)])
#D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10

no_cores <- detectCores()
registerDoParallel(no_cores[1]-1)

for (var in c(3,2,6,28,7,10)){
  new_df <- colMeans(Eco2C.data)
  new_df <- as.data.frame(lapply(new_df, rep, 100))
  new_df[,var] <-
seq(min(Eco2C.data[,var]),quantile(Eco2C.data[,var],0.999),length.out=100)

  head(new_df)
  set.seed(131)

  # How many iterations do you want?
  iter <- 20
  # Selects different plots for cross validate using different training and testing data
  d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(Eco2C.data),
size=floor(.1*nrow(Eco2C.data)), replace=F)))

  PDP_Eco2C <- foreach(sample=iter(d, by="column"), # number of rows (to create training
and testing data) that you will select for each iteration
                       i=1:iter, # Number of iterations
                       .combine = rbind, # you will add the outputs of each iteration as...
                       .multicombine=TRUE,
                       .packages=c("randomForest") # libraries that you will need in foreach
function
  ) %dopar% {
    # Create training and testing datasets
    train_rf <- Eco2C.data[-sample, ]
    #test_rf  <- Eco2C.data[ sample, ]

    # Train the model
    #rf <- randomForest(C ~ ., data = train_rf) # You can add more parameters...
    rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=23 ,nodesize=2)

    # predict in testing dataset
    pred_RF <-  predict(rf, new_df, type="response")
```

```
      data.frame(pred=pred_RF, iteration=i)
      # pred_RF$iter = i
      # write.csv(pred_RF, paste("Eco2C_",var, '_', i,".csv", sep = ""), row.names = T)

   }

   PDP.Eco2C<-as.data.frame(matrix(PDP_Eco2C$pred,100,20))
   rownames(PDP.Eco2C)<-
seq(min(Eco2C.data[,var]),quantile(Eco2C.data[,var],0.999),length.out=100)
   colnames(PDP.Eco2C) <- paste0("loop", i=1:20)

   write.csv(PDP.Eco2C, paste("Eco2C_",var,".csv", sep = ""), row.names = T)

   print(var)

}
```

3 Eco3----Plot ceteris paribus effect of important variables on C
Plot ceteris paribus effect of important variables on C in Eco3

```
# parallel package
library(parallel)
library(foreach)
library(doParallel)
library(randomForest)

# 3 Cross validation ----Eco3C
# read csv file
Eco3C_data <- read.csv("Eco3_C.csv")
str(Eco3C_data)
#Eco3C.data<-Eco3C_data[,c(1:3,4:5,6,10,21,36,40,52)]
Eco3C.data<-Eco3C_data
str(Eco3C.data)
head(Eco3C.data[,c(3,2,6,28,7,10)])
#D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10

no_cores <- detectCores()
registerDoParallel(no_cores[1]-1)

for (var in c(3,2,6,28,7,10)){
   new_df <- colMeans(Eco3C.data)
   new_df <- as.data.frame(lapply(new_df, rep, 100))
   new_df[,var]                                                    <-
seq(min(Eco3C.data[,var]),quantile(Eco3C.data[,var],0.999),length.out=100)
```

```
head(new_df)
set.seed(131)

# How many iterations do you want?
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(Eco3C.data), size=floor(.1*nrow(Eco3C.data)), replace=F)))

PDP_Eco3C <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                     i=1:iter, # Number of iterations
                     .combine = rbind, # you will add the outputs of each iteration as...
                     .multicombine=TRUE,
                     .packages=c("randomForest") # libraries that you will need in foreach function
) %dopar% {
  # Create training and testing datasets
  train_rf <- Eco3C.data[-sample, ]
  #test_rf  <- Eco3C.data[ sample, ]

  # Train the model
  #rf <- randomForest(C ~ ., data = train_rf) # You can add more parameters...
  rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=30 ,nodesize=3)

  # predict in testing dataset
  pred_RF <-  predict(rf, new_df, type="response")

  data.frame(pred=pred_RF, iteration=i)
  # pred_RF$iter = i
  # write.csv(pred_RF, paste("Eco3C_",var, '_', i,".csv", sep = ""), row.names = T)

