Package ‘tame’

April 2, 2025

Title Timing, Anatomical, Therapeutic and Chemical Based Medication Clustering

Version 0.2.0

Description Agglomerative hierarchical clustering with a bespoke distance measure based on medication similarities in the Anatomical Therapeutic Chemical Classification System, medication timing and medication amount or dosage. Tools for summarizing, illustrating and manipulating the cluster objects are also available.

License GPL (>= 3) | file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr (>= 1.0.0), fuzzyjoin, magrittr, purrr (>= 1.0.0), Rfast, rlang (>= 1.0.0), stats, stringr, tibble (>= 3.0.0), tidyr (>= 1.2.0), tidyselect, Rcpp (>= 1.0.8), ggplot2 (>= 3.3.0), scales, utils

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

BugReports `https://github.com/Laksafoss/tame/issues`

Config/testthat/edition 3

Depends R (>= 4.2)

LazyData true

LinkingTo Rcpp

NeedsCompilation yes

Author Anna Laksafoss [aut, cre] (<`https://orcid.org/0000-0002-9898-2924`>)

Maintainer Anna Laksafoss <`adls@ssi.dk`>

Repository CRAN

Date/Publication 2025-04-02 14:20:02 UTC

Contents

`cluster_frequency` *The Frequency of Assignment to Each Cluster*

Description

The function `cluster_frequency()` calculates the number and frequency of individuals assigned to each cluster.

Usage

```
cluster_frequency(
  object,
  only = NULL,
  clusters = NULL,
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Details

`cluster_frequency()` calculates the number of individuals assigned to each cluster and the associated frequency of assignment.

Value

`cluster_frequency()` returns a data frame with class `cluster_frequency`.

- `Clustering` the name of the clustering.
- `Cluster` the cluster name.
- `Count` the number of individuals assigned to the cluster.
- `Percent` the percent of individuals assigned to the cluster.

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3:5)

# make frequency tables
cluster_frequency(clust, k == 5)
cluster_frequency(clust, k < 5, I:III)
```

`comedication_count` *Frequency tables for medication amount*

Description

The function `comedication_count()` calculates the number of unique medications for each individual and presents the count frequencies by cluster.

Usage

```
comedication_count(
  object,
  only = NULL,
  clusters = NULL,
  count_grouper = function(x) {
     cut(x, breaks = c(0, 1, 2, Inf), labels = c("1",
    "2", "3+"))
 },
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`count_grouper`	A function for grouping counts. As a standard it groups counts as 1 medication, 2 medications, and 3+ medications.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Details

`comedication_count()` calculates the number of ATC codes an individual has, and then outputs the number of individuals within a cluster that has that many ATC codes. Moreover, various relevant percentages or calculated. See Value below for more details on these percentages.

Value

`comedication_count()` returns a data frame of class `comedication_count`

- `Clustering` the name of the clustering.
- `Cluster` the name of the cluster.
- `Medication Count` a number of medications. The numbers or groups are given by the `count_grouper()` function.
- `Number of People` the number of individuals in cluster who has `Medication Count` number of comedications in study.
- `Number of medications` the number of medications of individuals who has `Medication Count` number of comedications in the cluster.
- `Percentage of All People` the percentage of individuals is study who has `Medication Count` number of comedications in the cluster.
- `Percentage of People in Cluster` the percentage of individuals in the cluster who has `Medication Count` number of comedications.
- `Percentage of All Medications` the percentage of medication in study from individuals who has `Medication Count` number of comedications in cluster.
- `Percentage of Medication in Cluster` the percentage of medication in cluster from individuals who has `Medication Count` number of comedications.
- `Percentage of People with the Same Medication Count` percentage of individuals among those with `Medication Count` number of comedications in this cluster.
- `Percentage of Medication with the Same Medication Count` percentage of medication among medication of individuals with `Medication Count` number of comedications in this cluster.

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3:5)

comedication_count(clust, k == 5, clusters = I:III)
```

`complications` *A Simulated Data Set About Pregnancy Complications*

Description

We use this data set in all the examples in the package.

Usage

```
complications
```

Format

An object of class `data.frame` with 149 rows and 8 columns.

`default_atc_groups` *Default ATC groups for summaries*

Description

This function finds the default ATC groups for the summaries. It is used in the `summary.medic` function.

Usage

```
default_atc_groups(object, min_n = 2)
```

Arguments

`object`	A `medic` object.
`min_n`	The minimum number of ATC groups to be found.

Value

A data frame with two columns: `regex` and `atc_groups`.

