Random Splitting Random Forest for Categorical Response

Mohammad Fayaz

2022-09-13

Setting Up

We download and install the **RSRF** in the Rstudio:

```
#library(devtools)
#install_github("mohammad-fayaz/RSRF")
library(RSRF)
library(data.table)
```

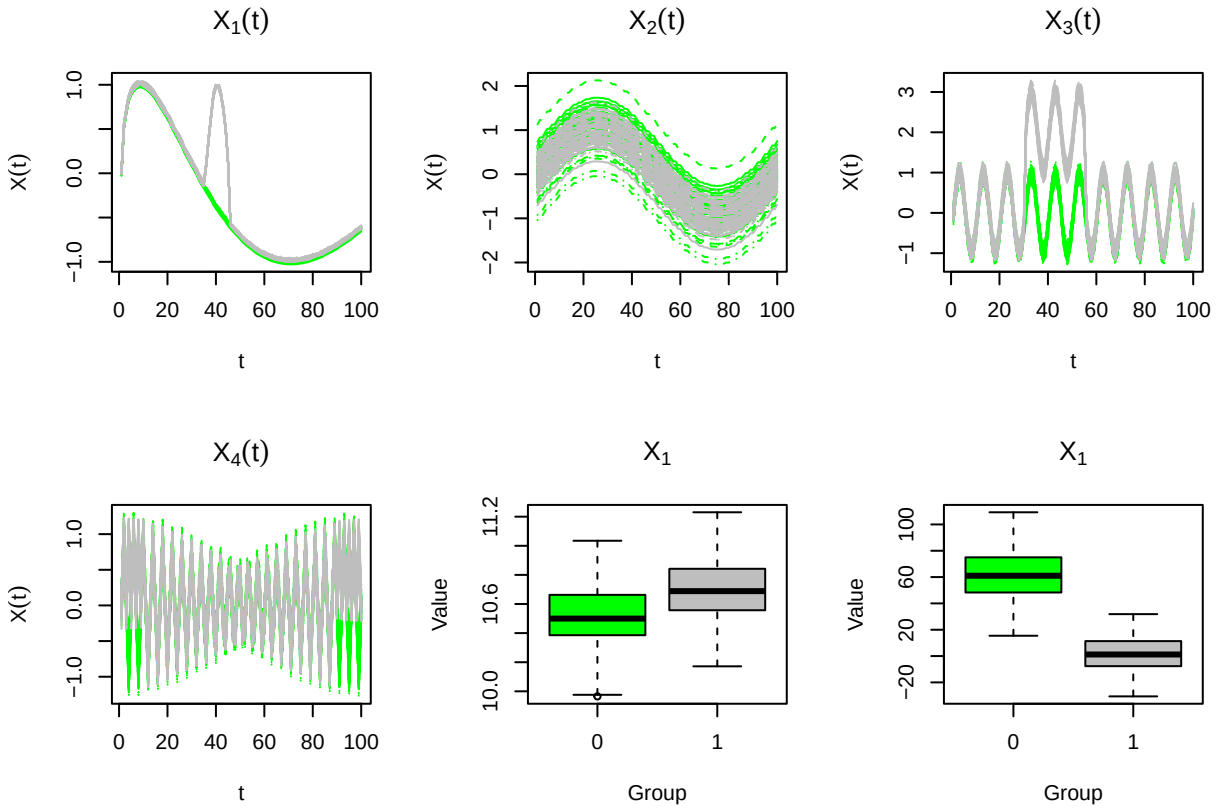
Simulating Data

We simulate the dataset with **makeSIMData** function. It has four arguments including: **nss_1** is the number of sample size of group 1,**nss_2** is the number of sample size of group 2, *responseType* is the type of response and **Seed** is the seed number in the memory.

```
nSample_1 = 100
nSample_2 = 100
Sim_Data <- makeSIMData(nSS_1 = 100 , nSS_2 = 100 , responseType = "Categorical", Seed=2022)
```

We plot the simulated data.

```
par(mfrow=c(2,3))
matplot(t(Sim_Data[[1]]),type="l",col=c(rep("Green",nSample_1),rep("Grey",nSample_2)),xlab = expression
matplot(t(Sim_Data[[2]]),type="l",col=c(rep("Green",nSample_1),rep("Grey",nSample_2)),xlab = expression
matplot(t(Sim_Data[[3]]),type="l",col=c(rep("Green",nSample_1),rep("Grey",nSample_2)),xlab = expression
matplot(t(Sim_Data[[4]]),type="l",col=c(rep("Green",nSample_1),rep("Grey",nSample_2)),xlab = expression
boxplot(Sim_Data[[5]][1][-(1:4),] ~ Sim_Data[[6]][-(1:4)] ,col=c("Green","Grey"),main=expression(X[1]),
boxplot(Sim_Data[[5]][2][-(1:4),] ~ Sim_Data[[6]][-(1:4)] ,col=c("Green","Grey"),main=expression(X[1]),
```



