```
# This is a script to reproduce results presented in the paper:
# A machine learning approach to analyzing the relationship between
# temperatures and multi-proxy tree-ring records.
# Authors: Jernej Jevsenak, Saso Dzeroski, Sasa Zavadlav, Tom Levanic

# To run this code, please instal listed packages: RWeka, brnn, dplyr,
 MLmetrics, reshape, RCurl ,foreign

# This code depende on rJava package, so users must have installed the
 appropriate Java, i.e., 32-bit Java
# for 32-bit R and 64-bit Java for the 64-bit R version. 64-bit Java
 for Wwindows could be downloaded from
# the following web page: https://www.java.com/en/download/manual.jsp
 (select Windows Offline (64-bit).
# rJava on Mac OS causes problems. So  we recommend using Windows or
 Linux operating system to run this code.

# Load the following R libraries
library(RWeka)
library(brnn)
library(dplyr)
library(MLmetrics)
library(reshape)
library(RCurl)
library(foreign)

# Downloading data from gitHub Site
URL_Spring <-
 "https://raw.githubusercontent.com/jernejjevsenak/Data/master/SpringMo
 del.csv"
URL_Summer <-
 "https://raw.githubusercontent.com/jernejjevsenak/Data/master/SummerMo
 del.csv"

Spring_connection <- getURL(URL_Spring)
Summer_connection <- getURL(URL_Summer)

Spring_data <- read.csv(textConnection(Spring_connection))
Summer_data <- read.csv(textConnection(Summer_connection))

# From Summer data, we delete the LateWood series, as described in the
 paper
Summer_data <- dplyr::select(Summer_data, -LW)
```

```r
# 1 we create a function that calculates performance metrics for train
 and test data
metrics <- function(train_predicted, test_predicted, train_observed,
 test_observed){

  # Calculating metrics for train (calibration data)
  train_cor <- cor(train_predicted, train_observed)
  train_RMSE <- MLmetrics::RMSE(train_predicted, train_observed)
  train_RRSE <- MLmetrics::RRSE(train_predicted, train_observed)
  train_CE <- 1 - (sum((train_observed - train_predicted) ^ 2) /
                     sum((train_observed - mean(train_observed)) ^ 2))
  train_RE <- 1 - (sum((train_observed - train_predicted) ^ 2) /
                     sum((train_observed - mean(test_observed)) ^ 2))

  train_metrics <- data.frame(cbind(train_cor, train_RMSE, train_RRSE,
                                    train_RE, train_CE))
  row.names(train_metrics) <- c(deparse(substitute(train_predicted)))
  colnames(train_metrics) <- c("cor", "RMSE", "RRSE", "RE", "CE")

  #Calculations for test (validation) data
  test_cor <- cor(test_observed, test_predicted)
  test_RMSE <- MLmetrics::RMSE(test_predicted, test_observed)
  test_RRSE <- MLmetrics::RRSE(test_predicted, test_observed)
  test_CE <- 1 - (sum((test_observed - test_predicted) ^ 2) /
                    sum((test_observed - mean(test_observed)) ^ 2))
  test_RE <- 1 - (sum((test_observed - test_predicted) ^ 2) /
                    sum((test_observed - mean(train_observed)) ^ 2))

  test_metrics <- data.frame(cbind(test_cor, test_RMSE, test_RRSE,
                                   test_RE, test_CE))
  row.names(test_metrics) <- c(deparse(substitute(test_predicted)))
  colnames(test_metrics) <- c("cor", "RMSE", "RRSE", "RE", "CE")

  df_metrics <- round(rbind(train_metrics, test_metrics), 4)
  row.names(df_metrics) <- c("train", "test")

  df_metrics
}

# 2 This is a small function that will be used to calculate ranks
count_ones = function(vector){
  sum(vector == 1)
}
```

```
# 3 Function that iterates and calculates performance metric for 5
 regression methods using 3-fold CV
iter = function(formula, dataset, multiply = 5) {

  set.seed(multiply * 5) # This is randomly selected number to ensure
 reproducible results. Later in a
  # loop, it is crucial to change the multiply number. If not, we get
 the same splits for each iteration

  # Here, idex of dependent variable is extracted and later used to
 locate the