}

PDP.Eco3C<-as.data.frame(matrix(PDP_Eco3C$pred,100,20))
rownames(PDP.Eco3C)<-seq(min(Eco3C.data[,var]),quantile(Eco3C.data[,var],0.999),length.out=100)
colnames(PDP.Eco3C) <- paste0("loop", i=1:20)

write.csv(PDP.Eco3C, paste("Eco3C_",var,".csv", sep = ""), row.names = T)

print(var)
```

```
}
## 4 Eco4----Plot ceteris paribus effect of important variables on C
## Plot ceteris paribus effect of important variables on C in Eco4

# parallel package
library(parallel)
library(foreach)
library(doParallel)
library(randomForest)


# 4 Cross validation ----Eco4C
# read csv file
Eco4C_data <- read.csv("Eco4_C.csv")
str(Eco4C_data)
#Eco4C.data<-Eco4C_data[,c(1:3,4:5,6,10,21,36,40,52)]
Eco4C.data<-Eco4C_data
str(Eco4C.data)
head(Eco4C.data[,c(3,2,6,28,7,10)])

no_cores <- detectCores()
registerDoParallel(no_cores[1]-1)

for (var in c(3,2,6,28,7,10)){
  new_df <- colMeans(Eco4C.data)
  new_df <- as.data.frame(lapply(new_df, rep, 100))
  new_df[,var]                                                                    <-
seq(min(Eco4C.data[,var]),quantile(Eco4C.data[,var],0.999),length.out=100)

  head(new_df)
  set.seed(131)

  # How many iterations do you want?
  iter <- 20
  # Selects different plots for cross validate using different training and testing data
  d      <-      as.data.frame(sapply(1:iter,      function(x)      sample.int(n=nrow(Eco4C.data),
size=floor(.1*nrow(Eco4C.data)), replace=F)))

  PDP_Eco4C <- foreach(sample=iter(d, by="column"), # number of rows (to create training
and testing data) that you will select for each iteration
                       i=1:iter, # Number of iterations
                       .combine = rbind, # you will add the outputs of each iteration as...
                       .multicombine=TRUE,
                       .packages=c("randomForest") # libraries that you will need in foreach
```

```
function
  ) %dopar% {
    # Create training and testing datasets
    train_rf <- Eco4C.data[-sample, ]
    #test_rf    <- Eco4C.data[ sample, ]

    # Train the model
    #rf <- randomForest(C ~ ., data = train_rf) # You can add more parameters...
    rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=11 ,nodesize=2)

    # predict in testing dataset
    pred_RF <-    predict(rf, new_df, type="response")

    data.frame(pred=pred_RF, iteration=i)
    # pred_RF$iter = i
    # write.csv(pred_RF, paste("Eco4C_",var, '_', i,".csv", sep = ""), row.names = T)

  }

  PDP.Eco4C<-as.data.frame(matrix(PDP_Eco4C$pred,100,20))
  rownames(PDP.Eco4C)<-
seq(min(Eco4C.data[,var]),quantile(Eco4C.data[,var],0.999),length.out=100)
  colnames(PDP.Eco4C) <- paste0("loop", i=1:20)

  write.csv(PDP.Eco4C, paste("Eco4C_",var,".csv", sep = ""), row.names = T)

  print(var)

}
```

5 Eco5----Plot ceteris paribus effect of important variables on C
Plot ceteris paribus effect of important variables on C in Eco5

```
# parallel package
library(parallel)
library(foreach)
library(doParallel)
library(randomForest)


# 5 Cross validation ----Eco5C
# read csv file
Eco5C_data <- read.csv("Eco5_C.csv")
```

```
str(Eco5C_data)
#Eco5C.data<-Eco5C_data[,c(1:3,4:5,6,10,21,36,40,52)]
Eco5C.data<-Eco5C_data
str(Eco5C.data)
head(Eco5C.data[,c(3,2,6,28,7,10)])
#D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10

no_cores <- detectCores()
registerDoParallel(no_cores[1]-1)

for (var in c(3,2,6,28,7,10)){
  new_df <- colMeans(Eco5C.data)
  new_df <- as.data.frame(lapply(new_df, rep, 100))
  new_df[,var] <-
seq(min(Eco5C.data[,var]),quantile(Eco5C.data[,var],0.999),length.out=100)

  head(new_df)
  set.seed(132)

