

# Package: DynForest (via r-universe)

October 24, 2024

**Title** Random Forest with Multivariate Longitudinal Predictors

**Version** 1.2.0

**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, Genuer & Proust-Lima (2023) <[doi:10.1177/09622802231206477](https://doi.org/10.1177/09622802231206477)>.

**Imports** DescTools, cli, cmprsk, doParallel, doRNG, foreach, ggplot2, lcm, methods, pbapply, pec, prodlim, stringr, survival, zoo

**Depends** R (>= 4.4.0)

**License** LGPL (>= 3)

**LazyData** true

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**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](mailto:anthony.devauxbarault@gmail.com)>

**Repository** CRAN

**Date/Publication** 2024-10-23 10:50:02 UTC

## Contents

compute_gvimp . . . . .	2
compute_ooberror . . . . .	4
compute_vardepth . . . . .	6
compute_vimp . . . . .	8
data_simu1 . . . . .	10
data_simu2 . . . . .	10
dynforest . . . . .	11
get_tree . . . . .	14
get_treenodes . . . . .	16
pbc2 . . . . .	18
plot.dynforest . . . . .	19
predict.dynforest . . . . .	22
print.dynforest . . . . .	24
summary.dynforest . . . . .	26
<b>Index</b>	<b>28</b>

---

compute_gvimp	<i>Compute the grouped importance of variables (gVIMP) statistic</i>
---------------	--

---

## 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_obj dynforest object
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 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

**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)

# Compute gVIMP statistic
res_dyn_gVIMP <- compute_gvimp(dynforest_obj = res_dyn,
                              group = list(group1 = c("serBilir", "SGOT"),
                                           group2 = c("albumin", "alkaline")),
                              ncores = 2, seed = 1234)

```

---

compute_ooberror	<i>Compute the Out-Of-Bag error (OOB error)</i>
------------------	---

---

## 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_obj dynforest object
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



```

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\_vardepth

*Extract characteristics from the trees building process*

---

### Description

Extract characteristics from the trees building process

### Usage

```
compute_vardepth(dynforest_obj)
```

### Arguments

dynforest\_obj    dynforest\_obj dynforest object

### Value

compute\_vardepth 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 compute_vardepth function

```

```
res_varDepth <- compute_vardepth(res_dyn)
```

---

compute_vimp	<i>Compute the importance of variables (VIMP) statistic</i>
--------------	---

---

### Description

Compute the importance of variables (VIMP) statistic

### Usage

```
compute_vimp(
  dynforest_obj,
  IBS.min = 0,
  IBS.max = NULL,
  ncores = NULL,
  seed = 1234
)
```

### Arguments

dynforest_obj	dynforest_obj dynforest object
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 predictors
Importance	A list of 3 elements: Longitudinal, Numeric and Factor. Each element contains a numeric vector of VIMP
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

### 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)

# Compute VIMP statistic
res_dyn_VIMP <- compute_vimp(dynforest_obj = res_dyn, ncores = 2, seed = 1234)

```

---

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\_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

**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 = 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"; .
nree	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
- 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 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.devauxbarault@gmail.com>)

**References**

- Devaux A., Helmer C., Genuer R., Proust-Lima C. (2023). Random survival forests with multivariate longitudinal endogenous covariates. SMMR doi:[10.1177/09622802231206477](https://doi.org/10.1177/09622802231206477)
- Devaux A., Proust-Lima C., Genuer R. (2023). Random Forests for time-fixed and time-dependent predictors: The DynForest R package. arXiv doi:[10.48550/arXiv.2302.02670](https://doi.org/10.48550/arXiv.2302.02670)

**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)

```

---

get\_tree

*Extract some information about the split for a tree by user*


---

## Description

Extract some information about the split for a tree by user

## Usage

```
get_tree(dynforest_obj, tree)
```



```

# 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 split information from tree 4
res_tree4 <- get_tree(dynforest_obj = res_dyn, tree = 4)

```

---

get\_treenodes

*Extract nodes identifiers for a given tree*


---

## Description

Extract nodes identifiers for a given tree

## Usage

```
get_treenodes(dynforest_obj, tree = NULL)
```

## Arguments

dynforest\_obj    dynforest\_obj dynforest object  
tree             Integer indicating the tree identifier

## Value

Extract nodes identifiers for a given tree

**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)

# Extract nodes identifiers for a given tree
get_treenodes(dynforest_obj = res_dyn, tree = 1)

```

---

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	<i>Plot function in dynforest</i>
----------------	-----------------------------------

