Package 'DynForest'

November 19, 2022

Title Random Forest with Multivariate Longitudinal Predictors

Version 1.1.0

Description Based on random forest principle, 'DynForest' is able to include multiple longitudinal predictors to provide individual predictions. Longitudinal predictors are modeled through the random forest. The methodology is fully described for a survival outcome in: Devaux, Helmer, Dufouil, Genuer & Proust-Lima (2022) <doi:10.48550/arXiv.2208.05801>.

Imports DescTools, cmprsk, doParallel, foreach, ggplot2, lcmm, methods, pbapply, pec, prodlim, stringr, survival, zoo

Depends R (>= 3.5.0)

License LGPL (>= 3)

LazyData true

Encoding UTF-8

RoxygenNote 7.2.1

URL https://github.com/anthonydevaux/DynForest

BugReports https://github.com/anthonydevaux/DynForest/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Anthony Devaux [aut, cre] (<https://orcid.org/0000-0002-8862-4218>)

Maintainer Anthony Devaux <anthony.devaux@u-bordeaux.fr>

Repository CRAN

Date/Publication 2022-11-19 00:10:02 UTC

R topics documented:

`compute_gVIMP` *Compute the grouped importance of variables (gVIMP) statistic*

Description

Compute the grouped importance of variables (gVIMP) statistic

Usage

```
compute_gVIMP(
  DynForest_obj,
  IBS.min = 0,
  IBS.max = NULL,
  group = NULL,
  ncores = NULL,
  seed = round(runif(1, 0, 10000))
)
```

Arguments

`DynForest_obj`	`DynForest` object containing the dynamic random forest used on train data
`IBS.min`	(Only with survival outcome) Minimal time to compute the Integrated Brier Score. Default value is set to 0.
`IBS.max`	(Only with survival outcome) Maximal time to compute the Integrated Brier Score. Default value is set to the maximal time-to-event found.
`group`	A list of groups with the name of the predictors assigned in each group
`ncores`	Number of cores used to grow trees in parallel. Default value is the number of cores of the computer-1.
`seed`	Seed to replicate results

Value

`compute_gVIMP()` function returns a list with the following elements:

Inputs	A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains the names of the predicto
gVIMP	A numeric vector containing the gVIMP for each group defined in group argument
tree_oob_err	A numeric vector containing the OOB error for each tree needed to compute the VIMP statistic
IBS.range	A vector containing the IBS min and max

Author(s)

Anthony Devaux (<anthony.devaux@u-bordeaux.fr>)

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))
```

```
# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Compute gVIMP statistic
res_dyn_gVIMP <- compute_gVIMP(DynForest_obj = res_dyn,
                               group = list(group1 = c("serBilir","SGOT"),
                                            group2 = c("albumin","alkaline")),
                               ncores = 2)
```

`compute_OOBerror` *Compute the Out-Of-Bag error (OOB error)*

Description

Compute the Out-Of-Bag error (OOB error)

Usage

```
compute_OOBerror(DynForest_obj, IBS.min = 0, IBS.max = NULL, ncores = NULL)
```

Arguments

`DynForest_obj`	`DynForest` object containing the dynamic random forest used on train data
`IBS.min`	(Only with survival outcome) Minimal time to compute the Integrated Brier Score. Default value is set to 0.
`IBS.max`	(Only with survival outcome) Maximal time to compute the Integrated Brier Score. Default value is set to the maximal time-to-event found.
`ncores`	Number of cores used to grow trees in parallel. Default value is the number of cores of the computer-1.

Value

`compute_OOBerror()` function return a list with the following elements:

`data`	A list containing the data used to grow the trees
`rf`	A table with each tree in column. Provide multiple characteristics about the tree building
`type`	Outcome type
`times`	A numeric vector containing the time-to-event for all subjects

cause	Indicating the cause of interest
causes	A numeric vector containing the causes indicator
Inputs	A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains the names of the p
Longitudinal.model	A list of longitudinal markers containing the formula used for modeling in the random forest
param	A list containing the hyperparameters
oob.err	A numeric vector containing the OOB error for each subject
oob.pred	Outcome prediction for all subjects
IBS.range	A vector containing the IBS min and max

Author(s)

Anthony Devaux (<anthony.devaux@u-bordeaux.fr>)

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
```

```
                    alkaline = list(fixed = alkaline ~ time,
                                    random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Compute OOB error
res_dyn_OOB <- compute_OOBerror(DynForest_obj = res_dyn, ncores = 2)
```

