# Package 'stCEG'

September 25, 2026

**Title** Fully Customizable Chain Event Graphs over Spatial Areas

**Version** 1.1.0

**Description** Enables the creation of Chain Event Graphs over spatial areas, with an optional 'Shiny' user interface. Allows users to fully customise both the structure and underlying model of the Chain Event Graph, offering a high degree of flexibility for tailored analyses. For more details on Chain Event Graphs, see Freeman, G., & Smith, J. Q. (2011) <doi:10.1016/j.jmva.2011.03.008>, Collazo R. A., Görgen C. and Smith J. Q. (2018, ISBN:9781498729604) and Barclay, L. M., Hutton, J. L., & Smith, J. Q. (2014) <doi:10.1214/13-BA843>.

**URL** https://github.com/holliecalley/stCEG

**BugReports** https://github.com/holliecalley/stCEG/issues

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** DT, dplyr, htmlwidgets, igraph, leaflet, magrittr, purrr, sf, shiny, shinyWidgets, shinyjs, viridis, visNetwork, zoo

**Suggests** randomcoloR, colourpicker

**Depends** R (>= 4.1.0)

**LazyData** true

**Config/roxygen2/version** 8.1.0

**NeedsCompilation** no

**Author** Hollie Calley [aut, cre] (ORCID: <https://orcid.org/0009-0002-1814-8428>), Daniel Williamson [ctb] (ORCID: <https://orcid.org/0000-0001-8917-3300>)

**Maintainer** Hollie Calley <hc629@exeter.ac.uk>

**Repository** CRAN

**Date/Publication** 2026-09-25 08:30:02 UTC

# Contents

ahc_colouring *Apply Agglomerative Hierarchical Clustering (AHC) Stage Colouring*

## Description

Applies an Agglomerative Hierarchical Clustering (AHC) stage-merging algorithm to an event tree or staged tree. The algorithm groups situations with equivalent floret structures and iteratively merges stages according to a Bayesian scoring criterion based on Dirichlet-Multinomial likelihoods.

## Usage

```
ahc_colouring(event_tree_obj, level_separation = 1000, node_distance = 300)
```

## Arguments

| | |
|---|---|
| `event_tree_obj` | An object of class `"event_tree"` or `"staged_tree"` containing node and edge information together with the underlying dataset stored in `$data`. |
| `level_separation` | Numeric value controlling the separation between levels when plotting the resulting staged tree. Included for compatibility with staged tree visualisation methods. Default is `1000`. |
| `node_distance` | Numeric value controlling the spacing between nodes when plotting the resulting staged tree. Included for compatibility with staged tree visualisation methods. Default is `300`. |

## Details

Nodes assigned to the same stage are coloured identically, and the resulting stage allocation is returned as a `"staged_tree"` object suitable for further analysis, visualisation, or CEG construction.

The algorithm only considers non-terminal situations when forming stages. Root and sink nodes are always assigned the default white colour and are not included in stage merging.

The procedure:

1. Extracts node, edge, and dataset information from the supplied event tree or staged tree object.
2. Identifies comparable situations based on their level and outgoing edge labels.
3. Constructs Dirichlet prior vectors and floret count vectors for each situation.
4. Iteratively merges candidate stages whenever doing so increases the Bayesian score.
5. Assigns a unique colour to each resulting stage using the **randomcoloR** package.
6. Returns a new `"staged_tree"` object containing the updated stage colouring.

A consistency check is performed before returning the staged tree to ensure that nodes sharing the same stage colour have identical outgoing edge structures. An error is raised if such conflicts are detected.

## Value

An object of class "staged_tree" containing:

- Updated node colours representing the inferred stage structure.
- Original edge information.
- The underlying dataset.
- Node-level method annotations identifying stages generated using the AHC procedure.

## See Also

create_staged_tree, plot.staged_tree, summary.staged_tree

## Examples

```
et <- create_event_tree(homicides, c(1:3))

st <- ahc_colouring(et)

print(st)
summary(st)
plot(st)
```

---

# bcu_shapefile *London Basic Command Units (BCUs)*

---

## Description

A simple features (sf) object containing the spatial boundaries of London's Basic Command Units (BCUs), used for policing. The dataset includes BCU names and their corresponding MULTIPOLYGON geometries. The data is projected in the British National Grid (EPSG:27700).

## Usage

```
bcu_shapefile
```

## Format

A sf object with 13 features and 1 field:

**BCU** Name of the Basic Command Unit (character)

**geometry** MULTIPOLYGON geometry column in BNG projection

## Source

Merged from Borough shapefile from London Datastore.

## Examples

```
library(sf)
plot(st_geometry(bcu_shapefile))
```

---

`borough_shapefile` *London Borough Boundaries*

---

## Description

A `sf` object containing the spatial boundaries of London boroughs, projected in the British National Grid (EPSG:27700). This dataset includes the borough names and associated MULTIPOLYGON geometries.

## Usage

```
borough_shapefile
```

## Format

A `sf` object with 33 features and 1 field:

**Borough** Name of the London Borough (character)

**geometry** MULTIPOLYGON geometry column in BNG projection

## Source

London Datastore

## Examples

```
library(sf)
plot(st_geometry(borough_shapefile))
```

`calculate_area_probabilities` *Calculate Area-Specific Conditional Probabilities*

## Description

Calculates a conditional probability for each geographical area represented within a Chain Event Graph.