  # observed values
  DepIndex <- grep(as.character(formula[[2]]), colnames(dataset))
  DepName <- as.character(formula[[2]])

  list_MLR=list()
  list_ANN=list()
  list_MT=list()
  list_BMT=list()
  list_RF=list()

  MLR_Npredicotrs = list() # This is a list, where the number of
 predictors for MLR is stored

  fold_key = seq(1:3)
  fold_key = paste('fold_',fold_key)

  #Randomly shuffle the data
  dataset<-dataset[sample(nrow(dataset)),]

  #Create 3 equally size folds
  folds <- cut(seq(1,nrow(dataset)),breaks = 3,labels = FALSE)

  #Perform 3-fold cross validation

  for(j in 1:3){
    #Segement the data by fold using the which() function
    testIndexes <- which(folds==j,arr.ind=TRUE)
    test <- dataset[testIndexes, ]
    train <- dataset[-testIndexes, ]

    #MLR Model
    MLR = step(lm(formula, data=train), direction = "backward")
    train_predicted = predict(MLR, train)
    test_predicted = predict(MLR, test)
    train_observed <- train[, DepIndex]
    test_observed <- test[, DepIndex]
    calculations = metrics(train_predicted, test_predicted, 
train_observed, test_observed)
    list_MLR[[j]] = calculations

    MLR_Npredicotrs[[j]] = ncol(MLR$model) - 1 # Here, the number of 
predictors is stored

    #ANN Model
    ANN = brnn(formula, data = train, neurons = 1)
    train_predicted = predict(ANN, train)
    test_predicted = predict(ANN, test)
    calculations = metrics(train_predicted, test_predicted, 
train_observed, test_observed)
    list_ANN[[j]] = calculations

    #MT Model
    MT_model = M5P(formula, data=train, control = Weka_control(M = 15, 
N = T))
    train_predicted = predict(MT_model, train)
    test_predicted = predict(MT_model, test)
    calculations = metrics(train_predicted, test_predicted, 
train_observed, test_observed)
    list_MT[[j]] = calculations

    #BMT Model
    BMT_model = Bagging(formula, data=train, control = Weka_control(P 
= 80, I = 100,
                  W = list("weka.classifiers.trees.M5P", N = T, M = 
15)))
    train_predicted = predict(BMT_model, train)
    test_predicted = predict(BMT_model, test)
    calculations = metrics(train_predicted, test_predicted, 
train_observed, test_observed)
    list_BMT[[j]] = calculations

    ##Random forest (RF) Model
    RF = make_Weka_classifier("weka/classifiers/trees/RandomForest")
    RF_model = RF(formula, data = train, control = Weka_control(P = 
100, I = 100, depth = 2))
    train_predicted = predict(RF_model, train)
    test_predicted = predict(RF_model, test)
```

```
   calculations = metrics(train_predicted, test_predicted, 
train_observed, test_observed)
   list_RF[[j]] = calculations

 }

 listVec <- lapply(list_MLR, c, recursive=TRUE)
 m <- do.call(cbind, listVec)
 middle_calculations <- apply(m, 1, mean)
 m = cbind(m,middle_calculations)
 df_MLR=data.frame(m)
 df_MLR=round(df_MLR, 4)
 colnames(df_MLR)=c(fold_key,'Average')
 rownames(df_MLR)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 
'RRSE_cal',
                    'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 
'CE_val')

 listVec <- lapply(list_ANN, c, recursive=TRUE)
 m <- do.call(cbind, listVec)
 middle_calculations <- apply(m, 1, mean)
 m = cbind(m,middle_calculations)
 df_ANN=data.frame(m)
 df_ANN=round(df_ANN, 4)
 colnames(df_ANN)=c(fold_key,'Average')
 rownames(df_ANN)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 
'RRSE_cal',
                       'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 
'CE_val')

 listVec <- lapply(list_MT, c, recursive=TRUE)
 m <- do.call(cbind, listVec)
 middle_calculations <- apply(m, 1, mean)
 m = cbind(m,middle_calculations)
 df_MT=data.frame(m)
 df_MT=round(df_MT, 4)
 colnames(df_MT)=c(fold_key,'Average')
 rownames(df_MT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 
'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 
'CE_val')

 listVec <- lapply(list_BMT, c, recursive=TRUE)