`compute_VIMP` *Compute the importance of variables (VIMP) statistic*

Description

Compute the importance of variables (VIMP) statistic

Usage

```
compute_VIMP(
  DynForest_obj,
  IBS.min = 0,
  IBS.max = NULL,
  ncores = NULL,
  seed = round(runif(1, 0, 10000))
)
```

Arguments

`DynForest_obj`	`DynForest` object containing the dynamic random forest used on train data
`IBS.min`	(Only with survival outcome) Minimal time to compute the Integrated Brier Score. Default value is set to 0.
`IBS.max`	(Only with survival outcome) Maximal time to compute the Integrated Brier Score. Default value is set to the maximal time-to-event found.
`ncores`	Number of cores used to grow trees in parallel. Default value is the number of cores of the computer-1.
`seed`	Seed to replicate results

Value

`compute_VIMP()` function returns a list with the following elements:

`Inputs`	A list of 3 elements: `Longitudinal`, `Numeric` and `Factor`. Each element contains the names of the predicto
`Importance`	A list of 3 elements: `Longitudinal`, `Numeric` and `Factor`. Each element contains a numeric vector of VIMI
`tree_oob_err`	A numeric vector containing the OOB error for each tree needed to compute the VIMP statistic
`IBS.range`	A vector containing the IBS min and max

Author(s)

Anthony Devaux (<anthony.devaux@u-bordeaux.fr>)

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])
```

```
# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Compute VIMP statistic
res_dyn_VIMP <- compute_VIMP(DynForest_obj = res_dyn, ncores = 2)
```

`data_simu1` *data_simu1 dataset*

Description

Simulated dataset 1 with continuous outcome

Format

Longitudinal dataset with 1200 rows and 13 columns for 200 subjects

id Subject identifier

time Time measurement

cont_covar1 Continuous time-fixed predictor 1

cont_covar2 Continuous time-fixed predictor 2

bin_covar1 Binary time-fixed predictor 1

bin_covar2 Binary time-fixed predictor 2

marker1 Continuous time-dependent predictor 1

marker2 Continuous time-dependent predictor 2

marker3 Continuous time-dependent predictor 3

marker4 Continuous time-dependent predictor 4

marker5 Continuous time-dependent predictor 5

marker6 Continuous time-dependent predictor 6

Y_res Continuous outcome

Examples

```
data(data_simu1)
```

data_simu2 *data_simu1 dataset*

Description

Simulated dataset 2 with continuous outcome

Format

Longitudinal dataset with 1200 rows and 13 columns for 200 subjects

id Subject identifier

time Time measurement

cont_covar1 Continuous time-fixed predictor 1

cont_covar2 Continuous time-fixed predictor 2

bin_covar1 Binary time-fixed predictor 1

bin_covar2 Binary time-fixed predictor 2

marker1 Continuous time-dependent predictor 1

marker2 Continuous time-dependent predictor 2

marker3 Continuous time-dependent predictor 3

marker4 Continuous time-dependent predictor 4

marker5 Continuous time-dependent predictor 5

marker6 Continuous time-dependent predictor 6

Y_res Continuous outcome

Examples

```
data(data_simu2)
```

DynForest *Random forest with multivariate longitudinal endogenous covariates*

Description

Build a random forest using multivariate longitudinal endogenous covariates

Usage

