Package: andorR (via r-universe)

May 21, 2026

Type Package

Title Optimisation of the Analysis of AND-OR Decision Trees

Version 0.3.1

Date 2025-10-01

Description A decision support tool to strategically prioritise evidence gathering in complex, hierarchical AND-OR decision trees. It is designed for situations with incomplete or uncertain information where the goal is to reach a confident conclusion as efficiently as possible (responding to the minimum number of questions, and only spending resources on generating improved evidence when it is of significant value to the final decision). The framework excels in complex analyses with multiple potential successful pathways to a conclusion ('OR' nodes). Key features include a dynamic influence index to guide users to the most impactful question, a system for propagating answers and semi-quantitative confidence scores (0-5) up the tree, and post-conclusion guidance to identify the best actions to increase the final confidence. These components are brought together in an interactive command-line workflow that guides the analysis from start to finish.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports data.tree, dplyr, yaml, jsonlite, cli, crayon, glue, rlang

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

URL https://epimundi.github.io/andorR/

Config/pak/sysreqs libicu-dev

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

Date/Publication 2025-10-22 12:54:11 UTC

RemoteUrl https://github.com/epimundi/andorr

RemoteRef HEAD

RemoteSha a175d07ca77ab1590bce9791aa7804d2457fb17c

Contents

`andorR_interactive` *Enter Interactive Analysis Mode*

Description

Iteratively prompts the user to answer questions to solve a decision tree. The function first presents the most impactful unanswered questions. Once the tree's root is solved, it presents questions that can increase the overall confidence of the conclusion.

Usage

```
andorR_interactive(tree, sort_by = "BOTH")
```

Arguments

`tree`	The `data.tree` object to be analysed.
`sort_by`	A character string indicating how the prioritised questions should be sorted. Options are: • "TRUE" : Sort by the product of the node true_index for all ancestors, which measures the influence of the question if it is answered TRUE • "FALSE" : Sort by the product of the node false_index for all ancestors, which measures the influence of the question if it is answered FALSE • "BOTH" : (Default) Sort by the sum of 'TRUE' and 'FALSE' values which measures the aggregate influence of the question before the answer is known

Details

This function provides a command-line interface (CLI) for working with the tree. It uses the `cli` package for formatted output and handles user input for quitting, saving, printing the tree state, or providing answers to specific questions (either by number or by name). All tree modifications are performed by calling the package's existing API functions:

- `set_answer()`
- `update_tree()`
- `get_highest_influence()`
- `get_confidence_boosters()`

The following key commands may be used during interactive mode:

- **h** : Show the help screen
- **p** : Print the current state of the tree
- **s** : Save the current state of the tree to an .rds file
- **q** : Quit (exit interactive mode)
- **n** : Specify a node to edit by name (case sensitive)
- **1, 2, ...** : Specify a node to edit from the numbered list

Value

The final, updated `data.tree` object.

Examples

```
# Load a tree
ethical_tree <- load_tree_df(ethical)

# Start interactive mode
if(interactive()){
  andorR_interactive(ethical_tree)
}
```

`calculate_tree` *Propagate Answers and Confidence Up the Tree*

Description

This function performs a full, bottom-up recalculation of the decision tree's state. It takes the user-provided answers and confidences at the leaf level and propagates the logical outcomes (`answer`) and aggregate confidence scores up to the parent nodes based on their `AND`/`OR` rules.

Usage

```
calculate_tree(tree)
```

Arguments

`tree`	The `data.tree` object to be calculated. The function modifies this object directly.

Details

This function is one of three called by `update_tree()`, which does a full recalculation of the decision tree result and optimisation indices.

The function first resets the `answer` and `confidence` of all non-leaf nodes to `NA` to ensure a clean calculation.

It then uses a **post-order traversal**, which is critical as it guarantees that a parent node is only processed after all of its children have been processed.

The logical rules are applied with short-circuiting:

OR Nodes: Become `TRUE` if any child is `TRUE`. Become `FALSE` only if all children are answered and none are `TRUE`.

AND Nodes: Become `FALSE` if any child is `FALSE`. Become `TRUE` only if all children are answered and none are `FALSE`.

The confidence calculation is based on the confidences of the children that determined the outcome (e.g., only the `TRUE` children for a resolved `OR` node).

Value

The modified `tree` object (returned invisibly).

