Package: mecit (via r-universe)

November 21, 2026

Type Package

Title Mixed-Effects Conditional Inference Trees as Identification Keys

Version 0.1.0

Description Fits decision trees to clustered data while a random effect absorbs the differences between clusters, so the tree splits only on variables that matter within clusters. Implements RE-EM trees with conditional inference trees (Fu and Simonoff 2015, <doi:10.1016/j.csda.2015.02.004>) and wraps the model-based version in 'glmertree' (Fokkema et al. 2018, <doi:10.3758/s13428-017-0971-x>). A fitted tree converts to a 'dimoose' dichotomous key, and expression data can be simulated on the human mitochondrial DNA phylogeny to test the method. A companion to the 'dimoose' package.

License GPL (>= 3)

URL https://github.com/leipzig/mecit

BugReports https://github.com/leipzig/mecit/issues

Encoding UTF-8

Depends R (>= 3.5.0)

Imports dimoose (>= 0.2.0), lme4, partykit, stats, utils, withr

Suggests glmertree, knitr, rmarkdown, testthat (>= 3.1.7)

Remotes leipzig/dimoose

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libicu-dev libxml2-dev libssl-dev

Repository https://leipzig.r-universe.dev

Date/Publication 2026-11-21 03:42:27 UTC

RemoteUrl https://github.com/leipzig/mecit

RemoteRef HEAD

RemoteSha d843b4c01c527d612d568d7bfc8e9cdf99201a5d

Contents

keyFromMecit 2
mecit 3
simulateExpression 5
splitVariables 6

Index **8**

`keyFromMecit` *Turn a mixed-effects tree into a key*

Description

Each split becomes a couplet and each terminal node a result, labelled with its stratum level and size. Splits on 0/1 variables read as "Carries 16311" / "Does not carry 16311". The leads carry machine-readable tests on the split variables, so `dimoose::classify()` can place new samples given their `dimoose::haplotypeMatrix()`, and `dimoose::exportWizard()` can draw the tree. Splits on factor columns become text-only couplets ("tissue is liver or lung"), which a person can follow but `dimoose::classify()` cannot; the function warns when the key has any.

Usage

```
keyFromMecit(fit, desc = NULL, digits = 2)
```

Arguments

`fit`	A `mecit()` fit.
`desc`	Title of the key.
`digits`	Decimal places for stratum levels.

Value

A dimoose::dimoose key.

Examples

```
sim <- simulateExpression(n = 600, genes = 6)
fit <- mecit(sim$expr[, 1], sim$X, sim$samples$macro)
key <- keyFromMecit(fit)
head(dimoose::classify(key, sim$X))
```

`mecit` *Mixed-effects conditional inference trees*

Description

Fits a tree that partitions observations on candidate split variables (for example which mutations each sample carries, from `dimoose::haplotypeMatrix()`) while a random effect absorbs differences between clusters (for example macrohaplogroups, from `dimoose::macroHaplogroup()`). Without the random effect, a tree also splits on variables that merely track the clusters, such as clade-defining markers; with it, those differences are removed before the tree looks for splits.

Usage

```
mecit(
  y,
  partition,
  cluster = NULL,
  method = c("ctree", "mob"),
  slope = NULL,
  alpha = 0.05,
  minbucket = 20L,
  maxdepth = Inf,
  ranefstart = TRUE,
  abstol = 0.001,
  maxit = 50L
)
```

Arguments

`y`	Numeric response, one value per observation.
`partition`	Matrix or data frame of candidate split variables, one row per observation, with column names (which may be non-syntactic, like `"16311"`).
`cluster`	Cluster of each observation, or `NULL` for no random effect.
`method`	`"ctree"` or `"mob"`.
`slope`	Optional names of `partition` columns to give random slopes within clusters, as in `(1 + slope | cluster)`.
`alpha`	Significance level for splitting (Bonferroni-adjusted over the candidate variables).
`minbucket`	Smallest allowed terminal node.
`maxdepth`	Maximum tree depth.
`ranefstart`	If `TRUE`, start from random effects estimated without a tree.
`abstol`, `maxit`	Convergence tolerance and the maximum number of iterations. For `"ctree"` the tolerance is on the largest change in any observation's random-effect contribution; for `"mob"` it is passed to `glmertree::lmertree()`, where it applies to the change in log-likelihood.

Details

`method = "ctree"` (the default) is the RE-EM algorithm of Sela & Simonoff (2012) with conditional inference trees, as proposed by Fu & Simonoff (2015). It alternates between (1) a `partykit::ctree()` on the response minus the current random-effect contribution and (2) a linear mixed model (`lme4::lmer()`) with one fixed effect per terminal node plus the random effects, until the random effects stop changing. `method = "mob"` uses model-based partitioning through `glmertree::lmertree()` (Fokkema et al. 2018). In both, `ranefstart = TRUE` estimates the random effects from an intercept-only mixed model before the first tree, which keeps that tree from splitting on cluster-level variables.

With `cluster = NULL` the result is a plain conditional inference tree, the baseline a mixed-effects tree is compared with.

Value

An object of class `mecit`: `tree` (a partykit tree), `lmer` (the final mixed model, or `NULL`), `strata` (data frame of `node`, `mean` and `n`; `mean` is the stratum level net of random effects), `node` (terminal node of each observation), `ranef` (random-effect contribution of each observation), `method`, `iterations` and `converged`.