  # How many iterations do you want?
  iter <- 20
  # Selects different plots for cross validate using different training and testing data
  d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(Eco5C.data),
size=floor(.1*nrow(Eco5C.data)), replace=F)))

  PDP_Eco5C <- foreach(sample=iter(d, by="column"), # number of rows (to create training
and testing data) that you will select for each iteration
                       i=1:iter, # Number of iterations
                       .combine = rbind, # you will add the outputs of each iteration as...
                       .multicombine=TRUE,
                       .packages=c("randomForest") # libraries that you will need in foreach
function
  ) %dopar% {
    # Create training and testing datasets
    train_rf <- Eco5C.data[-sample, ]
    #test_rf  <- Eco5C.data[ sample, ]

    # Train the model
    #rf <- randomForest(C ~ ., data = train_rf) # You can add more parameters...
    rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=18 ,nodesize=3)

    # predict in testing dataset
    pred_RF <-  predict(rf, new_df, type="response")
```

```
        data.frame(pred=pred_RF, iteration=i)
        # pred_RF$iter = i
        # write.csv(pred_RF, paste("Eco5C_",var, '_', i,".csv", sep = ""), row.names = T)

    }

    PDP.Eco5C<-as.data.frame(matrix(PDP_Eco5C$pred,100,20))
    rownames(PDP.Eco5C)<-
seq(min(Eco5C.data[,var]),quantile(Eco5C.data[,var],0.999),length.out=100)
    colnames(PDP.Eco5C) <- paste0("loop", i=1:20)

    write.csv(PDP.Eco5C, paste("Eco5C_",var,".csv", sep = ""), row.names = T)

    print(var)

}

#########################################################################
    # Make sensitivity analysis plots
    # Make sure there is a folder named "Images" under the main directory

    library(ggplot2)
    library(hexbin)
    library(ggpubr)
    library(matrixStats)


## 1 D ###
    # Chose the variable that you want to plot (select CSV file)
    # Manually do one variable at a time
    #D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10
    SA_1 <- read.csv("Eco1C_3.csv");
    SA_2 <- read.csv("Eco2C_3.csv");
    SA_3 <- read.csv("Eco3C_3.csv");
    SA_4 <- read.csv("Eco4C_3.csv");
    SA_5 <- read.csv("Eco5C_3.csv");


    for(i in c(1:5)){

        if(i == 1){ SA_i=SA_1};if(i == 1){ text <- "D" }
        if(i == 2){ SA_i=SA_2};if(i == 2){ text <- "D" }
        if(i == 3){ SA_i=SA_3};if(i == 3){ text <- "D" }
        if(i == 4){ SA_i=SA_4};if(i == 4){ text <- "D" }
```

```
    if(i == 5){ SA_i=SA_5};if(i == 5){ text <- "D" }



  SA_i$Mean_SA   <- rowMeans(SA_i[, c(2: 21)])
  SA_i$SdDev_SA <- rowSds(as.matrix(SA_i[, c(2: 21)]))
  SA_i$SdErr_SA <- SA_i$SdDev_SA / sqrt(20) * 1.96

    plot <- ggplot(SA_i)+
      geom_line(aes(x=X, y=Mean_SA))+
      geom_ribbon(aes(x=X,   ymin=Mean_SA-SdErr_SA,   ymax=Mean_SA+SdErr_SA),
alpha = 0.2, fill="blue")+
      theme_bw() +
      labs(x=NULL, y=NULL) +
      annotate("text", x = max(SA_i$X),
                  y = max(SA_i$Mean_SA+SA_i$SdErr_SA)+
                    (max(SA_i$Mean_SA+SA_i$SdErr_SA)-
                       min(SA_i$Mean_SA-SA_i$SdErr_SA))/8*0.95,
                  label = text,
                  hjust = 0.5, vjust=0.5,family="serif") +
      theme(legend.position="none", panel.grid = element_blank(),
            plot.subtitle=element_text(hjust = 0.4,family="serif",colour="black")); plot

    if(i == 1){   pl1 <- plot   }; if(i == 2){   pl2 <- plot   }; if(i == 3){   pl3 <- plot   }
    if(i == 4){   pl4 <- plot   }; if(i == 5){   pl5 <- plot   }

