Package: TreatmentPatterns (via r-universe)

March 10, 2025

Type Package

Title Analyzes Real-World Treatment Patterns of a Study Population of Interest

Version 3.0.1

Maintainer Maarten van Kessel <m.l.vankessel@erasmusmc.nl>

Description Computes treatment patterns within a given cohort using the Observational Medical Outcomes Partnership (OMOP) common data model (CDM). As described in Markus, Verhamme, Kors, and Rijnbeek (2022) <doi:10.1016/j.cmpb.2022.107081>.

URL https://github.com/darwin-eu/TreatmentPatterns, https://darwin-eu-dev.github.io/TreatmentPatterns/

BugReports https://github.com/darwin-eu/TreatmentPatterns/issues

Language en-US

Depends R (>= 4.2)

Imports checkmate, dplyr, stringr, utils, stats, Andromeda, tidyr, R6, sunburstR, networkD3, htmlwidgets, ggplot2, dbplyr, jsonlite

Suggests knitr, rmarkdown, tibble, testthat (>= 3.0.0), usethis, Eunomia, CDMConnector, DatabaseConnector (>= 6.0.0), SqlRender, CohortGenerator, ResultModelManager, webshot2, CirceR, duckdb, DBI, withr, plotly, PaRe

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Collate 'CDMInterface.R' 'TreatmentPatterns-package.R' 'TreatmentPatternsResults.R' 'attrition.R' 'computePathways.R' 'constructPathways.R' 'createSankeyDiagram.R' 'createSunburstPlot.R' 'executeTreatmentPatterns.R' 'export.R'

'getResultsDataModelSpecification.R' 'ggSunburst.R'
'plotEventDuration.R'

NeedsCompilation no

Author Aniek Markus [aut] (<https://orcid.org/0000-0001-5779-4794>),
Maarten van Kessel [cre]
(<https://orcid.org/0009-0006-8832-6030>)

Repository CRAN

Date/Publication 2025-03-10 15:40:02 UTC

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

Contents

`computePathways` *computePathways*

Description

Compute treatment patterns according to the specified parameters within specified cohorts.

Usage

```
computePathways(
  cohorts,
  cohortTableName,
  cdm = NULL,
  connectionDetails = NULL,
  cdmSchema = NULL,
  resultSchema = NULL,
  analysisId = 1,
  description = "",
  tempEmulationSchema = NULL,
  includeTreatments = "startDate",
  indexDateOffset = 0,
  minEraDuration = 0,
```

```
  splitEventCohorts = NULL,
  splitTime = NULL,
  eraCollapseSize = 30,
  combinationWindow = 30,
  minPostCombinationDuration = 30,
  filterTreatments = "First",
  maxPathLength = 5
)
```

Arguments

`cohorts` (`data.frame()`)
Data frame containing the following columns and data types:

cohortId `numeric(1)` Cohort ID's of the cohorts to be used in the cohort table.

cohortName `character(1)` Cohort names of the cohorts to be used in the cohort table.

type `character(1)` **["target", "event', "exit"**] Cohort type, describing if the cohort is a target, event, or exit cohort

`cohortTableName`
(`character(1)`)
Cohort table name.

`cdm` (`CDMConnector::cdm_from_con()`: `NULL`)
Optional; Ignores `connectionDetails`, `cdmSchema`, and `resultSchema`.

`connectionDetails`
(`DatabaseConnector::createConnectionDetails()`: `NULL`)
Optional; In congruence with `cdmSchema` and `resultSchema`. Ignores `cdm`.

`cdmSchema` (`character(1)`: `NULL`)
Optional; In congruence with `connectionDetails` and `resultSchema`. Ignores `cdm`.

`resultSchema` (`character(1)`: `NULL`)
Optional; In congruence with `connectionDetails` and `cdmSchema`. Ignores `cdm`.

`analysisId` (`character(1)`) Identifier for the TreatmentPatterns analysis.

`description` (`character(1)`) Description of the analysis.

`tempEmulationSchema`
Schema used to emulate temp tables

`includeTreatments`
(`character(1)`: `"startDate"`)

`"startDate"` Include treatments after the target cohort start date and onwards.

`"endDate"` Include treatments before target cohort end date and before.

`indexDateOffset`
(`integer(1)`: `0`)
Offset the index date of the `Target` cohort.