`eczema` *A Simulated Data Set About Eczema*

Description

A Simulated Data Set About Eczema

Usage

```
eczema
```

Format

An object of class `data.frame` with 50644 rows and 7 columns.

`employ` *Employ a Clustering to New Data*

Description

Employ a clustering to new data

Usage

```
employ(
  object,
  new_data,
  only = NULL,
  additional_data = NULL,
  assignment_method = "nearest_cluster",
  parallel = FALSE,
  ...
)
```

Arguments

`object`	A `medic` clustering object for which employment is desired.
`new_data`	A data frame in which to look for variables with
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data` and specifies which clusterings should be employed to the new data.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`assignment_method`	A character naming the employment method. The default assignment method `"nearest_cluster"` matches people in `new_data` to their nearest cluster in the chosen clusterings from `object`. As finding exact matches (the next assignment method) is contained within this strategy the `"exact_only"` matches are also reported in additional columns in the output. The assignment method `"exact_only"` only matches a person from `new_data` to a cluster if they are a perfect match to anyone in `object`. Thus, people from `new_data` are not guaranteed assignment to a cluster.
`parallel`	A logical or an integer. If `FALSE`, the default, no parallelization is done. If `TRUE` or an integer larger than 2L parallelization is implemented via parLapply from the **parallel** package. When `parallel` is `TRUE` the number of clusters is set to detectCores - 1, and when `parallel` is an integer then the number of clusters is set to `parallel`. For more details on the parallelization method see parallel::parLapply.
`...`	Additional arguments affecting the employment procedure.

Value

`employ` returns a medic object.

Examples

```
part1 <- complications[1:100,]
part2 <- complications[101:149,]

clust <- medic(part1, id = id, atc = atc, k = 3)

# Nearest cluster matching
employ(clust, part2)

# Only exact matching
employ(clust, part2, assignment_method = "exact_only")
```

`enrich` *Enrich Clustering Parameter*

Description

Enrich the parameter information in a clustering with user-defined data.

Usage

```
enrich(object, additional_data = NULL, by = NULL)
```

Arguments

`object`	A medic object for enrichment.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`.
`by`	A character vector of variables to join by. This variables is passed to the by term in a `dplyr::left_join()` and inherits its behavior: If `NULL`, the default, the join will perform a natural join, using all variables in common across the `parameters` and `additional_data`. To join by different variables on `parameters` and `additional_data`, use a named vector. For example, by = `c("k" = "cluster_size")` will match `parameters$k` to `additional_data$cluster_size`. To join by multiple variables, use a vector with length > 1. For example, by = `c("k", "summation_method")` will match `parameters$k` to `additional_data$k` and `parameters$summation_method` to `additional_data$summation_method`. Use a named vector to match different variables in `parameters` and `additional_data`. For example, by = `c("k" = "cluster_size", "summation_method" = "sm")` will match `parameters$k` to `additional_data$cluster_size` and `parameters$summation_method` to `additional_data$sm`.

Details

The `enrich()` function is a joining function used for enriching the clustering characteristics with user-defined data. This function is used in all of the investigative functions with a `additional_data` statement such as `summary()`, `cluster_frequency()` and `medication_frequency()`.

Value

An object of class *medic*.

Examples

```
clust <- medic(
   complications,
   id = id,
   atc = atc,
   timing = first_trimester:third_trimester,
   k = 3:5
)

new_parameters <- data.frame(k = 3:5, size = c("small", "small", "large"))

enrich(clust, new_parameters)
```

`is.medic` *Test if an object is a medic-object*

Description

Test if an object is a medic-object

Usage

```
is.medic(object)
```

Arguments

`object`	Any object.

Value

`TRUE` is the object inherits from the `medic` class and has the required elements.

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3)
is.medic(clust)
```

`medic` *Medication clustering (based on ATC and timing)*

Description

The `medic` method uses agglomerative hierarchical clustering with a bespoke distance measure based on medication ATC codes similarities, medication timing and medication amount or dosage.

Usage

```
medic(
  data,
  k = 5,
  id,
  atc,
  timing,
  base_clustering,
  linkage = "complete",
  summation_method = "sum_of_minima",
  alpha = 1,
  beta = 1,
  gamma = 1,
  p = 1,
  theta = (5:0)/5,
  parallel = FALSE,
  return_distance_matrix = FALSE,
  set_seed = FALSE,
  ...
)

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

Arguments

`data`	A data frame containing all the variables for the clustering.
`k`	a vector specifying the number of clusters to identify.
`id`	<`tidy-select`> An unquoted expression naming the variable in `data` describing person id.
`atc`	<`tidy-select`> An unquoted expression naming the variable in `data` containing ATC codes.
`timing`	<`tidy-select`> An unquoted expression naming the variable or variables in `data` describing medication timing. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables. Moreover, pattern matching selection helpers such as `starts_with` or `num_range` may also be used to select timing variables.