  }

  nt.plot <- ggarrange(pl1, pl2, pl3, pl4, pl5, ncol=5, nrow=1); nt.plot

  annotate_figure(nt.plot,   left     = "Predicted Eco_C")

  ggsave("pdp_Eco_D.png", width=9.0, height=1.8, dpi=1000)
  ggsave("pdp_Eco_D.tiff", width=9.0, height=1.8, dpi=1000)

 # dev.off()



  ## 2 DH ###
  # Chose the variable that you want to plot (select CSV file)
  # Manually do one variable at a time
  #D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10
  SA_1 <- read.csv("Eco1C_2.csv");
```

```
SA_2 <- read.csv("Eco2C_2.csv");
SA_3 <- read.csv("Eco3C_2.csv");
SA_4 <- read.csv("Eco4C_2.csv");
SA_5 <- read.csv("Eco5C_2.csv");

for(i in c(1:5)){

   if(i == 1){ SA_i=SA_1};if(i == 1){ text <- "DH" }
   if(i == 2){ SA_i=SA_2};if(i == 2){ text <- "DH" }
   if(i == 3){ SA_i=SA_3};if(i == 3){ text <- "DH" }
   if(i == 4){ SA_i=SA_4};if(i == 4){ text <- "DH" }
   if(i == 5){ SA_i=SA_5};if(i == 5){ text <- "DH" }



   SA_i$Mean_SA    <- rowMeans(SA_i[, c(2: 21)])
   SA_i$SdDev_SA <- rowSds(as.matrix(SA_i[, c(2: 21)]))
   SA_i$SdErr_SA <- SA_i$SdDev_SA / sqrt(20) * 1.96

   plot <- ggplot(SA_i)+
      geom_line(aes(x=X, y=Mean_SA))+
      geom_ribbon(aes(x=X,    ymin=Mean_SA-SdErr_SA,    ymax=Mean_SA+SdErr_SA),
alpha = 0.2, fill="blue")+
      theme_bw() +
      labs(x=NULL, y=NULL) +
      annotate("text", x = max(SA_i$X),
                  y = max(SA_i$Mean_SA+SA_i$SdErr_SA)+
                     (max(SA_i$Mean_SA+SA_i$SdErr_SA)-
                        min(SA_i$Mean_SA-SA_i$SdErr_SA))/8*0.95,
                  label = text,
                  hjust = 0.5, vjust=0.5,family="serif") +
      theme(legend.position="none", panel.grid = element_blank(),
            plot.subtitle=element_text(hjust = 0.4,family="serif",colour="black")); plot

   if(i == 1){    pl1 <- plot    }; if(i == 2){    pl2 <- plot    }; if(i == 3){    pl3 <- plot    }
   if(i == 4){    pl4 <- plot    }; if(i == 5){    pl5 <- plot    }

}

nt.plot <- ggarrange(pl1, pl2, pl3, pl4, pl5, ncol=5, nrow=1); nt.plot

annotate_figure(nt.plot,    left      = "Predicted Eco_DH")

ggsave("pdp_Eco_DH.png", width=9.0, height=1.8, dpi=1000)
ggsave("pdp_Eco_DH.tiff", width=9.0, height=1.8, dpi=1000)
```

```
# dev.off()