`minEraDuration` (`integer(1)`: `0`)
: Minimum time an event era should last to be included in analysis

`splitEventCohorts`
: (`character(n)`: `""`)
Specify event cohort to split in acute (< X days) and therapy (>= X days)

`splitTime` (`integer(1)`: `30`)
: Specify number of days (X) at which each of the split event cohorts should be split in acute and therapy

`eraCollapseSize`
: (`integer(1)`: `30`)
Window of time between which two eras of the same event cohort are collapsed into one era

`combinationWindow`
: (`integer(1)`: `30`)
Window of time two event cohorts need to overlap to be considered a combination treatment

`minPostCombinationDuration`
: (`integer(1)`: `30`)
Minimum time an event era before or after a generated combination treatment should last to be included in analysis

`filterTreatments`
: (`character(1)`: `"First"` ["first", "Changes", "all"])
Select first occurrence of ('First'); changes between ('Changes'); or all event cohorts ('All').

`maxPathLength` (`integer(1)`: `5`)
: Maximum number of steps included in treatment pathway

Value

(`Andromeda::andromeda()`) andromeda object containing non-sharable patient level data outcomes.

Examples

```
ableToRun <- all(
  require("CirceR", character.only = TRUE, quietly = TRUE),
  require("CDMConnector", character.only = TRUE, quietly = TRUE),
  require("TreatmentPatterns", character.only = TRUE, quietly = TRUE),
  require("dplyr", character.only = TRUE, quietly = TRUE)
)

if (ableToRun) {
  library(TreatmentPatterns)
  library(CDMConnector)
  library(dplyr)

  withr::local_envvar(
    R_USER_CACHE_DIR = tempfile(),
    EUNOMIA_DATA_FOLDER = Sys.getenv("EUNOMIA_DATA_FOLDER", unset = tempfile())
  )
```

```
  tryCatch({
    if (Sys.getenv("skip_eunomia_download_test") != "TRUE") {
      CDMConnector::downloadEunomiaData(overwrite = TRUE)
    }
  }, error = function(e) NA)

  con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomiaDir())
  cdm <- cdmFromCon(con, cdmSchema = "main", writeSchema = "main")

  cohortSet <- readCohortSet(
    path = system.file(package = "TreatmentPatterns", "exampleCohorts")
  )

  cdm <- generateCohortSet(
    cdm = cdm,
    cohortSet = cohortSet,
    name = "cohort_table"
  )

  cohorts <- cohortSet %>%
    # Remove 'cohort' and 'json' columns
    select(-"cohort", -"json") %>%
   mutate(type = c("event", "event", "event", "event", "exit", "event", "event", "target")) %>%
    rename(
      cohortId = "cohort_definition_id",
      cohortName = "cohort_name",
    ) %>%
    select("cohortId", "cohortName", "type")

  outputEnv <- computePathways(
    cohorts = cohorts,
    cohortTableName = "cohort_table",
    cdm = cdm
  )

  Andromeda::close(outputEnv)
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

---

createSankeyDiagram *createSankeyDiagram*

Description

Create sankey diagram.

Usage

```
createSankeyDiagram(
  treatmentPathways,
  groupCombinations = FALSE,
  colors = NULL,
  ...
)
```

Arguments

treatmentPathways	(data.frame()) The contents of the treatmentPathways.csv-file as a data.frame().
groupCombinations	(logical(1): FALSE) TRUE Group all combination treatments in category "Combination". FALSE Do not group combination treatments.
colors	(character(n)) Vector of hex color codes.
...	Paramaters for sankeyNetwork.

Value

(htmlwidget)

Examples

```
# Dummy data, typically read from treatmentPathways.csv
treatmentPathways <- data.frame(
  pathway = c("Acetaminophen", "Acetaminophen-Amoxicillin+Clavulanate",
           "Acetaminophen-Aspirin", "Amoxicillin+Clavulanate", "Aspirin"),
  freq = c(206, 6, 14, 48, 221),
  sex = rep("all", 5),
  age = rep("all", 5),
  index_year = rep("all", 5)
)

createSankeyDiagram(treatmentPathways)
```

---

createSunburstPlot *createSunburstPlot*

Description

New sunburstPlot function

Usage

```
createSunburstPlot(treatmentPathways, groupCombinations = FALSE, ...)
```

Arguments

treatmentPathways	(data.frame()) The contents of the treatmentPathways.csv-file as a data.frame().
groupCombinations	(logical(1): FALSE) TRUE Group all combination treatments in category "Combination". FALSE Do not group combination treatments.
...	Paramaters for sunburst.

Value

(htmlwidget)

Examples

