Package 'OdysseusPathwayModule'

April 21, 2026

Title Cohort Pathway Analysis with Pre-Index Event Support

Version 0.0.1

Maintainer Alexander Alexeyuk <AlexanderAlexeyuk@gmail.com>

Description Provides cohort pathway analysis for Observational Medical Outcomes Partnership (OMOP) Common Data Model databases, including both standard (post-index) and pre-index pathway analyses. The pre-index analysis identifies sequences of events occurring in a lookback window before the target cohort index date. Built on the 'CohortPathways' analysis framework originally developed by Christopher Knoll and the Observational Health Data Sciences and Informatics community through 'WebAPI'. Methodological background and the originating implementation are described in <https://github.com/OHDSI/CohortPathways>.

License Apache License (>= 2)

Depends R (>= 4.1.0)

Encoding UTF-8

RoxygenNote 7.3.3

Imports DatabaseConnector, SqlRender

Suggests Eunomia, igraph, knitr, rmarkdown, testthat

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Alexander Alexeyuk [aut, cre]

Repository CRAN

Date/Publication 2026-04-21 19:22:15 UTC

Contents

buildEventSequenceGraph 2
executeCohortPathways 3
plot.event_sequence_graph 6

Index **8**

`buildEventSequenceGraph` *Build an event sequence graph from pathway analysis results.*

Description

Takes the output of `executeCohortPathways` and a cohort name mapping, decodes the bitmask-encoded combo IDs into human-readable event names, and builds a directed **igraph** graph representing event transitions across pathway steps.

Nodes in the graph represent unique *(event, step)* combinations. Edges represent directed transitions between consecutive steps, weighted by patient counts and annotated with transition probabilities.

The returned object includes:

- **graph**: A directed, weighted `igraph` object. Vertex attributes: `name`, `eventName`, `step`, `count`, `share`. Edge attributes: `weight` (= patient count), `probability`, `sourceStep`, `targetStep`.
- **sequences**: The fully decoded pathway sequences with a human-readable `pathway` string column.
- **summary**: High-level statistics (total pathways, patients, depth, unique events, vertex/edge counts).
- **eventNameLookup**: Named character vector mapping combo IDs to event names.

Usage

```
buildEventSequenceGraph(
  cpResults,
  generationSet,
  maxSteps = NULL,
  minCount = 1
)
```

Arguments

`cpResults`	A list returned by `executeCohortPathways`. Must contain at least `pathwaysAnalysisPathsData`, `pathwayAnalysisCodesLong`, and `isCombo`.
`generationSet`	A data frame with at least two columns: `cohortId` (integer) and `cohortName` (character). Used to map event cohort IDs to descriptive names.
`maxSteps`	(Optional) Maximum number of steps to include. If `NULL` (default), all non-empty steps are used.
`minCount`	(Default = 1) Minimum patient count for a pathway to be included in the output.

Value

A list of class `"event_sequence_graph"` with components `graph` (igraph object), `sequences`, `summary`, and `eventNameLookup`.

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
  cpResults <- list(
    pathwaysAnalysisPathsData = data.frame(
      pathwayAnalysisGenerationId = c(1L, 1L),
      targetCohortId = c(10L, 10L),
      step1 = c(2L, 4L),
      step2 = c(4L, NA_integer_),
      countValue = c(10L, 5L)
    ),
    pathwayAnalysisCodesLong = data.frame(
      pathwayAnalysisGenerationId = c(1L, 1L),
      code = c(2L, 4L),
      targetCohortId = c(10L, 10L),
      eventCohortId = c(1L, 2L),
      isCombo = c(0L, 0L),
      numberOfEvents = c(1L, 1L)
    ),
    isCombo = data.frame(
      targetCohortId = c(10L, 10L),
      comboId = c(2L, 4L),
      numberOfEvents = c(1L, 1L),
      isCombo = c(0L, 0L)
    )
  )

  generationSet <- data.frame(
    cohortId = c(1L, 2L, 10L),
    cohortName = c("Celecoxib", "Diclofenac", "NSAIDs")
  )

  esg <- buildEventSequenceGraph(cpResults, generationSet)
  esg$summary
  igraph::gorder(esg$graph)
}
```

`executeCohortPathways` *Execute cohort pathway analysis.*

Description

Runs the cohort pathways on all instantiated combinations of target and event cohorts. Assumes the cohorts have already been instantiated.

Supports two analysis types:

- `"post-index"` (default): events occurring **after** the target cohort index date (within the target cohort period).

- `"pre-index"`: events occurring **before** the target cohort index date, in a configurable lookback window. For example, what events occurred in the 365 days before cohort entry.

Target and event cohorts can reside in different schemas/tables:

- `cohortDatabaseSchema` / `cohortTableName` for target cohorts.
- `outcomeDatabaseSchema` / `outcomeTableName` for event (outcome) cohorts. If not specified, defaults to the same as the target.

Usage