## 3 Elev ###
# Chose the variable that you want to plot (select CSV file)
# Manually do one variable at a time
#D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10
SA_1 <- read.csv("Eco1C_6.csv");
SA_2 <- read.csv("Eco2C_6.csv");
SA_3 <- read.csv("Eco3C_6.csv");
SA_4 <- read.csv("Eco4C_6.csv");
SA_5 <- read.csv("Eco5C_6.csv");


for(i in c(1:5)){

   if(i == 1){ SA_i=SA_1};if(i == 1){ text <- "Elev" }
   if(i == 2){ SA_i=SA_2};if(i == 2){ text <- "Elev" }
   if(i == 3){ SA_i=SA_3};if(i == 3){ text <- "Elev" }
   if(i == 4){ SA_i=SA_4};if(i == 4){ text <- "Elev" }
   if(i == 5){ SA_i=SA_5};if(i == 5){ text <- "Elev" }



   SA_i$Mean_SA   <- rowMeans(SA_i[, c(2: 21)])
   SA_i$SdDev_SA <- rowSds(as.matrix(SA_i[, c(2: 21)]))
   SA_i$SdErr_SA <- SA_i$SdDev_SA / sqrt(20) * 1.96

   plot <- ggplot(SA_i)+
      geom_line(aes(x=X, y=Mean_SA))+
      geom_ribbon(aes(x=X,   ymin=Mean_SA-SdErr_SA,   ymax=Mean_SA+SdErr_SA),
alpha = 0.2, fill="blue")+
      theme_bw() +
      labs(x=NULL, y=NULL) +
      annotate("text", x = max(SA_i$X),
                  y = max(SA_i$Mean_SA+SA_i$SdErr_SA)+
                     (max(SA_i$Mean_SA+SA_i$SdErr_SA)-
                        min(SA_i$Mean_SA-SA_i$SdErr_SA))/8*0.95,
                  label = text,
                  hjust = 0.5, vjust=0.5,family="serif") +
      theme(legend.position="none", panel.grid = element_blank(),
            plot.subtitle=element_text(hjust = 0.4,family="serif",colour="black")); plot
```

```
    if(i == 1){    pl1 <- plot    }; if(i == 2){    pl2 <- plot    }; if(i == 3){    pl3 <- plot    }
    if(i == 4){    pl4 <- plot    }; if(i == 5){    pl5 <- plot    }

}

nt.plot <- ggarrange(pl1, pl2, pl3, pl4, pl5, ncol=5, nrow=1); nt.plot

annotate_figure(nt.plot,    left      = "Predicted Eco_Elev")

ggsave("pdp_Eco_Elev.png", width=9.0, height=1.8, dpi=1000)
ggsave("pdp_Eco_Elev.tiff", width=9.0, height=1.8, dpi=1000)

# dev.off()


## 4 H1 ###
# Chose the variable that you want to plot (select CSV file)
# Manually do one variable at a time
#D,DH,Elev,H1,C1,C4: 3,2,6,28,7,10
SA_1 <- read.csv("Eco1C_28.csv");
SA_2 <- read.csv("Eco2C_28.csv");
SA_3 <- read.csv("Eco3C_28.csv");
SA_4 <- read.csv("Eco4C_28.csv");
SA_5 <- read.csv("Eco5C_28.csv");


for(i in c(1:5)){

    if(i == 1){ SA_i=SA_1};if(i == 1){ text <- "H1" }
    if(i == 2){ SA_i=SA_2};if(i == 2){ text <- "H1" }
    if(i == 3){ SA_i=SA_3};if(i == 3){ text <- "H1" }
    if(i == 4){ SA_i=SA_4};if(i == 4){ text <- "H1" }
    if(i == 5){ SA_i=SA_5};if(i == 5){ text <- "H1" }


    SA_i$Mean_SA    <- rowMeans(SA_i[, c(2: 21)])
    SA_i$SdDev_SA <- rowSds(as.matrix(SA_i[, c(2: 21)]))
    SA_i$SdErr_SA <- SA_i$SdDev_SA / sqrt(20) * 1.96

    plot <- ggplot(SA_i)+
      geom_line(aes(x=X, y=Mean_SA))+
      geom_ribbon(aes(x=X,    ymin=Mean_SA-SdErr_SA,    ymax=Mean_SA+SdErr_SA),
alpha = 0.2, fill="blue")+
      theme_bw() +
```