Models

Model Specification

First we define the model arguments:

```
Response  <- as.factor(c(Sim_Data[[6]]))

## Functional Covariate Specifications
FDList <- list()
FDList[[1]] <- Sim_Data[[1]]
FDList[[2]] <- Sim_Data[[2]]
FDList[[3]] <- Sim_Data[[3]]
FDList[[4]] <- Sim_Data[[4]]

 ## Parameter Specification for each Functioanl Covariate
Min <- list()
Max <- list()
PARAMS_1 <- list()
DIST_1 <- list()
TYPE <- list()
PARAMS_2 <- list()
STATS <- list()
DIST_2 <- list()
```

```
### Functional Covariate 1 Specificaion
Min[[1]] <- 1                    ## minimum time domain
Max[[1]] <- 100                  ## maximum time domain
DIST_1[[1]] <- c("Exponential")  ## The Random splitting distribution
PARAMS_1[[1]] <- c(0.1,0,0)      ## The parameter for the distribution
TYPE[[1]] <- c("Disjoint")       ## Type of splitting
DIST_2[[1]] <- c("")             ## if type is overlap , the distribution of the overlap
PARAMS_2[[1]] <- c(0,0,0)        ## if type is overlap , the parameter of the distribution
STATS[[1]]<- c("mean")          ## the statistics for each interval

### Functional Covariate 2 Specificaion
Min[[2]] <- 1                    ## minimum time domain
Max[[2]] <- 100                  ## maximum time domain
DIST_1[[2]] <- c("Exponential")  ## The Random splitting distribution
PARAMS_1[[2]] <- c(0.1,0,0)      ## The parameter for the distribution
TYPE[[2]] <- c("Disjoint")       ## Type of splitting
DIST_2[[2]] <- c("")             ## if type is overlap , the distribution of the overlap
PARAMS_2[[2]] <- c(0,0,0)        ## if type is overlap , the parameter of the distribution
STATS[[2]]<- c("mean")          ## the statistics for each interval

### Functional Covariate 3 Specificaion
Min[[3]] <- 1                    ## minimum time domain
Max[[3]] <- 100                  ## maximum time domain
DIST_1[[3]] <- c("Exponential")  ## The Random splitting distribution
PARAMS_1[[3]] <- c(0.1,0,0)      ## The parameter for the distribution
TYPE[[3]] <- c("Disjoint")       ## Type of splitting
DIST_2[[3]] <- c("")             ## if type is overlap , the distribution of the overlap
PARAMS_2[[3]] <- c(0,0,0)        ## if type is overlap , the parameter of the distribution
STATS[[3]]<- c("mean")          ## the statistics for each interval

### Functional Covariate 4 Specificaion
Min[[4]] <- 1                    ## minimum time domain
Max[[4]] <- 100                  ## maximum time domain
DIST_1[[4]] <- c("Exponential")  ## The Random splitting distribution
PARAMS_1[[4]] <- c(0.1,0,0)      ## The parameter for the distribution
TYPE[[4]] <- c("Disjoint")       ## Type of splitting
DIST_2[[4]] <- c("")             ## if type is overlap , the distribution of the overlap
PARAMS_2[[4]] <- c(0,0,0)        ## if type is overlap , the parameter of the distribution
STATS[[4]]<- c("mean")          ## the statistics for each interval

## Non-Functional Covariate Specifications
Covariates_all <- data.frame(c_01= Sim_Data[[5]][,1],c_02= Sim_Data[[5]][,2])
## General Settings
mtree0 <- 100
k0 <- 500
```