`base_clustering`	<`tidy-select`> An unquoted expression naming the variable in `data` that gives an initial clustering to start the `medic` from or `NULL`.
`linkage`	The agglomeration method to be used in the clustering. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). See stats::hclust for more information. For a discussion of linkage criterion choice see *details* below.
`summation_method`	The summation method used in the distance measure. This should be either "double_sum" or "sum_of_minima". See *details* below for more information.
`alpha`	A number giving the tuning of the normalization. See *details* below for more information.
`beta`	A number giving the power of the individual medication combinations. See *details* below for more information.
`gamma`	A number giving the weight of the timing terms. See *details* below for more information.
`p`	The power of the Minkowski distance used in the timing-specific distance. See *details* below for more information.
`theta`	A vector of length 6 specifying the tuning of the ATC measure. See *details* below for more information.
`parallel`	A logical or an integer. If `FALSE`, the default, no parallelization is done. If `TRUE` or an integer larger than 2L parallelization is implemented via parLapply from the **parallel** package. When `parallel` is `TRUE` the number of clusters is set to detectCores - 1, and when `parallel` is an integer then the number of clusters is set to `parallel`. For more details on the parallelization method see parallel::parLapply.
`return_distance_matrix`	A logical.
`set_seed`	A logical or an integer.
`...`	Additional arguments not currently in use.
`x`	A `medic` object for printing.

Details

The `medic` method uses agglomerative hierarchical clustering with a bespoke distance measure based on medication ATC codes and timing similarities to assign medication pattern clusters to people.

Two versions of the distance measure are available:

The *double sum*:

$$d(p_i, p_j) = N_\alpha(M_i \times M_j) \sum_{m \in M_i} \sum_{n \in M_j} ((1 + D_\theta(m,n))(1 + \gamma T_p(t_{im}, t_{jn})) - 1)^\beta.$$

and the *sum of minima*:

$$d(p_i, p_j) = \frac{1}{2}(N_\alpha(M_i) \sum_{m \in M_i} \min_{n \in M_j} ((1+D_\theta(m,n))(1+\gamma T_p(t_{im}, t_{jn}))-1)^\beta + N_\alpha(M_j) \sum_{n \in M_j} \min_{m \in M_i} ((1+D_\theta(m,n))(1+\gamma T_p($$

Normalization:

$$N_\alpha(x) = |x|^{-\alpha}$$

If the normalization tuning, `alpha`, is 0, then no normalization is preformed and the distance measure becomes highly dependent on the number of distinct medications given. That is, people using more medication will have larger distances to others. If the normalization tuning, `alpha`, is 1 - the default - then the summation is normalized with the number of terms in the sum, in other words, the average is calculated.

ATC distance:
The central idea of this method, namely the ATC distance, is given as

$$D_\theta(x, y) = \sum_{i=1,\ldots,5} 1\{xandymatchonleveli, butnotleveli + 1\}\theta_i$$

The ATC distance is tuned using the vector `theta`.
Note that two ATC codes are said to match at level i when they are identical at level i. E.g. the two codes N06AB01 and N06AA01 match on level 1, 2, and 3 as they are both "N" at level 1, "N06" at level 2, and "N06A" at level 3, but at level 4 they differ ("N06AB" and "N06AA" are not the same).

Timing distance:
The timing distance is a simple Minkowski distance:

$$T(x, y) = (\sum_{t \in T} |x_t - y_t|^p)^{1/p}.$$

When `p` is 1, the default, the Manhattan distance is used.

Value

An object of class *medic* which describes the clusters produced the hierarchical clustering process. The object is a list with components:

data the inputted data frame `data` with the cluster assignments appended at the end.

clustering a data frame with the person id as given by `id`, the `.analysis_order` and the clusters found.

variables a list of the variables used in the clustering.

parameters a data frame with all the inputted clustering parameters and the corresponding method names. These method names correspond to the column names for each cluster in the `clustering` data frame described right above.

key a list of keys used internally in the function to keep track of simplified versions of the data.

distance_matrix the distance matrices for each method if `return_distance_matrix` is `TRUE` otherwise `NULL`.

call the matched call.

Methods (by generic)

- `print(medic)`: Print method for medic-objects

See Also

summary.medic for summaries and plots.

employ for employing an existing clustering to new data.

enrich for enriching the meta data in the `medic` object with additional data.