```
executeCohortPathways(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTableName = "cohort",
  outcomeDatabaseSchema = cohortDatabaseSchema,
  outcomeTableName = cohortTableName,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  targetCohortIds,
  eventCohortIds,
  analysisType = "post-index",
  lookbackStartDay = -365,
  lookbackEndDay = -1,
  minCellCount = 5,
  allowRepeats = FALSE,
  maxDepth = 5,
  collapseWindow = 30
)
```

Arguments

`connectionDetails`
: An object of type connectionDetails as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
: An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`cohortDatabaseSchema`
: Schema name where your **target** cohort table resides. For SQL Server this should include both database and schema, e.g. 'scratch.dbo'.

`cohortTableName`
: The name of the **target** cohort table (default: "cohort").

`outcomeDatabaseSchema`
: Schema name where your **event/outcome** cohort table resides. Defaults to `cohortDatabaseSchema` if not specified.

`outcomeTableName`
: The name of the **event/outcome** cohort table. Defaults to `cohortTableName` if not specified.

`tempEmulationSchema`
: Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

`targetCohortIds`
: A vector of one or more Cohort Ids corresponding to target cohort(s).

`eventCohortIds`
: A vector of one or more Cohort Ids corresponding to event cohort(s).

`analysisType`
: Character. Either `"post-index"` (default) to analyze events after the index date, or `"pre-index"` to analyze events before the index date.

`lookbackStartDay`
: (Only used when `analysisType = "pre-index"`.) Start of the lookback window in days relative to the index date (default: -365). Must be negative.

`lookbackEndDay`
: (Only used when `analysisType = "pre-index"`.) End of the lookback window in days relative to the index date (default: -1). Must be negative.

`minCellCount`
: (Default = 5) The minimum cell count for fields containing person counts or fractions.

`allowRepeats`
: (Default = FALSE) Allow cohort events/combos to appear multiple times in the same pathway.

`maxDepth`
: (Default = 5) Maximum number of steps in a given pathway.

`collapseWindow`
: (Default = 30) Any dates found within the specified collapse days will be reassigned the earliest date.

Value

`A list of data frames containing pathway analysis results.`

Examples

```
if (requireNamespace("Eunomia", quietly = TRUE) &&
    exists("createCohorts", where = asNamespace("Eunomia"), inherits = FALSE)) {
  connectionDetails <- Eunomia::getEunomiaConnectionDetails()
  createCohorts <- get("createCohorts", envir = asNamespace("Eunomia"))
  createCohorts(connectionDetails)

  postIndexResults <- executeCohortPathways(
    connectionDetails = connectionDetails,
    cohortDatabaseSchema = "main",
    cohortTableName = "cohort",
    targetCohortIds = 4,
    eventCohortIds = c(1, 2, 3),
    maxDepth = 3
  )

  preIndexResults <- executeCohortPathways(
    connectionDetails = connectionDetails,
```

```
  cohortDatabaseSchema = "main",
  cohortTableName = "cohort",
  targetCohortIds = 4,
  eventCohortIds = c(1, 2, 3),
  analysisType = "pre-index",
  lookbackStartDay = -365,
  lookbackEndDay = -1,
  maxDepth = 3
)

names(postIndexResults)
names(preIndexResults)
}
```

`plot.event_sequence_graph` *Plot an event sequence graph.*

Description

Produces a layered plot of the event sequence graph using `plot.igraph`. Nodes are laid out by step (left to right), with edge widths proportional to patient counts and node sizes proportional to event counts within each step.

Usage

```
## S3 method for class 'event_sequence_graph'
plot(
  x,
  colorPalette = NULL,
  edgeWidthRange = c(0.5, 8),
  vertexSizeRange = c(8, 25),
  vertexLabelCex = 0.7,
  main = "Event Sequence Graph",
  ...
)
```

Arguments

`x`	An `event_sequence_graph` object (returned by `buildEventSequenceGraph`).
`colorPalette`	Character vector of hex colors for nodes. If `NULL` (default), a built-in palette is used. Colors are mapped by unique event name (same event = same color across steps).
`edgeWidthRange`	Numeric vector of length 2. Min and max edge widths (default: `c(0.5, 8)`).
`vertexSizeRange`	Numeric vector of length 2. Min and max vertex sizes (default: `c(8, 25)`).

vertexLabelCex	Label size multiplier (default: 0.7).
main	Plot title (default: "Event Sequence Graph").
...	Additional arguments passed to plot.igraph.

Value

Invisibly returns the igraph object.

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
  cpResults <- list(
    pathwaysAnalysisPathsData = data.frame(
      pathwayAnalysisGenerationId = c(1L, 1L),
      targetCohortId = c(10L, 10L),
      step1 = c(2L, 4L),
      step2 = c(4L, NA_integer_),
      countValue = c(10L, 5L)
    ),
    pathwayAnalysisCodesLong = data.frame(
      pathwayAnalysisGenerationId = c(1L, 1L),
      code = c(2L, 4L),
      targetCohortId = c(10L, 10L),
      eventCohortId = c(1L, 2L),
      isCombo = c(0L, 0L),
      numberOfEvents = c(1L, 1L)
    ),
    isCombo = data.frame(
      targetCohortId = c(10L, 10L),
      comboId = c(2L, 4L),
      numberOfEvents = c(1L, 1L),
      isCombo = c(0L, 0L)
    )
  )

  generationSet <- data.frame(
    cohortId = c(1L, 2L, 10L),
    cohortName = c("Celecoxib", "Diclofenac", "NSAIDs")
  )

  esg <- buildEventSequenceGraph(cpResults, generationSet)
  plot(esg)
}
```

Index

buildEventSequenceGraph, 2, *6*

connect, *4*
createConnectionDetails, *4*

executeCohortPathways, *2*, 3

igraph, *2*

plot.event_sequence_graph, 6
plot.igraph, *6, 7*