 m <- do.call(cbind, listVec)
```

```
  middle_calculations <- apply(m, 1, mean)
  m = cbind(m,middle_calculations)
  df_BMT=data.frame(m)
  df_BMT=round(df_BMT, 4)
  colnames(df_BMT)=c(fold_key,'Average')
  rownames(df_BMT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                       'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')

  listVec <- lapply(list_RF, c, recursive=TRUE)
  m <- do.call(cbind, listVec)
  middle_calculations <- apply(m, 1, mean)
  m = cbind(m,middle_calculations)
  df_RF=data.frame(m)
  df_RF=round( df_RF, 4)
  colnames(df_RF)=c(fold_key,'Average')
  rownames(df_RF)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal', 'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 'CE_val')

  listVec <- lapply(MLR_Npredicotrs, c, recursive=TRUE)
  m <- do.call(cbind, listVec)
  middle_calculations <- apply(m, 1, mean)
  MLR_pred = middle_calculations

  final = list(df_MLR, df_ANN, df_MT, df_BMT, df_RF, MLR_pred)

}

#### CALCULATIONS FOR THE SPRING MODEL #####

list_MLR=list()
list_ANN=list()
list_MT=list()
list_BMT=list()
list_RF=list()

list_MLR_predictors = list() # List to store the number of predictors

for (m in 1:100){
  temp_list = iter(formula = T_Spring ~ ., dataset = Spring_data ,
 multiply = m)
  list_MLR[[m]]=temp_list[[1]][4]
  list_ANN[[m]]=temp_list[[2]][4]
  list_MT[[m]]=temp_list[[3]][4]
  list_BMT[[m]]=temp_list[[4]][4]
  list_RF[[m]]=temp_list[[5]][4]

  list_MLR_predictors[[m]]=temp_list[[6]]
}

listVec <- lapply( list_MLR, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_MLR=data.frame(m)
rownames(df_MLR)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1, 100), sep='')
colnames(df_MLR)=c(fold_key,'Mean')

listVec <- lapply( list_ANN, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_ANN=data.frame(m)
rownames(df_ANN)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_ANN)=c(fold_key,'Mean')

listVec <- lapply(list_MT, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_MT=data.frame(m)
rownames(df_MT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 'RRSE_cal',
                  'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_MT)=c(fold_key,'Mean')

listVec <- lapply(list_BMT, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
```

```
m = cbind(m,middle_calculations)
df_BMT=data.frame(m)
rownames(df_BMT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                      'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_BMT) = c(fold_key,'Mean')

listVec <- lapply(list_RF, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_RF=data.frame(m)
rownames(df_RF)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_RF)=c(fold_key,'Mean')


df_all = round(rbind(df_MLR, df_ANN, df_MT,df_BMT, df_RF),4)
df_all$sd=apply(df_all[,c(1:100)], 1,sd)

r_cal = df_all[c(seq(1,50, by = 10)),c(1:100)]
r_val = df_all[c(seq(2,50, by = 10)),c(1:100)]

RMSE_cal = df_all[c(seq(3,50, by = 10)),c(1:100)]
RMSE_val = df_all[c(seq(4,50, by = 10)),c(1:100)]

RRSE_cal = df_all[c(seq(5,50, by = 10)),c(1:100)]
RRSE_val = df_all[c(seq(6,50, by = 10)),c(1:100)]

RE_cal = df_all[c(seq(7,50, by = 10)),c(1:100)]
RE_val = df_all[c(seq(8,50, by = 10)),c(1:100)]

CE_cal = df_all[c(seq(9,50, by = 10)),c(1:100)]
CE_val = df_all[c(seq(10,50, by = 10)),c(1:100)]

AVG_rank = data.frame(rowMeans(apply(-r_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-r_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
r_cal_ranks = cbind(AVG_rank, shareOne)
names(r_cal_ranks) = c('Average Rank', 'Share of Rank 1')
```

```
AVG_rank = data.frame(rowMeans(apply(-r_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-r_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
r_val_ranks = cbind(AVG_rank, shareOne)
names(r_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RMSE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RMSE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RMSE_cal_ranks = cbind(AVG_rank, shareOne)