Examples

```
# A simple clustering based only on ATC
clust <- medic(complications, id = id, atc = atc, k = 3)

# A simple clustering with both ATC and timing
clust <- medic(
  complications,
  id = id,
  atc = atc,
  timing = first_trimester:third_trimester,
  k = 3
)
```

`medication_frequency` *ATC Code Frequency Within Clusters*

Description

The function `medications()` calculates the frequency of the different unique ATC codes within each cluster.

Usage

```
medication_frequency(
  object,
  only = NULL,
  clusters = NULL,
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Details

`medication_frequency()` calculates the number of individuals with a specific ATC code within a cluster. Moreover, it calculates the percentage of people with this medication assigned to this cluster and the percent of people within the cluster with this medication.

Value

`medication_frequency()` returns a data frame with class `medication_frequency`.

- `Clustering` the name of the clustering.
- `Cluster` the cluster name.
- *atc* ATC codes.
- `Count` number of individuals with this ATC code in this cluster.
- `Percent of All Medication` the percentage of individuals in the study with this ATC code and cluster.
- `Percent of Medication in Cluster` the percent of individuals in the cluster with this ATC code.

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3:5)

medication_frequency(clust, k == 5, clusters = I:III)
```

---

parameters_constructor

Internal option constructor

Description

Given the input of the `medic` this function checks the input and constructs a data frame with the analysis parameters specified by the user.

Usage

```
parameters_constructor(
  data,
  id,
  k = 5,
  atc,
  timing,
  base_clustering,
  linkage = "complete",
  summation_method = "sum_of_minima",
  alpha = 1,
  beta = 1,
  gamma = 1,
  p = 1,
  theta = (5:0)/5,
  ...
)
```

Arguments

`data`	A data frame containing all the variables for the clustering.
`id`	<`tidy-select`> An unquoted expression naming the variable in `data` describing person id.
`k`	a vector specifying the number of clusters to identify.
`atc`	<`tidy-select`> An unquoted expression naming the variable in `data` containing ATC codes.
`timing`	<`tidy-select`> An unquoted expression naming the variable or variables in `data` describing medication timing. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables. Moreover, pattern matching selection helpers such as `starts_with` or `num_range` may also be used to select timing variables.
`base_clustering`	<`tidy-select`> An unquoted expression naming the variable in `data` that gives an initial clustering to start the `medic` from or `NULL`.

`linkage`	The agglomeration method to be used in the clustering. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). See stats::hclust for more information. For a discussion of linkage criterion choice see *details* below.
`summation_method`	The summation method used in the distance measure. This should be either "double_sum" or "sum_of_minima". See *details* below for more information.
`alpha`	A number giving the tuning of the normalization. See *details* below for more information.
`beta`	A number giving the power of the individual medication combinations. See *details* below for more information.
`gamma`	A number giving the weight of the timing terms. See *details* below for more information.
`p`	The power of the Minkowski distance used in the timing-specific distance. See *details* below for more information.
`theta`	A vector of length 6 specifying the tuning of the ATC measure. See *details* below for more information.
`...`	Additional arguments not currently in use.

Value

A data.frame with the parameters for clustering.

Examples

```
parameters_constructor(
   data = complications,
   k = 3,
   id = id,
   atc = atc
)
```

`plot_cluster_frequency`

Plot Cluster Frequency

Description

This function plots the cluster frequency.

Usage

```
plot_cluster_frequency(object, ...)

## S3 method for class 'medic'
plot_cluster_frequency(object, ...)

## S3 method for class 'summary.medic'
plot_cluster_frequency(object, ...)

## S3 method for class 'cluster_frequency'
plot_cluster_frequency(object, scale = "percent", with_population = FALSE, ...)
```

Arguments

`object`	The object containing the cluster frequency data.
`...`	Additional arguments passed to the plotting functions.
`scale`	The scale of the y-axis. Must be either "percent" or "count".
`with_population`	Logical value indicating whether to include the population cluster.

Value

A ggplot object.

See Also

`cluster_frequency`

`plot_medication_frequency`

`plot_comedication_count`

`plot_timing_trajectory`

`plot_timing_atc_group`

`plot_summary`

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3)
clust |> plot_cluster_frequency()
clust |> cluster_frequency() |> plot_cluster_frequency()
clust |> summary() |> plot_cluster_frequency()
```

`plot_comedication_count` *Plot Comedication Count*

Description

This function plots the comedication count.

Usage

```
plot_comedication_count(object, ...)