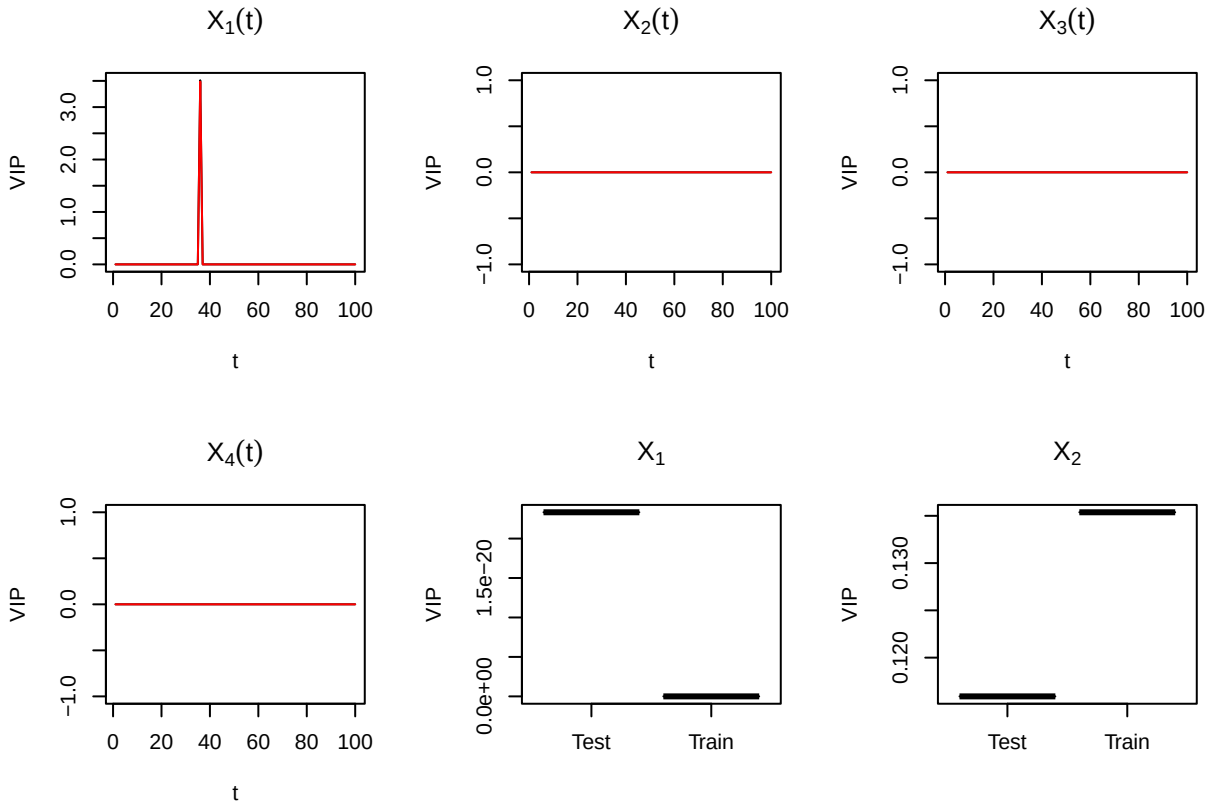
Bagging We run the bagging model. In this reagrd **RFMethod = c("Bagging")** and **VIP_Algorithm = c("Normal")** :

```
######## Running the model
BG_NO_Result <- RS_RF_SRC_H(
  ResponseVar=Response,
  Covariates=Covariates_all,
```

```
  RawData=FDList,
  Stat=STATS ,
  min=Min,
  max=Max,
  Params=PARAMS_1,
  Distribution=DIST_1,
  Params2=PARAMS_2,
  Distribution2=DIST_2,
  type=TYPE,
  m=mtree0,
  k=k0,
  ResponseType="Categorical",
  RFMethod = c("Bagging"),
  NUM_Covariates=1,
  Block.Size = 10 ,
  SplitRule = c("gini"),
  Importance= TRUE,
  Ensemble = c("all"),
  Proximity = TRUE,
  VIP_Algorithm = c("Normal"))
```

We plot the Variable Importance Plot for each covariate:

```
BG_NO_Result_IMP <- data.table(BG_NO_Result$IMP_VALUE)
par(mfrow=c(2,3))
## X1(t)
plot(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][3:102]$V1, xlab = expression(t),ylab = "VIP"
lines(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS][3:102]$V1, xlab = expression(t),ylab = "VIP"
## X2(t)
plot(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][103:202]$V1, xlab = expression(t),ylab = "VI
lines(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS][103:202]$V1, xlab = expression(t),ylab = "VI
## X3(t)
plot(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][203:302]$V1, xlab = expression(t),ylab = "VI
lines(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS][203:302]$V1, xlab = expression(t),ylab = "VI
## X4(t)
plot(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][303:402]$V1, xlab = expression(t),ylab = "VI
lines(x=1:100,y=BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS][303:402]$V1, xlab = expression(t),ylab = "VI
## X1
boxplot(c(BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][1]$V1, BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS]
## X2
boxplot(c(BG_NO_Result_IMP[,mean(VIMP_Train),by=VARS][2]$V1, BG_NO_Result_IMP[,mean(VIMP_Test),by=VARS]
```



We see only the first functional covariate between 35 and 40 has high VIP and the second non-functional covariate in both train (Black) and test(Red). We have following summary statistics **Accuracy**, **Sensitivity** and **Specificity** with **RS_Summary()** function:

```
RS_Summary(Dataset =BG_NO_Result$Result , Outcome_Type = c("Categorical"))
```

```
##     Train_ACC_Mean Train_ACC_sd Train_ACC_Q1 Train_ACC_Q3 Test_ACC_Mean
## 25%              1            0            1            1     0.9941667
##     Test_ACC_sd Test_ACC_Q1 Test_ACC_Q3 Train_SEN_Mean Train_SEN_sd
## 25%  0.01428722           1           1              1            0
##     Train_SEN_Q1 Train_SEN_Q3 Test_SEN_Mean Test_SEN_sd Test_SEN_Q1 Test_SEN_Q3
## 25%            1            1     0.9974375  0.01846575           1           1
##     Train_SPE_Mean Train_SPE_sd Train_SPE_Q1 Train_SPE_Q3 Test_SPE_Mean
## 25%              1            0            1            1     0.9911775
##     Test_SPE_sd Test_SPE_Q1 Test_SPE_Q3
## 25%   0.0171091           1           1
```

Random Forest We run the random forest model. In this reagrd **RFMethod = c("RandomForest")** and **VIP_Algorithm = c("Normal")** . In this regard, we use only **2** covariate among **6** covariates in each tree by setting **NUM_Covariates=2**:

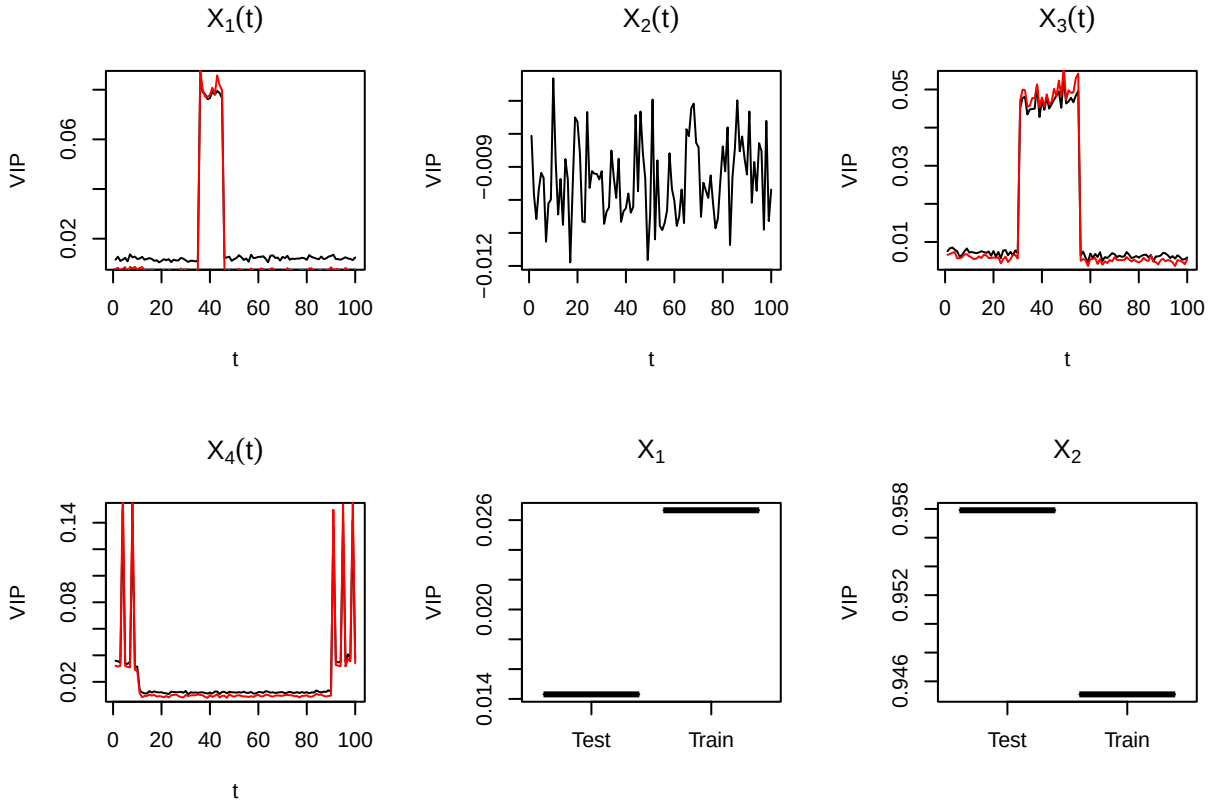
```
######## Running the model
RF_NO_Result <- RS_RF_SRC_H(
  ResponseVar=Response,
  Covariates=Covariates_all,
```

```
  RawData=FDList,
  Stat=STATS ,
  min=Min,
  max=Max,
  Params=PARAMS_1,
  Distribution=DIST_1,
  Params2=PARAMS_2,
  Distribution2=DIST_2,
  type=TYPE,
  m=mtree0,
  k=k0,
  ResponseType="Categorical",
  RFMethod = c("RandomForest"),
  NUM_Covariates=2,
  Block.Size = 10 ,
  SplitRule = c("gini"),
  Importance= TRUE,
  Ensemble = c("all"),
  Proximity = TRUE,
  VIP_Algorithm = c("Normal"))
```