```
      labs(x=NULL, y=NULL) +
      annotate("text", x = max(SA_i$X),
                       y = max(SA_i$Mean_SA+SA_i$SdErr_SA)+
                          (max(SA_i$Mean_SA+SA_i$SdErr_SA)-
                               min(SA_i$Mean_SA-SA_i$SdErr_SA))/8*0.95,
                       label = text,
                       hjust = 0.5, vjust=0.5,family="serif") +
      theme(legend.position="none", panel.grid = element_blank(),
              plot.subtitle=element_text(hjust = 0.4,family="serif",colour="black")); plot

   if(i == 1){    pl1 <- plot    }; if(i == 2){    pl2 <- plot    }; if(i == 3){    pl3 <- plot    }
   if(i == 4){    pl4 <- plot    }; if(i == 5){    pl5 <- plot    }; if(i == 6){    pl6 <- plot    }

}

nt.plot <- ggarrange(pl1, pl2, pl3, pl4, pl5,    ncol=5, nrow=1); nt.plot

annotate_figure(nt.plot,    left      = "Predicted Eco_H1")

ggsave("pdp_Eco_H1.png", width=9.0, height=1.8, dpi=1000)
ggsave("pdp_Eco_H1.tiff", width=9.0, height=1.8, dpi=1000)
```

4. Predict 1km data for DH, D, W, C

 # 1 DH---- Predict 1km data

```
################################################################################
###########
library(parallel)
library(foreach)
library(doParallel)
################################################################################
##########
library(matrixStats)
library(sp)
library(raster)
library(caret)
library(randomForest)

# load train data and predict EA climate data
ORDH_data <- read.csv("OR_DH.csv")
str(ORDH_data)
ORDH.data<-ORDH_data[,-c(5:6,29:33,36:37,41:43,45,47,49,52,54)]
str(ORDH.data)

NEA.climate<-read.csv("05032020_NEAclimate.csv",header=T)
newdata<-NEA.climate[,-c(1,2)]
str(newdata)

# Deterc the number of cores of your computer
no_cores <- detectCores()
# How many iterations do you want?
set.seed(132) #124
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORDH.data),
size=floor(.1*nrow(ORDH.data)), replace=F)))


registerDoParallel(no_cores[1]-1)
data.list = list()

Pred_DH <- foreach(sample=iter(d, by="column"), # number of rows (to create training and
testing data) that you will select for each iteration
                   i=1:iter, # Number of iterations
```

```
                        .combine = rbind, # you will add the outputs of each iteration as...
                        .multicombine=TRUE,
                        .packages=c("randomForest", "caret") # libraries that you will need in
foreach function
) %dopar% {
   # Create training and testing datasets
   train_rf <- ORDH.data[-sample, ]
   #test_rf    <- ORDH.data[ sample, ]
   # Train the model
   rf <- randomForest(DH ~ ., data = train_rf,ntree=1000,mtry=13
                              ,nodesize=2)

   # ,ntree=1000,mtry=20 ,nodesize=2), # r2=0.539, rmse=2.77


   #    predict in testing dataset
   pred_RF <-    predict(rf, newdata, type="response")

   data.frame(pred=pred_RF, iteration=i)


}

stopImplicitCluster()

# split predict data to data frame

Pred.DH<-as.data.frame(matrix(Pred_DH$pred,541149,20))
colnames(Pred.DH) <- paste0("loop", i=1:20)

head(Pred.DH,5)


DH_mean <- rowMeans(Pred.DH) # Mean
DH_SE<- rowSds(as.matrix(Pred.DH))/sqrt(ncol(Pred.DH)) # Standard Error

Pred.RF_DH<-cbind.data.frame(NEA.climate,DH_mean,DH_SE)

write.csv(Pred.RF_DH,"Pred.RF_DH.csv",row.names=FALSE)
##############################################################
```

 # 2 D---- Predict 1km data