```
# Dummy data, typically read from treatmentPathways.csv
treatmentPatwhays <- data.frame(
  pathway = c("Acetaminophen", "Acetaminophen-Amoxicillin+Clavulanate",
           "Acetaminophen-Aspirin", "Amoxicillin+Clavulanate", "Aspirin"),
  freq = c(206, 6, 14, 48, 221),
  sex = rep("all", 5),
  age = rep("all", 5),
  index_year = rep("all", 5)
)

createSunburstPlot(treatmentPatwhays)
```

---

executeTreatmentPatterns

executeTreatmentPatterns

Description

Compute treatment patterns according to the specified parameters within specified cohorts. For more customization, or investigation of patient level outcomes, you can run computePathways and export separately.

Usage

```
executeTreatmentPatterns(
  cohorts,
  cohortTableName,
  cdm = NULL,
  connectionDetails = NULL,
  cdmSchema = NULL,
  resultSchema = NULL,
  tempEmulationSchema = NULL,
  minEraDuration = 0,
  eraCollapseSize = 30,
  combinationWindow = 30,
  minCellCount = 5
)
```

Arguments

cohorts	(data.frame()) Data frame containing the following columns and data types: **cohortId** numeric(1) Cohort ID's of the cohorts to be used in the cohort table. **cohortName** character(1) Cohort names of the cohorts to be used in the cohort table. **type** character(1) **["target", "event', "exit"**] Cohort type, describing if the cohort is a target, event, or exit cohort
cohortTableName	(character(1)) Cohort table name.
cdm	(CDMConnector::cdm_from_con(): NULL) Optional; Ignores connectionDetails, cdmSchema, and resultSchema.
connectionDetails	(DatabaseConnector::createConnectionDetails(): NULL) Optional; In congruence with cdmSchema and resultSchema. Ignores cdm.
cdmSchema	(character(1): NULL) Optional; In congruence with connectionDetails and resultSchema. Ignores cdm.
resultSchema	(character(1): NULL) Optional; In congruence with connectionDetails and cdmSchema. Ignores cdm.
tempEmulationSchema	(character(1)) Schema to emulate temp tables.
minEraDuration	(integer(1): 0) Minimum time an event era should last to be included in analysis
eraCollapseSize	(integer(1): 30) Window of time between which two eras of the same event cohort are collapsed into one era

combinationWindow
: (integer(1): 30)
Window of time two event cohorts need to overlap to be considered a combination treatment

minCellCount
: (integer(1): 5)
Minimum count required per pathway. Censors data below x as <x. This minimum value will carry over to the sankey diagram and sunburst plot.

Value

TreatmentPatternsResults

Examples

```
ableToRun <- all(
  require("CirceR", character.only = TRUE, quietly = TRUE),
  require("CDMConnector", character.only = TRUE, quietly = TRUE),
  require("TreatmentPatterns", character.only = TRUE, quietly = TRUE),
  require("dplyr", character.only = TRUE, quietly = TRUE)
)

if (require("CirceR", character.only = TRUE, quietly = TRUE)) {
  library(TreatmentPatterns)
  library(CDMConnector)
  library(dplyr)

  withr::local_envvar(
    R_USER_CACHE_DIR = tempfile(),
    EUNOMIA_DATA_FOLDER = Sys.getenv("EUNOMIA_DATA_FOLDER", unset = tempfile())
  )

  tryCatch({
    if (Sys.getenv("skip_eunomia_download_test") != "TRUE") {
      CDMConnector::downloadEunomiaData(overwrite = TRUE)
    }
  },
  error = function(e) NA)

  con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomiaDir())
  cdm <- cdmFromCon(con, cdmSchema = "main", writeSchema = "main")

  cohortSet <- readCohortSet(
    path = system.file(package = "TreatmentPatterns", "exampleCohorts")
  )

  cdm <- generateCohortSet(
    cdm = cdm,
    cohortSet = cohortSet,
    name = "cohort_table"
  )

  cohorts <- cohortSet %>%
```

```
    # Remove 'cohort' and 'json' columns
    select(-"cohort", -"json") %>%
   mutate(type = c("event", "event", "event", "event", "exit", "event", "event", "target")) %>%
    rename(
      cohortId = "cohort_definition_id",
      cohortName = "cohort_name",
    ) %>%
    select("cohortId", "cohortName", "type")

  executeTreatmentPatterns(
    cohorts = cohorts,
    cohortTableName = "cohort_table",
    cdm = cdm
  )

  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

`export` *export*

Description

Export andromeda generated by computePathways object to sharable csv-files and/or a zip archive.

