Package 'efdm'

May 8, 2026

Title Simulate Forest Resources with the European Forestry Dynamics Model

Version 0.2.3

Description An implementation of European Forestry Dynamics Model (EFDM) and an estimation algorithm for the transition probabilities.
The EFDM is a large-scale forest model that simulates the development of the forest and estimates volume of wood harvested for any given forested area. This estimate can be broken down by, for example, species, site quality, management regime and ownership category.
See Packalen et al. (2015) <doi:10.2788/153990>.

URL https://github.com/mikkoku/efdm

BugReports https://github.com/mikkoku/efdm/issues

License GPL-2

Encoding UTF-8

LazyData true

Imports stats, utils, data.table

Suggests dplyr, ggplot2, gridExtra, knitr, rmarkdown, sf, testthat

RoxygenNote 7.3.3

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Mikko Kuronen [aut, cre] (ORCID: <https://orcid.org/0000-0002-8089-7895>), Minna Räty [aut] (ORCID: <https://orcid.org/0000-0001-9898-8712>)

Maintainer Mikko Kuronen <mikko.kuronen@luke.fi>

Repository CRAN

Date/Publication 2026-04-27 05:30:03 UTC

Contents

`define_activity` *Define an activity*

Description

Define an activity

Usage

```
define_activity(name, dynamicvariables, transprobs = NULL, probname = name)
```

Arguments

`name`	Name of activity used in reporting
`dynamicvariables`	Names of variables where changes happen
`transprobs`	Transition probabilities (`data.frame`)
`probname`	(optional) Name of activity in activity probabilities data

Details

The set of activities in EFDM defines all possible alternatives for a forest stratum to develop during a scenario run step. Therefore activities are not only limited to forest treatments and management actions such as thinnings and final fellings but should also include 'no management' i.e. growth, if applicable. An activity may also be something else affecting the development, for example, a forest hazard: snow, wind, drought, pest damage etc.

`name` is used as the name of the activity in the result of `runEFDM` and it is used to identify the activity probability. If multiple activities should be reported under a common name, `actprobname` can be used to specify the activity name in activity probability data.

`dynamicvariables` are those (state space) variables which change as a result of the activity Typically an activity affects on the age, volume or stem count of the forest, but an activity may also, for example, change land-use and then a variable related to land-use categories is essential.

The `transprobs` should be a `data.frame` with columns:

- pairs of dynamic variables. For example, if the dynamic variables are `vol` and `age`, then transprobs should have columns `age0`, `age1`, `vol0`, `vol1`.
- the probability of transition `prob`
- and possibly other variables affecting the transition probabilities. For example `soiltype`, `treespecies`, etc.

`estimatetransprobs` can be used to estimate transition probabilities from a similar set of pair data of observations.

Value

An activity definition

Examples

```
define_activity("nomanagement", c("vol", "age"))
```

`estimatetransprobs` *Estimate Transition Probabilities from Pairdata*

Description

Estimate Transition Probabilities from Pairdata

Usage

```
estimatetransprobs(
  dynamicvariables,
  pairdata,
  statespace,
  factors = character(),
  by = character(),
  prior = "nochange"
)
```

Arguments

`dynamicvariables`	The names of the dynamic variables, `character` vector
`pairdata`	`data.frame` Observed transitions
`statespace`	`data.frame` or a `list` of two `data.frames` if the state space changes as a result of the activity
`factors`	`character` Variables used by the activity
`by`	`character` Variables that split the state space
`prior`	function or character

Details

Transition probabilities 'move' the forest areas allocated in the cells of state matrix from the initial states in the beginning of a EFDM run step to the end position. This end position will be the initial state of the next EFDM step. Length of a step (=time) in the EFDM run is typically determined by the pairdata. It is the time difference of tree observations. Note that the pairdata can be also constructed from single observation, if the other (pair) observation is estimated or modelled.

Each activity needs to have a transition probability. If no pairdata is available, transition probability matrices can be based entirely on a prior defined with expert knowledge.

The estimation uses an iterative Bayesian algorithm that is explained in https://github.com/ec-jrc/efdm/blob/master/documents/EFDMinstructions/Seija_Mathematics_behind_EFDM.pdf. The transition probability estimate is the proportion of observed transitions divided by the number of all transitions from the same starting state. `prior` gives the number of prior transitions. For each `factor` variable the transitions are counted in classes of all factors before the current factor. The "most important" observations (having all classes right) is counted `length(factors)` times, the second most important observations are counted `length(factors)-1` times and so on.

If pairdata is NULL prior is used by itself.

Observations should have 'factor' and 'by' variables and statepairs with 0 and 1 suffixes to indicate before and after observations.

The estimation algorithm uses information across 'factor' variables, but not across 'by' variables.

`prior` can either `character` or `function`.

- "nochange" implies that there is one observation where state doesn't change
- "uninformative" when no observations are given all states are as likely
- `function(A, dynvar1, dynvar0)` where A is an array of zeros with dimnames(A) <- c(dynvar1, dynvar0). The function should fill A with the number of prior transitions and return it.

The statespace is used to fill in the transitions where there are no observations. In special cases the statespace may change as a result of the activity. For example changing the tree species might lead to change in the volume classification used.