## Usage

```
calculate_area_probabilities(
  path_df,
  unique_values,
  selected_indices,
  last_group,
  shapefile_vals
)
```

## Arguments

| | |
|---|---|
| `path_df` | Output from `calculate_path_products`. |
| `unique_values` | Character vector of conditioning values. |
| `selected_indices` | Numeric vector indicating grouping structure for the conditioning values. |
| `last_group` | Character string specifying the outcome of interest. |
| `shapefile_vals` | Character vector containing area identifiers. |

## Details

The function evaluates a specified conditional probability separately for each area and returns the resulting probabilities as a named list.

## Value

A named list of conditional probabilities indexed by area.

## See Also

`calculate_conditional_prob`

## Examples

```
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

path_df <- calculate_path_products(
  ceg$nodes,
  ceg$edges
)

probs <- calculate_area_probabilities(
  path_df,
  unique_values = c("Adult"),
  selected_indices = c(2),
  last_group = "Female",
  shapefile_vals = c("West", "South East")
)
```

---

# calculate_conditional_prob

*Calculate a Conditional Probability from CEG Paths*

---

## Description

Computes a conditional probability using path probabilities obtained from a Chain Event Graph.

## Usage

```
calculate_conditional_prob(
  path_df,
  unique_values,
  selected_indices,
  last_group
)
```

## Arguments

| | |
|---|---|
| path_df | Output from calculate_path_products. |
| unique_values | Character vector of values appearing in the CEG paths. |
| selected_indices | Numeric vector indicating which values belong to the same conditioning group. |
| last_group | Character string corresponding to the event whose conditional probability is required. |

## Details

Given a set of conditioning variables and a target outcome, the function evaluates:

$$P(\text{last\_group} \mid \text{conditions})$$

by summing the probabilities of all relevant paths.

## Value

A numeric value between 0 and 1.

## See Also

calculate_path_products

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

path_df <- calculate_path_products(
  ceg$nodes,
  ceg$edges
)

head(path_df)

calculate_conditional_prob(
  path_df,
```

```
  unique_values = c("Adult", "Male"),
  selected_indices = c(1, 2),
  last_group = "Shooting"
)
```

---

`calculate_path_products` *Calculate Path Probabilities in a Chain Event Graph*

---

## Description

Calculates the probability associated with every root-to-leaf path in a Chain Event Graph (CEG) by recursively multiplying the posterior transition probabilities along each path.

## Usage

```
calculate_path_products(nodes_df, edges_df, root_node = "w0")
```

## Arguments

| | |
|---|---|
| `nodes_df` | A node data frame from a `"ceg"` object. |
| `edges_df` | An edge data frame from a `"ceg"` object containing posterior transition probabilities in the `posterior_mean` column. |
| `root_node` | Character string identifying the root node. Default is `"w0"`. |

## Details

The resulting output can be used for conditional probability calculations and geographical probability mapping.

## Value

A data frame containing:

- `path`: sequence of edge labels defining the path.
- `product`: probability associated with the path.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)
```

```
st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

path_df <- calculate_path_products(
  ceg$nodes,
  ceg$edges
)

head(path_df)
```

---

`compare_ceg_models` *Compare Two Chain Event Graph Models*

---

## Description

Compares two fitted Chain Event Graph (CEG) models using their log marginal likelihoods and calculates the corresponding Bayes Factor.

## Usage

```
compare_ceg_models(ceg1, ceg2)
```

## Arguments

| | |
|---|---|
| `ceg1` | An object of class `"ceg"`. |
| `ceg2` | An object of class `"ceg"`. |

## Details

Evidence in favour of each model is quantified and classified according to Jeffreys' scale for Bayes Factors.

The Bayes Factor is computed as:

$$BF = \exp(\log p(D \mid M_1) - \log p(D \mid M_2))$$

where $p(D \mid M)$ denotes the marginal likelihood of a model.

The resulting Bayes Factor is interpreted using Jeffreys' evidence scale.

## Value

An object of class "compare_ceg_models" containing:

- Log marginal likelihoods for both models.
- Log Bayes Factor.
- Bayes Factor.
- Preferred model.
- Evidence measures for each model.
- Jeffreys evidence category.

## See Also

summary.compare_ceg_models

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg_model1 <- compute_ceg(st_priors)

priors2 <- specify_priors(
  st,
  prior_type = "Phantom"
)

st_priors2 <- compute_staged_tree_priors(
  st,
  priors2
)

ceg_model2 <- compute_ceg(st_priors2)

comparison <- compare_ceg_models(
  ceg_model1,
  ceg_model2
)

print(comparison)
```

```
summary(comparison)

## End(Not run)
```

---

`compute_ceg` *Compute a Chain Event Graph (CEG)*

---

## Description

Constructs a Chain Event Graph (CEG) from a `"staged_tree_priors"` object by contracting situations that share equivalent future developments. The resulting graph contains contracted vertices, aggregated edge information, posterior and prior summaries, and a stage-level summary table.

## Usage