Usage

```
export(
  andromeda,
  outputPath = NULL,
  ageWindow = 10,
  minCellCount = 5,
  censorType = "minCellCount",
  archiveName = NULL,
  nonePaths = FALSE,
  stratify = FALSE
)
```

Arguments

`andromeda`	(`Andromeda::andromeda()`) Andromeda object.
`outputPath`	(`character`: `NULL`) Output path where to write output files to. When set to `NULL` no files will be written, and only the results object is returned.
`ageWindow`	(`integer(n)`: `10`) Number of years to bin age groups into. It may also be a vector of integers. I.e. `c(0, 18, 150)` which will results in age group `0-18` which includes subjects `< 19`. And age group `18-150` which includes subjects `> 18`.

`minCellCount` (integer(1): 5)
Minimum count required per pathway. Censors data below x as <x. This minimum value will carry over to the sankey diagram and sunburst plot.

`censorType` (character(1))

"minCellCount" Censors pathways <minCellCount to minCellCount.

"remove" Censors pathways <minCellCount by removing them completely.

"mean" Censors pathways <minCellCount to the mean of all frequencies below minCellCount

`archiveName` (character(1): NULL)
If not NULL adds the exported files to a ZIP-file with the specified archive name.

`nonePaths` (logical(1)) Should None paths be included? This will fetch all persons included in the target cohort and assign them a "None" pathway. Significantly impacts performance.

`stratify` (logical(1)) Should pathways be stratified? This will perform pairwise stratification between age, sex, and index year. Significantly impacts performance.

Value

TreatmentPatternsResults object

Examples

```
ableToRun <- all(
  require("CirceR", character.only = TRUE, quietly = TRUE),
  require("CDMConnector", character.only = TRUE, quietly = TRUE),
  require("TreatmentPatterns", character.only = TRUE, quietly = TRUE),
  require("dplyr", character.only = TRUE, quietly = TRUE)
)

if (ableToRun) {
  library(TreatmentPatterns)
  library(CDMConnector)
  library(dplyr)

  withr::local_envvar(
    R_USER_CACHE_DIR = tempfile(),
    EUNOMIA_DATA_FOLDER = Sys.getenv("EUNOMIA_DATA_FOLDER", unset = tempfile())
  )

  tryCatch({
    if (Sys.getenv("skip_eunomia_download_test") != "TRUE") {
      CDMConnector::downloadEunomiaData(overwrite = TRUE)
    }
  }, error = function(e) NA)

  con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomiaDir())
  cdm <- cdmFromCon(con, cdmSchema = "main", writeSchema = "main")
```

```
  cohortSet <- readCohortSet(
    path = system.file(package = "TreatmentPatterns", "exampleCohorts")
  )

  cdm <- generateCohortSet(
    cdm = cdm,
    cohortSet = cohortSet,
    name = "cohort_table"
  )

  cohorts <- cohortSet %>%
    # Remove 'cohort' and 'json' columns
    select(-"cohort", -"json") %>%
   mutate(type = c("event", "event", "event", "event", "exit", "event", "event", "target")) %>%
    rename(
      cohortId = "cohort_definition_id",
      cohortName = "cohort_name",
    ) %>%
    select("cohortId", "cohortName", "type")

  outputEnv <- computePathways(
    cohorts = cohorts,
    cohortTableName = "cohort_table",
    cdm = cdm
  )

  results <- export(
    andromeda = outputEnv
  )

  Andromeda::close(outputEnv)
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

`getResultsDataModelSpecifications` *getResultsDataModelSpecifications*

Description

Gets the results data model specifications of `TreatmentPatterns`.

Usage

```
getResultsDataModelSpecifications()
```

Value

`data.frame`

Examples

```
{
getResultsDataModelSpecifications()
}
```

---

ggSunburst *ggSunburst*

Description

ggSunburst

Usage

```
ggSunburst(treatmentPathways, groupCombinations = FALSE, unit = "percent")
```

Arguments

treatmentPathways	(data.frame()) The contents of the treatmentPathways.csv-file as a data.frame().
groupCombinations	(logical(1): FALSE) TRUE Group all combination treatments in category "Combination". FALSE Do not group combination treatments.
unit	(character(1)) Either "count" or "percent", to scale the plot to.

Value

(gg, ggplot)

Examples