## S3 method for class 'medic'
plot_comedication_count(object, ...)

## S3 method for class 'summary.medic'
plot_comedication_count(object, ...)

## S3 method for class 'comedication_count'
plot_comedication_count(
  object,
  scale = "percent",
  scope = "cluster",
  focus = "people",
  with_population = FALSE,
  ...
)
```

Arguments

`object`	The object containing the comedication count data.
`...`	Additional arguments passed to the plotting functions.
`scale`	The scale of the y-axis. Must be either "percent" or "count".
`scope`	The scope of the plot. Must be one of "cluster", "global" or "medication count".
`focus`	The focus of the plot. Must be either "people" or "medication".
`with_population`	Logical value indicating whether to include the population cluster.

Value

A ggplot object.

See Also

`comedication_count`

`plot_cluster_frequency`

`plot_medication_frequency`

`plot_timing_trajectory`

`plot_timing_atc_group`

`plot_summary`

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3)

clust |> plot_comedication_count()
clust |> comedication_count() |> plot_comedication_count()
clust |> summary() |> plot_comedication_count()
```

---

plot_medication_frequency

Plot Medication Frequency

Description

This function plots the medication frequency.

Usage

```
plot_medication_frequency(object, ...)

## S3 method for class 'medic'
plot_medication_frequency(object, ...)

## S3 method for class 'summary.medic'
plot_medication_frequency(object, ...)

## S3 method for class 'medication_frequency'
plot_medication_frequency(
  object,
  scale = "percent",
  scope = "cluster",
  with_population = FALSE,
  ...
)
```

Arguments

`object`	The object containing the medication frequency data.
`...`	Additional arguments passed to the plotting functions.
`scale`	The scale of the y-axis. Must be either "percent" or "count".
`scope`	The scope of the plot. Must be one of "cluster", "global" or "medication".
`with_population`	Logical value indicating whether to include the population cluster.

Value

A ggplot object.

See Also

`medication_frequency`

`plot_cluster_frequency`

`plot_comedication_count`

`plot_timing_trajectory`

`plot_timing_atc_group`

`plot_summary`

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3)

clust |> plot_medication_frequency()
clust |> medication_frequency() |> plot_medication_frequency()
clust |> summary() |> plot_medication_frequency()
```

`plot_summary` *Plot Summary*

Description

This function plots the summary of the clustering results.

Usage

```
plot_summary(object, ...)

## S3 method for class 'medic'
plot_summary(object, only = NULL, clusters = NULL, additional_data = NULL, ...)

## S3 method for class 'summary.medic'
```

```
plot_summary(
  object,
  n_breaks = 5,
  plot_individual = FALSE,
  labels = FALSE,
  alpha_individual = 0.1,
  label_y_value = 0.1,
  ...
)
```

Arguments

`object`	The object containing the summary data.
`...`	Additional arguments passed to the plotting functions.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`n_breaks`	The number of breaks for the time scale.
`plot_individual`	Logical value indicating whether to plot individual trajectories.
`labels`	Logical value indicating whether to include labels.
`alpha_individual`	The alpha value for the individual trajectories.
`label_y_value`	A number between 0 and 1 that defines the height of the label text hight.

Value

A ggplot object.

See Also

`summary`

`plot_cluster_frequency`

`plot_medication_frequency`

`plot_comedication_count`

`plot_timing_trajectory`

`plot_timing_atc_group`

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3,
  timing = first_trimester:third_trimester
)

clust |> plot_summary()
clust |> summary() |> plot_summary()

# If the clustering object contains more than one clustering, it is necessary
# to filter the clustering, as only one clustering can be plotted at a time.
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)
clust |> plot_summary(only = k == 4)
clust |> summary(only = k == 4) |> plot_summary()
```

`plot_timing_atc_group` *Plot Timing ATC Group*

Description

This function plots the timing ATC group.

Usage

```
plot_timing_atc_group(object, ...)

## S3 method for class 'medic'
plot_timing_atc_group(object, ...)

## S3 method for class 'summary.medic'
plot_timing_atc_group(object, ...)

## S3 method for class 'timing_atc_group'
plot_timing_atc_group(
  object,
  focus = "average",
  with_population = FALSE,
  max_lines = 50,
  ...
)
```

Arguments

`object`	The object containing the timing ATC group data.
`...`	Additional arguments passed to the plotting functions.
`focus`	The focus of the plot. Must be either "average", "individual" or "both".
`with_population`	Logical value indicating whether to include the population cluster.
`max_lines`	The maximum number of lines to plot.

Value

A ggplot object.