Examples

```
# Load the data
ethical_tree <- load_tree_df(ethical)

# Answer some questions
set_answer(ethical_tree, "FIN2", TRUE, 4)
```

```
set_answer(ethical_tree, "ENV2", TRUE, 3)
set_answer(ethical_tree, "SOC2", TRUE, 4)
set_answer(ethical_tree, "GOV2", FALSE, 1)

# Calculate the tree
ethical_tree <- calculate_tree(ethical_tree)

# View the result
print_tree(ethical_tree)
```

`ethical` *Ethical investment decision tree for a fictional company - data frame format*

Description

This dataframe represents a decision tree in relational format, in which hierarchical relationships are indicated by a value indicating the parent of each node.

The decision tree is a hypothetical tool to standardise the process of making ethical investments. It was developed to illustrate the functionality of this package.

Usage

```
ethical
```

Format

A data frame with 5 variables and 34 rows

Each row represents a node or leaf in the tree and the columns represent attributes of those nodes. The columns are:

id A unique sequential numeric identifier for each node

name A short, unique alphanumeric code or name for nodes. For leaf nodes (questions), a short code is used. For higher nodes, a descriptive phrase is used.

question The full text of the question for leaves, or NA for higher nodes.

rule The logical rule for nodes, either **AND** or **OR**, and NA for leaves.

parent The numeric id of the parent node, and NA for the root node.

Details

A `data.tree` object is created from the dataframe using the `read_tree_df()` function.

Source

This is a simple hypothetical decision tree created solely to illustrate the use of the analytical approach.

Examples

```
# Read the data into a data.tree object for analysis
tree <- load_tree_df(ethical)

# View the tree
print_tree(tree)
```

`ethical_nl` *Ethical investment decision tree for a fictional company in hierarchical format*

Description

This dataframe represents a decision tree in hierarchical format, in which hierarchical relationships are indicated by a nested list

The decision tree is a hypothetical tool to standardise the process of making ethical investments. It was developed to illustrate the functionality of this package.

Usage

```
ethical_nl
```

Format

A nested node list of the ethical investment dataset, in which hierarchical relationships are indicated by a nested list

Each list element represents a node or leaf in the tree and has the following members:

name A short, unique alphanumeric code or name for nodes. For leaf nodes (questions), a short code is used. For higher nodes, a descriptive phrase is used.

rule The logical rule for nodes, either **AND** or **OR**, and NA for leaves.

question (Optional) For leaf nodes, the associated question.

nodes A list of nested nodes

Details

A `data.tree` object is created from the nested list using the `read_tree_node_list()` function.

Source

This is a simple hypothetical decision tree created solely to illustrate the use of the analytical approach.

Examples

```
# Read the data into a data.tree object for analysis
tree <- load_tree_node_list(ethical_nl)

# View the tree
print_tree(tree)
```

`get_confidence_boosters` *Find Actions to Most Effectively Boost Confidence*

Description

Performs a sensitivity analysis on the tree to find which actions (answering a new question or increasing confidence in an old one) will have the greatest positive impact on the root node's final confidence score.

Usage

```
get_confidence_boosters(tree, top_n = 5, verbose = TRUE)
```

Arguments

`tree`	The current data.tree object, typically after a conclusion is reached.
`top_n`	The number of suggestions to return.
`verbose`	Logical value (default TRUE) determining the level of output.

Value

A data.frame of the top_n suggested actions, ranked by potential gain.

Examples

```
# Load a tree
ethical_tree <- load_tree_df(ethical)

# Answer some questions
set_answer(ethical_tree, "FIN2", TRUE, 4)
set_answer(ethical_tree, "FIN4", TRUE, 3)
set_answer(ethical_tree, "FIN5", TRUE, 2)
set_answer(ethical_tree, "ENV5", TRUE, 3)
set_answer(ethical_tree, "SOC2", TRUE, 4)
set_answer(ethical_tree, "GOV1", TRUE, 1)
set_answer(ethical_tree, "GOV2", TRUE, 2)
set_answer(ethical_tree, "GOV3", TRUE, 1)
set_answer(ethical_tree, "GOV4", TRUE, 1)
set_answer(ethical_tree, "GOV5", TRUE, 1)
```

```
# Updated tree
ethical_tree <- update_tree(ethical_tree)

# View the tree
print_tree(ethical_tree)

# Get guidance on how to improve the confidence ---
guidance <- get_confidence_boosters(ethical_tree, verbose = FALSE)
print(guidance)
```

`get_highest_influence` *Identify the Most Influential Question(s)*

Description

Scans all leaf nodes in the tree to find the questions that currently have the highest `influence_index`.