names(RMSE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RMSE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RMSE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RMSE_val_ranks = cbind(AVG_rank, shareOne)
names(RMSE_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RRSE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RRSE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RRSE_cal_ranks = cbind(AVG_rank, shareOne)
names(RRSE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RRSE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RRSE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RRSE_val_ranks = cbind(AVG_rank, shareOne)
names(RRSE_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-RE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-RE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RE_cal_ranks = cbind(AVG_rank, shareOne)
names(RE_cal_ranks) = c('Average Rank', 'Share of Rank 1')
```

```
AVG_rank = data.frame(rowMeans(apply(-RE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-RE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RE_val_ranks = cbind(AVG_rank, shareOne)
names(RE_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-CE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-CE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
CE_cal_ranks = cbind(AVG_rank, shareOne)
names(CE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-CE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-CE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
CE_val_ranks = cbind(AVG_rank, shareOne)
names(CE_val_ranks) = c('Average Rank', 'Share of Rank 1')

ranks_together = rbind(r_cal_ranks,r_val_ranks,
                       RMSE_cal_ranks, RMSE_val_ranks,
                       RRSE_cal_ranks, RRSE_val_ranks,
                       RE_cal_ranks, RE_val_ranks,
                       CE_cal_ranks, CE_val_ranks)

ranks_together$Method = c('MLR', 'ANN', 'MT', 'BMT', 'RF')
ranks_together$Period =
 c('cal','cal','cal','cal','cal','val','val','val','val','val')
ranks_together$Measure = c('r','r','r','r','r','r','r','r','r','r',

 'RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE',

 'RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE',

 'RE','RE','RE','RE','RE','RE','RE','RE','RE','RE',

 'CE','CE','CE','CE','CE','CE','CE','CE','CE','CE')

colnames(ranks_together)[1]='Avg_rank'
togeter_AVG_rank = cast(ranks_together, formula = Measure + Period ~
 Method, value =c('Avg_rank'))
```

```
togeter_AVG_rank$Measure  <- factor(togeter_AVG_rank$Measure, levels =
 c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
togeter_AVG_rank=togeter_AVG_rank[order(togeter_AVG_rank$Measure), ]
togeter_AVG_rank = dplyr::select(togeter_AVG_rank, Measure, Period,
 MLR, ANN, MT, BMT, RF)

colnames(ranks_together)[2]='Share_rank1'
together_share1 = cast(ranks_together, formula = Measure + Period ~
 Method, value =c('Share_rank1'))
together_share1$Measure  <- factor(together_share1$Measure, levels =
 c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
together_share1=together_share1[order(together_share1$Measure), ]
together_share1 = dplyr::select(together_share1, Measure, Period, MLR,
 ANN, MT, BMT, RF)

#
df_means_sd = rbind(df_MLR, df_ANN, df_MT, df_BMT, df_RF)
df_means_sd$sd=apply(df_means_sd[,c(1:100)], 1, sd)
df_means_sd$Method = c('MLR', 'MLR', 'MLR', 'MLR', 'MLR', 'MLR',
 'MLR', 'MLR', 'MLR', 'MLR',
                       'ANN',
 'ANN','ANN','ANN','ANN','ANN','ANN','ANN','ANN','ANN',
                       'MT', 'MT', 'MT', 'MT', 'MT', 'MT', 'MT', 'MT',
 'MT', 'MT',

 'BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT',

 'RF','RF','RF','RF','RF','RF','RF','RF','RF','RF')
df_means_sd$Period = c('cal','val')
df_means_sd$Measure = c('r','r', 'RMSE','RMSE', 'RRSE', 'RRSE',