See Also

`timing_atc_group`

`plot_cluster_frequency`

`plot_medication_frequency`

`plot_comedication_count`

`plot_timing_trajectory`

`plot_summary`

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)

clust |> plot_timing_atc_group()
clust |> timing_atc_group() |> plot_timing_atc_group()
clust |> summary() |> plot_timing_atc_group()
```

`plot_timing_trajectory`

Plot Timing Trajectory

Description

This function plots the timing trajectory.

Usage

```
plot_timing_trajectory(object, ...)

## S3 method for class 'medic'
plot_timing_trajectory(object, ...)

## S3 method for class 'summary.medic'
plot_timing_trajectory(object, ...)

## S3 method for class 'timing_trajectory'
plot_timing_trajectory(
  object,
  focus = "average",
  with_population = FALSE,
  max_lines = 50,
  ...
)
```

Arguments

`object`	The object containing the timing trajectory data.
`...`	Additional arguments passed to the plotting functions.
`focus`	The focus of the plot. Must be either "average", "individual" or "both".
`with_population`	Logical value indicating whether to include the population cluster.
`max_lines`	The maximum number of lines to plot.

Value

A ggplot object.

See Also

`timing_trajectory`

`plot_cluster_frequency`

`plot_medication_frequency`

`plot_comedication_count`

`plot_timing_atc_group`

`plot_summary`

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
```

```
  k = 3:5,
  timing = first_trimester:third_trimester
)

clust |> plot_timing_trajectory()
clust |> timing_trajectory() |> plot_timing_trajectory()
clust |> summary() |> plot_timing_trajectory()
```

`print.summary.medic` *Print Summary of Medication*

Description

This function prints a summary of medication information.

Usage

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

Arguments

`x`	An object of class `summary.medic`.
`...`	currently only included for compatibility with generic. Has no effect.

Details

This function prints various information about medication, including cluster frequency, medication frequency, number of different medication taken in the study period, average exposure trajectories, and average exposure trajectories by ATC groups.

Value

The function is called for its side effects and does not return any value.

`refactor` *Refactor Cluster Levels*

Description

Refactor the levels of the chosen clusters.

Usage

```
refactor(object, ..., inheret_parameters = TRUE)
```

Arguments

`object`	A `medic` object.
`...`	<`data-masking`> Name-value pairs. `...` is passed to `dplyr::mutate`, and therefor inherits its behavior: The name gives the name of the new clustering in the output. The value can be: • A vector of length 1, which will be recycled to the correct length. • A function of another clustering. When a recording uses the name of an existing clustering, this new clustering will overwrite the existing one.
`inheret_parameters`	A logical. If `TRUE` a new clustering overwriting an existing clustering inherits the `parameters` of the old.

Value

A `medic` object with relevant clusterings refactored.

Examples

```
clust <- medic(complications, id = id, atc = atc, k = 3:4)

# Refactor one clustering
refactor(
  clust,
  `cluster_1_k=4` = dplyr::recode(`cluster_1_k=4`, IV = "III")
)

# Refactor all clusterings
refactor(
  clust,
  dplyr::across(
    dplyr::everything(),
    ~dplyr::recode(., IV = "III")
  )
)
```

`str.summary.medic` *Summary of a medic-object using str function*

Description

Summary of a medic-object using str function

Usage

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

Arguments

`object`	A `medic` object.
`...`	Additional arguments passed to `str.default`. This function provides a summary of an object by using the `str` function. It is a modified version of the `str.default` function from the `utils` package, with the maximum level set to 2.

`summary.medic` *Summary of medic object*

Description

Make cluster characterizing summaries.

Usage

```
## S3 method for class 'medic'
summary(
  object,
  only = NULL,
  clusters = NULL,
  outputs = "all",
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`outputs`	A character vector naming the desired characteristics to output. The default names all possible output types.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Value

A list of clustering characteristics of class `summary.medic` is returned. It can contain any of the following characteristics:

Cluster Frequencies:

The number of individuals assigned to each cluster and the associated frequency of assignment.

Medication Frequencies:

The number of individuals with a specific ATC code within a cluster. Moreover, it calculates the percentage of people with this medication assigned to this cluster and the percent of people within the cluster with this medication.

Comedication Count:

The number of ATC codes an individual has, and then outputs the number of individuals within a cluster that has that many ATC codes. Moreover, various relevant percentages or calculated. See Value below for more details on these percentages.

Timing Trajectories:

The number of unique timing trajectories in each cluster, and the average timing trajectories in each cluster.

Timing and ATC group interactions:

The number of people with unique timing trajectory and ATC group, as given by `atc_groups`, in each cluster.

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)
summary(clust)
```

`summary_crop` *Crop Clustering Summary*

Description

Functions for cropping summarized cluster data.