Usage

```
get_highest_influence(tree, top_n = 5, sort_by = "BOTH")
```

Arguments

`tree`	The main `data.tree` object for the analysis.
`top_n`	The number of top-ranked questions to return.
`sort_by`	A character string indicating how the prioritised questions should be sorted. Options are: • "TRUE" : Sort by the product of the node true_index for all ancestors, which measures the influence of the question if it is answered TRUE • "FALSE" : Sort by the product of the node false_index for all ancestors, which measures the influence of the question if it is answered FALSE • "BOTH" : (Default) Sort by the sum of 'TRUE' and 'FALSE' values which measures the aggregate influence of the question before the answer is known

Value

A `data.frame` (tibble) containing the `name`, `question`, the components of the influence index (`influence_if_true`, `influence_if_false`), and the total `influence_index` for the highest-influence leaf/leaves, sorted by influence.

get_questions *Get a Data Frame Summary of All Leaf Questions*

Description

Traverses the tree to find all leaf nodes (questions) and compiles their key attributes into a single, tidy data frame. This is useful for getting a complete overview of the analysis state or for creating custom reports.

Usage

```
get_questions(tree)
```

Arguments

tree	The data.tree object to be summarised.

Value

A data.frame with one row for each leaf node and the following columns: name, question, answer, confidence (on a 0-5 scale), and influence_index.

Examples

```
# Load the example 'ethical' dataset
data(ethical)

# Build and initialise the tree object
ethical_tree <- load_tree_df(ethical)
ethical_tree <- update_tree(ethical_tree)

# Get the summary data frame of all questions
questions_df <- get_questions(ethical_tree)

# Display the first few rows
head(questions_df)
```

load_tree_csv *Load a decision tree from a CSV file (Relational Format)*

Description

Reads a CSV file from a given path and constructs a tree. This function expects the CSV to define the tree in a relational format with id and parent columns defining the hierarchy and name, question (for leaves) and rule (for nodes) columns for the decision tree attributes.

Usage

```
load_tree_csv(file_path)
```

Arguments

file_path	The path to the .csv file.

Value

A data.tree object, fully constructed and initialised with answer and confidence attributes set to NA.

See Also

load_tree_df() for the underlying constructor function.

Examples

```
# Load data from the `ethical.csv` file included with this package
path <- system.file("extdata", "ethical.csv", package = "andorR")
ethical_tree <- load_tree_csv(path)

# View the tree
print_tree(ethical_tree)
```

---

load_tree_csv_path *Load a decision tree from a CSV file (Path String Format)*

Description

Reads a CSV file from a given path and constructs a tree. This function expects the CSV to define the tree in a path string format, with each node's hierarchy defined in a column named path.

Usage

```
load_tree_csv_path(file_path, delim = "/")
```

Arguments

file_path	The path to the .csv file.
delim	The character used to separate nodes in the path string. Defaults to "/".

Value

A data.tree object.

See Also

load_tree_df_path() for the underlying constructor function.

Examples

```
#' # Load data from the `ethical_path.csv` file included with this package
path <- system.file("extdata", "ethical_path.csv", package = "andorR")
ethical_tree <- load_tree_csv_path(path)

# View the tree
print_tree(ethical_tree)
```

---

load_tree_df *Build a decision tree from a relational data frame*

Description

Constructs and initialises a tree from a data frame that is already in memory, where the hierarchy is defined in a relational (ID/parent) format.

Usage

```
load_tree_df(df)
```

Arguments

df	A data frame with columns: id, name, question, rule, parent.

Details

This is a core constructor function. It may be used to load one of the example datasets in relational format. It is called by the load_tree_csv() wrapper, which handles reading the data from a file.

Value

A data.tree object, fully constructed and initialised with answer and confidence attributes set to NA.

See Also

load_tree_csv() to read this format from a file.

Examples

```
# Load a tree from the 'ethical' dataframe included in this package
ethical_tree <- load_tree_df(ethical)

# View the tree structure
## Not run:
print_tree(ethical_tree)

## End(Not run)
```

`load_tree_df_path` *Build a decision tree from a path-string data frame*

Description

Constructs a tree from a data frame that is already in memory, where the hierarchy is defined using a path string for each node (e.g., "Root/Branch/Leaf").