```
compute_ceg(staged_tree_priors)
```

## Arguments

`staged_tree_priors`
: An object of class `"staged_tree_priors"` created by `compute_staged_tree_priors`.

## Details

During construction, nodes with equivalent stage colours and downstream structures are recursively merged to form the final CEG representation. Edge counts, prior values, posterior values, and corresponding probability summaries are aggregated across contracted situations.

The procedure:

1. Assigns contraction identifiers to nodes based on stage colours and downstream structure.
2. Contracts equivalent situations into CEG vertices.
3. Aggregates edge counts, prior information and posterior information.
4. Calculates prior and posterior transition probabilities for each stage.
5. Creates a stage-level summary table suitable for downstream model inspection and comparison.

## Value

An object of class `"ceg"` containing:

- `nodes`: contracted CEG vertices.
- `edges`: aggregated transition edges with counts, priors, posteriors and probability summaries.
- `table`: stage-level summary table containing data, prior, posterior and probability information.

## See Also

plot.ceg, summary.ceg

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

print(ceg)
summary(ceg)
```

---

# compute_deleted_nodes *Compute an Event Tree After Node Deletion*

---

## Description

Creates a modified event tree by removing one or more specified nodes and reconnecting the remaining tree structure.

## Usage

```
compute_deleted_nodes(tree_obj, nodes_to_delete)
```

## Arguments

| | |
|---|---|
| tree_obj | An object of class "event_tree" or "staged_tree". |
| nodes_to_delete | Character vector containing the node IDs to remove. |

## Details

For each deleted node, incoming edges are redirected to its descendants so that paths through the tree remain connected where possible. After deletion, edge counts are aggregated, orphaned nodes are removed, and node identifiers are renumbered sequentially.

The function accepts both "event_tree" and "staged_tree" objects and returns the modified structure as a new event tree.

The function:

1. Identifies the specified nodes for deletion.
2. Redirects incoming edges to the deleted node's descendants.
3. Removes deleted nodes and associated edges.
4. Aggregates duplicate edges sharing the same originating node and edge label.
5. Removes unused nodes.
6. Renumbers remaining nodes sequentially.

The returned object contains the modified graph structure while preserving the original dataset.

## Value

An object of class "event_tree" containing:

- nodes: updated node information.
- edges: updated edge information.
- data: the original dataset.

## See Also

create_event_tree, plot.event_tree

## Examples

```
et <- create_event_tree(homicides, c(1:3))

modified_et <- compute_deleted_nodes(et, nodes_to_delete = c("s11", "s12"))

print(modified_et)
summary(modified_et)
plot(modified_et)
```

---

# compute_reduced_ceg *Compute a Reduced Chain Event Graph*

---

## Description

Extracts one or more florets from a Chain Event Graph (CEG) beginning at specified edge labels and returns the downstream subgraph as a reduced CEG.

## Usage

```
compute_reduced_ceg(ceg, start_labels)
```

## Arguments

| | |
|---|---|
| ceg | An object of class "ceg" created by compute_ceg. |
| start_labels | A character vector containing one or more edge labels from which the reduced CEG should begin. |

## Details

For each supplied edge label, the function identifies all matching edges and recursively traverses descendant situations and transitions. The resulting graph contains only the nodes and edges reachable from the selected starting labels.

The function:

1. Locates edges whose label1 values match the supplied start_labels.
2. Extracts the descendant floret associated with each matching edge.
3. Recursively traverses all reachable downstream situations.
4. Combines and deduplicates nodes and edges from all extracted florets.

## Value

An object of class "reduced_ceg" containing:

- nodes: nodes contained in the reduced graph.
- edges: edges contained in the reduced graph.
- start_labels: labels used to generate the reduction.

## See Also

compute_ceg, plot.reduced_ceg

## Examples

```
## Not run:
data("Medical_Trial")

et <- create_event_tree(Medical_Trial)
st <- ahc_colouring(et)
st_priors <- compute_staged_tree_priors(st)
ceg <- compute_ceg(st_priors)

reduced_ceg <- compute_reduced_ceg(
  ceg,
  start_labels = "Recovered"
)

print(reduced_ceg)
summary(reduced_ceg)

## End(Not run)
```

`compute_staged_tree_priors` *Compute Prior-Adjusted Values for a Staged Tree*

## Description

Combines a staged tree and a stage-level prior specification to create a staged tree with prior, prior mean and prior variance information attached to both nodes and edges.

## Usage

```
compute_staged_tree_priors(staged_tree_obj, prior_table)
```

## Arguments

| | |
|---|---|
| `staged_tree_obj` | An object of class `"staged_tree"`. |
| `prior_table` | An object of class `"prior_table"` created by `specify_priors`. |

## Details

The function distributes stage-level Dirichlet priors across the nodes belonging to each stage and computes the corresponding prior means and variances. Prior information is then propagated to outgoing edges, creating edge-level prior labels suitable for visualisation and subsequent Chain Event Graph construction.

The function:

1. Matches stages using stage colour and tree level.
2. Allocates stage-level Dirichlet parameters across nodes within each stage.
3. Computes prior means and Dirichlet variances.
4. Assigns prior information to outgoing edges.
5. Creates labels suitable for visualisation using `plot.staged_tree_priors`.

The resulting object is typically used as input to `compute_ceg`.