```
DynForest(
  timeData = NULL,
  fixedData = NULL,
  idVar = NULL,
  timeVar = NULL,
  timeVarModel = NULL,
  Y = NULL,
  ntree = 200,
  mtry = NULL,
  nodesize = 1,
  minsplit = 2,
  cause = 1,
  nsplit_option = "quantile",
  ncores = NULL,
  seed = round(runif(1, 0, 10000)),
  verbose = TRUE
)
```

Arguments

timeData	A data.frame containing the id and time measurements variables and the time-dependent predictors.
fixedData	A data.frame containing the id variable and the time-fixed predictors. Categorical variables should be characterized as factor.
idVar	A character indicating the name of variable to identify the subjects
timeVar	A character indicating the name of time variable
timeVarModel	A list for each time-dependent predictors containing a list of formula for fixed and random part from the mixed model
Y	A list of output which should contain: type defines the nature of the outcome, can be "surv", "numeric" or "factor"; .
ntree	Number of trees to grow. Default value set to 200.
mtry	Number of candidate variables randomly drawn at each node of the trees. This parameter should be tuned by minimizing the OOB error. Default is defined as the square root of the number of predictors.
nodesize	Minimal number of subjects required in both child nodes to split. Cannot be smaller than 1.
minsplit	(Only with survival outcome) Minimal number of events required to split the node. Cannot be smaller than 2.
cause	(Only with competing events) Number indicates the event of interest.
nsplit_option	A character indicates how the values are chosen to build the two groups for the splitting rule (only for continuous predictors). Values are chosen using deciles (nsplit_option="quantile") or randomly (nsplit_option="sample"). Default value is "quantile".

ncores	Number of cores used to grow trees in parallel. Default value is the number of cores of the computer-1.
seed	Seed to replicate results
verbose	A logical controlling the function progress. Default is TRUE

Details

The function currently supports survival (competing or single event), continuous or categorical outcome.

FUTUR IMPLEMENTATIONS:

- Continuous longitudinal outcome

Value

DynForest function return a list with the following elements:

data	A list containing the data used to grow the trees
rf	A table with each tree in column. Provide multiple characteristics about the tree building
type	Outcome type
times	A numeric vector containing the time-to-event for all subjects
cause	Indicating the cause of interest
causes	A numeric vector containing the causes indicator
Inputs	A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains the names of the p
Longitudinal.model	A list of longitudinal markers containing the formula used for modeling in the random forest
param	A list containing the hyperparameters
comput.time	Computation time

Author(s)

Anthony Devaux (<anthony.devaux@u-bordeaux.fr>)

References

Devaux A., Helmer C., Dufouil C., Genuer R., Proust-Lima C. (2022). Random survival forests for competing risks with multivariate longitudinal endogenous covariates. arXiv <doi: 10.48550/arXiv.2208.05801>

See Also

summary.DynForest compute_OOBerror compute_VIMP compute_gVIMP predict.DynForest plot.DynForest

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)
```

pbc2 *pbc2 dataset*

Description

pbc2 data from Mayo clinic

Format

Longitudinal dataset with 1945 rows and 19 columns for 312 patients

id Patient identifier

time Time measurement

ascites Presence of ascites (Yes/No)

hepatomegaly Presence of hepatomegaly (Yes/No)

spiders Blood vessel malformations in the skin (Yes/No)

edema Edema levels (No edema/edema no diuretics/edema despite diuretics)

serBilir Level of serum bilirubin

serChol Level of serum cholesterol

albumin Level of albumin

alkaline Level of alkaline phosphatase

SGOT Level of aspartate aminotransferase

platelets Platelet count

prothrombin Prothrombin time

histologic Histologic stage of disease

drug Drug treatment (D-penicillmain/Placebo)

age Age at enrollment

sex Sex of patient

years Time-to-event in years

event Event indicator: 0 (alive), 1 (transplanted) and 2 (dead)

Source

pbc2 joineRML

Examples

```
data(pbc2)
```

plot.DynForest *Plot results about the most predictive variables used in DynForest*

Description

This function displays a plot of the most predictive variables with the minimal depth (for class DynForestVarDepth), the variable importance (for class DynForestVIMP) or the grouped variable importance (for class DynForestgVIMP).