Usage

```
summary_crop(object, ...)

## S3 method for class 'cluster_frequency'
summary_crop(object, top_n = 5L, min_count = 0, min_percent = 0, ...)

## S3 method for class 'medication_frequency'
summary_crop(
  object,
  top_n = 5L,
  min_count = 0,
  min_percent = 0,
  scope = "cluster",
  ...
)

## S3 method for class 'comedication_count'
summary_crop(object, ...)

## S3 method for class 'timing_trajectory'
summary_crop(object, sample_n_individual = 100L, weighted_sample = TRUE, ...)

## S3 method for class 'timing_atc_group'
summary_crop(
  object,
  sample_n_individual = 100L,
  weighted_sample = TRUE,
  min_count = 0L,
  ...
)

## S3 method for class 'summary.medic'
summary_crop(object, which = "all", ...)
```

Arguments

`object`	The summary object to be cropped.
`...`	Additional arguments to be passed to the specific method.
`top_n`	integer. In the case of `cluster_frequency` it is the number of clusters to keep. In the case of `medication_frequency` it is the number of medications to keep. If `inf`, all clusters or medications are kept.
`min_count`	integer. The minimum count of a cluster or medication to keep it in the summary. If 0, the default, the minimum count is zero, i.e. there is not a minimum count.
`min_percent`	numeric. The minimum percentage of a cluster or medication to keep it in the summary. If 0, the default, the minimum percentage is zero, i.e. there is not a minimum percentage.

`scope`	character. The scope of the summary crops `top_n`, `min_count` and `min_percent`. The options are "cluster" and "global". The default is "cluster". If "cluster", the crop is based on the percentage of medication in the cluster. If "global", the crop is based on the percentage of all medication.
`sample_n_individual`	a logical or integer. If FALSE, no individual timing trajectories are sampled. If integer, `sample_n_individual` is the number of individual timing trajectories to sample. To sample all individual timing trajectories, set `sample_n_individual` to `Inf`.
`weighted_sample`	a logical, but only used if `sample_n_individual` is an integer. If TRUE, the individual timing trajectories are sampled weighted by the number of medications in the individual timing trajectory. If FALSE, the individual timing trajectories are sampled uniformly.
`which`	A character vector specifying which summaries to crop. The options are "cluster_frequency", "medication_frequency", "comedication_count", "timing_trajectory", and "timing_atc_group". The default is "all".

Value

A summary object, which is a modified version of the input summary object.

`cluster_frequency` summary crop

Extracts the top `top_n` clusters by count. If `top_n` is `Inf`, all clusters are kept. If `min_count` is greater than 0, clusters with a count less than `min_count` are removed. If `min_percent` is greater than 0, clusters with a percentage less than `min_percent` are removed. The remaining clusters are grouped into a "Remaining" cluster.

`medication_frequency` summary crop

Extracts the top `top_n` medications by count. If `top_n` is `Inf`, all medications are kept. If `min_count` is greater than 0, medications with a count less than `min_count` are removed. If `min_percent` is greater than 0, medications with a percentage less than `min_percent` are removed. The remaining medications are grouped into a "Remaining" cluster.

The `scope` argument determines the scope of the crop. If `scope` is "cluster", the crop is based on the percentage of medication in the cluster. If `scope` is "global", the crop is based on the percentage of all medication.

`comedication_count` summary crop

TO DO

`timing_trajectory` summary crop

Samples `sample_n_individual` individual timing trajectories. If `sample_n_individual` is `Inf`, all individual timing trajectories are kept. If `weighted_sample` is `TRUE`, the individual timing trajectories are sampled weighted by the number of medications in the individual timing trajectory.

`timing_atc_group` **summary crop**

Samples `sample_n_individual` individual timing trajectories. If `sample_n_individual` is `Inf`, all individual timing trajectories are kept. If `weighted_sample` is `TRUE`, the individual timing trajectories are sampled weighted by the number of medications in the individual timing trajectory.

`summary.medic` **summary crop**

Crops multiple summaries. The `which` argument is a character vector specifying which summaries to crop. The options are "cluster_frequency", "medication_frequency", "comedication_count", "timing_trajectory", and "timing_atc_group". If `which` is "all", all summaries are cropped.

The `...` argument is passed to the specific methods, e.g. `top_n` and `min_count` are passed to `cluster_frequency` and `medication_frequency`.