## Value

An object of class `"staged_tree_priors"` containing:

- `nodes`: node data with prior information.
- `edges`: edge data with prior values and prior means.
- `prior_table`: the stage-level prior table.

## See Also

`specify_priors`, `compute_ceg`

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

print(st_priors)
summary(st_priors)
```

---

`create_event_tree` *Create an Event Tree from a Dataset*

---

## Description

Constructs an event tree from a categorical dataset. Each unique path through the selected variables is represented as a sequence of situations and edges, with edge counts corresponding to observed frequencies in the data.

## Usage

```
create_event_tree(dataset, columns = seq_along(dataset))
```

## Arguments

| | |
|---|---|
| `dataset` | A data frame containing categorical variables. |
| `columns` | A vector of column names or column indices specifying which variables should be used to construct the event tree. Defaults to all columns in the dataset. |

## Details

The resulting object can be visualised using `plot()` and inspected using `print()` and `summary()`.

The function:

1. Constructs a rooted event tree from the selected variables.
2. Creates one edge for each possible transition between levels.
3. Calculates frequencies for all observed and unobserved paths.
4. Stores node and edge information in a format suitable for visualisation with **visNetwork**.

## Value

An object of class "event_tree" containing:

- nodes: node information for the event tree.
- edges: edge information and frequencies.
- data: the filtered dataset used to construct the tree.
- variables: variables used in the tree.
- n_nodes: total number of nodes.
- n_edges: total number of edges.

## See Also

plot.event_tree, summary.event_tree

## Examples

```
et <- create_event_tree(homicides, c(1:3))

print(et)
summary(et)
```

---

# create_staged_tree *Create a Staged Tree*

---

## Description

Creates an object of class "staged_tree" from node and edge data produced by a staged-tree colouring procedure.

## Usage

```
create_staged_tree(nodes, edges, filtereddf, method)
```

## Arguments

| | |
|---|---|
| nodes | A data frame containing node information, including stage colours and identifiers. |
| edges | A data frame containing edge information inherited from the event tree. |
| filtereddf | The dataset used to construct the original event tree. |
| method | Character string describing the primary colouring method used to generate the staged tree. |

### Details

A staged tree is an event tree in which situations have been grouped into stages. Situations assigned to the same stage are represented by a common node colour and are assumed to share the same conditional transition structure.

This function is typically called internally by stage-colouring methods such as `ahc_colouring` or manual colouring procedures rather than directly by users.

### Value

An object of class `"staged_tree"` containing:

- `nodes`: staged node information.
- `edges`: edge information.
- `data`: underlying dataset.
- `metadata`: information describing the staged tree.

### See Also

`ahc_colouring`, `plot.staged_tree`

### Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)
```

---

## edit_priors *Edit Priors in a Prior Table*

---

### Description

Modifies one or more rows of a prior table created by `specify_priors`.

### Usage

```
edit_priors(prior_table, rows, new_priors)
```

### Arguments

| | |
|---|---|
| `prior_table` | An object of class `"prior_table"`. |
| `rows` | Integer vector specifying rows to modify. |
| `new_priors` | List containing replacement prior specifications. |

### Details

The function validates that the number of supplied Dirichlet parameters matches the number of outgoing edges associated with each stage.

## Value

An updated object of class "prior_table".

## See Also

specify_priors

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)
priors <- specify_priors(st, "Uniform")

priors <- edit_priors(
  priors,
  rows = 1,
  new_priors = list("2,3,4,5")
)

print(priors)
```

---

# generate_CEG_map *Generate a Chain Event Graph Probability Map*

---

## Description

Creates an interactive leaflet map displaying area-level probabilities derived from a Chain Event Graph (CEG).

## Usage

```
generate_CEG_map(
  shapefile,
  ceg_object,
  conditionals = unique(ceg_object$edges$label1),
  colour_by = NULL,
  color_palette = "viridis"
)
```

## Arguments

| | |
|---|---|
| shapefile | An sf object containing polygon geometries. |
| ceg_object | An object of class "ceg". |
| conditionals | Character vector specifying the conditioning variables. By default all unique edge labels are used. |

| | |
|---|---|
| colour_by | Character string specifying the outcome label whose conditional probability should be visualised. If NULL, the first label appearing at the deepest level of the CEG is used. |
| color_palette | Character string specifying the viridis palette option. |

## Details

Conditional probabilities are calculated for each geographical region and displayed using a colour scale. Areas that do not appear in the CEG are reported separately and excluded from colouring.

The function:

1. Calculates all root-to-leaf path probabilities.
2. Computes conditional probabilities for each geographical region.
3. Joins probabilities to the supplied shapefile.
4. Creates an interactive leaflet map with colour-coded polygons.

## Value

An object of class "ceg_map" containing:

- map: leaflet map object.
- conditional_probabilities: probability table.
- colour_by: outcome used for colouring.
- conditionals: conditioning variables.
- excluded_polygons: polygons not present in the CEG.

## See Also

plot.ceg_map, summary.ceg_map

## Examples

```
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

map_obj <- generate_CEG_map(
  shapefile = bcu_shapefile,
```

```
  ceg_object = ceg,
  colour_by = "Female"
)
```

---

`homicides` *Homicides Dataset*

---

## Description

A dataset containing homicides recorded by the Metropolitan Police from 2003-2023

## Usage