 'RE','RE','CE', 'CE')

together_means_sd = cast(df_means_sd, formula = Measure + Period ~
 Method, value =c('Mean'))
together_means_sd$Measure  <- factor(together_means_sd$Measure, levels
 = c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
together_means_sd=together_means_sd[order(together_means_sd$Measure),
 ]
together_means = dplyr::select(together_means_sd, Measure, Period,
 MLR, ANN, MT, BMT, RF)

together_means_sd = cast(df_means_sd, formula = Measure + Period ~
 Method, value =c('sd'))
```

```
together_means_sd$Measure  <- factor(together_means_sd$Measure, levels
 = c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
together_means_sd=together_means_sd[order(together_means_sd$Measure),
 ]
together_sd = dplyr::select(together_means_sd, Measure, Period, MLR,
 ANN, MT, BMT, RF)

colnames(together_means) =c("Measure", "Period", "MLR_M", "ANN_M",
 "MT_M", "BMT_M","RF_M")
colnames(together_sd) = c("Measure_SD", "Period_SD", "MLR_SD",
 "ANN_SD", "MT_SD", "BMT_SD","RF_SD")
colnames(togeter_AVG_rank) = c("Measure_AR", "Period_AR", "MLR_AR",
 "ANN_AR", "MT_AR", "BMT_AR","RF_AR")
colnames(together_share1) = c("Measure_S1", "Period_S1", "MLR_S1",
 "ANN_S1", "MT_S1", "BMT_S1","RF_S1")

TOGETHER =
 cbind(together_means,together_sd,togeter_AVG_rank,together_share1)
TOGETHER_MEAN_SD_SPRING = dplyr::select(TOGETHER, Measure, Period,
 MLR_M,MLR_SD,
                                ANN_M,ANN_SD,
                                MT_M, MT_SD,
                                BMT_M,BMT_SD,
                                RF_M,RF_SD)

TOGETHER_RANKS_SPRING = dplyr::select(TOGETHER, Measure, Period,
 MLR_AR,MLR_S1,
                              ANN_AR,ANN_S1,
                              MT_AR,MT_S1,
                              BMT_AR,BMT_S1,
                              RF_AR,RF_S1)

# On average, how many predictors did MLR model use?
listVec <- lapply(list_MLR_predictors, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
N_predicotrs_spring <- round(middle_calculations, 2)

##### Calculations for the Summer Model #######

list_MLR=list()
list_ANN=list()
list_MT=list()
list_BMT=list()
```

```
list_RF=list()

list_MLR_predictors = list() # List to store the number of predictors

for (m in 1:100){
  temp_list = iter(formula = T_Summer ~ ., dataset = Summer_data ,
 multiply = m)
  list_MLR[[m]]=temp_list[[1]][4]
  list_ANN[[m]]=temp_list[[2]][4]
  list_MT[[m]]=temp_list[[3]][4]
  list_BMT[[m]]=temp_list[[4]][4]
  list_RF[[m]]=temp_list[[5]][4]

  list_MLR_predictors[[m]]=temp_list[[6]]
}

listVec <- lapply( list_MLR, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_MLR=data.frame(m)
rownames(df_MLR)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1, 100), sep='')
colnames(df_MLR)=c(fold_key,'Mean')

listVec <- lapply( list_ANN, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_ANN=data.frame(m)
rownames(df_ANN)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_ANN)=c(fold_key,'Mean')

listVec <- lapply(list_MT, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
```

```
df_MT=data.frame(m)
rownames(df_MT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 'RRSE_cal',
                  'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_MT)=c(fold_key,'Mean')

listVec <- lapply(list_BMT, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_BMT=data.frame(m)
rownames(df_BMT)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val',
 'RRSE_cal',
                   'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal',
 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_BMT) = c(fold_key,'Mean')

listVec <- lapply(list_RF, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
m = cbind(m,middle_calculations)
df_RF=data.frame(m)
rownames(df_RF)=c('r_cal', 'r_val','RMSE_cal', 'RMSE_val', 'RRSE_cal',
                  'RRSE_val',  'RE_cal', 'RE_val', 'CE_cal', 'CE_val')