Usage

```
## S3 method for class 'DynForestVarDepth'
plot(x, plot_level = c("predictor", "feature"), ...)

## S3 method for class 'DynForestVIMP'
plot(x, PCT = FALSE, ordering = TRUE, ...)

## S3 method for class 'DynForestgVIMP'
plot(x, PCT = FALSE, ...)
```

Arguments

x	Object inheriting from classes DynForestVarDepth, DynForestVIMP or DynForestgVIMP, to respectively plot the minimal depth, the variable importance or grouped variable importance.
plot_level	For DynForestVarDepth object, compute the statistic at predictor (plot_level="predictor") or feature (plot_level="feature") level
...	Optional parameters to be passed to the low level function
PCT	For DynForestVIMP or DynForestgVIMP object, display VIMP statistic in percentage. Default value is FALSE.
ordering	For DynForestVIMP object, order predictors according to VIMP value. Default value is TRUE.

Value

plot() function displays:

With DynForestVarDepth	the minimal depth for each predictor/feature
With DynForestVIMP	the VIMP for each predictor
With DynForestVarDepth	the grouped-VIMP for each given group

See Also

DynForest var_depth compute_VIMP compute_gVIMP

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Run var_depth function
res_varDepth <- var_depth(res_dyn)
```

```
# Plot minimal depth
plot(x = res_varDepth, plot_level = "feature")

# Compute VIMP statistic
res_dyn_VIMP <- compute_VIMP(DynForest_obj = res_dyn, ncores = 2)

# Plot VIMP
plot(x = res_dyn_VIMP, PCT = TRUE)

# Compute gVIMP statistic
res_dyn_gVIMP <- compute_gVIMP(DynForest_obj = res_dyn,
                               group = list(group1 = c("serBilir","SGOT"),
                                            group2 = c("albumin","alkaline")),
                               ncores = 2)

# Plot gVIMP
plot(x = res_dyn_gVIMP, PCT = TRUE)
```

`plot_CIF` *Plot the individual Cumulative Incidence Function (CIF) for the interest cause*

Description

Plot the individual Cumulative Incidence Function (CIF) for the interest cause

Usage

```
plot_CIF(DynForestPred_obj, id = NULL)
```

Arguments

`DynForestPred_obj`	An DynForestPred object from `predict()` function
`id`	Identifiers for the selected subjects

Value

Display the CIF for selected subjects

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Sample 5 subjects to predict the event
set.seed(123)
id_pred <- sample(id, 5)

# Create predictors objects
pbc2_pred <- pbc2[which(pbc2$id%in%id_pred),]
```

```
timeData_pred <- pbc2_pred[,c("id", "time", "serBilir", "SGOT", "albumin", "alkaline")]
fixedData_pred <- unique(pbc2_pred[,c("id","age","drug","sex")])

# Predict the CIF function for the new subjects with landmark time at 4 years
pred_dyn <- predict(object = res_dyn,
                    timeData = timeData_pred, fixedData = fixedData_pred,
                    idVar = "id", timeVar = "time",
                    t0 = 4)

# Display CIF for subjects 26 and 110
plot_CIF(DynForestPred_obj = pred_dyn,
         id = c(26, 110))
```

`predict.DynForest` *Prediction using dynamic random forests*

Description

Prediction using dynamic random forests

Usage

```
## S3 method for class 'DynForest'
predict(
  object,
  timeData = NULL,
  fixedData = NULL,
  idVar,
  timeVar,
  t0 = NULL,
  ...
)
```

Arguments

`object`	`DynForest` object containing the dynamic random forest used on train data
`timeData`	A data.frame containing the id and time measurements variables and the time-dependent predictors.
`fixedData`	A data.frame containing the id variable and the time-fixed predictors. Non-continuous variables should be characterized as factor.
`idVar`	A character indicating the name of variable to identify the subjects
`timeVar`	A character indicating the name of time variable
`t0`	Landmark time
`...`	Optional parameters to be passed to the low level function