References

Sela RJ, Simonoff JS (2012). RE-EM trees: a data mining approach for longitudinal and clustered data. *Machine Learning* 86, 169-207. doi:10.1007/s1099401152583

Fu W, Simonoff JS (2015). Unbiased regression trees for longitudinal and clustered data. *Computational Statistics & Data Analysis* 88, 53-74. doi:10.1016/j.csda.2015.02.004

Fokkema M, Smits N, Zeileis A, Hothorn T, Kelderman H (2018). Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. *Behavior Research Methods* 50, 2016-2034. doi:10.3758/s134280170971x

See Also

`splitVariables()`, `keyFromMecit()`, `simulateExpression()`

Examples

```
sim <- simulateExpression(n = 600, genes = 6)
fit <- mecit(sim$expr[, 1], sim$X, sim$samples$macro)
fit
splitVariables(fit)
key <- keyFromMecit(fit)
key$leads[, c("Statement", "Character", "Next", "Taxon")]
## Not run:
dimoose::exportWizard(key, "mecit.html")

## End(Not run)
```

`simulateExpression` *Simulate expression data on PhyloTree*

Description

Simulates per-sample expression for a set of genes, where recurrent mutations (variants that arise independently on many branches) shift expression by fixed amounts on top of a baseline that differs between macrohaplogroups. It is the ground truth for testing `mecit()`: a good method finds the mutation effects as splits and does not split on clade markers that only track the macrohaplogroup baselines.

Usage

```
simulateExpression(
  n = 1000,
  genes = 20,
  effectSites = c("16362", "150", "709", "16093"),
  propMutation = 0.4,
  propBaseline = 0.3,
  sitesPerGene = 1:2,
  effect = 1,
  macroSD = 1,
  slopeSD = 0,
  noise = 1,
  decoys = NULL,
  macroLevel = "letter",
  seed = 1
)
```

Arguments

`n`	Number of samples.
`genes`	Number of genes.
`effectSites`	Sites that can carry mutation effects.
`propMutation`, `propBaseline`	Proportions of mutation and baseline genes; the rest are null.
`sitesPerGene`	Possible numbers of effect sites per mutation gene.
`effect`	Size of each mutation effect (its sign is random).
`macroSD`	Standard deviation of the macrohaplogroup baselines.
`slopeSD`	If positive, each mutation effect also varies between macrohaplogroups with this standard deviation (a random slope).
`noise`	Residual standard deviation.
`decoys`	Extra candidate split sites with no effect. By default the most recurrent other sites plus the markers of the four most common macrohaplogroups in the sample.

macroLevel	Passed to dimoose::macroHaplogroup() as level.
seed	Random seed.

Details

Samples are drawn uniformly from the leaf haplogroups of dimoose::phylotree17, so their macrohaplogroups are as unbalanced as the tree itself (H is far more common than L5). For sample *i* and gene *g*:

```
expr[i, g] = b[g, macro(i)] + sum_s beta[g, s] * X[i, s] + slope + noise
```

Genes come in three kinds: "mutation" genes have a baseline and one or more mutation effects, "baseline" genes have only the macrohaplogroup baseline, and "null" genes are pure noise.

The default effect sites (16362, 150, 709, 16093) are highly recurrent but carried by a minority of lineages across many macrohaplogroups. The most-cited recurrent sites (152, 16311, 146, 16189) are poor effect sites in RSRS orientation: they are also on deep branches (16311 is an L3 marker), so most lineages carry them and carrying them largely tracks clade membership. They are included as decoys instead.

Value

A list with expr (samples x genes), samples (data frame of sample, haplogroup, macro), X (samples x sites, 0/1, from dimoose::haplotypeMatrix()), truth (beta, genes x sites; b, genes x macrohaplogroups; geneType; and slope when slopeSD > 0), effectSites, markerSites and decoySites.

See Also

mecit(), dimoose::haplotypeMatrix()

Examples

```
sim <- simulateExpression(n = 300, genes = 10)
table(sim$truth$geneType)
```

---

splitVariables *Variables a fitted tree splits on*

Description

Variables a fitted tree splits on

Usage

```
splitVariables(fit)
```

Arguments

fit	A mecit() fit.

Value

The names (as given in `partition`) of the variables used in any split, in the order they first appear; `character(0)` if the tree has no splits.

Examples

```
sim <- simulateExpression(n = 600, genes = 6)
splitVariables(mecit(sim$expr[, 1], sim$X, sim$samples$macro))
```

Index

dimoose::classify(), *2*
dimoose::dimoose, *2*
dimoose::exportWizard(), *2*
dimoose::haplotypeMatrix(), *2, 3*, *6*
dimoose::macroHaplogroup(), *3*, *6*
dimoose::phylotree17, *6*

glmertree::lmertree(), *3, 4*

keyFromMecit, 2
keyFromMecit(), *4*

lme4::lmer(), *4*

mecit, 3
mecit(), *2*, *5, 6*

partykit::ctree(), *4*

simulateExpression, 5
simulateExpression(), *4*
splitVariables, 6
splitVariables(), *4*