Package ‘DynForest’

March 22, 2024

Title Random Forest with Multivariate Longitudinal Predictors
Version 1.1.3
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, Genuer & Proust-Lima (2023) <doi:10.1177/09622802231206477>.
Imports DescTools, cmprsk, doParallel, doRNG, foreach, ggplot2, lme4, methods, pbapply, pec, prodlim, stringr, survival, zoo
Depends R (>= 4.3.0)
License LGPL (>= 3)
LazyData true
Encoding UTF-8
RoxygenNote 7.2.3
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>), Robin Genuer [aut] (<https://orcid.org/0000-0002-0981-3943>), Cécile Proust-Lima [aut] (<https://orcid.org/0000-0002-9884-955X>), Louis Capitaine [aut] (<https://orcid.org/0000-0001-6800-2342>)
Maintainer Anthony Devaux <anthony.devauxbarault@gmail.com>
Repository CRAN
Date/Publication 2024-03-22 11:30:05 UTC
R topics documented:

compute_gVIMP ................................. 2
compute_OOBerror ................................ 4
compute_VIMP .................................... 6
data_simu1 ....................................... 9
data_simu2 ...................................... 10
DynForest ....................................... 10
dynGetTree ...................................... 14
gtGetTreeNodes ................................. 15
pbc2 ............................................. 17
plot.DynForest ................................. 18
predict.DynForest .............................. 21
print.DynForest ................................ 23
summary.DynForest ............................ 25
var_depth ....................................... 27

Index 29

compute_gVIMP ................................. 2
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 = 1234
)

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
compute_gVIMP

**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 predictors

**group**
A list of each group defined in group argument

**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

**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")]

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
**compute_OOBerror**

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

**Description**

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

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DynForest_obj</code></td>
<td>DynForest object containing the dynamic random forest used on train data</td>
</tr>
<tr>
<td><code>IBS.min</code></td>
<td>(Only with survival outcome) Minimal time to compute the Integrated Brier Score. Default value is set to 0.</td>
</tr>
<tr>
<td><code>IBS.max</code></td>
<td>(Only with survival outcome) Maximal time to compute the Integrated Brier Score. Default value is set to the maximal time-to-event found.</td>
</tr>
<tr>
<td><code>ncores</code></td>
<td>Number of cores used to grow trees in parallel. Default value is the number of cores of the computer-1.</td>
</tr>
</tbody>
</table>

**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
cause
causes
Inputs
Longitudinal.model
param
oob.err
oob.pred
IBS.range

Examples

```r
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**

```r
compute_VIMP(  
    DynForest_obj,  
    IBS.min = 0,  
    IBS.max = NULL,  
    ncores = NULL,  
    seed = 1234  
)
```

**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
compute_VIMP

Value

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

**Inputs**
A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains the names of the predictors.

**Importance**
A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains a numeric vector of VIMP statistic.

**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.

**Examples**

```r
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")]

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",
...)
```

data_simu

```r
timeVarModel = timeVarModel, Y = Y, ntree = 50, nodesize = 5, msplit = 5, cause = 2, ncores = 2, seed = 1234)

# Compute VIMP statistic
res_dyn_VIMP <- compute_VIMP(DynForest_obj = res_dyn, ncores = 2, seed = 1234)
```

---

data_simu

data_simu 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_simu2 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
ccont_covar1 Continuous time-fixed predictor 1
ccont_covar2 Continuous time-fixed predictor 2
cbin_covar1 Binary time-fixed predictor 1
cbin_covar2 Binary time-fixed predictor 2
cmarker1 Continuous time-dependent predictor 1
cmarker2 Continuous time-dependent predictor 2
cmarker3 Continuous time-dependent predictor 3
cmarker4 Continuous time-dependent predictor 4
cmarker5 Continuous time-dependent predictor 5
cmarker6 Continuous time-dependent predictor 6
cY_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
DynForest

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 = 1234,
    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".
DynForest

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
• Functional data analysis

Value

DynForest function returns 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 predictor
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.devauxbarault@gmail.com>)

References


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")]

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)
**getTree**

Extract some information about the split for a tree by user

**Description**

Extract some information about the split for a tree by user

**Usage**

