Package: causalASCEND (via r-universe)

November 10, 2026

Type Package

Title Scalable Two-Tier Causal Discovery for Multi-Omic Data

Version 0.99.5

Description Causal ASCEND (Ancestral Scalable Causal discovEry via iNherited Descent) infers ancestral (cause-before-effect) relations among downstream molecular variables, such as gene expression, when a layer of upstream variables, such as genotypes or methylation sites, is known to precede them. Each conditional independence test conditions only on a small, iteratively refined set of nearest ancestors instead of the whole upstream layer, so the number of tests grows polynomially with the number of variables. The package accepts matrices, SummarizedExperiment objects or a MultiAssayExperiment, and returns the ancestral relations with the rule and evidence behind each one.

Depends R (>= 4.5.0)

License GPL-3

Encoding UTF-8

Imports methods, stats, SummarizedExperiment, utils

Suggests BiocStyle, knitr, MultiAssayExperiment, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews GraphAndNetwork, NetworkInference, Network, SystemsBiology, GeneExpression, Transcriptomics, SNP, GeneticVariability, MultipleComparison, Software

URL https://github.com/SparkAILab/causalASCEND

BugReports https://github.com/SparkAILab/causalASCEND/issues

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Collate 'AllClasses.R' 'accessors.R' 'ascend.R' 'causalASCEND-package.R' 'engine.R' 'evaluate.R' 'simulate.R'

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-11-06 13:39:42 UTC

RemoteUrl https://github.com/bioc/causalASCEND

RemoteRef HEAD

RemoteSha 9104c90774ead6bbda65320363ca5ef0c136e922

Contents

ascend 2
ascendMultiAssay 4
AscendResult-accessors 5
AscendResult-class 6
evaluateAncestral 7
simulateTwoTier 8
trueAncestral 9

Index **10**

`ascend` *Infer ancestral relations with Causal ASCEND*

Description

`ascend()` infers which foreground variables (for example genes) are causal ancestors of which, using a background layer (for example genotypes) that is known to precede the foreground. It tests conditional independence with a Fisher-z partial-correlation test, conditioning each test only on the nearest ancestors of the two variables, which it learns and refines over successive sweeps.

Usage

```
ascend(
  background,
  foreground,
  alpha = 0.05,
  alphaMB = 0.05,
  fdr = TRUE,
  minVotes = 1,
  prescreen = 0.3,
  maxIter = 10,
  verbose = FALSE,
  backgroundAssay = 1,
  foregroundAssay = 1
)
```

Arguments

background	Background (upstream) data: a numeric matrix or data frame with features in rows and samples in columns, or a SummarizedExperiment object.
foreground	Foreground (downstream) data, in the same form as background and with the same samples.
alpha	Significance level of the orientation rules.
alphaMB	Significance level of the Markov-blanket (nearest ancestor) searches.
fdr	Logical: correct the pairwise tests of each sweep with the Benjamini-Hochberg procedure.
minVotes	Number of agreeing witnesses required before a direction is committed. Values above 1 trade coverage for precision.
prescreen	Benjamini-Hochberg threshold of the marginal prescreen that limits each foreground variable's background candidates. Set to 1 to keep every background variable.
maxIter	Maximum number of sweeps.
verbose	Logical: report progress with message().
backgroundAssay, foregroundAssay	Assay name or index used when background or foreground is a SummarizedExperiment.

Details

The method assumes that no foreground variable causes a background variable, and that the data are Markov and faithful to a directed acyclic graph. Not every background variable needs to be measured, but missing background variables act as hidden confounders and lower accuracy. Missing values are handled with pairwise-complete correlations.

Each pair of foreground variables is resolved by one of three rules, with background or already ordered foreground variables as witnesses: R1 (deactivation) gives an ancestor relation, R2 (activation) a non-descendant relation and R3 (conditional independence) an "unrelated" call. A pair holding only a non-descendant relation is tested again in later sweeps, and if R2 also holds in the other direction the pair is called unrelated (symmetry). Relations are then closed transitively, the nearest ancestors are re-learned and the sweep repeats until nothing changes.

Value

An AscendResult object. See ancestralMatrix(), ancestralEdges(), pairPValues() and ascendStats().

References

Asiedu S, Watson D. Causal ASCEND: scalable two-tier causal discovery on high-dimensional multi-omics data.

See Also

ascendMultiAssay() for a MultiAssayExperiment, simulateTwoTier() for test data.

Examples

```
set.seed(123)
sim <- simulateTwoTier(n = 1000, nBackground = 20, nForeground = 8,
    r2 = 0.7, sparsity = 0.7, pCross = 0.1, effect = 1.5)
res <- ascend(sim$background, sim$foreground)
res
head(ancestralEdges(res))
evaluateAncestral(res, trueAncestral(sim))
```

`ascendMultiAssay` *Run ASCEND on two experiments of a MultiAssayExperiment*

Description

Extracts the background and foreground experiments, keeps the samples (primary units) present in both and passes them to `ascend()`.