Usage

```
load_tree_df_path(df, delim = "/")
```

Arguments

`df`	A data frame with a column named `path` containing the node paths, and other optional attribute columns like `question` and `rule`.
`delim`	The character used to separate nodes in the path string. Defaults to "/".

Details

This is a core constructor function, typically called by a wrapper like `load_tree_csv_path()`, which handles reading the data from a file. The node's name is inferred from the last element of its path.

Value

A `data.tree` object.

See Also

`load_tree_csv_path()` to read this format from a file.

Examples

```
# Create a sample data frame in path format
path_df <- data.frame(
  path = c("Root", "Root/Branch1", "Root/Branch1/LeafA", "Root/Branch2"),
  rule = c("AND", "OR", NA, NA),
  question = c(NA, "Is Branch1 relevant?", "Is LeafA true?", "Is Branch2 true?")
)

# Build the tree
my_tree <- load_tree_df_path(path_df)
print(my_tree)
```

`load_tree_json` *Load a decision tree from a JSON file (Hierarchical Format)*

Description

Reads a JSON file from a given path and constructs a tree. This function expects the JSON to define the tree in a hierarchical (nested) format. It uses `load_tree_node_list` to construct the tree object.

Usage

```
load_tree_json(file_path)
```

Arguments

`file_path`	The path to the .jsn or .json file.

Value

A `data.tree` object, fully constructed and initialised with `answer` and `confidence` attributes set to `NA`.

See Also

`load_tree_node_list()` for the underlying constructor function.

Examples

```
#' # Load data from the `ethical.json` file included with this package
path <- system.file("extdata", "ethical.json", package = "andorR")
ethical_tree <- load_tree_json(path)

# View the tree
print_tree(ethical_tree)
```

load_tree_node_list *Build a decision tree from a hierarchical list*

Description

Constructs a tree from a nested R list, where the hierarchy is defined by the list's structure. It also initialises the `answer` and `confidence` attributes required for the analysis.

Usage

```
load_tree_node_list(data_list)
```

Arguments

`data_list`	A nested R list representing the tree structure. Each list element should have a `name` and can have other attributes like `question`, `rule`, and a sub-list named `nodes` containing its children.

Details

This is a core constructor function, typically called by the `load_tree_yaml()` wrapper, which handles parsing the YAML file into a list.

Value

A `data.tree` object, fully constructed and initialised with `answer` and `confidence` attributes set to `NA`.

See Also

`load_tree_yaml()` to read this format from a file.

Examples

```
# 1. Define the tree structure as a nested list
my_data_list <- list(
  name = "Root",
  rule = "OR",
  nodes = list(
    list(name = "Leaf A", question = "Is A true?"),
    list(name = "Branch B",
         rule = "AND",
         nodes = list(
           list(name = "Leaf B1", question = "Is B1 true?"),
           list(name = "Leaf B2", question = "Is B2 true?")
         )
    )
  )
)
```

```
# 2. Build the tree from the list
my_tree <- load_tree_node_list(my_data_list)

# 3. Print the resulting tree
print_tree(my_tree)
```

`load_tree_yaml` *Load a decision tree from a YAML file (Hierarchical Format)*

Description

Reads a YAML file from a given path and constructs a tree. This function expects the YAML to define the tree in a hierarchical (nested) format. It uses `load_tree_node_list` to construct the tree object.

Usage

```
load_tree_yaml(file_path)
```

Arguments

`file_path`	The path to the .yml or .yaml file.

Value

A `data.tree` object, fully constructed and initialised with `answer` and `confidence` attributes set to `NA`.

See Also

`load_tree_node_list()` for the underlying constructor function.

Examples

```
#' # Load data from the `ethical.yml` file included with this package
path <- system.file("extdata", "ethical.yml", package = "andorR")
ethical_tree <- load_tree_yaml(path)

# View the tree
print_tree(ethical_tree)
```

`print_tree` *Print a Styled, Formatted Summary of the Decision Tree*

Description

Displays a clean, perfectly aligned, color-coded summary of the tree's current state, based on pre-calculated `answer` attributes.

Usage

```
print_tree(tree)
```

Arguments

`tree`	The `data.tree` object (the root `Node`) to be printed.