```
homicides
```

## Format

A data frame with 2670 rows and 9 variables:

**Age_Group** Age group of the victim. One of: `"Adult"`, `"Adolescent/Young Adult"`, `"Child"`, `"Elderly"`.

**Sex** Sex of the victim. `"Male"` or `"Female"`

**Method_of_Killing** Recorded method of killing, e.g., `"Knife or Sharp Implement"`, `"Blunt Implement"`, etc.

**Domestic_Abuse** Indicates whether the incident was flagged as domestic abuse.

**Solved_Status** Whether the case has been solved. One of: `"Solved"`, `"Unsolved"`

**Borough** London Borough where the homicide occurred

**Ethnicity** Recorded ethnicity of the victim. One of `"Asian"`, `"Black"`, `"Not Reported/Not Known"`, `"Other"`, `"White"`.

**Year** Year in which the incident was recorded.

**BCU** Basic Command Unit where the homicide occurred (Homicide units changed from Boroughs to BCUs in 2017).

## Source

Data taken and filtered from https://www.met.police.uk/police-forces/metropolitan-police/areas/stats-and-data/stats-and-data/met/homicide-dashboard/

plot.ceg *Plot a Chain Event Graph*

## Description

Produces an interactive visualisation of a Chain Event Graph (CEG) using **visNetwork**. Edge labels can display observed counts, prior values, posterior values, prior probabilities, or posterior probabilities.

## Usage

```
## S3 method for class 'ceg'
plot(
  x,
  label = "posterior_mean",
  level_separation = 1200,
  node_distance = 400,
  ...
)
```

## Arguments

| | |
|---|---|
| x | An object of class "ceg". |
| label | Character string specifying the edge label type to display. One of:<br>• "posterior_mean" (default)<br>• "posterior"<br>• "prior_mean"<br>• "prior"<br>• any other value displays the original edge labels |
| level_separation | Numeric value controlling spacing between graph levels. Default is 1200. |
| node_distance | Numeric value controlling spacing between nodes. Default is 400. |
| ... | Additional arguments passed to S3 methods. |

## Details

Selecting a node highlights incoming and outgoing edges, allowing local graph structure to be explored interactively.

## Value

A **visNetwork** htmlwidget representing the Chain Event Graph.

## See Also

compute_ceg, summary.ceg

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

plot(ceg)

plot(
  ceg,
  label = "posterior_mean"
)

plot(
  ceg,
  label = "prior_mean"
)

plot(
  ceg,
  label = "posterior"
)

## End(Not run)
```

---

`plot.ceg_map` *Plot a CEG Probability Map*

---

## Description

Displays the interactive leaflet map stored within a `"ceg_map"` object.

## Usage

```
## S3 method for class 'ceg_map'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class "ceg_map". |
| ... | Additional arguments passed to S3 methods. |

### Value

A leaflet map widget.

### Examples

```
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

map_obj <- generate_CEG_map(
  shapefile = bcu_shapefile,
  ceg_object = ceg,
  colour_by = "Female"
)

plot(map_obj)
```

---

## plot.event_tree *Plot an Event Tree*

---

### Description

Produces an interactive visualisation of an event tree using **visNetwork**.

### Usage

```
## S3 method for class 'event_tree'
plot(x, label_type = "both", level_separation = 1600, node_distance = 400, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class "event_tree". |
| label_type | Character string specifying the edge label format. Supported values are:<br>• "both" (default), displaying labels and frequencies.<br>• "names", displaying labels only. |
| level_separation | Numeric value controlling spacing between levels. Default is 1600. |
| node_distance | Numeric value controlling spacing between nodes. Default is 400. |
| ... | Additional arguments passed to S3 methods. |

### Details

Edge labels may display either transition names only or both transition names and observed frequencies.

### Value

A **visNetwork** htmlwidget.

### See Also

create_event_tree

### Examples

```
## Not run:
et <- create_event_tree(homicides, c(1:3))

plot(et)

## End(Not run)
```

---

## plot.reduced_ceg *Plot a Reduced Chain Event Graph*

---

### Description

Produces an interactive visualisation of a reduced Chain Event Graph using **visNetwork**.

## Usage

```
## S3 method for class 'reduced_ceg'
plot(
  x,
  label_type = "posterior_mean",
  level_separation = 1200,
  node_distance = 400,
  font_size = 80,
  ...
)
```

## Arguments

| | |
|---|---|
| x | An object of class "reduced_ceg". |
| label_type | Character string specifying which edge labels to display. Supported values are:<br>• "posterior_mean" (default)<br>• "posterior"<br>• "prior_mean"<br>• "prior"<br>• any other value displays the original edge labels |
| level_separation | Numeric value controlling spacing between graph levels. Default is 1200. |
| node_distance | Numeric value controlling spacing between nodes. Default is 400. |
| font_size | Numeric value controlling edge-label font size. Default is 80. |
| ... | Additional arguments passed to S3 methods. |

## Details

Edge labels may display observed counts, prior values, posterior values, prior probabilities, posterior probabilities or the original edge labels.

Selecting a node highlights incoming and outgoing transitions to aid interpretation of the local graph structure.