Value

Return the outcome of interest for the new subjects: matrix of probability of event of interest in survival mode, average value in regression mode and most likely value in classification mode

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Sample 5 subjects to predict the event
```

```
set.seed(123)
id_pred <- sample(id, 5)

# Create predictors objects
pbc2_pred <- pbc2[which(pbc2$id%in%id_pred),]
timeData_pred <- pbc2_pred[,c("id", "time", "serBilir", "SGOT", "albumin", "alkaline")]
fixedData_pred <- unique(pbc2_pred[,c("id","age","drug","sex")])

# Predict the CIF function for the new subjects with landmark time at 4 years
pred_dyn <- predict(object = res_dyn,
                    timeData = timeData_pred, fixedData = fixedData_pred,
                    idVar = "id", timeVar = "time",
                    t0 = 4)
```

`summary.DynForest` *Display the summary of DynForest*

Description

Display the summary of DynForest

Usage

```
## S3 method for class 'DynForest'
summary(object, ...)
```

Arguments

`object`	`DynForest` or `DynForest_OOB` object
`...`	Optional parameters to be passed to the low level function

Value

Return some information about the random forest

Author(s)

Anthony Devaux (<anthony.devaux@u-bordeaux.fr>)

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
```

```
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
                                    random = ~ time),
                     alkaline = list(fixed = alkaline ~ time,
                                     random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Compute OOB error
res_dyn_OOB <- compute_OOBerror(DynForest_obj = res_dyn, ncores = 2)

# DynForest summary
summary(object = res_dyn_OOB)
```

---

var_depth *Extract characteristics from the trees building process*

Description

Extract characteristics from the trees building process

Usage

```
var_depth(DynForest_obj)
```

Arguments

DynForest_obj	DynForest object

Value

var_depth function return a list with the following elements:

min_depth	A table providing for each feature in row: the average depth and the rank
var_node_depth	A table providing for each tree in column the minimal depth for each feature in row. NA indicates that the
var_count	A table providing for each tree in column the number of times where the feature is used (in row). 0 value i

See Also

DynForest

Examples

```
data(pbc2)

# Get Gaussian distribution for longitudinal predictors
pbc2$serBilir <- log(pbc2$serBilir)
pbc2$SGOT <- log(pbc2$SGOT)
pbc2$albumin <- log(pbc2$albumin)
pbc2$alkaline <- log(pbc2$alkaline)

# Sample 100 subjects
set.seed(1234)
id <- unique(pbc2$id)
id_sample <- sample(id, 100)
id_row <- which(pbc2$id%in%id_sample)

pbc2_train <- pbc2[id_row,]

timeData_train <- pbc2_train[,c("id","time",
                                "serBilir","SGOT",
                                "albumin","alkaline")]

# Create object with longitudinal association for each predictor
timeVarModel <- list(serBilir = list(fixed = serBilir ~ time,
                                     random = ~ time),
                     SGOT = list(fixed = SGOT ~ time + I(time^2),
                                 random = ~ time + I(time^2)),
                     albumin = list(fixed = albumin ~ time,
```

```
                                  random = ~ time),
                      alkaline = list(fixed = alkaline ~ time,
                                      random = ~ time))

# Build fixed data
fixedData_train <- unique(pbc2_train[,c("id","age","drug","sex")])

# Build outcome data
Y <- list(type = "surv",
          Y = unique(pbc2_train[,c("id","years","event")]))

# Run DynForest function
res_dyn <- DynForest(timeData = timeData_train, fixedData = fixedData_train,
                     timeVar = "time", idVar = "id",
                     timeVarModel = timeVarModel, Y = Y,
                     ntree = 50, nodesize = 5, minsplit = 5,
                     cause = 2, ncores = 2, seed = 1234)

# Run var_depth function
res_varDepth <- var_depth(res_dyn)
```

Index