Usage

```
ascendMultiAssay(
  mae,
  background,
  foreground,
  backgroundAssay = 1,
  foregroundAssay = 1,
  ...
)
```

Arguments

`mae`	A `MultiAssayExperiment` object.
`background`, `foreground`	Names (or indices) of the background and foreground experiments in `mae`.
`backgroundAssay`, `foregroundAssay`	Assay used when an experiment is a `SummarizedExperiment`.
`...`	Further arguments passed to `ascend()`.

Value

An AscendResult object.

Examples

```
if (requireNamespace("MultiAssayExperiment", quietly = TRUE)) {
    set.seed(2)
    sim <- simulateTwoTier(n = 300, nBackground = 10, nForeground = 5)
    ids <- paste0("s", seq_len(300))
    colnames(sim$background) <- colnames(sim$foreground) <- ids
    mae <- MultiAssayExperiment::MultiAssayExperiment(
        experiments = list(snp = sim$background, rna = sim$foreground))
    ascendMultiAssay(mae, background = "snp", foreground = "rna")
}
```

AscendResult-accessors

Accessors for AscendResult objects

Description

Extract the inferred ancestral relations and the evidence behind them from the result of `ascend()`.

Usage

```
ancestralMatrix(x)

## S4 method for signature 'AscendResult'
ancestralMatrix(x)

pairPValues(x)

## S4 method for signature 'AscendResult'
pairPValues(x)

ascendStats(x)

## S4 method for signature 'AscendResult'
ascendStats(x)

ancestralEdges(x, include = c("ancestor", "non-descendant"))

## S4 method for signature 'AscendResult'
ancestralEdges(x, include = c("ancestor", "non-descendant"))

## S4 method for signature 'AscendResult'
show(object)
```

Arguments

x	An AscendResult object.

include — Which relations ancestralEdges() returns: any of "ancestor", "non-descendant", "unrelated" and "unresolved".

object — An AscendResult object.

Value

- ancestralMatrix(): the numeric ancestrality matrix (1 ancestor, 0.5 non-descendant, 0 unrelated, NA unresolved), row variable relative to column variable.
- pairPValues(): matrix of the p-value of the last pairwise independence test of each pair, a continuous score for ranking pairs.
- ascendStats(): list with the numbers of conditional independence tests by type (nPairTests, nWitnessTests, nBlanketTests), the time spent in them, conditioning-set sizes and the number of sweeps.
- ancestralEdges(): data frame with one row per relation, with columns from, to, relation, rule, sweep, votes and pValue. Ancestor and non-descendant relations are listed once, from the earlier to the later variable; unrelated and unresolved pairs are listed once with from before to in the matrix order.

Examples

```
set.seed(1)
sim <- simulateTwoTier(n = 500, nBackground = 10, nForeground = 6)
res <- ascend(sim$background, sim$foreground)
ancestralMatrix(res)
ancestralEdges(res)
ascendStats(res)$nPairTests
```

AscendResult-class *The result of an ASCEND run*

Description

AscendResult holds the ancestral relations that ascend() inferred among the foreground variables, together with the evidence behind each one and an account of the work done. Use the accessors listed below rather than the slots.

Value

An object of class AscendResult.

Slots

ancestral Numeric matrix, foreground by foreground. Entry [a, b] is 1 if a is an ancestor of b, 0.5 if a is not a descendant of b, 0 if neither is an ancestor of the other, and NA if the pair is unresolved. Rows and columns follow a topological order of the claims.

pValues Numeric matrix: p-value of the last pairwise (rule R3) independence test of each pair.

`rule` Character matrix: the step that committed each pair (`"R1"`, `"R2"`, `"R3"`, `"transitivity"`, `"symmetry"`, `"R3_final"`), or `NA` if unresolved.

`ruleSweep` Integer matrix: the sweep in which the pair was committed.

`ruleVotes` Integer matrix: for R1 and R2 commits, the number of agreeing witnesses.

`stats` List: numbers of conditional independence tests by type, time spent in them, conditioning-set sizes and number of sweeps.

`params` List: the arguments of the call and the data dimensions.

See Also

`ascend()`, `ancestralMatrix()`, `ancestralEdges()`, `pairPValues()`, `ascendStats()`

Examples

```
set.seed(1)
sim <- simulateTwoTier(n = 500, nBackground = 10, nForeground = 6)
res <- ascend(sim$background, sim$foreground)
res
```

---

evaluateAncestral *Score inferred ancestral relations against the truth*

Description

Each ordered pair `(a, b)` is a claim that `a` precedes `b` (entry 1 or 0.5) or not (entry 0). A reversed claim therefore counts as both a false positive and a false negative. Unresolved pairs are counted separately and are not scored.