fold_key = paste('CV_',seq(1,100), sep='')
colnames(df_RF)=c(fold_key,'Mean')


df_all = round(rbind(df_MLR, df_ANN, df_MT,df_BMT, df_RF),4)
df_all$sd=apply(df_all[,c(1:100)], 1,sd)

r_cal = df_all[c(seq(1,50, by = 10)),c(1:100)]
r_val = df_all[c(seq(2,50, by = 10)),c(1:100)]

RMSE_cal = df_all[c(seq(3,50, by = 10)),c(1:100)]
RMSE_val = df_all[c(seq(4,50, by = 10)),c(1:100)]

RRSE_cal = df_all[c(seq(5,50, by = 10)),c(1:100)]
RRSE_val = df_all[c(seq(6,50, by = 10)),c(1:100)]

RE_cal = df_all[c(seq(7,50, by = 10)),c(1:100)]
RE_val = df_all[c(seq(8,50, by = 10)),c(1:100)]
```

```r
CE_cal = df_all[c(seq(9,50, by = 10)),c(1:100)]
CE_val = df_all[c(seq(10,50, by = 10)),c(1:100)]

AVG_rank = data.frame(rowMeans(apply(-r_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-r_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
r_cal_ranks = cbind(AVG_rank, shareOne)
names(r_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-r_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-r_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
r_val_ranks = cbind(AVG_rank, shareOne)
names(r_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RMSE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RMSE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RMSE_cal_ranks = cbind(AVG_rank, shareOne)
names(RMSE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RMSE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RMSE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RMSE_val_ranks = cbind(AVG_rank, shareOne)
names(RMSE_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RRSE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RRSE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RRSE_cal_ranks = cbind(AVG_rank, shareOne)
names(RRSE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(RRSE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(RRSE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RRSE_val_ranks = cbind(AVG_rank, shareOne)
names(RRSE_val_ranks) = c('Average Rank', 'Share of Rank 1')
```

```
AVG_rank = data.frame(rowMeans(apply(-RE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-RE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RE_cal_ranks = cbind(AVG_rank, shareOne)
names(RE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-RE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-RE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
RE_val_ranks = cbind(AVG_rank, shareOne)
names(RE_val_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-CE_cal, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-CE_cal, 2, rank,
 ties.method='first'), 1, count_ones)/100)
CE_cal_ranks = cbind(AVG_rank, shareOne)
names(CE_cal_ranks) = c('Average Rank', 'Share of Rank 1')

AVG_rank = data.frame(rowMeans(apply(-CE_val, 2, rank,
 ties.method='first')))
shareOne = data.frame(apply(apply(-CE_val, 2, rank,
 ties.method='first'), 1, count_ones)/100)
CE_val_ranks = cbind(AVG_rank, shareOne)
names(CE_val_ranks) = c('Average Rank', 'Share of Rank 1')

ranks_together = rbind(r_cal_ranks,r_val_ranks,
                       RMSE_cal_ranks, RMSE_val_ranks,
                       RRSE_cal_ranks, RRSE_val_ranks,
                       RE_cal_ranks, RE_val_ranks,
                       CE_cal_ranks, CE_val_ranks)

ranks_together$Method = c('MLR', 'ANN', 'MT', 'BMT', 'RF')
ranks_together$Period =
 c('cal','cal','cal','cal','cal','val','val','val','val','val')
ranks_together$Measure = c('r','r','r','r','r','r','r','r','r','r',

 'RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE','RMSE',

 'RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE','RRSE',
 'RE','RE','RE','RE','RE','RE','RE','RE','RE','RE',

 'CE','CE','CE','CE','CE','CE','CE','CE','CE','CE')

colnames(ranks_together)[1]='Avg_rank'
togeter_AVG_rank = cast(ranks_together, formula = Measure + Period ~
 Method, value =c('Avg_rank'))