```
# Dummy data, typically read from treatmentPathways.csv
treatmentPatwhays <- data.frame(
  pathway = c("Acetaminophen", "Acetaminophen-Amoxicillin+Clavulanate",
           "Acetaminophen-Aspirin", "Amoxicillin+Clavulanate", "Aspirin"),
  freq = c(206, 6, 14, 48, 221),
  sex = rep("all", 5),
  age = rep("all", 5),
  index_year = rep("all", 5)
)

ggSunburst(treatmentPatwhays)
```

plotEventDuration *plotEventDuration*

Description

plotEventDuration

Usage

```
plotEventDuration(
  eventDurations,
  minCellCount = 0,
  treatmentGroups = "both",
  eventLines = NULL,
  includeOverall = TRUE
)
```

Arguments

eventDurations	(data.frame) Contents of summaryEventDuration.csv file.
minCellCount	(numeric(1): 0) Min Cell Count per event group.
treatmentGroups	(character(1): "both") "group": Only mono-, and combination-events. "individual": Only individual (combination) events. "both": Both mono-, and combination-events, and individual (combination) events.
eventLines	(numeric(n): NULL) Event lines to include, i.e. c(1, 2, 3) includes first (1), second (2), and third (3) lines of events. NULL will include all eventLines.
includeOverall	(logical(1): TRUE) TRUE: Include an overall column with the eventLines. FALSE: Exclude the overall column.

Value

ggplot

Examples

```
ableToRun <- all(
  require("CirceR", character.only = TRUE, quietly = TRUE),
  require("CDMConnector", character.only = TRUE, quietly = TRUE),
  require("TreatmentPatterns", character.only = TRUE, quietly = TRUE),
  require("dplyr", character.only = TRUE, quietly = TRUE)
)

if (ableToRun) {
  withr::local_envvar(
    R_USER_CACHE_DIR = tempfile(),
    EUNOMIA_DATA_FOLDER = Sys.getenv("EUNOMIA_DATA_FOLDER", unset = tempfile())
```

```
  )

  tryCatch({
    if (Sys.getenv("skip_eunomia_download_test") != "TRUE") {
      CDMConnector::downloadEunomiaData(overwrite = TRUE)
    }
  }, error = function(e) NA)

  con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomiaDir())
  cdm <- cdmFromCon(con, cdmSchema = "main", writeSchema = "main")

  cohortSet <- readCohortSet(
    path = system.file(package = "TreatmentPatterns", "exampleCohorts")
  )

  cdm <- generateCohortSet(
    cdm = cdm,
    cohortSet = cohortSet,
    name = "cohort_table"
  )

  cohorts <- cohortSet %>%
    # Remove 'cohort' and 'json' columns
    select(-"cohort", -"json") %>%
   mutate(type = c("event", "event", "event", "event", "exit", "event", "event", "target")) %>%
    rename(
      cohortId = "cohort_definition_id",
      cohortName = "cohort_name",
    ) %>%
    select("cohortId", "cohortName", "type")

  outputEnv <- computePathways(
    cohorts = cohorts,
    cohortTableName = "cohort_table",
    cdm = cdm
  )

  results <- export(outputEnv)

  plotEventDuration(
    eventDurations = results$summary_event_duration,
    minCellCount = 5,
    treatmentGroups = "group",
    eventLines = 1:4,
    includeOverall = FALSE
  )

  Andromeda::close(outputEnv)
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

TreatmentPatternsResults

TreatmentPatternsResults Class

Description

Houses the results of a TreatmentPatterns analysis. Each field corresponds to a file. Plotting methods are provided.

Active bindings

attrition (data.frame)

metadata (data.frame)

treatment_pathways (data.frame)

summary_event_duration (data.frame)

counts_age (data.frame)

counts_sex (data.frame)

counts_year (data.frame)

cdm_source_info (data.frame)

analyses (data.frame)

arguments (list)

Methods

Public methods:

- TreatmentPatternsResults$new()
- TreatmentPatternsResults$saveAsZip()
- TreatmentPatternsResults$saveAsCsv()
- TreatmentPatternsResults$uploadResultsToDb()
- TreatmentPatternsResults$load()
- TreatmentPatternsResults$plotSunburst()
- TreatmentPatternsResults$plotSankey()
- TreatmentPatternsResults$plotEventDuration()
- TreatmentPatternsResults$clone()

Method new(): Initializer method

Usage:

```
TreatmentPatternsResults$new(
  attrition = NULL,
  metadata = NULL,
  treatmentPathways = NULL,
  summaryEventDuration = NULL,
```

```
  countsAge = NULL,
  countsSex = NULL,
  countsYear = NULL,
  cdmSourceInfo = NULL,
  analyses = NULL,
  arguments = NULL,
  filePath = NULL
)
```

Arguments:

`attrition` (`data.frame`) attrition result.