This function assumes that the dynamic variables are coded as integers starting with 1. Other variables are not restricted.

Value

Transition probabilities as a `data.frame`

Examples

```
# Estimation can use observed transitions with different levels of factors.
statespace <- expand.grid(a=1:2, b=1:2, vol=1:5)
pairdata <- data.frame(a=c(1,1,2,2), b=c(1,2,1,2), vol0=c(1,1,1,1), vol1=c(2,3,4,5))
state0 <- statespace
actprob <- statespace
actprob$test <- 1
state0$area <- 0
state0$area[1] <- 1
```

```
# With by=c("a", "b") there are two observations: one from prior and the other
# from the exact combination of class levels.
probs <- estimatetransprobs("vol", pairdata, statespace, by=c("a", "b"), prior="nochange")
act1 <- define_activity("test", c("vol"), probs)
runEFDM(state0, actprob, list(act1), 1)

probs <- estimatetransprobs("vol", pairdata, statespace, factors="a", by="b", prior="nochange")
act2 <- define_activity("test", c("vol"), probs)
runEFDM(state0, actprob, list(act2), 1)

probs <- estimatetransprobs("vol", pairdata, statespace, factors="b", by="a", prior="nochange")
act3 <- define_activity("test", c("vol"), probs)
runEFDM(state0, actprob, list(act3), 1)

# The order of variables in factors argument specifies the order of importance.
# Observation that differ in the first variable are counted more times.
probs <- estimatetransprobs("vol", pairdata, statespace, factors=c("a", "b"), prior="nochange")
act4 <- define_activity("test", c("vol"), probs)
runEFDM(state0, actprob, list(act4), 1)

probs <- estimatetransprobs("vol", pairdata, statespace, factors=c("b", "a"), prior="nochange")
act5 <- define_activity("test", c("vol"), probs)
runEFDM(state0, actprob, list(act5), 1)
```

`example` *Example dataset.*

Description

A list containing the necessary datasets for EFDM forest scenario example.

Usage

```
example
```

Format

A list of data frames:

actprob Activity probabilities

noman_pairs Pair data for growth without management activities

thin_pairs Pair data for thinning

initial_state Initial forest state

drain_coef Coefficient to transform harvested areas into harvest accumulation by timber assortments

vol_coef Coefficients to transform volume classes into volumes m3/ha

income_coef Coefficients to transform harvest accumulation into income

MetsaKasvVyoh *Finnish bio-geographical regions*

Description

A low resolution copy of the finnish bio-geographical regions. The original shapefile was provided by Finnish Environment Institute. See https://ckan.ymparisto.fi/fi/dataset/metsakasvillisuusvyohykkeet.

Usage

```
MetsaKasvVyoh
```

Format

An sf-object:

region Factor with three levels: North, Middle, South

geometry Polygon, each region is composed of multiple polygons

prior_ff *Priors for* estimatetransprobs

Description

Priors for estimatetransprobs

Usage

```
prior_ff()

prior_grow(variable, howmuch = 1)
```

Arguments

variable	Name of the variable to grow
howmuch	Amount of growth

Details

prior_ff moves the forest area to the smallest classes of the given dynamic variables of the forest stratum.

prior_grow moves the forest to another class given by increasing variable by howmuch.

Value

Return value is used by `estimatetransprobs` to provide prior information on the transition probabilities.

Examples

```
statespace <- expand.grid(a=1:2, b=1:2, vol=1:15, age=1:15)
act <- define_activity("test", c("vol", "age"))
act1 <- estimatetransprobs(c("vol", "age"), NULL, statespace, by=c("a", "b"),
  prior=prior_ff())
act2 <- estimatetransprobs(c("vol", "age"), NULL, statespace, by=c("a", "b"),
  prior=prior_grow("age"))
```

---

runEFDM *Run European Forestry Dynamics Model*

Description

Run European Forestry Dynamics Model

Usage

```
runEFDM(state0, actprob, activities, n, check = TRUE, format = 1)
```

Arguments

`state0`	`data.frame` Initial state
`actprob`	`data.frame` Activity probabilities
`activities`	`list` A list of activities
`n`	`integer` Number of time steps required
`check`	Check input arguments for consistency.
`format`	??

Details

This is the actual scenario running function, which projects the initial forest state n time steps to the future.

Activities are defined using `define_activity`.

Value

`data.frame` State of each time step divided by activities

transprobs<- *Transition probabilities of an activity*

Description

Functions to get or set the transition probabilities of an activity

Usage

```
transprobs(act) <- value

transprobs(act)
```

Arguments

act	Activity definition
value	data.frame of transition probabilities

Details

The value should be a data.frame defining the transition probabilities between dynamic variables of the activity. See define_activity for details

Value

data.frame where prob is the transition probability from current state (with suffix 0) to next state (with suffix 1).

Examples

```
act1 <- define_activity("test", c("vol"))
transprobs(act1) <- data.frame(vol0 = 1:5, vol1=c(2:5, 5), prob=1)
transprobs(act1)

if(require("ggplot2")) {
  ggplot(transprobs(act1)) + geom_raster(aes(x=vol0, y=vol1, fill=prob))
}
```

Index