## Value

A **visNetwork** htmlwidget.

## See Also

compute_reduced_ceg

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)
```

```
priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

reduced_ceg <- compute_reduced_ceg(
  ceg,
  start_labels = "Adult"
)

plot(reduced_ceg)

plot(reduced_ceg, label_type = "posterior")

plot(reduced_ceg, label_type = "prior_mean")

## End(Not run)
```

---

`plot.staged_tree` *Plot a Staged Tree*

---

## Description

Produces an interactive visualisation of a staged tree using **visNetwork**.

## Usage

```
## S3 method for class 'staged_tree'
plot(x, label_type = "both", level_separation = 1600, node_distance = 400, ...)
```

## Arguments

| | |
|---|---|
| `x` | An object of class `"staged_tree"`. |
| `label_type` | Character string specifying the edge label format. Supported values include:<br>• `"both"` (default), displaying transition labels and counts.<br>• `"names"`, displaying transition labels only.<br>• `"priormeans"`, displaying prior means.<br>• `"priorfrac"`, displaying prior fractions. |
| `level_separation` | Numeric value controlling spacing between graph levels. Default is `1600`. |

| | |
|---|---|
| node_distance | Numeric value controlling spacing between nodes. Default is 400. |
| ... | Additional arguments passed to S3 methods. |

## Details

Nodes belonging to the same stage share a common colour. Edge labels may display transition names, observed frequencies or prior information, depending on the selected label type.

## Value

A **visNetwork** htmlwidget representing the staged tree.

## See Also

create_event_tree, ahc_colouring

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

plot(st)
```

---

# plot.staged_tree_priors

*Plot a Staged Tree with Priors*

---

## Description

Produces an interactive visualisation of a staged tree containing prior information using **visNetwork**.

## Usage

```
## S3 method for class 'staged_tree_priors'
plot(
  x,
  level_separation = 1600,
  node_distance = 400,
  label_type = "priors",
  ...
)
```

## Arguments

| | |
|---|---|
| x | An object of class "staged_tree_priors". |
| level_separation | Numeric value controlling spacing between graph levels. Default is 1600. |
| node_distance | Numeric value controlling spacing between nodes. Default is 400. |
| label_type | Character string specifying the edge labels to display. Supported values are:<br>• "priors" (default)<br>• "priormeans"<br>• "names" |
| ... | Additional arguments passed to S3 methods. |

## Details

Node tooltips display prior distributions, prior means and prior variances. Edge labels may display prior values or prior means.

## Value

A **visNetwork** htmlwidget.

## See Also

compute_staged_tree_priors, compute_ceg

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

plot(st_priors)
```

print.ceg *Print a Chain Event Graph*

## Description

Prints a concise overview of a Chain Event Graph including the number of vertices, edges and stages.

## Usage

```
## S3 method for class 'ceg'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "ceg". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied "ceg" object, invisibly.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

print(ceg)
```

print.ceg_map *Print a CEG Probability Map*

## Description

Prints a concise description of a "ceg_map" object including the outcome being mapped, probability range and the number of excluded geographical regions.

## Usage

```
## S3 method for class 'ceg_map'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "ceg_map". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied "ceg_map" object, invisibly.

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

map_obj <- generate_CEG_map(
  shapefile = bcu_shapefile,
  ceg_object = ceg,
  colour_by = "Female"
)

print(map_obj)

## End(Not run)
```

## print.event_tree *Print an Event Tree*

### Description

Prints a concise summary of an event tree including the variables used, number of nodes and number of edges.

### Usage

```
## S3 method for class 'event_tree'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class "event_tree". |
| ... | Additional arguments passed to S3 methods. |

### Value

The supplied "event_tree" object, invisibly.

### Examples

```
et <- create_event_tree(homicides, c(1:3))

print(et)
```

## print.prior_table *Print a Prior Table*

### Description

Prints the stage-level prior table contained within a "prior_table" object.

### Usage

```
## S3 method for class 'prior_table'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class "prior_table". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied object, invisibly.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)
priors <- specify_priors(st, "Uniform")

print(priors)
```

---

# print.reduced_ceg *Print a Reduced Chain Event Graph*

---

## Description

Prints a concise summary of a reduced Chain Event Graph including the number of nodes, edges and starting labels used to construct the graph.

## Usage

```
## S3 method for class 'reduced_ceg'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "reduced_ceg". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied "reduced_ceg" object, invisibly.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)
```

```
ceg <- compute_ceg(st_priors)

reduced_ceg <- compute_reduced_ceg(
  ceg,
  start_labels = "Adult"
)

print(reduced_ceg)
```

---

`print.staged_tree` *Print a Staged Tree*

---

## Description

Prints a concise summary of a staged tree, including the number of nodes, edges and colouring methods used.

## Usage

```
## S3 method for class 'staged_tree'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | An object of class `"staged_tree"`. |
| `...` | Additional arguments passed to S3 methods. |

## Value

The supplied `"staged_tree"` object, invisibly.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

print(st)
```

print.staged_tree_priors *Print a Staged Tree with Priors*

## Description

Prints a concise overview of a "staged_tree_priors" object, including the number of nodes, edges and prior type.