```
library(parallel)
library(foreach)
```

```
library(doParallel)
##############################################################################
##########
library(matrixStats)
library(sp)
library(raster)
library(caret)
library(randomForest)

# load train data and predict EA climate data
ORD_data <- read.csv("OR_D.csv")
str(ORD_data)
ORD.data<-ORD_data[,-c(5:6,29:33,36:37,41:43,45,47,49,52,54)]
str(ORD.data)

NEA.climate<-read.csv("05032020_NEAclimate.csv",header=T)
newdata<-NEA.climate[,-c(1,2)]
str(newdata)

# Deterc the number of cores of your computer
no_cores <- detectCores()
# How many iterations do you want?
set.seed(132) #125
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORD.data),
size=floor(.1*nrow(ORD.data)), replace=F)))


registerDoParallel(no_cores[1]-1)
data.list = list()

Pred_D <- foreach(sample=iter(d, by="column"), # number of rows (to create training and
testing data) that you will select for each iteration
                  i=1:iter, # Number of iterations
                  .combine = rbind, # you will add the outputs of each iteration as...
                  .multicombine=TRUE,
                  .packages=c("randomForest", "caret") # libraries that you will need in
foreach function
) %dopar% {
  # Create training and testing datasets
  train_rf <- ORD.data[-sample, ]
  #test_rf  <- ORD.data[ sample, ]
  # Train the model
```

```
  rf <- randomForest(D ~ ., data = train_rf,ntree=1000,mtry=14
                                ,nodesize=2)

  # ,ntree=1000,mtry=11,nodesize=2,    r2=0.163,rmse=4.06
  # ,ntree=1000,mtry=15,nodesize=2,    r2=0.165,rmse=4.03


  #    predict in testing dataset
  pred_RF <-    predict(rf, newdata, type="response")

  data.frame(pred=pred_RF, iteration=i)


}

stopImplicitCluster()

# split predict data to data frame

Pred.D<-as.data.frame(matrix(Pred_D$pred,541149,20))
colnames(Pred.D) <- paste0("loop", i=1:20)

head(Pred.D,5)


D_mean <- rowMeans(Pred.D) # Mean
D_SE<- rowSds(as.matrix(Pred.D))/sqrt(ncol(Pred.D)) # Standard Error

Pred.RF_D<-cbind.data.frame(NEA.climate,D_mean,D_SE)

write.csv(Pred.RF_D,"Pred.RF_D.csv",row.names=FALSE)

##########################################
```

 # 3 C---- Predict 1km data

```
library(parallel)
library(foreach)
library(doParallel)
##############################################################################
##########
library(matrixStats)
library(sp)
library(raster)
```

```
library(caret)
library(randomForest)

# load train data and predict EA climate data
# read csv file
ORCd_data<- read.csv("OR_C.csv")
str(ORCd_data)
ORCd.data<-ORCd_data[,-c(7:8,31:35,38:39,43:45,47,49,51,54,56)]
str(ORCd.data)

# Read DH,D,Climate
Pred.RF_DH<-read.csv("Pred.RF_DH.csv")
Pred.RF_D<-read.csv("Pred.RF_D.csv")
NEA.climate<-read.csv("05032020_NEAclimate.csv",header=T)


# Add DH,D
NEA.climate$DH<-Pred.RF_DH$DH_mean
NEA.climate$D<-Pred.RF_D$D_mean
str(NEA.climate)

newdata<-NEA.climate[,-c(1,2)]
new.data<-newdata[,c(38,39,1:37)]
str(new.data)

# Deterc the number of cores of your computer
no_cores <- detectCores()
# How many iterations do you want?
set.seed(132) #132
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORCd.data), size=floor(.1*nrow(ORCd.data)), replace=F)))


registerDoParallel(no_cores[1]-1)
data.list = list()

Pred_C <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                  i=1:iter, # Number of iterations
                  .combine = rbind, # you will add the outputs of each iteration as...
                  .multicombine=TRUE,
                  .packages=c("randomForest", "caret") # libraries that you will need in
```