We plot the Variable Importance Plot for each covariate:

```
RF_NO_Result_IMP <- data.table(RF_NO_Result$IMP_VALUE)
RF_NO_Result_IMP <- RF_NO_Result_IMP[is.na(COV_NUM) == FALSE]

RF_NO_Result_IMP_Train <- RF_NO_Result_IMP[,mean(VIMP_Train),by=VARS]
RF_NO_Result_IMP_Test  <- RF_NO_Result_IMP[,mean(VIMP_Test),by=VARS]

## Train VIP
VIP_C1_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "c_01"),]
VIP_C2_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "c_02"),]

VIP_X1_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "x_01"),]
VIP_X2_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "x_02"),]
VIP_X3_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "x_03"),]
VIP_X4_Train <- RF_NO_Result_IMP_Train[which(substr(RF_NO_Result_IMP_Train$VARS,1,4) == "x_04"),]

VIP_X1_Train$Time <- as.numeric(substr(VIP_X1_Train$VARS,6,9))
VIP_X2_Train$Time <- as.numeric(substr(VIP_X2_Train$VARS,6,9))
VIP_X3_Train$Time <- as.numeric(substr(VIP_X3_Train$VARS,6,9))
VIP_X4_Train$Time <- as.numeric(substr(VIP_X4_Train$VARS,6,9))

VIP_X1_Train <- VIP_X1_Train[order(VIP_X1_Train$Time,decreasing = FALSE)]
VIP_X2_Train <- VIP_X2_Train[order(VIP_X2_Train$Time,decreasing = FALSE)]
VIP_X3_Train <- VIP_X3_Train[order(VIP_X3_Train$Time,decreasing = FALSE)]
VIP_X4_Train <- VIP_X4_Train[order(VIP_X4_Train$Time,decreasing = FALSE)]

## Test VIP
VIP_C1_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "c_01"),]
VIP_C2_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "c_02"),]

VIP_X1_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "x_01"),]
```

```r
VIP_X2_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "x_02"),]
VIP_X3_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "x_03"),]
VIP_X4_Test <- RF_NO_Result_IMP_Test[which(substr(RF_NO_Result_IMP_Test$VARS,1,4) == "x_04"),]

VIP_X1_Test$Time <- as.numeric(substr(VIP_X1_Test$VARS,6,9))
VIP_X2_Test$Time <- as.numeric(substr(VIP_X2_Test$VARS,6,9))
VIP_X3_Test$Time <- as.numeric(substr(VIP_X3_Test$VARS,6,9))
VIP_X4_Test$Time <- as.numeric(substr(VIP_X4_Test$VARS,6,9))

VIP_X1_Test <- VIP_X1_Test[order(VIP_X1_Test$Time,decreasing = FALSE)]
VIP_X2_Test <- VIP_X2_Test[order(VIP_X2_Test$Time,decreasing = FALSE)]
VIP_X3_Test <- VIP_X3_Test[order(VIP_X3_Test$Time,decreasing = FALSE)]
VIP_X4_Test <- VIP_X4_Test[order(VIP_X4_Test$Time,decreasing = FALSE)]

par(mfrow=c(2,3))
## X1(t)
plot(x=1:100,y= VIP_X1_Train$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[1](t)),type="l")
lines(x=1:100,y=VIP_X1_Test$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[1](t)),type="l",c
## X2(t)
plot(x=1:100,y=VIP_X2_Train$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[2](t)),type="l")
lines(x=1:100,y=VIP_X2_Test$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[2](t)),type="l",c
## X3(t)
plot(x=1:100,y=VIP_X3_Train$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[3](t)),type="l")
lines(x=1:100,y=VIP_X3_Test$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[3](t)),type="l",c
## X4(t)
plot(x=1:100,y=VIP_X4_Train$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[4](t)),type="l")
lines(x=1:100,y=VIP_X4_Test$V1, xlab = expression(t),ylab = "VIP" ,main= expression(X[4](t)),type="l",c
## X1
boxplot(c(VIP_C1_Train$V1, VIP_C1_Test$V1) ~ c("Train","Test"), xlab = c(""),ylab = "VIP" ,main= expres
## X2
boxplot(c(VIP_C2_Train$V1, VIP_C2_Test$V1) ~ c("Train","Test"), xlab = c(""),ylab = "VIP" ,main= expres
```



As we expected the VIP plot for all covariates are made. The summary statistics of this model is:

```
RS_Summary(Dataset =RF_NO_Result$Result , Outcome_Type = c("Categorical"))
```

```
##     Train_ACC_Mean Train_ACC_sd Train_ACC_Q1 Train_ACC_Q3 Test_ACC_Mean
## 25%              1            0            1            1        0.9785
##     Test_ACC_sd Test_ACC_Q1 Test_ACC_Q3 Train_SEN_Mean Train_SEN_sd
## 25%  0.03215903   0.9666667           1              1            0
##     Train_SEN_Q1 Train_SEN_Q3 Test_SEN_Mean Test_SEN_sd Test_SEN_Q1 Test_SEN_Q3
## 25%            1            1     0.9872641  0.02947529           1           1
##     Train_SPE_Mean Train_SPE_sd Train_SPE_Q1 Train_SPE_Q3 Test_SPE_Mean
## 25%              1            0            1            1     0.9689505
##     Test_SPE_sd Test_SPE_Q1 Test_SPE_Q3
## 25%  0.04671908   0.9480249           1
```