## Usage

```
## S3 method for class 'staged_tree_priors'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "staged_tree_priors". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied object, invisibly.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

print(st_priors)
```

print.summary_ceg_map *Print a Summary of a CEG Probability Map*

## Description

Prints a concise summary of a "summary_ceg_map" object.

## Usage

```
## S3 method for class 'summary_ceg_map'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "summary_ceg_map". |
| ... | Additional arguments passed to S3 methods. |

## Value

The supplied summary object, invisibly.

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

map_obj <- generate_CEG_map(
  shapefile = bcu_shapefile,
  ceg_object = ceg,
  colour_by = "Female"
)

print(summary(map_obj))

## End(Not run)
```

`run_stceg` *Launch the stCEG Shiny Application*

## Description

Launches the interactive stCEG Shiny application for constructing, visualising and analysing Event Trees, Staged Trees and Chain Event Graphs (CEGs), including spatial visualisation through interactive maps.

## Usage

```
run_stceg()
```

## Details

The application provides a graphical workflow for:

- Uploading and filtering datasets.
- Constructing event trees.
- Manual and automatic stage colouring.
- Prior specification and editing.
- Generating staged trees and Chain Event Graphs.
- Interactive spatial analysis using shapefiles.
- Computing conditional probability maps from CEGs.
- Exploring reduced CEGs and florets.

The application supports both manual stage specification and automated stage discovery using Agglomerative Hierarchical Clustering (AHC).

Users can:

1. Upload tabular datasets and shapefiles.
2. Select variables and spatial regions for analysis.
3. Create and modify event trees interactively.
4. Specify prior distributions for stages.
5. Generate Chain Event Graphs and posterior summaries.
6. Visualise probabilities on interactive leaflet maps.

The application is intended as a graphical interface to the functionality provided throughout the package.

## Value

Launches a Shiny application and returns a `shinyApp` object.

## See Also

create_event_tree, ahc_colouring, compute_ceg, compute_reduced_ceg, generate_CEG_map

## Examples

```
## Not run:
  run_stceg()

## End(Not run)
```

---

`specify_priors` *Specify Stage Priors for a Staged Tree*

---

## Description

Generates prior distributions for each stage of a staged tree.

## Usage

```
specify_priors(staged_tree_obj, prior_type = "Uniform", custom_priors = NULL)
```

## Arguments

| | |
|---|---|
| `staged_tree_obj` | An object of class `"staged_tree"`. |
| `prior_type` | Character string specifying the prior construction method. Supported values are:<br>• `"Uniform"`: assigns equal Dirichlet parameters to each outgoing edge.<br>• `"Phantom"`: computes a Phantom Individuals prior based on the staged tree structure. |
| `custom_priors` | Optional named list of custom prior vectors. Names must correspond to stage names (e.g. `"u1"`, `"u2"`). |

## Details

Prior distributions are specified at the stage level and may be generated automatically using a Uniform prior, a Phantom Individuals prior, or supplied manually through custom Dirichlet parameters.

The resulting object is used when constructing posterior distributions and Chain Event Graphs.

The function groups situations into stages and constructs a stage-level prior table containing:

- Stage identifiers.
- Stage colours.
- Tree levels.
- Number of outgoing edges.
- Dirichlet prior parameters.
- Corresponding prior means.

If `custom_priors` is supplied, the supplied values override the selected `prior_type`.

## Value

An object of class "prior_table" containing a stage-level summary of prior distributions.

## See Also

edit_priors, compute_ceg

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

print(priors)
summary(priors)
```

---

summary.ceg *Summarise a Chain Event Graph*

---

## Description

Generates a concise summary of a Chain Event Graph including the number of contracted vertices, edges, stages, levels and available prior and posterior information.

## Usage

```
## S3 method for class 'ceg'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class "ceg". |
| ... | Additional arguments passed to S3 methods. |

## Value

An object of class "summary_ceg" containing:

- Number of vertices.
- Number of edges.
- Graph levels.
- Stage colours.
- Stage identifiers.
- Prior type information.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

summary(ceg)
```

---

`summary.ceg_map` *Summarise a CEG Probability Map*

---

## Description

Produces a summary of a `"ceg_map"` object, including the number of mapped areas, probability range and any excluded polygons.

## Usage

```
## S3 method for class 'ceg_map'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `"ceg_map"`. |
| `...` | Additional arguments passed to S3 methods. |

## Value

An object of class `"summary_ceg_map"`.

## Examples

```
## Not run:
et <- create_event_tree(homicides, c(9,1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
```

```
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg <- compute_ceg(st_priors)

map_obj <- generate_CEG_map(
  shapefile = bcu_shapefile,
  ceg_object = ceg,
  colour_by = "Female"
)

summary(map_obj)

## End(Not run)
```

---

`summary.compare_ceg_models` *Summarise a CEG Model Comparison*

---

## Description

Produces a summary of a Bayes Factor comparison between two Chain Event Graph models.