togeter_AVG_rank$Measure  <- factor(togeter_AVG_rank$Measure, levels =
 c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
togeter_AVG_rank=togeter_AVG_rank[order(togeter_AVG_rank$Measure), ]
togeter_AVG_rank = dplyr::select(togeter_AVG_rank, Measure, Period,
 MLR, ANN, MT, BMT, RF)

colnames(ranks_together)[2]='Share_rank1'
together_share1 = cast(ranks_together, formula = Measure + Period ~
 Method, value =c('Share_rank1'))
together_share1$Measure  <- factor(together_share1$Measure, levels =
 c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
together_share1=together_share1[order(together_share1$Measure), ]
together_share1 = dplyr::select(together_share1, Measure, Period, MLR,
 ANN, MT, BMT, RF)

#
df_means_sd = rbind(df_MLR, df_ANN, df_MT, df_BMT, df_RF)
df_means_sd$sd=apply(df_means_sd[,c(1:100)], 1, sd)
df_means_sd$Method = c('MLR', 'MLR', 'MLR', 'MLR', 'MLR', 'MLR',
 'MLR', 'MLR', 'MLR', 'MLR',
                       'ANN',
 'ANN','ANN','ANN','ANN','ANN','ANN','ANN','ANN','ANN',
                       'MT', 'MT', 'MT', 'MT', 'MT', 'MT', 'MT', 'MT',
 'MT', 'MT',

 'BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT','BMT',

 'RF','RF','RF','RF','RF','RF','RF','RF','RF','RF')
df_means_sd$Period = c('cal','val')
df_means_sd$Measure = c('r','r', 'RMSE','RMSE', 'RRSE', 'RRSE',
 'RE','RE','CE', 'CE')

together_means_sd = cast(df_means_sd, formula = Measure + Period ~
 Method, value =c('Mean'))
together_means_sd$Measure  <- factor(together_means_sd$Measure, levels
 = c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
```

```
together_means_sd=together_means_sd[order(together_means_sd$Measure),
 ]
together_means = dplyr::select(together_means_sd, Measure, Period,
 MLR, ANN, MT, BMT, RF)

together_means_sd = cast(df_means_sd, formula = Measure + Period ~
 Method, value =c('sd'))
together_means_sd$Measure  <- factor(together_means_sd$Measure, levels
 = c('r', 'RMSE', 'RRSE', 'RE', 'CE'))
together_means_sd=together_means_sd[order(together_means_sd$Measure),
 ]
together_sd = dplyr::select(together_means_sd, Measure, Period, MLR,
 ANN, MT, BMT, RF)

colnames(together_means) =c("Measure", "Period", "MLR_M", "ANN_M",
 "MT_M", "BMT_M","RF_M")
colnames(together_sd) = c("Measure_SD", "Period_SD", "MLR_SD",
 "ANN_SD", "MT_SD", "BMT_SD","RF_SD")
colnames(togeter_AVG_rank) = c("Measure_AR", "Period_AR", "MLR_AR",
 "ANN_AR", "MT_AR", "BMT_AR","RF_AR")
colnames(together_share1) = c("Measure_S1", "Period_S1", "MLR_S1",
 "ANN_S1", "MT_S1", "BMT_S1","RF_S1")

TOGETHER =
 cbind(together_means,together_sd,togeter_AVG_rank,together_share1)
TOGETHER_MEAN_SD_SUMMER = dplyr::select(TOGETHER, Measure, Period,
 MLR_M,MLR_SD,
                                         ANN_M,ANN_SD,
                                         MT_M, MT_SD,
                                         BMT_M,BMT_SD,
                                         RF_M,RF_SD)

TOGETHER_RANKS_SUMMER = dplyr::select(TOGETHER, Measure, Period,
 MLR_AR,MLR_S1,
                                      ANN_AR,ANN_S1,
                                      MT_AR,MT_S1,
                                      BMT_AR,BMT_S1,
                                      RF_AR,RF_S1)

# On average, how many predictors did MLR model use?
listVec <- lapply(list_MLR_predictors, c, recursive=TRUE)
m <- do.call(cbind, listVec)
middle_calculations <- apply(m, 1, mean)
N_predicotrs_summer <- round(middle_calculations, 2)
```

```
#########################################
### The final Results are listed here ###
#########################################

TOGETHER_MEAN_SD_SPRING # Table 3A - upper part
TOGETHER_RANKS_SPRING # Table 3A - lower part
TOGETHER_MEAN_SD_SUMMER #Table 3B - upper part
TOGETHER_RANKS_SUMMER # Table 3B - lower part

N_predicotrs_spring # average number of predictors for spring MLR
 model
N_predicotrs_summer # average number of predictors for summer MLR
 model
```