```r
getTree(DynForest_obj, tree)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DynForest_obj</code></td>
<td>DynForest object containing the dynamic random forest used on train data</td>
</tr>
<tr>
<td><code>tree</code></td>
<td>Integer indicating the tree identifier</td>
</tr>
</tbody>
</table>

**Value**

A table sorted by the node/leaf identifier with each row representing a node/leaf. Each column provides information about the splits:

- **type**: The nature of the predictor (Longitudinal for longitudinal predictor, Numeric for continuous predictor or Factor for categorical predictor)
- **var_split**: The predictor used for the split defined by its order in `timeData` and `fixedData`
- **feature**: The feature used for the split defined by its position in random statistic
- **threshold**: The threshold used for the split (only with Longitudinal and Numeric). No information is returned for Factor
- **N**: The number of subjects in the node/leaf
- **Nevent**: The number of events of interest in the node/leaf (only with survival outcome)
- **depth**: the depth level of the node/leaf

**See Also**

`DynForest summary.DynForest`

**Examples**

```r
data(pbc2)
# Get Gaussian distribution for longitudinal predictors
```
getTreeNodes <- 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, nnodesize = 5, minsplit = 5,
cause = 2, ncores = 2, seed = 1234)

# Extract split information from tree 4
res_tree4 <- getTree(DynForest_obj = res_dyn, tree = 4)

---

**getTreeNodes**

Extract nodes identifiers for a given tree

**Description**

Extract nodes identifiers for a given tree
Usage

getTreeNodes(DynForest_obj, tree = NULL)

Arguments

DynForest_obj A DynForest object from DynForest() function
tree Integer indicating the tree identifier

Value

Extract nodes identifiers for a given tree

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)

# Extract nodes identifiers for a given tree
getTreeNodes(DynForest_obj = res_dyn, tree = 1)

---

**pbc2 dataset**

**Description**

pbc2 data from Mayo clinic

**Format**

Longitudinal dataset with 1945 rows and 19 columns for 312 patients

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Patient identifier</td>
</tr>
<tr>
<td>time</td>
<td>Time measurement</td>
</tr>
<tr>
<td>ascites</td>
<td>Presence of ascites (Yes/No)</td>
</tr>
<tr>
<td>hepatomegaly</td>
<td>Presence of hepatomegaly (Yes/No)</td>
</tr>
<tr>
<td>spiders</td>
<td>Blood vessel malformations in the skin (Yes/No)</td>
</tr>
<tr>
<td>edema</td>
<td>Edema levels (No edema/edema no diuretics/edema despite diuretics)</td>
</tr>
<tr>
<td>serBilir</td>
<td>Level of serum bilirubin</td>
</tr>
<tr>
<td>serChol</td>
<td>Level of serum cholesterol</td>
</tr>
<tr>
<td>albumin</td>
<td>Level of albumin</td>
</tr>
<tr>
<td>alkaline</td>
<td>Level of alkaline phosphatase</td>
</tr>
<tr>
<td>SGOT</td>
<td>Level of aspartate aminotransferase</td>
</tr>
<tr>
<td>platelets</td>
<td>Platelet count</td>
</tr>
<tr>
<td>prothrombin</td>
<td>Prothrombin time</td>
</tr>
<tr>
<td>histologic</td>
<td>Histologic stage of disease</td>
</tr>
<tr>
<td>drug</td>
<td>Drug treatment (D-penicillmain/Placebo)</td>
</tr>
<tr>
<td>age</td>
<td>Age at enrollment</td>
</tr>
<tr>
<td>sex</td>
<td>Sex of patient</td>
</tr>
<tr>
<td>years</td>
<td>Time-to-event in years</td>
</tr>
<tr>
<td>event</td>
<td>Event indicator: 0 (alive), 1 (transplanted) and 2 (dead)</td>
</tr>
</tbody>
</table>
Source

pbc2 joineRML

Examples

data(pbc2)

---

**plot.DynForest**

*Plot function in DynForest*

**Description**

This function displays a plot of CIF for a given node and tree (for class DynForest), 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**

```r
## S3 method for class 'DynForest'
plot(x, tree = NULL, nodes = NULL, id = NULL, max_tree = NULL, ...)

## 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, ...)

## S3 method for class 'DynForestPred'
plot(x, id = NULL, ...)
```