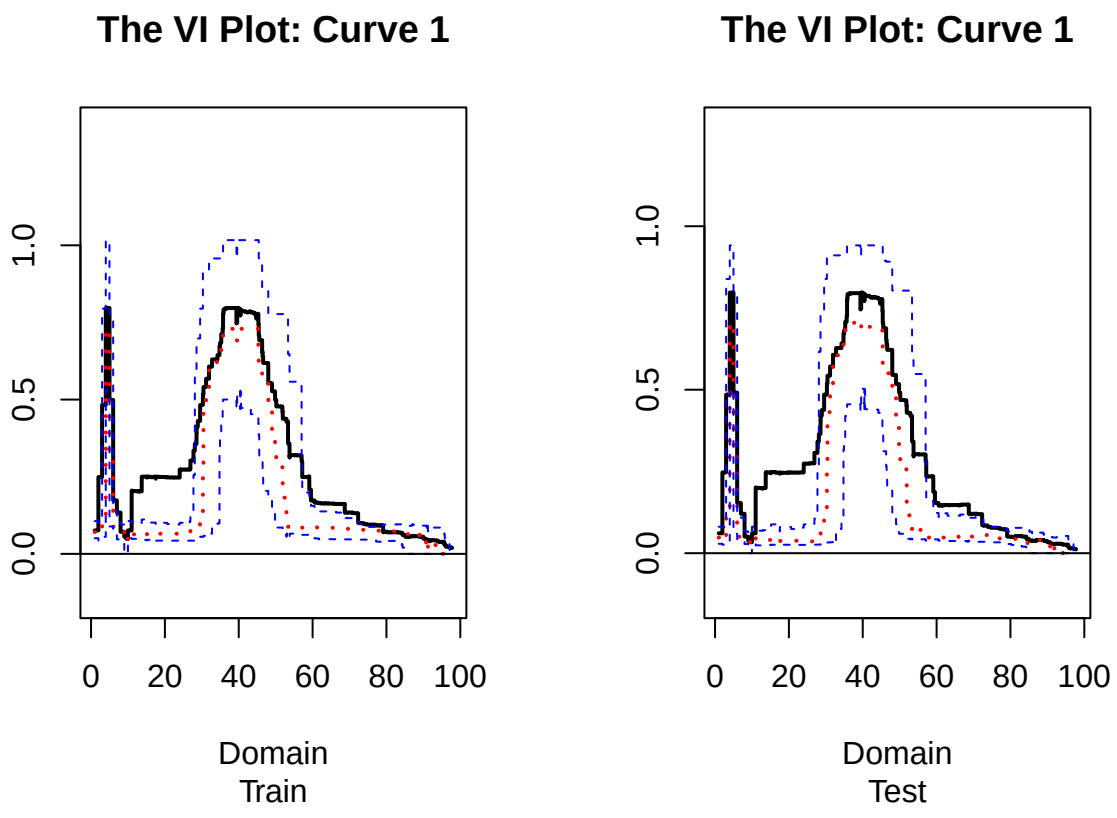
Random Splitting Random Forest We run the random splitting random forest model. In this reagrd **RFMethod = c("RandomForest")** and **VIP_Algorithm = c("RS")**. In this regard, we use only **2** covariate among **6** covariates in each tree by setting **NUM_Covariates=2**:

```
######## Running the model
RF_RS_Result <- RS_RF_SRC_H(
  ResponseVar=Response,
  Covariates=Covariates_all,
  RawData=FDList,
  Stat=STATS ,
```

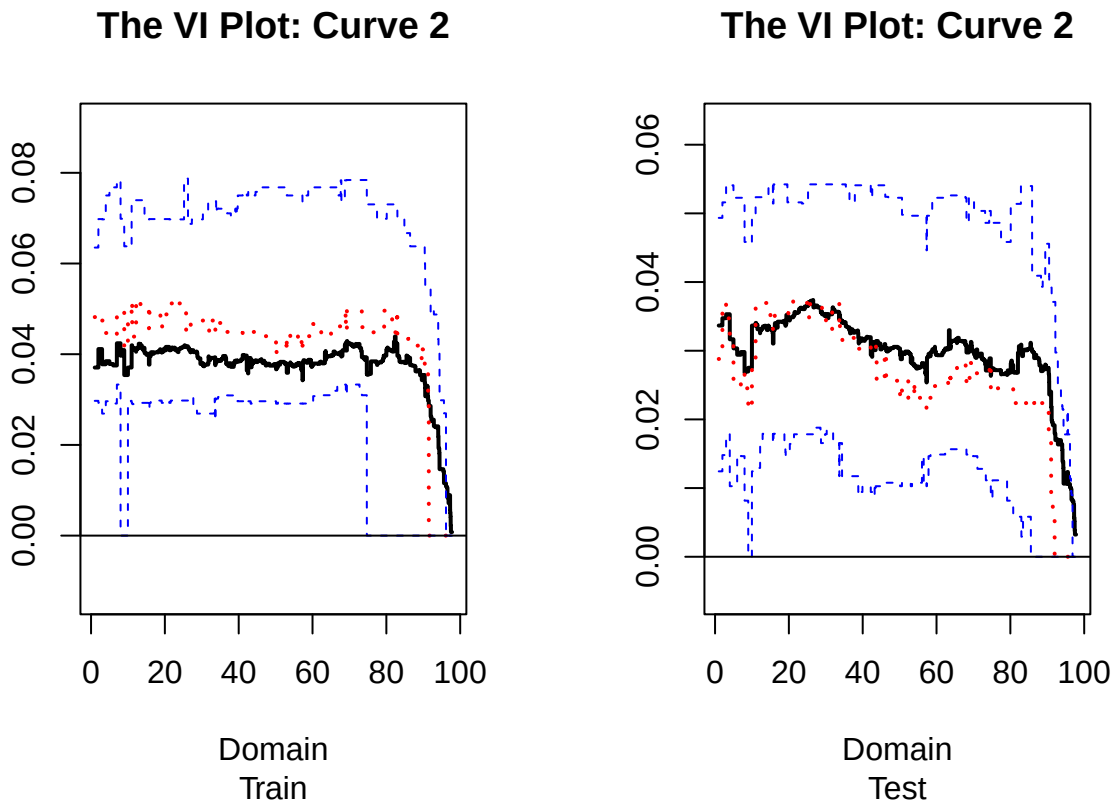
```
  min=Min,
  max=Max,
  Params=PARAMS_1,
  Distribution=DIST_1,
  Params2=PARAMS_2,
  Distribution2=DIST_2,
  type=TYPE,
  m=mtree0,
  k=k0,
  ResponseType="Categorical",
  RFMethod = c("RandomForest"),
  NUM_Covariates=2,
  Block.Size = 10 ,
  SplitRule = c("gini"),
  Importance= TRUE,
  Ensemble = c("all"),
  Proximity = TRUE,
  VIP_Algorithm = c("RS"))
```