Details

An alternative approach to inspect internal attributes is to use the `data.tree` print() function with named attributes. See the example below.

Available attributes include:

- rule : AND or OR for a node
- name : The name of the node or leaf
- question : The question for leaves
- answer : The response provided for leaves or the calculated status of nodes
- confidence : The confidence score provided for leaves (0 - 5) or the probability that the answer is correct (50% to 100%) for nodes
- true_index : Influence the node has on the overall conclusion, if the response is TRUE
- false_index : Influence the node has on the overall conclusion, if the response is FALSE
- influence_if_true: Influence the leaf has on the overall conclusion, if the response is TRUE. This is the product of the ancestor values of true_index
- influence_if_false: Influence the leaf has on the overall conclusion, if the response is FALSE. This is the product of the ancestor values of false_index
- influence_index : The sum of influence_if_true and influence_if_false for each unanswered leaf

Value

The original `tree` object (returned invisibly).

Examples

```
# Load a tree
ethical_tree <- load_tree_df(ethical)

# View the tree - initially all 'plain' as no answers
print_tree(ethical_tree)

# Set an answer for leaf 'FIN2' and update the tree
ethical_tree <- set_answer(ethical_tree, "FIN2", TRUE, 3)
ethical_tree <- update_tree(ethical_tree) # Crucial: update the tree to propagate answers
print_tree(ethical_tree)

# Alternative approach to inspect internal attributes using `data.tree::print()
# First, recalculate the internal indices
update_tree(ethical_tree)

# Then print the tree, renaming column headings if required
print(ethical_tree, "rule", "true_index", "false_index", influence = "influence_index")
```

`set_answer` *Set an Answer and Confidence for a Leaf Node*

Description

This is the primary function for providing evidence to the tree. It finds a specific leaf node by its name and updates its `answer` and `confidence` attributes based on user input.

Usage

```
set_answer(tree, node_name, response, confidence_level, verbose = TRUE)
```

Arguments

`tree`	The `data.tree` object to be modified.
`node_name`	A character string specifying the `name` of the leaf node to update.
`response`	A logical value, `TRUE` or `FALSE`, representing the answer.
`confidence_level`	A numeric value from 0 to 5 representing the user's confidence in the answer. Confidence levels are semi-quantitative and map to the following probabilities: • 0 : 50% • 1 : 60% • 2 : 70% • 3 : 80% • 4 : 90% • 5 : 100%
`verbose`	An optional logical value controlling output. Default is TRUE.

Details

The function takes a 0-5 confidence level from the user and converts it to an internal score between 0.5 (uncertain) and 1.0 (certain) using the formula: `score = 0.5 + (confidence_level / 10)`.

It includes validation to ensure the target node exists, is a leaf, and that the provided response is a valid logical value. A confirmation message is printed to the console upon successful update.

Value

Returns the modified `tree` object invisibly, which allows for function chaining.

Examples

```
# Load a tree
ethical_tree <- load_tree_df(ethical)

# View the tree
print_tree(ethical_tree)

# Set an answer for leaf 'A1'
ethical_tree <- set_answer(ethical_tree, "FIN2", TRUE, 3)
print_tree(ethical_tree)
```

`update_tree` *Update a Tree Based on Answers Provided*

Description

Propagate the results up to the tree nodes based on the answers provided, and update the influence index to identify most important questions.

Usage

```
update_tree(tree)
```

Arguments

`tree`	The `data.tree` object to be modified.

Value

Returns the modified `tree` object invisibly, which allows for function chaining.

Examples

```
# Load a tree
ethical_tree <- load_tree_df(ethical)

# Internal indices before update
print(ethical_tree, "rule", "true_index", "false_index", influence = "influence_index")

ethical_tree <- update_tree(ethical_tree)

# Updated indices
print(ethical_tree, "rule", "true_index", "false_index", influence = "influence_index")

# Answer some questions
set_answer(ethical_tree, "FIN2", TRUE, 4)
set_answer(ethical_tree, "ENV2", TRUE, 3)
set_answer(ethical_tree, "SOC2", TRUE, 4)
set_answer(ethical_tree, "GOV2", FALSE, 1)

# Updated again
ethical_tree <- update_tree(ethical_tree)

# Updated indices
print(ethical_tree, "rule", "true_index", "false_index", influence = "influence_index")

# Updated results
print_tree(ethical_tree)
```

Index