Usage

```
evaluateAncestral(estimate, truth)
```

Arguments

`estimate`	An AscendResult object or an ancestrality matrix.
`truth`	The output of `trueAncestral()`.

Value

A list with `tp`, `fp`, `fn`, `tn`, `unresolved`, `precision`, `recall`, `f1`, `orientationAccuracy` (fraction of claims on truly related pairs that point the right way) and `coverage` (fraction of pairs with any verdict).

Examples

```
set.seed(123)
sim <- simulateTwoTier(n = 1000, nBackground = 20, nForeground = 8,
    r2 = 0.7, sparsity = 0.7, pCross = 0.1, effect = 1.5)
res <- ascend(sim$background, sim$foreground)
evaluateAncestral(res, trueAncestral(sim))
```

`simulateTwoTier` *Simulate two-tier data*

Description

Draws data from a linear-Gaussian structural equation model with a background layer that precedes a foreground layer: background variables form a random DAG, each foreground variable receives edges from random background variables, and foreground variables form a random DAG among themselves. No foreground variable affects a background variable.

Usage

```
simulateTwoTier(
  n,
  nBackground,
  nForeground,
  r2 = 0.5,
  sparsity = 0.9,
  pCross = 0.05,
  effect = 0.8,
  linear = 1,
  scaleBackground = TRUE
)
```

Arguments

`n`	Number of samples.
`nBackground, nForeground`	Numbers of background and foreground variables.
`r2`	Fraction of each foreground variable's variance explained by its parents.
`sparsity`	Sparsity of the within-layer DAGs (larger is sparser; an edge appears with probability `1 - sparsity`, kept in `[0.01, 0.4]`).
`pCross`	Probability of each background-to-foreground edge.
`effect`	Size of the foreground-to-foreground effects, relative to the parent's standard deviation.
`linear`	Fraction of parent effects that are linear; the rest are squared, square-root, softplus or ReLU transforms of the parent.
`scaleBackground`	Logical: standardise each background variable to unit variance as it is drawn. Without it, background variances grow along background paths and swamp the foreground signal when the background is large.

Details

Call `set.seed()` first for reproducible data.

Value

A list with `background` and `foreground` (numeric matrices, features in rows, samples in columns), `adjacency` (the true foreground graph, `adjacency[parent, child] = 1`) and `params`.

See Also

`trueAncestral()`, `ascend()`

Examples

```
set.seed(1)
sim <- simulateTwoTier(n = 200, nBackground = 5, nForeground = 4)
dim(sim$background)
sim$adjacency
```

---

trueAncestral *True ancestral relations of a simulated graph*

Description

True ancestral relations of a simulated graph

Usage

```
trueAncestral(x)
```

Arguments

`x`	The output of `simulateTwoTier()`, or a 0/1 adjacency matrix with `x[parent, child] = 1`.

Value

A 0/1 matrix with entry `[a, b] = 1` if `a` is an ancestor of `b` (diagonal `NA`), in a topological order.

Examples

```
set.seed(3)
sim <- simulateTwoTier(n = 100, nBackground = 5, nForeground = 5,
    sparsity = 0.5)
trueAncestral(sim)
```

Index

ancestralEdges
        *(*AscendResult-accessors*)*, 5
ancestralEdges(), *3*, *7*
ancestralEdges,AscendResult-method
        *(*AscendResult-accessors*)*, 5
ancestralMatrix
        *(*AscendResult-accessors*)*, 5
ancestralMatrix(), *3*, *7*
ancestralMatrix,AscendResult-method
        *(*AscendResult-accessors*)*, 5
ascend, 2
ascend(), *4–7*, *9*
ascendMultiAssay, 4
ascendMultiAssay(), *3*
AscendResult, *3–7*
AscendResult-accessors, 5
AscendResult-class, 6
ascendStats *(*AscendResult-accessors*)*, 5
ascendStats(), *3*, *7*
ascendStats,AscendResult-method
        *(*AscendResult-accessors*)*, 5

evaluateAncestral, 7

message(), *3*
MultiAssayExperiment, *4*

pairPValues *(*AscendResult-accessors*)*, 5
pairPValues(), *3*, *7*
pairPValues,AscendResult-method
        *(*AscendResult-accessors*)*, 5

set.seed(), *9*
show,AscendResult-method
        *(*AscendResult-accessors*)*, 5
simulateTwoTier, 8
simulateTwoTier(), *3*, *9*
SummarizedExperiment, *3*

trueAncestral, 9
trueAncestral(), *7*, *9*