## Usage

```
## S3 method for class 'compare_ceg_models'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `"compare_ceg_models"`. |
| `...` | Additional arguments passed to S3 methods. |

## Value

An object of class `"summary_compare_ceg_models"` containing Bayes Factor statistics, preferred model information and evidence measures.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

ceg_model1 <- compute_ceg(st_priors)

priors2 <- specify_priors(
  st,
  prior_type = "Phantom"
)

st_priors2 <- compute_staged_tree_priors(
  st,
  priors2
)

ceg_model2 <- compute_ceg(st_priors2)

comparison <- compare_ceg_models(
  ceg_model1,
  ceg_model2
)

summary(comparison)
```

---

`summary.event_tree` *Summarise an Event Tree*

---

## Description

Produces a summary of an event tree including variable names, graph size, and previews of node and edge information.

## Usage

```
## S3 method for class 'event_tree'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | An object of class "event_tree". |
| ... | Additional arguments passed to S3 methods. |

### Value

A list containing:

- Variables used to construct the tree.
- Number of nodes.
- Number of edges.
- Preview of node information.
- Preview of edge information.

### Examples

```
et <- create_event_tree(homicides, c(1:3))

summary(et)
```

---

## summary.prior_table *Summarise a Prior Table*

---

### Description

Produces a summary of a prior table including the number of stages, levels, colours and prior types represented.

### Usage

```
## S3 method for class 'prior_table'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | An object of class "prior_table". |
| ... | Additional arguments passed to S3 methods. |

### Value

An object of class "summary_prior_table".

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)
priors <- specify_priors(st, "Uniform")

summary(priors)
```

---

summary.reduced_ceg *Summarise a Reduced Chain Event Graph*

---

## Description

Produces a summary of a reduced Chain Event Graph including graph size, levels present and stage colours represented in the extracted subgraph.

## Usage

```
## S3 method for class 'reduced_ceg'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class "reduced_ceg". |
| ... | Additional arguments passed to S3 methods. |

## Value

An object of class "summary_reduced_ceg" containing:

- Number of nodes.
- Number of edges.
- Starting labels used for extraction.
- Levels represented in the reduced graph.
- Stage colours present in the reduced graph.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
```

```
  st,
  priors
)

ceg <- compute_ceg(st_priors)

reduced_ceg <- compute_reduced_ceg(
  ceg,
  start_labels = "Adult"
)
summary(reduced_ceg)
```

---

`summary.staged_tree` *Summarise a Staged Tree*

---

## Description

Produces a summary of a staged tree, including graph size, colouring methods and previews of node and edge information.

## Usage

```
## S3 method for class 'staged_tree'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `"staged_tree"`. |
| `...` | Additional arguments passed to S3 methods. |

## Value

A list containing:

- Number of nodes.
- Number of edges.
- Preview of node information.
- Preview of edge information.
- Colouring methods used.
- Primary colouring method.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

summary(st)
```

summary.staged_tree_priors *Summarise a Staged Tree with Priors*

## Description

Produces a summary of a "staged_tree_priors" object including graph size, prior types and stage information.

## Usage

```
## S3 method for class 'staged_tree_priors'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class "staged_tree_priors". |
| ... | Additional arguments passed to S3 methods. |

## Value

An object of class "summary_staged_tree_priors" containing:

- Number of nodes.
- Number of edges.
- Prior type information.
- Numbers of uniform and custom priors.
- Levels represented in the tree.
- Stage colours.

## Examples

```
et <- create_event_tree(homicides, c(1:3))
st <- ahc_colouring(et)

priors <- specify_priors(
  st,
  prior_type = "Uniform"
)

st_priors <- compute_staged_tree_priors(
  st,
  priors
)

summary(st_priors)
```

update_node_colours *Update Node Colours in an Event Tree*

## Description

Assigns stage colours to nodes in an event tree or staged tree and returns an updated staged tree object.

## Usage

```
update_node_colours(
  event_tree_obj,
  node_groups,
  colours,
  level_separation = 1000,
  node_distance = 300
)
```

## Arguments

| | |
|---|---|
| event_tree_obj | An object of class "event_tree" or "staged_tree". |
| node_groups | A list of character vectors. Each vector contains the node identifiers belonging to a single stage. |
| colours | Character vector of colour values corresponding to node_groups. The lengths of node_groups and colours must be equal. |
| level_separation | Numeric value controlling spacing between graph levels. Included for consistency with earlier versions. |
| node_distance | Numeric value controlling spacing between nodes. Included for consistency with earlier versions. |

## Details

Nodes assigned the same colour are interpreted as belonging to the same stage. The function validates that nodes sharing a colour also share the same outgoing edge structure, ensuring consistency with staged tree definitions.

The function:

1. Assigns colours to the specified node groups.
2. Prevents nodes appearing in multiple groups.
3. Computes outgoing edge labels for every node.
4. Verifies that nodes sharing a colour have identical outgoing edge labels.
5. Returns the result as an object of class "staged_tree".

If a colour is assigned to nodes with different outgoing edge structures, an error is produced.

## Value

An object of class "staged_tree" containing:

- nodes: node information including assigned colours.
- edges: edge information inherited from the original tree.
- data: the original dataset.
- metadata: staged tree summary information.

## See Also

create_event_tree, create_staged_tree, ahc_colouring

## Examples

```
et <- create_event_tree(homicides, c(1:3))

st <- update_node_colours(
  et,
  node_groups = list(c("s1", "s2")),
  colours = "#BBA0CA"
)

print(st)
summary(st)
plot(st)
```

# Index