`metadata` (`data.frame`)) metadata result.

`treatmentPathways` (`data.frame`)) treatmentPathways result.

`summaryEventDuration` (`data.frame`)) summaryEventDuration result.

`countsAge` (`data.frame`)) countsAge result.

`countsSex` (`data.frame`)) countsSex result.

`countsYear` (`data.frame`)) countsYear result.

`cdmSourceInfo` (`data.frame`) cdmSourceInfo result.

`analyses` (`data.frame`) Analyses result.

`arguments` (`list`) Named list of arguments used.

`filePath` (`character`) File path to either a directory or zip-file, containing the csv-files.

Method `saveAsZip()`**:** Save the results as a zip-file.

Usage:

```
TreatmentPatternsResults$saveAsZip(path, name, verbose = TRUE)
```

Arguments:

`path` (`character(1)`) Path to write to.

`name` (`character(1)`) File name.

`verbose` (`logical`: `TRUE`) Verbose messaging.

Returns: `self`

Method `saveAsCsv()`**:** Save the results as csv-files.

Usage:

```
TreatmentPatternsResults$saveAsCsv(path, verbose = TRUE)
```

Arguments:

`path` (`character(1)`) Path to write to.

`verbose` (`logical`: `TRUE`) Verbose messaging.

Returns: `self`

Method `uploadResultsToDb()`**:** Upload results to a resultsDatabase using `ResultModelManager`.

Usage:

```
TreatmentPatternsResults$uploadResultsToDb(
  connectionDetails,
  schema,
  prefix = "tp_",
  overwrite = TRUE,
  purgeSiteDataBeforeUploading = FALSE
)
```

Arguments:

`connectionDetails` (`ConnectionDetails`) ConnectionDetails object from `DatabaseConnector`.

`schema` (`character(1)`) Schema to write tables to.

`prefix` (`character(1)`: `"tp_"`) Table prefix.

`overwrite` (`logical(1)`: `TRUE`) Should tables be overwritten?

`purgeSiteDataBeforeUploading` (`logical`: `FALSE`) Should site data be purged before uploading?

Returns: `self`

Method `load()`**:** Load data from files.

Usage:

```
TreatmentPatternsResults$load(filePath)
```

Arguments:

`filePath` (`character(1)`) Path to a directory or zip-file containing the result csv-files.

Returns: `self`

Method `plotSunburst()`**:** Wrapper for `TreatmentPatterns::createSunburstPlot()`, but with data filtering step.

Usage:

```
TreatmentPatternsResults$plotSunburst(
  age = "all",
  sex = "all",
  indexYear = "all",
  nonePaths = FALSE,
  ...
)
```

Arguments:

`age` (`character(1)`) Age group.

`sex` (`character(1)`) Sex group.

`indexYear` (`character(1)`) Index year group.

`nonePaths` (`logical(1)`) Should `None` paths be included?

`...` Parameters for `TreatmentPatterns::createSunburstPlot()`

Returns: `htmlwidget`

Method `plotSankey()`**:** Wrapper for `TreatmentPatterns::createSankeyDiagram()`, but with data filtering step.

Usage:

```
TreatmentPatternsResults$plotSankey(
  age = "all",
  sex = "all",
  indexYear = "all",
  nonePaths = FALSE,
  ...
)
```

Arguments:

`age` (`character(1)`) Age group.

`sex` (`character(1)`) Sex group.

`indexYear` (`character(1)`) Index year group.

`nonePaths` (`logical(1)`) Should `None` paths be included?

`...` Parameters for `TreatmentPatterns::createSankeyDiagram()`

Returns: `htmlwidget`

Method `plotEventDuration()`**:** Wrapper for `TreatmentPatterns::plotEventDuration()`.

Usage:

```
TreatmentPatternsResults$plotEventDuration(...)
```

Arguments:

`...` Parameters for `TreatmentPatterns::plotEventDuration()`

Returns: `ggplot`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TreatmentPatternsResults$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index