```
foreach function
) %dopar% {
   # Create training and testing datasets
   train_rf <- ORCd.data[-sample, ]
   #test_rf    <- ORCd.data[ sample, ]
   # Train the model
   rf <- randomForest(C ~ ., data = train_rf,ntree=1000,mtry=13 ,nodesize=3)

   # ntree=1000,mtry=21,nodesize=2,sampsize=round(0.62*nrow(train_rf)))


   #    predict in testing dataset
   pred_RF <-    predict(rf, new.data, type="response")

   data.frame(pred=pred_RF, iteration=i)


}

stopImplicitCluster()

# split predict data to data frame

Pred.C<-as.data.frame(matrix(Pred_C$pred,541149,20))
colnames(Pred.C) <- paste0("loop", i=1:20)

head(Pred.C,5)


C_mean <- rowMeans(Pred.C) # Mean
C_SE<- rowSds(as.matrix(Pred.C))/sqrt(ncol(Pred.C)) # Standard Error

Pred.RF_C<-cbind.data.frame(NEA.climate,C_mean,C_SE)

write.csv(Pred.RF_C,"Pred.RF_C.csv",row.names=FALSE)

##########################################
```

 # 4 W---- Predict 1km data

```
library(parallel)
library(foreach)
library(doParallel)
library(matrixStats)
library(sp)
```

```
library(raster)
library(caret)
library(randomForest)

# load train data and predict EA climate data
ORW_data<- read.csv("OR_W.csv")
str(ORW_data)
ORW.data<-ORW_data[,-c(7:8,31:35,38:39,43:45,47,49,51,54,56)]
str(ORW.data)

# Read DH,D,Climate
Pred.RF_DH<-read.csv("Pred.RF_DH.csv")
Pred.RF_D<-read.csv("Pred.RF_D.csv")
NEA.climate<-read.csv("05032020_NEAclimate.csv",header=T))

# Add DH,D
NEA.climate$DH<-Pred.RF_DH$DH_mean
NEA.climate$D<-Pred.RF_D$D_mean
str(NEA.climate)

newdata<-NEA.climate[,-c(1,2)]
new.data<-newdata[,c(38,39,1:37)]
str(new.data)
# Deterc the number of cores of your computer
no_cores <- detectCores()
# How many iterations do you want?
set.seed(132) #131
iter <- 20
# Selects different plots for cross validate using different training and testing data
d <- as.data.frame(sapply(1:iter, function(x) sample.int(n=nrow(ORW.data), size=floor(.1*nrow(ORW.data)), replace=F)))


registerDoParallel(no_cores[1]-1)
data.list = list()

Pred_W <- foreach(sample=iter(d, by="column"), # number of rows (to create training and testing data) that you will select for each iteration
                  i=1:iter, # Number of iterations
                  .combine = rbind, # you will add the outputs of each iteration as...
                  .multicombine=TRUE,
                  .packages=c("randomForest", "caret") # libraries that you will need in foreach function
) %dopar% {
```

```
  # Create training and testing datasets
  train_rf <- ORW.data[-sample, ]
  #test_rf    <- ORW.data[ sample, ]
  # Train the model
  rf <- randomForest(W ~ ., data = train_rf,ntree=1000,mtry=13 ,nodesize=3)

  # ntree=1000,mtry=21,nodesize=2,sampsize=round(0.62*nrow(train_rf)))


  #    predict in testing dataset
  pred_RF <-    predict(rf, newdata, type="response")

  data.frame(pred=pred_RF, iteration=i)


}

stopImplicitCluster()

# split predict data to data frame

Pred.W<-as.data.frame(matrix(Pred_W$pred,541149,20))
colnames(Pred.W) <- paste0("loop", i=1:20)

head(Pred.W,5)


W_mean <- rowMeans(Pred.W) # Mean
W_SE<- rowSds(as.matrix(Pred.W))/sqrt(ncol(Pred.W)) # Standard Error

Pred.RF_W<-cbind.data.frame(NEA.climate,W_mean,W_SE)

write.csv(Pred.RF_W,"Pred.RF_W.csv",row.names=FALSE)

##############################################

## End of Code
```