**Arguments**

- **x** Object inheriting from classes DynForest, DynForestVarDepth, DynForestVIMP or DynForestgVIMP, to respectively plot the CIF, the minimal depth, the variable importance or grouped variable importance.
- **tree** For DynForest class, integer indicating the tree identifier
- **nodes** For DynForest class, identifiers for the selected nodes
- **id** For DynForest and DynForestPred classes, identifier for a given subject
- **max_tree** For DynForest class, integer indicating the number of tree to display while using id argument
- **...** Optional parameters to be passed to the low level function
plot.DynForest

plot_level
For DynForestVarDepth class, compute the statistic at predictor (plot_level="predictor") or feature (plot_level="feature") level

PCT
For DynForestVIMP or DynForestgVIMP class, display VIMP statistic in percentage. Default value is FALSE.

ordering
For DynForestVIMP class, 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 DynForestgVIMP: 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))
plot.DynForest

```r
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)

# Plot estimated CIF at nodes 17 and 32
plot(x = res_dyn, tree = 1, nodes = c(17,32))

# 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)

# 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,

```
predict.DynForest

\[
\text{timeData} = \text{timeData}\_\text{pred}, \text{fixedData} = \text{fixedData}\_\text{pred}, \\
\text{idVar} = "id", \text{timeVar} = "time", \\
t_0 = 4)
\]

# Plot predicted CIF for subjects 26 and 110
plot(x = pred\_dyn, id = c(26, 110))

---

**predict.DynForest**

**Prediction using dynamic random forests**

**Description**

Prediction using dynamic random forests

**Usage**

```r
## 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)

print.DynForest

**Print function**

**Description**

This function displays a brief summary regarding the trees (for class DynForest), a data frame with variable importance (for class DynForestVIMP) or the grouped variable importance (for class DynForestgVIMP).

**Usage**

```r
## S3 method for class 'DynForest'
print(x, ...)

## S3 method for class 'DynForestVIMP'
print(x, ...)

## S3 method for class 'DynForestgVIMP'
print(x, ...)

## S3 method for class 'DynForestVarDepth'
print(x, ...)

## S3 method for class 'DynForestOOB'
print(x, ...)

## S3 method for class 'DynForestPred'
print(x, ...)
```

**Arguments**

- `x` Object inheriting from classes DynForest, DynForestVIMP or DynForestgVIMP.
- `...` Optional parameters to be passed to the low level function

**See Also**

- `DynForest var_depth compute_VIMP compute_gVIMP compute_OOBerror`
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)

# Print function
print(res_dyn)

# Compute VIMP statistic
res_dyn_VIMP <- compute_VIMP(DynForest_obj = res_dyn, ncores = 2, seed = 1234)
summary.DynForest

Display the summary of DynForest

Description
Display the summary of DynForest

Usage

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

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

Arguments

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

Value
Return some information about the random forest
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, nnode = 5, msplit = 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

```r
var_depth(DynForest_obj)
```

### Arguments

- `DynForest_obj`  
  DynForest object

### Value

- `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 feature was not used for the corresponding tree

- `var_count`  
  A table providing for each tree in column the number of times where the feature is used (in row). 0 value indicates that the feature was not used for the corresponding tree

### See Also

- `DynForest`

### Examples

```r
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, nnode = 5, minsplit = 5,
    cause = 2, ncores = 2, seed = 1234)

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

compute_gVIMP, 2, 13, 19, 23
compute_OOBerror, 4, 13, 23
compute_VIMP, 6, 13, 19, 23

data_simu1, 9
data_simu2, 10
DynForest, 10, 14, 19, 23, 27

getTree, 14
getTreeNodes, 15

pbc2, 17
plot.DynForest, 13, 18
plot.DynForestgVIMP (plot.DynForest), 18
plot.DynForestPred (plot.DynForest), 18
plot.DynForestVarDepth
  (plot.DynForest), 18
plot.DynForestVIMP (plot.DynForest), 18
predict.DynForest, 13, 21
print.DynForest, 23
print.DynForestgVIMP (print.DynForest), 23
print.DynForestOOB (print.DynForest), 23
print.DynForestPred (print.DynForest), 23
print.DynForestVarDepth
  (print.DynForest), 23
print.DynForestVIMP (print.DynForest), 23

summary.DynForest, 13, 14, 25
summary.DynForestOOB
  (summary.DynForest), 25

var_depth, 19, 23, 27