We plot the Variable Importance Plot for each covariate:

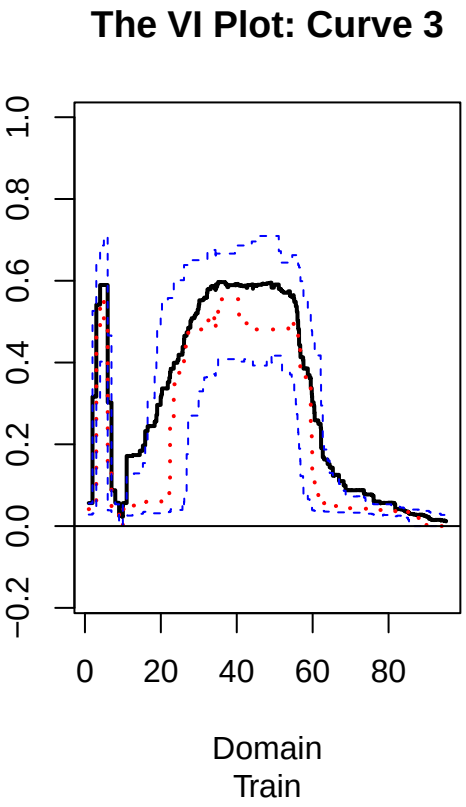
```
## X1(t)
par(mfrow=c(1,2))
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
```

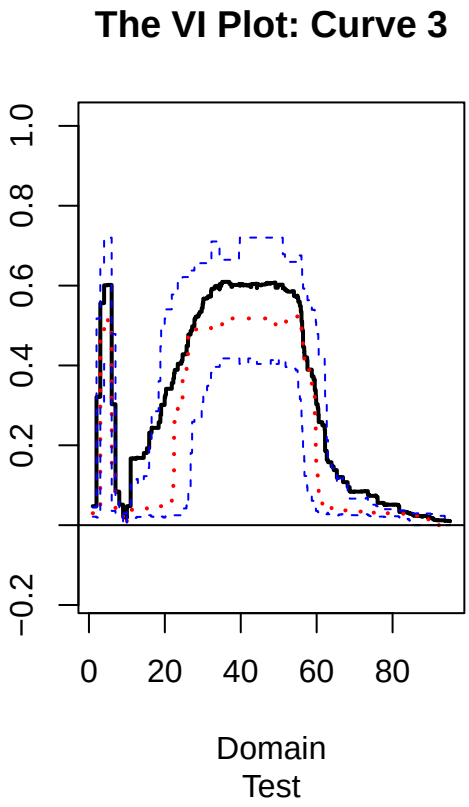


```
## X2(t)
par(mfrow=c(1,2))
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
```

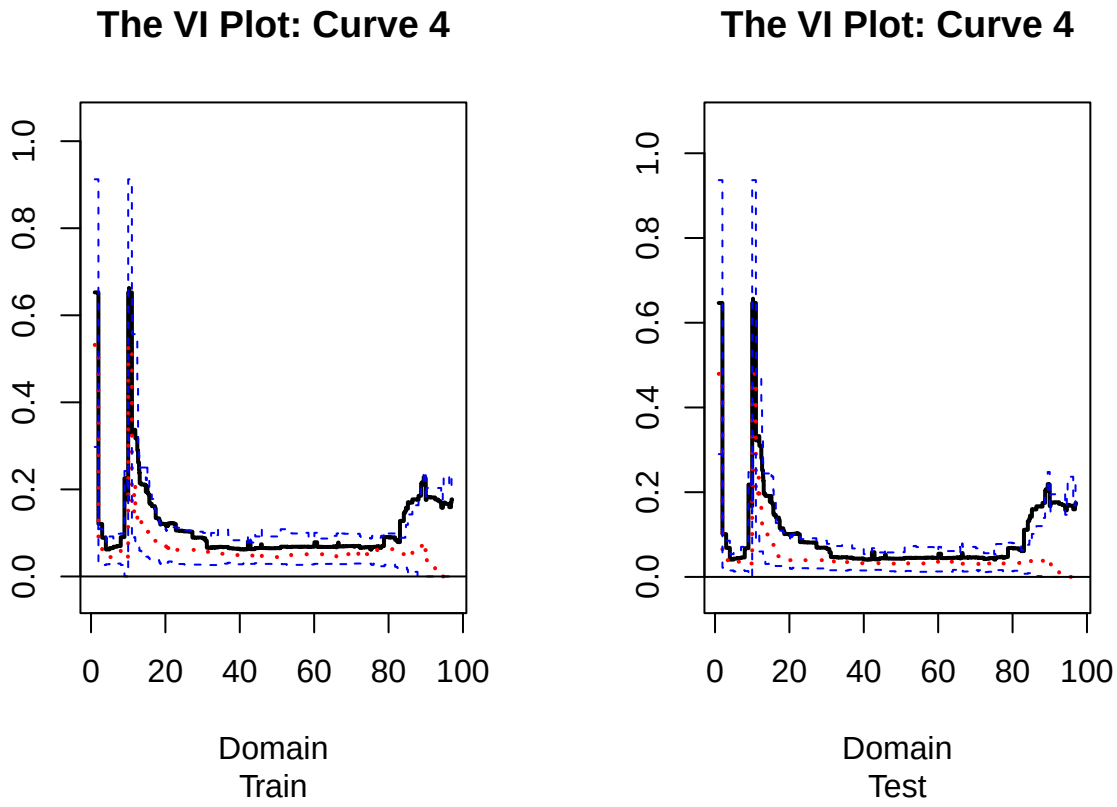



```
## X3(t)
par(mfrow=c(1,2))
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
```