---

### 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

```
## 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

<code>x</code>	Object inheriting from classes <code>dynforest</code> , <code>dynforestvardepth</code> , <code>dynforestvimp</code> or <code>dynforestgvimp</code> , to respectively plot the CIF, the minimal depth, the variable importance or grouped variable importance.
<code>tree</code>	For <code>dynforest</code> class, integer indicating the tree identifier
<code>nodes</code>	For <code>dynforest</code> class, identifiers for the selected nodes
<code>id</code>	For <code>dynforest</code> and <code>dynforestpred</code> classes, identifier for a given subject
<code>max_tree</code>	For <code>dynforest</code> class, integer indicating the number of tree to display while using <code>id</code> argument
<code>...</code>	Optional parameters to be passed to the low level function
<code>plot_level</code>	For <code>dynforestvardepth</code> class, compute the statistic at predictor ( <code>plot_level="predictor"</code> ) or feature ( <code>plot_level="feature"</code> ) level
<code>PCT</code>	For <code>dynforestvimp</code> or <code>dynforestgvimp</code> class, display VIMP statistic in percentage. Default value is <code>FALSE</code> .
<code>ordering</code>	For <code>dynforestvimp</code> class, order predictors according to VIMP value. Default value is <code>TRUE</code> .

**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\(\)](#) [compute\\_ooberror\(\)](#) [compute\\_vimp\(\)](#) [compute\\_gvimp\(\)](#) [compute\\_vardepth\(\)](#)

**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)

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

# Run var_depth function
res_varDepth <- compute_vardepth(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,
                   timeData = timeData_pred, fixedData = fixedData_pred,
                   idVar = "id", timeVar = "time",
                   t0 = 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

```
## 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

## 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)

# 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

```

## 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\(\)](#) [compute\\_ooberror\(\)](#) [compute\\_vimp\(\)](#) [compute\\_gvimp\(\)](#) [compute\\_vardepth\(\)](#)  
[predict.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)

# Print function
print(res_dyn)

# Compute VIMP statistic
res_dyn_VIMP <- compute_vimp(dynforest_obj = res_dyn, ncores = 2, seed = 1234)

# Print function

```

```
print(res_dyn_VIMP)

# Compute gVIMP statistic
res_dyn_gVIMP <- compute_gvimp(dynforest_obj = res_dyn,
                              group = list(group1 = c("serBilir", "SGOT"),
                                           group2 = c("albumin", "alkaline")),
                              ncores = 2, seed = 1234)

# Print function
print(res_dyn_gVIMP)

# Run var_depth function
res_varDepth <- compute_vardepth(res_dyn)

# Print function
print(res_varDepth)
```

---

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 dynforest00B object  
...                Optional parameters to be passed to the low level function

## Value

Return some information about the random forest

## 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)

# Compute OOB error
res_dyn_OOB <- compute_ooberror(dynforest_obj = res_dyn, ncores = 2)

# dynforest summary
summary(object = res_dyn_OOB)

```

# Index

compute\_gvimp, [2](#)  
compute\_gvimp(), [13](#), [20](#), [24](#)  
compute\_ooberror, [4](#)  
compute\_ooberror(), [13](#), [20](#), [24](#)  
compute\_vardepth, [6](#)  
compute\_vardepth(), [20](#), [24](#)  
compute\_vimp, [8](#)  
compute\_vimp(), [13](#), [20](#), [24](#)

data\_simu1, [10](#)  
data\_simu2, [10](#)  
dynforest, [11](#)  
dynforest(), [3](#), [5](#), [7](#), [8](#), [15](#), [17](#), [20](#), [22](#), [24](#), [26](#)

get\_tree, [14](#)  
get\_treenodes, [16](#)

pbc2, [18](#)  
plot.dynforest, [19](#)  
plot.dynforest(), [13](#)  
plot.dynforestgvimp(plot.dynforest), [19](#)  
plot.dynforestpred(plot.dynforest), [19](#)  
plot.dynforestvardepth  
    (plot.dynforest), [19](#)  
plot.dynforestvimp(plot.dynforest), [19](#)  
predict.dynforest, [22](#)  
predict.dynforest(), [13](#), [24](#)  
print.dynforest, [24](#)  
print.dynforestgvimp(print.dynforest),  
    [24](#)  
print.dynforesttoob(print.dynforest), [24](#)  
print.dynforestpred(print.dynforest),  
    [24](#)  
print.dynforestvardepth  
    (print.dynforest), [24](#)  
print.dynforestvimp(print.dynforest),  
    [24](#)

summary.dynforest, [26](#)  
summary.dynforest(), [13](#)  
summary.dynforesttoob  
    (summary.dynforest), [26](#)