See Also

`summary`, `cluster_frequency`, `medication_frequency`, `comedication_count`, `timing_trajectory`, `timing_atc_group`

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)


# Crop the cluster frequency summary
clust |>
  cluster_frequency() |>
  summary_crop(top_n = 3)

clust |>
  summary() |>
  summary_crop(which = "cluster_frequency", top_n = 3)


# Crop the medication frequency summary
clust |>
  medication_frequency() |>
  summary_crop(top_n = 3)

clust |>
  summary() |>
  summary_crop(which = "medication_frequency", top_n = 3)


# Crop the co-medication count summary
clust |>
```

```
    comedication_count() |>
    summary_crop(min_count = 10)

  clust |>
    summary() |>
    summary_crop(which = "comedication_count", min_count = 10)


  # crop the timing trajectory summary
  clust |>
    timing_trajectory() |>
    summary_crop()

  clust |>
    summary() |>
    summary_crop(which = "timing_trajectory")


  # crop the timing ATC group summary
  clust |>
    timing_atc_group() |>
    summary_crop()

  clust |>
    summary() |>
    summary_crop(which = "timing_atc_group")

  # crop multiple summaries
  clust |>
    summary() |>
    summary_crop(
      which = c("cluster_frequency", "medication_frequency"),
      top_n = 3
    )
```

`timing_atc_group` *Timing and ATC pattern interactions*

Description

The function `timing_atc_group()` calculates the frequencies of distinct timing and ATC combinations within clusters.

Usage

```
timing_atc_group(
  object,
  only = NULL,
  clusters = NULL,
```

```
  atc_groups = default_atc_groups,
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.
`atc_groups`	A data.frame specifying the ATC groups to summaries by or a funciton that returns such a data.frame. The data.frame must have two columns: • `regex` giving regular expressions specifying the wanted ATC groups. • `atc_groups` the name of this ATC grouping. As a standard the anatomical level (first level) of the ATC codes is used.
`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Details

`timing_atc_group()` calculates both the number of people with unique timing trajectory and ATC group, as given by `atc_groups`, in each cluster.

Value

`timing_atc_group()` returns a list of class `timing_atc_group` with two data frames:

average:

- `Clustering` the name of the clustering.
- `Cluster` the name of the cluster.
- `ATC Groups` the name of the ATC group. The groups are given by the `atc_groups` input.
- *timing variables* the average timing value in the ATC group and cluster.
- `Number of Medications` the number of medications in the ATC group in the cluster.
- `Percentage of Medications` the percentage of medication in the cluster with this ATC group.
- `Number of Distinct Timing Trajectories` the number of unique timing trajectories in the ATC group in the cluster.

individual:

- `Clustering` the name of the clustering.
- `Cluster` the name of the cluster.
- *timing variables* a unique timing pattern in the ATC group and cluster.
- `Number of Medications with Timing Trajectory` the number of medications with this unique timing trajectory and ATC group.

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)

timing_atc_group(clust, k == 5, clusters = I:III)
```

`timing_trajectory` *Timing pattern frequency within clusters*

Description

`timing_trajectory()` calculates the average timing paths within clusters.

Usage

```
timing_trajectory(
  object,
  only = NULL,
  clusters = NULL,
  additional_data = NULL,
  ...
)
```

Arguments

`object`	An object for which a summary is desired.
`only`	<`data-masking`> Expressions that return a logical value, and are defined in terms of the variables in `object` and/or `additional_data`. The default `NULL` selects all clusterings in `object`.
`clusters`	<`tidy-select`> An unquoted expression naming the cluster or clusters in `object` one wants to see summaries of. Names can be used as if they were positions in the data frame, so expressions like I:IV can be used to select a range of clusters. The default `NULL` selects all clusters in the chosen clusterings of `object`.

`additional_data`	A data frame with additional data that may be (left-)joined onto the `parameters` in `object`. This is often used in conjuction with `only` to select specific clusterings based on `additional_data`.
`...`	Additional arguments passed to the specific summary sub-function.

Details

`timing_trajectory()` calculates both the number of unique timing trajectories in each cluster and the average timing trajectories in each cluster.

Value

`timing_trajectory()` returns a list of class `timing_trajectory` with two data frames:

average:

- `Clustering` the name of the clustering.
- `Cluster` the cluster name.
- *timing variables* the average timing value in the cluster.
- `Count` the number of people in the cluster.

individual:

- `Clustering` the name of the clustering.
- `Cluster` the cluster name.
- *timing variables* unique timing pattern in the cluster.
- `Count` number of people with this unique timing pattern.

Examples

```
clust <- medic(
  complications,
  id = id,
  atc = atc,
  k = 3:5,
  timing = first_trimester:third_trimester
)

timing_trajectory(clust, k == 5, clusters = I:III)
```

Index