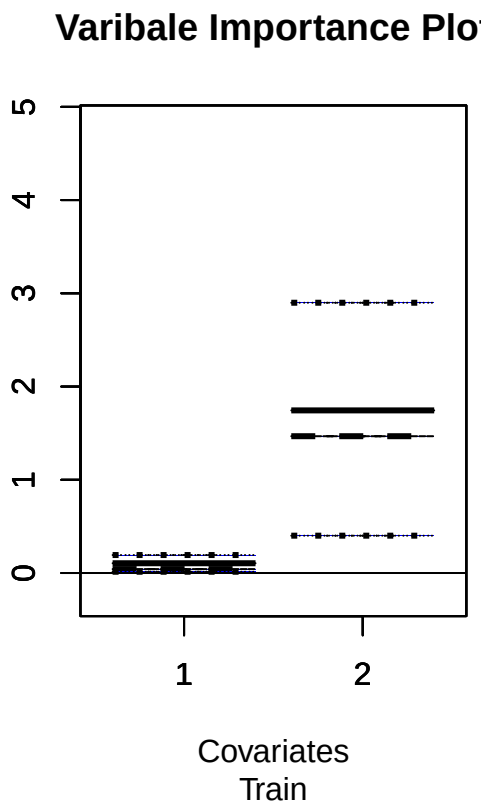


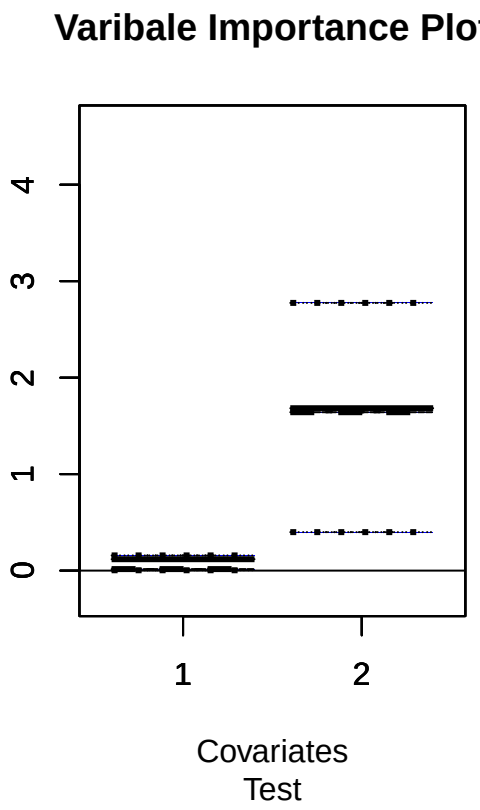



```
## X4(t)
par(mfrow=c(1,2))
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Functiona
```




```
## X1 and X2
par(mfrow=c(1,2))
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Non-Funct
RS_VARIMP_SRC_PLOT(Dataset = RF_RS_Result$IMP_VALUE,VIP_Algorithm  =  c("RS"), COVAR_TYPE = c("Non-Funct
```





As we expected the VIP plot for all co variates are made. The summary statistics of this model is:

```
RS_Summary(Dataset =RF_RS_Result$Result , Outcome_Type = c("Categorical"))
```

```
##     Train_ACC_Mean Train_ACC_sd Train_ACC_Q1 Train_ACC_Q3 Test_ACC_Mean
## 25%              1            0            1            1          0.98
##     Test_ACC_sd Test_ACC_Q1 Test_ACC_Q3 Train_SEN_Mean Train_SEN_sd
## 25%  0.03408249   0.9833333           1              1            0
##     Train_SEN_Q1 Train_SEN_Q3 Test_SEN_Mean Test_SEN_sd Test_SEN_Q1 Test_SEN_Q3
## 25%            1            1       0.98538  0.03465776           1           1
##     Train_SPE_Mean Train_SPE_sd Train_SPE_Q1 Train_SPE_Q3 Test_SPE_Mean
## 25%              1            0            1            1     0.9737139
##     Test_SPE_sd Test_SPE_Q1 Test_SPE_Q3
## 25%  0.05332915   0.9655172           1
```

Conclusion

The VIP for **random splitting random forest** is more informative than the VIP for **random forest** and **bagging**.